Blocks of Code (v3)

Reimagine the Building Blocks of Programming

Group 28 Authors:

Jordan Merkel	Camilla Torres	Annesley Kolb	Destiny Ellenwood
Computer Engineering	*Electrical Engineering*	*Electrical Engineering*	*Computer Engineering*

Reviewer Committee:

Dr. Mike Borowczak	ECE	Associate Professor
Dr. Shady Elashhab	ECE	Lecturer
Dr. Sonali Das	ECE	Lecturer
Dr. Nazanin Rahnavard	ECE	Lecturer

Advisors:

Dr. Mike Borowczak	ECE	Associate Professor
Dr. Arthur Weeks	ECE	Senior Lecturer
Dr. Sreeram Sundaresh	ECE	Lecturer

Table of Contents

Chapter 1 – Executive Summary 14
Chapter 2 – Project Description 15
2.1 Motivation and Background 15
2.2 Existing Products, Past Projects, and Prior Related Work 16
2.2.1 Prior Related Work 16
2.2.2 Existing Commercial Products 16
2.2.3 Existing Coding Platforms 17
2.2.4 Summary and Relevance 18
2.3 Goals 18
2.3.1 Basic Goals 18
2.3.2 Advanced Goals 18
2.3.3 Stretch Goals 19
2.4 Objectives 19
2.4.1 Basic Objectives 19
2.4.2 Advanced Objectives 19
2.4.3 Stretch Objectives 20
2.5 Description of Features and Functionalities 20
2.5.1 Physical Block Functionalities 20
2.6 Prototype Illustration 22
2.7 House of Quality 23
2.8 Engineering Specifications 23
2.9 Budget and Financing 24
2.10 Milestones 25
2.11 Software Flowchart 27
2.12 Hardware Flowchart 28
Chapter 3 – Research and Investigation 28
3.1 Microcontroller 28
3.1.1 ESP32-C3 28
3.1.2 ESP32-S3 29
3.1.3 Raspberry Pi 2040 29

3.1.4 Arduino ATMega....29
3.1.5 ESP32-WROOM-32....30
3.1.6 MCU Selection....30
3.1.7 Senior Design 2 Selection....32
3.2 Voltage Regulation....32
3.2.1 Linear Regulator: TLV733P-3.3 (TI)....33
3.2.2 Switching Regulator: TPS63060 (TI)....33
3.2.3 Buck Regulator: LM2576T-ADJ....34
3.2.4 Boost Regulator: TPS61023....35
3.2.5 Voltage Regulator Selection....36
3.3 Charging Interfaces....37
3.3.1 USB-C: hiBCTR B0FPCL44K7....37
3.3.2 Magnetic pogo-pins: Attend 303C-C4115-30-04....38
3.3.3 Qi Wireless: Adafruit Qi Wireless Receiver Module....38
3.3.4 Charging Interface selection....39
3.4 Charging IC....40
3.4.1 MCP73831T....40
3.4.2 MCP73832T....41
3.4.3 TP4056....41
3.4.4 ME4057....42
3.4.5 TP5100....42
3.4.6 Charging IC Selection....43
3.5 Display....44
3.5.1 OLED Display....44
3.5.2 Segment LCDs....45
3.5.3 LED Matrix Display....46
3.5.4 TTL TFT Display....47
3.5.5 Display Selection....48
3.5.6 Senior Design 2 Selection....50
3.6 LEDs and Lighting....50
3.6.1 Discrete LEDs....50
3.6.2 Addressable RGB LEDs....51
3.6.3 Lighting Selection....52

3.6.4 Senior Design 2 Selection 53
3.7 Speakers 53
3.7.1 Same Sky (CUI) CMS-2580-05SP 54
3.7.2 PUI audio AS03608MR-10-R 54
3.7.3 Soberton SP-2305L 55
3.7.4 Speaker Selection 55
3.8 Amplifiers 56
3.8.1 PAM 8302A: Class-D (Analog/PWM input) 56
3.8.2 MAX98357A: Class-D ($I2S$ + integrated DAC) 57
3.8.3 TPA2028D1 (TI): Class-D (AGC/DRC over $I2C$) 57
3.8.4 TAS2562 (TI): Class-D with integrated boost (digital I²S TDM input) 58
3.8.5 Amplifier selection 58
3.8.6 Senior Design 2 Selection 60
3.9 Filters 61
3.9.1 Input High-Pass (AC-Coupling) 61
3.9.2 PWM smoothing Low-Pass 61
3.9.3 Output EMI Tidy for Filterless Class-D 62
3.9.4 Filter Selection 63
3.10 Microphone 64
3.10.1 Adafruit I²S MEMS Microphone Breakout - SPH0645LM4H 64
3.10.2 Adafruit Electret Microphone Amplifier - MAX4466 65
3.10.3 INMP441 I²S Microphone 65
3.10.4 Microphone Selection 65
3.10.5 Senior Design 2 Selection 66
3.11 Connectors and Interfacing 66
3.11.1 Magnetic Pogo-Pin Connectors: Adafruit Industries LLC - 5358 67
3.11.2 Board-to-Board and Header Connectors 67
3.11.3 Magnetic Coupling and Contactless Interfaces: ADuM1201 68
3.11.4 Connector and Interfacing Selection 69
3.12 Power Supply Unit 70
3.12.1 Battery: 18650 Li-ion Cell (Protected) 71
3.12.2 Wall Outlet 71
3.12.3 Miniature Power Banks 72

3.12.4 Power Supply Selection 72
3.13 Communication Protocols 73
3.13.1 I²C (Inter-Integrated Circuit) 73
3.13.2 SPI (Serial Peripheral Interface) 74
3.13.3 Wi-Fi/BLE 76
3.13.4 UART (Universal Asynchronous Receiver-Transmitter) 77
3.13.5 Protocol Selection 78
3.13.6 Senior Design 2 Selection 80
3.14 Memory Storage 80
3.14.1 SRAM 80
3.14.2 Flash ROM 81
3.14.3 EEPROM 81
3.14.4 Memory Storage Selection 81
3.14.5 Senior Design 2 Selection 82
3.15 Version Control Systems 82
3.15.1 Git 83
3.15.2 Github 83
3.15.3 BitBucket 83
3.15.4 Version Control System Selection 83
3.16 Embedded Framework 84
3.16.1 ESP-IDF 84
3.16.2 Arduino 84
3.16.3 PicoSDK 85
3.16.4 Embedded Framework Selection 85
3.16.5 Senior Design 2 Selection 85
3.17 Companion Application Framework 86
3.17.1 React Native 86
3.17.2 Unity 86
3.17.3 Flutter 86
3.17.4 Companion App Framework Selection 87
Chapter 4 – Standards and Design Constraints 87
4.1 Industrial Standards 88
4.1.1 Electrical Standards 88

4.1.2 Communication Standards....89
4.1.4 Mechanical and Manufacturing Standards....89
4.2 Design Constraints....89
4.2.1 Power Constraints....89
4.2.2 Size and Layout Constraints....90
4.2.3 Memory Constraints....91
4.2.4 Communication Constraints....91
4.3 Practical Constraints....92
4.3.1 Time....92
4.3.2 Budget....93
4.3.3 Environmental....93
4.3.4 Manufacturability....94
4.4 Other Constraints....95
4.4.1 Sustainability....95
4.4.2 Political....96
4.4.3 Ethical....96
4.4.4 Health and Safety....97
4.4.5 Scalability....97
Chapter 5 – Application of ChatGPT and Other Similar Platforms....98
5.1 Introduction....98
5.2 Benefits of Using Generative AI....98
5.2.1 Time Efficiency....98
5.2.2 Expanded Research Scope....99
5.2.3 Accessibility and Learning Support....100
5.2.4 Brainstorming and Design Exploration....100
5.2.5 Writing and Communication Clarity....101
5.3 Limitations and Challenges....101
5.3.1 Lack of Physical Validation....101
5.3.2 Inaccuracy and Fabricated Information....101
5.3.3 Limited Project Context Retention....101
5.3.4 Confirmation Bias in AI Responses....102
5.3.5 Over-Reliance....102
5.4 Case Studies....102

5.4.1 Case Study 1 - Using ChatGPT for Deep Research and Component Sourcing 102
5.4.2 Case Study 2 - Using ChatGPT to Decide the Audio Path (Amplifier/Speaker/Filters) 103
5.4.3 Case Study 3 - ChatGPT Confirmation Bias When Deciding Charging Interfaces 104
5.5 Ethical and Academic Considerations 104
5.5.1 Accuracy and Verification 105
5.5.2 Academic Integrity 105
5.5.3 Ethical Engineering Practice 105
5.5.4 Long-term Learning 105
Chapter 6 – Hardware Design 106
6.1 System Hardware Overview 106
6.1.1 Brain PCB 106
6.1.2 Child PCB 110
6.1.3 Senior Design 2 selection 112
6.2 Subsystem Hardware Design 113
6.2.1 Power Delivery and Charging 113
6.2.2 Display and Touch 114
6.2.3 Audio 115
6.2.4 Programmer, POGO pins, EN and BOOT 116
6.2.5 Amplifier 117
6.2.6 OLED LCD Display 118
6.2.7 Push Button 119
6.2.8 Numpad 120
6.2.9 GPIO Expander 120
6.2.10 4X4 Matrix Display 122
6.2.11 1602 LCD Display 122
6.2.12 Amplifier Pin Header 123
6.2.7 Power Distribution System 124
6.3 Power Distribution Table 126
6.3.1 Senior Design 2 Power Distribution System 128
Chapter 7 – Software Design 129
7.1 System Software Overview 129

7.1.1 Firmware Overview....130
7.1.2 Companion App Overview....130
7.2 Subsystem and Components....131
7.2.1 Software Block Diagram....131
7.2.2 Component Interaction Description....131
7.3 Communication and Data Transfer....132
7.3.1 I2C Communication....132
7.3.2 SPI Communication....133
7.3.3 Wi-Fi Communication....133
7.3.4 Timing and Synchronization....134
7.4 State Diagrams....135
7.4.1 Brain Block State Machine....135
7.4.2 Child Block State Machine....136
7.4.3 Companion App State Machine....137
7.4.4 Senior Design 2 Brain Block State Machine....138
7.4.5 Senior Design 2 Child Block State Machine....139
7.4.6 Senior Design 2 Companion App State Machine....141
7.5 Data Structures and Storage....142
7.6 User Interface Design....143
7.6.1 Senior Design 2 Implementation....144
7.7 Error Handling....145
7.7.1 Communication Faults....145
7.7.2 Hardware Faults:....145
7.7.3 Invalid Block Configurations....145
7.7.4 Watchdog Timer (WDT)....146
Chapter 8 – System Fabrication/Prototype Construction....147
8.1 Brain Block PCB....147
8.2 Child Block PCB....150
8.2.1 Senior Design 2 Implementation....153
8.3 Amplifier....155
8.4 Power Distribution System....158
8.5 Block Enclosure Diagram....163
8.5.1 Exterior Enclosure....164

8.5.2 Interior Enclosure 165
Chapter 9 – System Testing and Evaluation 167
9.1 Hardware Testing 167
9.1.1 ESP32-WROOM-32 167
9.1.2 3.3V and 5V Regulator 167
9.1.3 LED Matrix 169
9.2 Software Testing 169
9.2.1 Wi-Fi Connectivity 169
9.2.2 TCP Client-Server 169
9.2.3 Flutter Application 169
9.3 Overall Integration 170
9.4 Senior Design II Results 171
Chapter 10 – Administrative Content 172
10.1 Budget 172
10.2 Bill of Materials 172
10.3 Distribution of Worktable 173
10.4 Milestones 173
Chapter 11 – Conclusion 176
11.1 Summary of Accomplishments 176
11.2 Technical Insights and Design Challenges 177
11.3 Closing Remarks 178
Appendices 179
Appendix A - References 179
Appendix B - Copyright Permissions 191
Appendix C - Data Sheet 191
Appendix D - Software Code 192
Appendix E - ChatGPT Prompts and Outcomes 196

List of Figures

Figure 2.1 SolidWorks rough draft of the cube prototype 22
Figure 2.2 Conditional logic diagram 22
Figure 2.3 House of Quality 23

Figure 2.4 Software Flowchart 27
Figure 2.5 Hardware Flowchart 28
Figure 6.1 Brain Block Hardware Diagram 106
Figure 6.2 Brain PCB Schematic 107
Figure 6.3 GPIO and Net Reference Table 108
Figure 6.4 Child Block Hardware Diagram 110
Figure 6.5 Child Block Schematics 111
Figure 6.6 Final Block Schematic 112
Figure 6.7 Power Delivery and Charging 113
Figure 6.8 TFT Display 114
Figure 6.9 LED Displays 115
Figure 6.10 Audio 116
Figure 6.11 Pins, BOOT, and EN Schematics 117
Figure 6.12 Amplifier Schematic 118
Figure 6.13 OLED LCD Display Schematic 119
Figure 6.14 Push Button Schematic 120
Figure 6.15 Numpad Schematic 121
Figure 6.16 GPIO Expander Schematic 122
Figure 6.17 4x4 Matrix Schematic 123
Figure 6.18 1602 LCD Display Schematic 124
Figure 6.19 Amplifier Pin Header Schematic 125
Figure 6.20 Charging Module 126
Figure 6.21 3.3V Voltage Regulator 127
Figure 6.22 5V Voltage Regulator 128
Figure 7.1 Software Block Diagram 130
Figure 7.2 I²C Communication 131

Figure 7.3 SPI Communication 132
Figure 7.4 Brain Block State Machine 134
Figure 7.5 Child Block State Machine 135
Figure 7.6 Companion App State Machine 136
Figure 7.7 Updated Brain Block State Machine……..138
Figure 7.8 Updated Child Block State Machine……..139
Figure 7.9 Updated Companion App State Machine……..……..141
Figure 8.1 Brain Block PCB Layout 148
Figure 8.2 Brain Block PCB 3D Model (Front) 149
Figure 8.3 Brain Block PCB 3D Model (Back) 149
Figure 8.4 Child Block PCB Layout 151
Figure 8.5 Child Block PCB 3D Model (Front) 152
Figure 8.6 Child Block PCB 3D Model (Back) 153
Figure 8.7 Final Brain/Child PCB Layout 154
Figure 8.8 Final Brain/Child PCB 3D Model (Front) 156
Figure 8.9 Final Brain/Child PCB 3D Model (Back) 157
Figure 8.10 Charging Module PCB Layout 158
Figure 8.11 Charging Module PCB 3D Model (Front) 159
Figure 8.12 Charging Module PCB 3D Model (Back) 160
Figure 8.13 Voltage Regulator PCB Layout 160
Figure 8.14 Voltage Regulator PCB 3D Model (Front) 161
Figure 8.15 Voltage Regulator PCB 3D Model (Back) 162
Figure 8.16 Enclosure Exterior Front View 163
Figure 8.17 Enclosure Exterior Back View 164
Figure 8.18 Enclosure Interior Left Side View 165
Figure 8.19 Enclosure Interior Right Side View ..... ….166

Figure 8.20 Enclosure Interior Right Side View……...166

Figure 9.1 Voltage Regulators Breadboarding 168

Figure 9.2 Electronic Load Displaying Output Voltage and Current from Regulators 168

List of Tables

Table 2.8 Component Specifications 24
Table 2.8.1 System Specifications 24

Table 2.9 Budget and Financing 25

Table 2.10 Milestones 26

Table 3.1 MCU Comparison 31

Table 3.2 Regulator Comparison 37

Table 3.3 Charging Interface Comparison 40

Table 3.4 Charging IC Comparison 44

Table 3.5 Display Comparison 50

Table 3.6 Lighting Comparison 53

Table 3.7 Speaker Comparison 56

Table 3.8 Amplifier Comparison 60

Table 3.9 Filter Comparison 64

Table 3.10 Microphone Comparison 66

Table 3.11 Connector Comparison 70

Table 3.12 PSU Comparison 73

Table 3.13 Communication Protocol Comparison 80

Table 3.14 Memory Storage Comparison 82

Table 3.15 Version Control System Comparison 84

Table 3.16 Embedded Framework Comparison 85

Table 3.17 Companion App Comparison 87

Table 6.2 Brain Block Power Distribution 127

Table 6.3 Child Block Power Distribution 128

Table 6.3.1 Block Power Distribution (Brain and Child, per block)……..……128

Table 6.3.2 System Summary……....129

Table 10.2.1 Prototype BOM 172

Table 10.2.2 Manufacturer BOM 173

Table 10.3 Worktable 173

Table 10.4.1 Milestones for SD1 174

Table 10.4.2 Milestones for SD2 175

Chapter 1 – Executive Summary

The *Blocks o' Code (v3)* project is a modular, interactive learning platform designed to teach fundamental programming concepts through tangible, magnetically connected physical blocks. Building on the foundation established by earlier versions of *Blocks o' Code,* this third iteration significantly expands the system's technical depth, interactivity, and educational value. The primary goal of version 3 is to combine low-cost embedded hardware, intuitive block-based interfaces, synchronized multimedia outputs, and a companion mobile application into a cohesive learning ecosystem suitable for classrooms, STEM camps, and at-home learning.

At its core, the system consists of a central **Brain Block** and multiple **Child Blocks**. Each block contains its own ESP32-WROOM, enabling distributed processing and flexible functionality. The blocks snap together using magnetic pogo-pin connectors that will carry power and I²C data, allowing users to physically "assemble" a program. When connected, the Brain Block detects the configuration, retrieves data from each Child Block, and executes the sequence based on the user-defined arrangement. This experience emphasizes hands-on exploration rather than writing lines of code. The students can directly create logic structures such as loops, delays, input events, and output actions using physical components.

This version 3 redesign focuses heavily on expanding both capability and user engagement. New block categories have been created such as note selection blocks for music creation, LED and display outputs, logic-control, and voice-responsive input modules. This will allow students to construct increasingly sophisticated programs. A key design goal is to ensure that these blocks are robust and intuitive to make the system scalable for larger classroom deployments. Additional advanced features such as synchronized music-and-light "disco mode," microphone integration, OLED displays, and potential wireless charging enhancements aim to push the system toward a richer interactive experience.

A companion mobile application forms the second major subsystem of the project. Using Wi-Fi communication with the Brain Block, the app provides users with a real-time visualization of the constructed block arrangement. It displays detected errors such as invalid connections, missing required blocks, or mis-ordered logic. When a configuration is valid, the app allows users to monitor behavior and view synchronized or music-driven visuals. This integration bridges physical and digital learning, reinforcing key computational concepts in a multimodal way.

The primary objectives of the project center on achieving functional, stable, and educationally meaningful prototypes of the Brain Block and multiple Child Blocks. These objectives include establishing a reliable I²C communication protocol, developing magnetically connecting enclosures that ensure proper alignment and signal integrity, implementing firmware for basic and advanced control-flow operations, and synchronizing audio-visual output across blocks. Higher-level objectives include expanding the range of block types, increasing interactivity through sensors, improving timing accuracy for coordinated outputs, and reducing the per-block cost through component optimization.

Stretch objectives explore future directions such as orientation agnostic connectors, wireless charging stations, microphones using machine learning, and enhanced companion app features.

By the conclusion of Senior Design 1, the team aims to deliver a fully documented, partially implemented system capable of demonstrating block-to-block communication, basic program execution, and coordinated outputs across modules. Hardware prototypes for the Brain and Child Blocks will be completed, along with PCB designs, and preliminary firmware. The companion app will be functional enough to provide a user interface for starting or stopping sequences. These deliverables collectively form the foundation for Senior Design 2, where full system integration, expanded block sets, audio-visual features, and refined user experience will be completed.

Ultimately, *Blocks o' Code (v3)* is designed to transform the way children engage with programming. By merging embedded systems, real time systems, interactive hardware, synchronized outputs, and user-friendly software, the project delivers an accessible platform that reimagines coding as a tactile, creative, and highly engaging experience.

Chapter 2 – Project Description

2.1 Motivation and Background

Block-based programming has become one of the most popular and effective ways to introduce children to coding. Platforms such as Scratch and Blockly emphasize drag-and-drop blocks over typed syntax, reducing barriers to entry and allowing beginners to focus on problem solving and creativity. This approach has been widely adopted in classrooms because it transforms programming into a more visual and approachable activity. Learners can see coding as the arrangement of colorful blocks that fit together like puzzle pieces, which helps them understand sequencing, logic, and cause-and-effect relationships without being overwhelmed by technical details.

Our project builds on this same philosophy but extends it into the physical world. Rather than dragging blocks on a screen, children will be able to snap together tangible coding blocks to form programs. Each block represents a different programming concept such as input, output, or control flow, and the process of assembling them becomes a literal, hands-on way of coding. Research in tangible programming interfaces (TPIs) shows that physically embodied coding tools can be especially effective in classrooms, giving children a more intuitive and engaging way to learn. This tangible approach lowers barriers even further, particularly for younger children and those with disabilities who may benefit from tactile interaction over screen-based systems.

In addition to accessibility, our design emphasizes multi-sensory feedback. When a child assembles a sequence of blocks, the outputs will be seen through LED lights and visual screen displays and heard through sound and music. By combining physical coding with multisensory outputs, our system provides immediate reinforcement of programming concepts in ways that can be understood and enjoyed by children with different abilities.

Comparable products in the market, such as LEGO Education SPIKE, Cubelets, and Cubetto, demonstrate the growing demand for screen-free coding experiences. However, most of these focus on robotics or visual problem solving, with limited attention to accessibility or multisensory inclusiveness. Our project fills this gap with a disco party theme, turning coding into a playful, accessible, and inclusive experience. By merging physical block programming with lights, music, and real-time app visualization, the system offers children a unique way to explore computational thinking by literally holding their programs in their hands while experiencing the results in a fun, multi-sensory way.

2.2 Existing Products, Past Projects, and Prior Related Work

2.2.1 Prior Related Work

2.2.1.1 Blocks o' Code Version 1 (V1).

The first version of *Blocks o' Code* was developed by Erebus Labs in 2015. Unlike the worksheet prototype, this version used real hardware. Each block was a four-layer PCB enclosed in acrylic, containing an ATtiny461a microcontroller, indicator LEDs, and a selector wheel connected to a potentiometer. Blocks communicated with each other using SPI for local connections and a TWI bus for global communication, with a BeagleBone Black acting as the main processor. The processor board ran a lexer, parser, and interpreter to process the block arrangements, and could drive different outputs like an LCD screen, RGB LEDs, a piezo buzzer, or a relay [1]. This made V1 a significant step toward tangible programming because it was interactive, but the system was expensive, somewhat fragile, and limited in scalability.

2.2.1.2 Blocks o' Code Version 2 (V2).

The second version was created by UCF Senior Design Group 13 in Fall 2023–Spring 2024. V2 took the worksheet idea and transformed it into physical blocks with embedded electronics. Each block had its own function, and when connected, the blocks formed programs that could be executed. Another big step was the introduction of a companion app. The app allowed users to visualize the block arrangements, see how they translated into commands, and interact digitally [2]. This combination of hardware and software was a strong milestone, but V2 still struggled with connection reliability, scalability, and an app that was basic compared to what learners are used to today.

2.2.2 Existing Commercial Products

2.2.2.1 MatataLab Musician Add-On.

MatataLab created a kit that teaches coding through music. With this add-on, students arrange blocks to compose songs, turning coding lessons into playful and creative activities. This shows the power of mixing logic with art and is especially important to our project since we are also focusing on music to teach coding [3]. The MatataLab musician kit proves that students are excited to learn when coding is combined with music, and this directly supports the direction we are taking with our project V3.

2.2.2.2 Cubroid Coding Blocks.

Cubroid is a wireless coding block system where each block has its own function, like motors, sensors, or lights. It demonstrates how coding can be hands-on and interactive in a

classroom setting [4]. Cubroid's strength is that it makes learning fun and immediate, but its reliance on proprietary modules makes it less customizable and harder to scale for more advanced engineering use.

2.2.2.3 Tynker Sound Blocks.

Tynker is a popular digital coding platform. One of its features is sound blocks, which let students add music, sound effects, or voice into their programs. Even though Tynker does not use physical blocks, it is highly relevant to our project because it shows how a companion app can keep learners engaged through multimedia features [5]. Since V2 already had a basic app, Tynker inspires us to expand ours with real-time audio feedback and stronger integration with the physical blocks.

2.2.2.4 LEGO Education SPIKE

LEGO SPIKE combines LEGO bricks with programmable hubs, sensors, and motors, all controlled through a block-based app [6]. It is widely used in STEM classrooms and competitions to teach problem solving and coding through play. SPIKE proves the effectiveness of combining hands-on construction with digital coding, and it serves as an important reference for how our project combines physical coding blocks with a companion app.

2.2.2.5 Cubelets

Cubelets are small robot blocks that connect magnetically, with each cube representing a different behavior such as sensing, logic, or output. By snapping them together, students can build robots without needing wiring or screens [7]. Cubelets highlight the appeal of simple, tactile system, though they lack deeper coding features and do not integrate music or app-based feedback like our project does.

2.2.2.6 Cubetto

Cubetto is a wooden robot designed for early learners that is programmed using physical blocks placed on a board. It is screen-free and introduces sequencing and logic in an approachable way for young children [8]. Cubetto shows how tangible programming can reach very young learners, but it is limited to very basic coding concepts and does not expand into multimedia experiences.

2.2.3 Existing Coding Platforms

2.2.3.1 Scratch

Scratch is one of the most popular visual coding platforms for kids. Students drag and drop blocks to create animations, games, and even music [9]. It emphasizes creativity and storytelling, which aligns with our vision for using music and playful interactions in our project V3.

2.2.3.2 Blocky

Blocky is Google's visual programming library that powers many educational apps [10]. It shows how block-based coding can be directly translated into real code, which inspires how our companion app will display the logic of physical arrangements.

2.2.4 Summary and Relevance

Looking back at earlier versions of *Blocks o' Code* and exploring commercial systems gave us a clear direction for Version 3. V1 and V2 established the foundation, showing that tangible blocks combined with an app can be an effective way to teach coding. MatataLab demonstrates how music can make coding playful and engaging, which matches directly with our vision of using music to teach. Cubroid shows how interactive block systems can succeed in classrooms but also reveals the limits of proprietary, closed systems. Tynker highlights the importance of companion apps with multimedia features, which motivates us to make our app more advanced and engaging.

By combining the lessons learned from V1 and V2 with inspiration from these commercial products, our Version 3 project will be more reliable, expandable, and fun. With stronger block hardware, integrated music features, and an enhanced app, we aim to create a system that not only continues the *Blocks o' Code* legacy but also pushes it further into a more creative and interactive direction.

Our team is developing the third version of *Blocks o' Code*. We are improving hardware reliability, making the system more expandable, and redesigning the companion app to be more powerful and fun. One of our biggest focuses is bringing music into action. While V2 has a basic app, our V3 app will provide real-time audio and visual feedback, inspired by how kids already enjoy music-based coding. By combining stronger block hardware with an app that integrates multimedia like sound and music, V3 will create a richer and more engaging experience than past iterations.

2.3 Goals

2.3.1 Basic Goals

- Design and implement at least 15 fully functional blocks, each containing a low-cost MCU (ESP32-WROOM-32) with sufficient GPIO pins for peripherals and I^2C communication capability.
- Support up to 15 blocks per configuration connected via magnetic connectors carrying power (3.3-5V) and I^2C data lines.
- Develop magnetic snap-together physical coding blocks that transfer data to a master "Brain" block MCU.
- Support core control-flow logic at the firmware level: Start/Stop execution flag, If/Then Boolean branching, Loop counters, and Delay timers.
- Provide visual indication of block functionality on each block.
- Include at least two output modalities:
  - LED driver block with PWM control (1-10 kHz)
  - Piezo speaker block with tone generation (100 Hz-2kHz)
- 3D-printable enclosures with embedded magnet alignment and PCB mounting.

2.3.2 Advanced Goals

- Introduce data blocks: Boolean (yes/no), Integer (counters, values).

- Implement synchronized music + disco effects across blocks using I^2C broadcast packets from Brain MCU.

- Ensure low-power operation with per-block consumption of <100mA during active mode

2.3.3 Stretch Goals

- Orientation-free block connectors with rotational symmetry to allow blocks to connect from any direction.
- Implement wireless charging via magnetic coupling for each block when stored in a charging drawer.
- Enable a companion app leveraging the ESP32 Wi-Fi module that shows a real-time visualization of block configuration and generates a live disco visualization.
- Add OLED displays to blocks on selected blocks for a richer output.
- Provide randomized sequence generation for music and LED patterns.
- Add more input options:
    - Microphone sensor block with ADC sampling (~8kHz)
    - Voice input keyword detection using an ML model

2.4 Objectives

2.4.1 Basic Objectives

- Select and program a low-cost MCU per block (Arduino R4 Minima for simplicity, or ESP32-WROOM-32 for wireless capability).
- Establish a reliable data-sharing protocol over I²C between blocks with the Brain MCU serving as master.
- Design and 3D-print magnetically connecting enclosures with pogo-pin data/power contacts.
- Implement firmware for core control blocks (Start, If/Then, Loop, Delay) and ensure correct sequencing across configurations.

2.4.2 Advanced Objectives

- Expand the block set to include audio input (microphone/voice), motion sensors, and data-handling blocks.
- Develop synchronized audio-light effects across multiple output blocks with millisecond-level timing accuracy.

- Implement power management strategies (sleep modes, duty cycling) to extend runtime.

2.4.3 Stretch Objectives

- Design orientation-independent magnetic connectors for maximum user flexibility.
- Integrate wireless charging coils + drawer station to simplify recharging.
- Extend firmware and app integration to a real-time visualization platform (via ESP32 Wi-Fi link).
- Add high-resolution OLED displays and optional randomized music/LED effect generation for extended engagement.

2.5 Description of Features and Functionalities

Blocks of Code will contain 15 or more 3D-printed physical programming blocks that serve different functionalities. When assembled into a configuration, the blocks will work together to create an interactive "disco-party" experience. This combination will also introduce children to programming concepts such as sequencing, logic, and iteration, while the music-and-lights theme keeps the experience engaging and fun.

2.5.1 Physical Block Functionalities

2.5.1.1 Input Blocks

- Start "Brain" Block
  - The Brain Block acts as the central controller and is required for any configuration. It manages Wi-Fi communication with the companion app, synchronizes data between Child Blocks, and serves as the main event handler. The Brain Block gathers data via I²C from connected Child Blocks, initializes the program sequence, and controls execution.
- Note Block
  - The Note Block allows users to select specific musical notes (A–F) using a numpad or pushbutton interface. When activated, it outputs a tone through the speaker module. It may be used individually or as part of a larger music sequence.
- Basic Input Block
  - This block detects user interactions through either button presses or clap/voice inputs, serving as a general-purpose trigger for conditional or sequential actions. The Basic Input Block can be used to activate events like note playback, LED flashes, or logic sequences.

2.5.1.2 Control Flow Blocks

- If/Then/EndIf Block Configuration
  - This block configuration introduces conditional logic by linking an input event to a specific output action. It receives signals from an input block (e.g., Microphone) and triggers the output block when conditions are met. The proper syntax is:
    - If → Input Block → Then → Output Block → EndIf
- Loop/EndLoop Configuration
  - The Loop Block repeats a chosen output action a specified number of times. Users can select loop count and repetition target using a rotary or numpad

input interface. It supports the concept of repetition and iteration within a music or lighting sequence. The proper syntax is:
        - Loop → Output Block (can be multiple) → EndLoop
- Delay
    - This block introduces a timed pause between sequential outputs. Users specify delay duration (in seconds) via pushbuttons or selection controls. The block includes feedback LEDs indicating when a delay is active. It ensures timing precision between note playback and LED flashes.

2.5.1.3 Output Blocks

- LED Color Flash Block
    - The LED Block provides colorful visual feedback using either a single RGB LED or an LED matrix. Users can select colors via pushbuttons or numpad controls. The block can operate in different visual modes (flash, fade, or pulse) to synchronize with the music sequence or respond to user input.
- Music Sequence Block
    - This versatile output block can play pre-configured note sequences to demonstrate programmed routines. It helps teach musical structure and sequencing concepts by combining logic, loops, and note control in a single module.

Along with the physical blocks, *Blocks o' Code (v3)* also includes a real-time companion app that mirrors the sequences created by the user. As blocks are added, removed, or rearranged, the app updates instantly to display the current configuration and visually represent the outputs. The companion app will not only display the real-time block configuration but will also generate a visual representation of the disco experience once the Start Block is activated. This means that while the physical blocks control lights, sounds, and vibrations in the real world, the app will simultaneously create a virtual party with animated lights, music cues, and effects that mirror the outputs.

2.6 Prototype Illustration

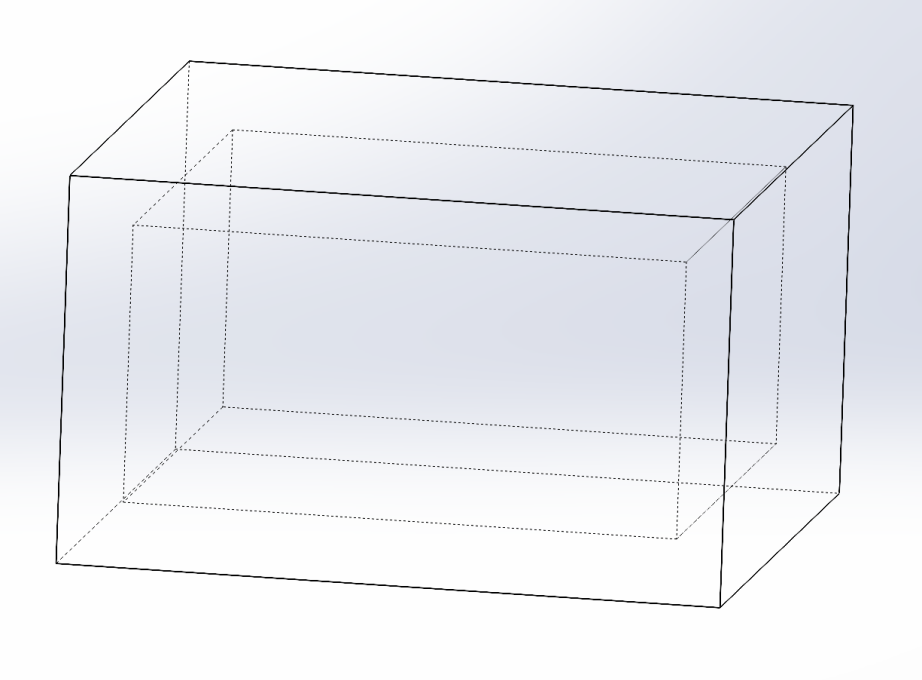


*Figure **2.1** SolidWorks rough draft of the cube prototype*

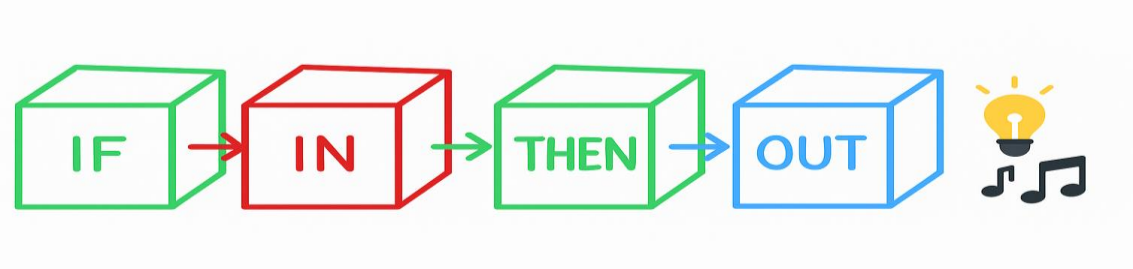


***Figure 2.2** Conditional logic diagram*

This rough draft of the prototype that we are developing for the Blocks of code (v3) system focuses on the design of a standardized cube enclosure. This will serve as the physical foundation for all block types. This cube was modeled in SolidWorks as a 50 × 50 × 50 mm hollow cube with 2mm wall thickness. This provides the appropriate balance between compactness while having enough room to house all the required electronics. While designing, the size was selected to ensure that the block provides internal volume for a printed circuit board (PCB), coin-cell battery, and other peripheral components, while staying compact and durable for repeated use.

This cube enclosure will be 3D-printed in a transparent PETG Filament. Transparent filament was chosen to allow users to visually connect physical electronics to the abstract programming functions. This also allows the visibility of the internal indicators such as LEDs or small displays to be visual. The goal is to get more kids interested in STEM due to visualization. To ensure that the assembly and durability of these cubes last, it is designed as a two-piece enclosure secured with screws threaded into reinforced bosses.

This ensures that the blocks can be opened for maintenance while remaining mechanically safe for student handling.

The power for each block is supplied by an internal rechargeable battery, similar to mag-safe charging. The enclosures will be designed with consideration for this compatibility. In the final design, blocks will be recharged when placed into a dedicated storage and charging station. This station will contain drawer trays with embedded inductive charging coils allowing for the block to be recharged while they are stored away.

2.7 House of Quality

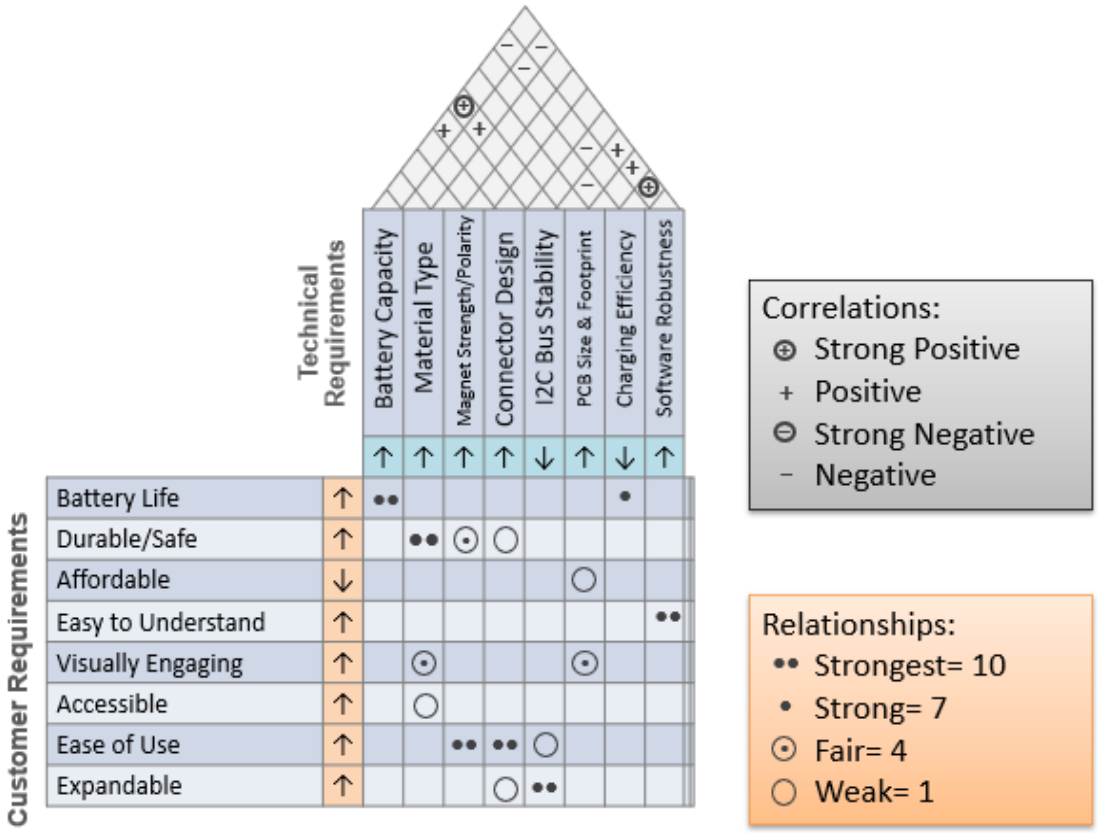


Figure 2.3 *House of Quality*

2.8 Engineering Specifications

Component/ Block Level Specifications

Component/ Block	Specification	Quantitative Measure	Verification Method
Gateway Block	MCU	ESP32-WROOM-32, 240 MHz, Wi-Fi	Functionality test
Function Block (Display)	Display hardware	2.8" TFT SPI, 240x320, 14-pin, touch	Power-on + SPI; logic analyzer

Function Block (Speaker)	Audio output	3 W into 8Ω @3.3V	Oscilloscope + SPL meter
Function Block (LED strip)	Addressable LED strips	WS2812B; chainable	Power-on+dataline; oscilloscope
Function Block (LED matrix)	LED matrix	4x4 RGB WS2812B	Power-on+scan; oscilloscope
Power Module	Battery	2600 mAh Li-ion, 3.7V	Charge/discharge test
Contact Resistance	Pad-to-pad	≤ 150mΩ per interface	4-wire ohmmeter

Table 2.8 *Component Specifications*

Overall System Specifications

Specification	Target	Measured	Verification Method
Block topology sync to app	≤ 2000 ms	50 ms	Logs; oscilloscope; logic analyzer
Maximum chain length	≥15 blocks (I^2C @ 100kHz)	15 blocks, ~426 ns	Oscilloscope (I^2C decode); rise time check
End-to-end latency (button to LED response)	≤ 50 ms	50 ms	Oscilloscope; logic analyzer
Active runtime	≥ 60 minutes typical use (music + LEDs)	≤ 240 min	Timer to battery cutoff

Table 2.8.1: *System Specifications*

*Highlighted = Demonstratable specification

2.9 Budget and Financing

Item	Price (USD)	Qty	Total Cost	Details
ESP32 MCU	$5.00	2	$10.00	Core MCU for Gateway block, Wi-Fi + I²C
RGB LED Matrix (4x4)	$4.00	2	$8.00	Visual output on Gateway
OLED Display	$3.00	2	$6.00	Display-based function blocks
Electret Microphone + Op-Amp	$1.50	3	$4.50	Mic-based function block for beat detection
Button Switches (6x6 mm)	$0.05	20	$1.00	Used in ladder/button function blocks

Class-D Audio Amp (PAM8302)	$1.00	3	$3.00	Speaker blocks
Mini 8 I^2C Speaker	$1.50	3	$4.50	Speaker output for audio blocks
Vibration Motor (Coin Type)	$2.00	2	$4.00	Haptic feedback blocks
Rechargeable Li-ion Battery (200 mAh)	$3.00	3	$9.00	Power modules
Buck/Boost Regulator (MCP1640)	$1.20	3	$3.60	Battery regulation for Power Puck
PCB Fabrication & Assembly	$25.00	4	$100.00	JLCPCB prototypes, assembly for Gateway and Function PCBs
Magnets (6 mm x 2 mm)	$0.10	50	$5.00	Magnetic block alignment + contacts
Gold-Plated Contact Pads	$0.05	50	$2.50	ENIG finish for bus pads
PETG Transparent Filament (1 kg spool)	$20.00	1	$20.00	3D printing for block enclosures
Miscellaneous (adhesives, wiring, shipping)	$15.00	1	$15.00	General project consumables
TOTAL			$196.10	

Table 2.9: *Budget and Financing*

2.10 Milestones

Milestone number	Milestone Description	Start Date	End Date (estimated)	Advisors and requirements [i]
1	Brainstorming	08/27/2025	09/04/2025	Meeting virtually to pitch ideas
2	Sponsor Meeting	08/27/2025	08/27/2025	Discuss project requirements with Dr. Mike and various ideas
3	Initial Document	08/26/2025	09/05/2025	First 10 pages of Divide and Conquer
4	Advisor Meeting	09/08/2025	09/08/2025	First meeting with Dr. Sundaresh reviewing Divide & Conquer paper and feedback
5	Revised Document	09/08/2025	09/12/2025	Final revision of Divide and Conquer paper
6	Sponsor Meeting	09/22/2025	09/22/2025	Reviewing project progress with Dr.

				Mike and Dr. Weeks for feedback
7	Reviewer Meeting	10/8/2025	10/8/2025	Meet with Dr. Mike for feedback and order parts
8	60-Page Milestone	10/7/2025	10/31/2025	First half of final document finished
9	Midterm report meeting	11/4/2025	11/4/2025	Meeting with Dr. Sundaresh for midterm report
10	Final Document	08/26/2025	11/25/2025	Final document of SD1 completed
11	Mini Demo Video	11/20/2025	11/25/2025	Demo of prototype

Table 2.10 Milestones

2.11 Software Flowchart

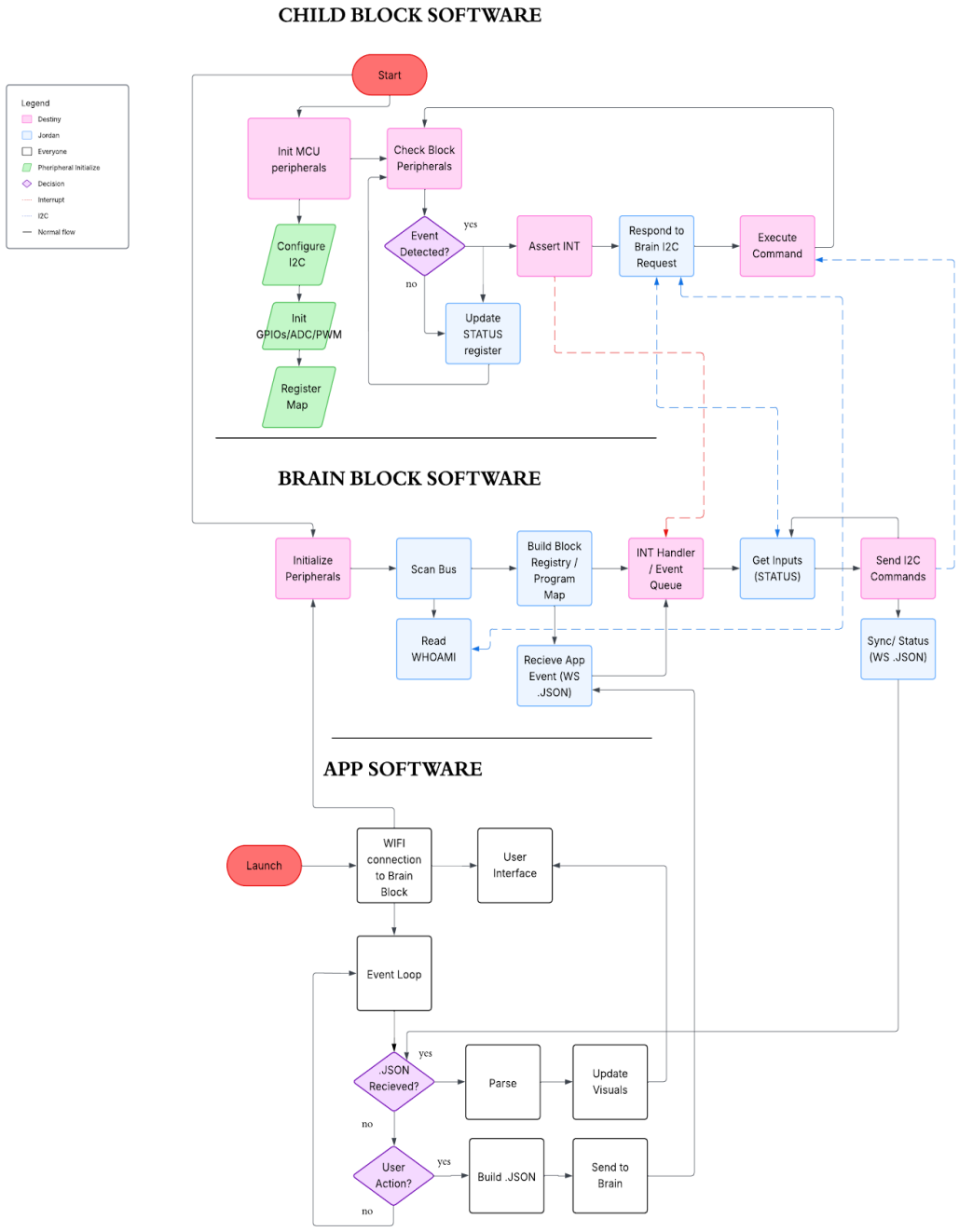


Figure 2.4: *Software Flowchart*

2.12 Hardware Flowchart

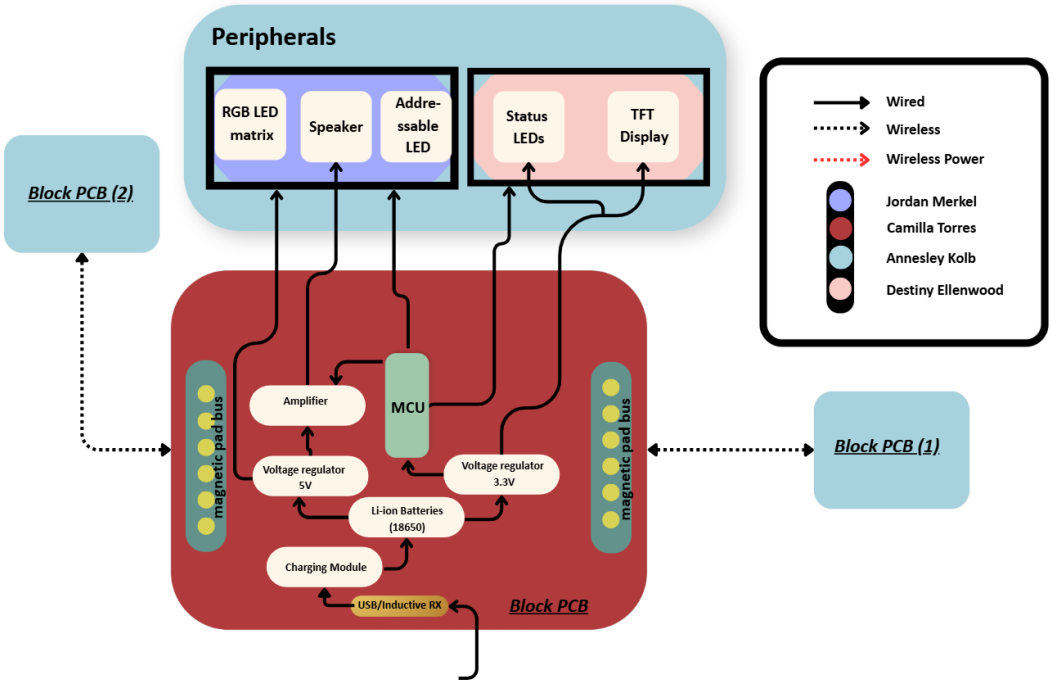


Figure 2.5: *Hardware Flowchart*

Chapter 3 – Research and Investigation

3.1 Microcontroller

This section is an investigation of different microcontrollers that were considered in the implementation of *Blocks o' Code (v3).* The microcontroller is a very important aspect of each block, as it is responsible for communication, data transfer, and interaction with peripheral components. Since there are two types of PCBs in this project, the Brain PCB and the Child PCB, two microcontrollers will be selected based on the different functional requirements of the PCBs.

3.1.1 ESP32-C3

The ESP32-C3, from Espressif, is a single-core microcontroller based on the 32-bit RISC-V architecture. It claims to try to balance cost, power, and I/O capabilities. It also has wireless communication with Wi-Fi and Bluetooth Low Energy (BLE). The ESP32-C3 operates at 160 MHz and features numerous peripherals, including I2C, SPI, PWM, ADC, and GPIO interfaces [19]. The IC itself costs approximately $1.00, making it a relatively low-cost device for its capabilities [23]. The main advantages of the ESP32-C3 are its wireless functionalities, its low power during modem-sleep mode, and it is also widely supported within the ESP-IDF and Arduino development frameworks. Some drawbacks of the ESP32-C3 are its small number of GPIO pins, and having only one processing core

could lead to issues with real-time tasks when needing to multitask. Given our project and the ESP32-C3's characteristics, this MCU is a contender for the Brain PCB.

3.1.2 ESP32-S3

The ESP32-S3, also from Espressif, is a dual-core microcontroller built on the Xtensa LX7 architecture. It operates at up to 240 MHz and is designed for high-performance applications that require both computation and connectivity. The device supports Wi-Fi and Bluetooth Low Energy (BLE) 5.0, providing robust wireless capabilities for real-time communication. It includes a wide range of peripherals such as UART, SPI, I²C, PWM, and ADC [25]. With approximately $2.00 per IC, it offers enhanced performance over the ESP32-C3 at a modest cost increase [26]. The ESP32-S3 also integrates vector instructions optimized for AI acceleration, making it well-suited for embedded intelligence and signal processing tasks [24]. Its advantages include higher clock speed, more GPIO pins, and dual-core performance, but its higher power consumption may be a drawback for battery-operated systems. Overall, the ESP32-S3 is a strong option for applications requiring wireless communication and higher computational throughput, another contender for the brain block.

3.1.3 Raspberry Pi 2040

The Raspberry Pi RP2040 is a dual-core ARM Cortex-M0+ microcontroller running at up to 133 MHz. Unlike the ESP32 series, it does not include built-in Wi-Fi or Bluetooth connectivity, focusing instead on flexible I/O control and deterministic performance. It supports a variety of peripherals such as UART, SPI, I²C, PWM, and ADC, and features a unique Programmable I/O (PIO) subsystem that allows developers to implement custom communication protocols in hardware [17]. The IC costs around $1.00, making it a competitive choice for the budget [27]. Advantages of the RP2040 include its reliable timing, lots of GPIO availability, and strong community support through the Raspberry Pi ecosystem. However, the lack of native wireless capability and relatively low processing power compared to the ESP32 family limit its use in connectivity-heavy applications. Given its balance between flexibility and cost, the RP2040 is better suited as a peripheral controller rather than the central processing unit in this project, making it a contender for the Child PCB.

3.1.4 Arduino ATMega

The Arduino ATmega328P, produced by Microchip Technology, is an 8-bit AVR RISC-based microcontroller commonly used in Arduino platforms. It operates at up to 20 MHz and supports peripherals such as UART, SPI, I²C, PWM, ADC, and GPIO [20]. The ATmega328P is known for its simplicity, reliability, and low power consumption, with an IC cost of approximately $2 [29]. It can operate at voltages ranging from 1.8 V to 5.5 V, making it versatile for various embedded designs; however, it is slower when operating at a smaller voltage. While the ATmega328P remains a proven and accessible option for general-purpose control tasks, its limited processing speed, memory, and lack of integrated wireless functionality make it less suited for complex or high-performance applications. In the context of this project, the ATmega328P serves as a contender for the Child PCB.

3.1.5 ESP32-WROOM-32

The ESP-WROOM-32, developed by Espressif Systems, is a Wi-Fi and Bluetooth-enabled module that integrates the ESP32 dual-core microcontroller based on the Xtensa LX6 architecture. Each core operates at up to 240 MHz, offering high performance for multitasking and communication applications. The module includes 520 KB of on-chip SRAM and typically 4 MB of external SPI flash memory, though variants support up to 16 MB. It supports a wide variety of peripherals, making it flexible for our project. Given its strong performance, wireless communication, and ease of integration, the ESP-WROOM-32 module presents a unified solution suitable for both the Brain and Child PCBs. A drawback is that it draws much higher power than the other MCUs being considered for the Child blocks and is also higher in cost [160].

3.1.6 MCU Selection

After evaluating the features, performance, and costs of multiple microcontrollers, the decision to use the **ESP32-WROOM-32** module for both the Brain PCB and the Child PCB in *Blocks o' Code (v3)* was made based on its versatility, integrated wireless capabilities, and ease of software development under a unified framework, we compared these MCUs based on *Table 3.1*.

Brain PCB:

The Brain PCB serves as the system's central controller, managing communication between blocks, executing real-time coordination, and handling wireless connectivity. The ESP32-WROOM-32 provides a powerful dual-core fast processor, sufficient SRAM for concurrent task management, and built-in Wi-Fi and Bluetooth connectivity. These capabilities support the Brain's need for responsive control, wireless communication, and data exchange with the companion app. Its robust processing power and communication interfaces make it well-suited for handling complex scheduling and coordination tasks.

Child PCB:

The Child PCB is responsible for managing localized inputs, outputs, and sensor feedback while maintaining communication with the Brain. Although the Child's computational and I/O requirements are lower, using the same ESP32-WROOM-32 module provides significant advantages in system consistency and simplicity. By standardizing a single MCU platform, the development process benefits from a unified firmware structure, easier debugging, and simplified hardware interfacing. The built-in Wi-Fi and BLE also enable future expandability without redesigning the board or firmware.

Choosing the **ESP32-WROOM-32** for both PCBs streamlines development by allowing the entire system to operate under the ESP-IDF framework, reducing code fragmentation and integration issues that often arise when using heterogeneous architectures. This uniformity simplifies firmware updates, testing, and peripheral driver management while minimizing development time and maintenance overhead. Although the WROOM module has a slightly higher cost compared to alternatives like the ESP32-C3 or RP2040, the

benefits of a single development environment, shared libraries, and consistent communication interfaces outweigh the marginal price increase. Overall, the ESP32-WROOM-32 ensures both high performance for the Brain PCB and design simplicity for the Child PCB.

	ESP32-C3	ESP32-S3	RP2040	ATMega	ESP32-WROOM-32
Cost	~$1.00	~$2.00	~$0.70	~$2.00	~$3.00
Architecture	RISC-V 32-bit	Xtensa 32-bit	ARM Cortex-M0	AVR 8-bit	Xtensa LX6 32-bit
Wi-Fi	Yes (2.4GHz)	Yes (2.4GHz)	No	No	Yes (2.4GHz)
Bluetooth	Yes (BLE 5.0)	Yes (BLE 5.0)	No	No	Yes (BLE 4.2)
UART	Yes (2)	Yes (3)	Yes (2)	Yes (USART)	Yes (3)
I2C	Yes (1)	Yes (2)	Yes (2)	Yes (TWI)	Yes (2)
SPI	Yes (3)	Yes (2)	Yes (2)	Yes (1)	Yes (4)
PWM	Yes (6 channels)	Yes (8 channels)	Yes (16 channels)	Yes (6 channels)	Yes (16 channels)
ADC	Yes (6 channels)	Yes (20 channels)	Yes (4 channels)	Yes (8 channels)	Yes (18 channels)
SRAM	400KB	512KB	264KB	2KB	520 KB
GPIO Count	22	45	30	23	32
Clock Speed	160MHz	240MHz	133MHz	16MHz	240MHz
Operating Voltage	3.3V	3.3V	3.3V	5V	3.3V
Current Draw	~84mA (Rx), ~300mA (Tx), ~25mA (Modem-sleep)	~88mA (Rx), ~300mA (Tx), ~65mA (Modem-sleep)	~100mA (max)	~1.5mA	~88mA (Rx), ~300mA (Tx), ~65mA (Modem-sleep)
Operating Temperature	-40°C to 105°C	-40°C to 105°C	-40°C to 85°C	-40°C to 125°C	-40°C to 85°C
Processing Cores	1	2	2	1	2

Table 3.1 *MCU Comparison*

3.1.7 Senior Design 2 Selection

The final system architecture uses the ESP32-WROOM-32 microcontroller for both the Brain Block and all Child Blocks. This decision was made to simplify the overall system design and avoid the added complexity of using multiple types of microcontrollers. Earlier in the design process, we considered using different MCUs for different roles, but this introduced issues such as increased communication overhead, more complicated firmware development, and potential integration problems between platforms.

By standardizing a single MCU, we were able to keep both the hardware and software more consistent across all blocks. Each block is responsible for handling its own local functionality, such as controlling LEDs, generating audio, or reading user input. These blocks communicate with the Brain Block over an I^2C bus, where the Brain Block acts as the master and coordinates the overall system behavior.

The ESP-IDF framework was used for firmware development on all blocks. This allowed us to stay within one development environment and made debugging and integration much easier. It also gave us better control over peripherals and built-in support for Wi-Fi, which is required for communication with the companion app.

The ESP32-WROOM-32 was a good fit for this project because it provides enough processing power, built-in Wi-Fi, and sufficient GPIO for all required peripherals. Using a single, capable MCU also reduced the need for additional components, which helped simplify the PCB design and keep costs down.

Overall, this approach made the system more reliable, easier to develop, and easier to expand in the future as more block types are added.

3.2 Voltage Regulation

Any project that receives a supply voltage and requires electrical components to operate needs a voltage regulator. To ensure reliable performance of low power educational devices, stable voltage regulation is essential for protecting components, minimizing noise, and improving battery efficiency. Since there are two types of PCBs in this project, the Brain PCB and the Child PCB, and each block has different electrical component configurations, this is why voltage regulators are a significant aspect of our project *Blocks o' Code (v3).*

Voltage regulators are important because even small fluctuations in supply voltage can cause erratic microcontroller behavior, audio distortion, data corruption in memory devices, or unstable sensor readings. These variations can happen due to battery discharge, sudden load changes, or differences in charging sources. Each voltage regulator selected for *Blocks o' Code (v3)* will play an essential role in maintaining system stability, extending

component lifespan, and ensuring predictable operation across all blocks by providing a constant and clean voltage output.

3.2.1 Linear Regulator: TLV733P-3.3 (TI)

A linear voltage regulator works by using an active component such as a transistor or MOSFET that is controlled by an operational amplifier. It keeps the output voltage steady by continuously comparing a built-in reference voltage with a sample of the output and adjusting the transistor's resistance until the difference is nearly zero. Because linear regulators can only reduce voltage, the output is always lower than the input [106].

In terms of efficiency, their main disadvantage is poor power conversion since excess energy is dissipated as heat, which causes a noticeable loss in power [108]. However, linear regulators perform exceptionally well when it comes to noise and ripple, producing a clean and steady output signal with very little interference [107]. From a thermal standpoint, the voltage difference between input and output becomes heat that must be managed, but they remain compact and practical for low current parts of a system where simplicity and low noise are more important than efficiency [117]. They are also easy to use and inexpensive, as they do not require inductors or complicated switching circuitry.

In *Blocks o' Code (v3)*, linear regulators are especially valuable for powering sensitive components. The microcontroller, audio amplifier, and sensor circuits require a steady and noise-free voltage supply to function correctly. Because linear regulators produce such a clean output, they are ideal for microphone input, communication lines, and other parts of the system where signal integrity is critical. Their simplicity also helps reduce design complexity in smaller blocks.

While researching linear regulators, we came across TLV733P-3.3. The TLV733P-3.3 is an excellent choice for *Blocks o' Code (v3)* because it delivers clean, low-noise voltage output, which is especially important for analog and audio circuits such as the microphone, amplifier, and sensor modules. It supports up to 300 mA of output current with a very low dropout voltage (125 mV at 300 mA) [120]. It is also ultra small and requires few external components, which helps reduce PCB complexity and size. The Brain and Child PCBs both include noise-sensitive elements, so TLV733P helps provide the stability needed to prevent audio distortion and interference with communication lines. Its low quiescent current (34 µA typical) also helps conserve energy in battery-powered blocks, balancing simplicity, efficiency, and noise performance [120].

3.2.2 Switching Regulator: TPS63060 (TI)

A switching regulator operates by rapidly turning a transistor on and off and controlling the energy stored in magnetic or electric fields to maintain a desired voltage level. These regulators can step voltage down, boost it up, or perform both, which makes them more versatile than linear regulators. Their operation depends heavily on proper selection of components and tuning of the feedback loop for stability, making layout and design a crucial part of achieving the desired performance [106].

In terms of efficiency, switching regulators are capable of reaching more than 95% efficiency depending on their design and load conditions [106]. This makes them highly effective for portable or battery powered systems that rely on energy conservation. However, the same switching action that enables high efficiency also introduces noise and ripple that can interfere with sensitive circuits if not carefully filtered [109]. From a thermal standpoint, switching regulators dissipate less heat than linear regulators due to their efficient energy transfer, allowing them to handle higher currents and output power. They are, however, more complex and require additional components such as inductors, capacitors, and transistors, which increase their overall size and cost [117].

In *Blocks o' Code (v3)*, switching regulators provide an efficient solution for managing power distribution between modules. Their ability to convert higher input voltages to lower regulated outputs without generating excessive heat makes them ideal for battery operated blocks. While they require careful layout to avoid interference with the audio or sensor circuits, their high efficiency directly supports the goal of extending battery life in the system.

When considering switching regulators in our project, the TPS63060 is a highly efficient synchronous DC-DC converter that makes it ideal for powering the main system logic in *Blocks o' Code (v3)*. It supports up to about 1.2A of continuous output current and operates with efficiencies above 90%, which helps extend battery life and reduce thermal stress on components [121]. This regulator's compact design and integrated MOSFETs minimize external parts, making it easy to implement in small PCBs without sacrificing stability. Because it can both step down and boost the input voltage from the battery or external power docks to a stable 3.3V or 5V rail, it serves as an efficient and reliable power supply for the ESP32-WROOM and other digital circuits. Its protection features, including overcurrent and thermal shutdown, also improve system reliability and safety for use in a children's educational device [121].

3.2.3 Buck Regulator: LM2576T-ADJ

A buck or step down converter is commonly used in power management systems to lower a higher input voltage, such as 12 or 28 volts, to a stable 5 or 3.3 volt supply suitable for low voltage electronics [110]. It operates by storing energy in an inductor and releasing it at a controlled rate, effectively reducing voltage while maintaining high efficiency. Because of this, buck converters are widely used in embedded systems and microcontroller based designs.

In terms of efficiency, buck converters perform well at moderate and heavy loads but tend to lose efficiency under very light load conditions, which is a common trade off in switching power supplies [118]. When considering noise and ripple, buck converters can achieve low output ripple when properly designed, though they may generate electromagnetic interference that must be minimized through filtering and good PCB layout [111]. From a thermal standpoint, their efficient energy transfer means they produce less heat than linear regulators, though they still require some level of heat management to maintain consistent performance. In terms of complexity and cost, buck converters

typically require a moderate number of components, offering a balance between simplicity, compact size, and efficiency [119].

In *Blocks o' Code (v3)*, the buck converter can be used to supply 3.3V to the ESP32-WROOM microcontroller when the system is powered by a higher input voltage. This ensures reliable operation of logic level components while keeping power losses to a minimum. Proper filtering will also help prevent switching noise from affecting communication and signal lines within the Brain PCB.

The buck regulator that we considered was the LM2576T-ADJ, this one is a compact and efficient buck converter that provides a stable 3.3V output from a 7V-40V input, making it a strong choice for the Brain PCB of *Blocks o' Code (v3).* It offers up to 3A of output current with efficiencies up to 75%, which ensures consistent performance for high-current components such as the microcontroller, display, or communication peripherals [122]. This regulator integrates internal MOSFETs and compensation circuits, which reduces both noise and design complexity. The low output ripple and built-in protection against overcurrent and thermal faults make it suitable for sensitive digital systems. Because of its small footprint and minimal external components, it helps optimize board space while maintaining efficiency and thermal balance, supporting both reliability and compact design goals for the project [122].

3.2.4 Boost Regulator: TPS61023

A boost converter is a type of DC-to-DC switching circuit that raises the input voltage to a higher output level by storing energy in an inductor when the switch is on and releasing it to the load when the switch is off [112]. These converters are useful in systems that require an output voltage greater than the available supply, allowing circuits to operate even when the power source is weak or partially discharged [114].

In terms of efficiency, boost converters can achieve efficiencies above 90% and, in optimized designs, even exceed 95% [115]. They offer fast transient response and voltage stability, maintaining less than 0.5% voltage error in experimental results [113]. However, the high frequency switching that allows this performance can also create output ripple and electromagnetic interference that must be carefully controlled through proper filtering and layout [116]. From a thermal perspective, boost converters operate efficiently but can experience higher stress and heat at large duty cycles or heavy loads. Additionally, they are somewhat more complex than linear regulators and require multiple external components, which increases cost and board space.

In *Blocks o' Code (v3)*, the boost regulator is critical for maintaining stable operation as the battery voltage drops. It allows the system to step up a low input from batteries and still provide enough voltage to power key components. This ensures that each block continues to function reliably throughout battery discharge, supporting consistent user performance and extended play time.

We came across the TPS61023, this one is an efficient step-up converter designed to boost low battery voltages to a stable output, making it perfect for the Child Blocks of *Blocks o'*

Code (v3). It can start with inputs as low as 0.5V, which allows the system to operate even as battery voltage drops. This feature ensures that the modules continue working reliably without sudden shutdowns, extending usability between charges or replacements. With efficiencies above 90% and a fast transient response, it can power LEDs, small displays, and sensors that require steady voltage despite battery variation [123]. The TPS61023 also operates with low noise and a small number of external components, which helps keep the circuit design compact and child safe. Overall, it ensures stable operation, energy efficiency, and long-lasting performance in battery-powered modules [123].

3.2.5 Voltage Regulator Selection

After comparing the performance, efficiency, and noise levels of different voltage regulators referenced in *Table 3.2*, the design for *Blocks o' Code (v3)* uses rechargeable batteries as the main power source. Each block runs on its own battery and recharges through USB-C or a wireless interface. The goal was to make the power system efficient, quiet, and stable under different loads so every module feels the same to a child whether it has been running for five minutes or an hour.

The **LM2576T-ADJ** buck regulator is used in both the Brain and Child Blocks to generate the required 3.3V and 5V rails from the same battery input. This regulator keeps both supplies steady even as the battery voltage changes during discharge. In the Brain Block, it provides reliable power for the ESP32-WROOM, display, and touch controller, preventing flicker or resets. In the Child Block, it powers LEDs, sensors, and displays consistently so they respond smoothly and keep their brightness and timing as the battery level drops.

By using a single **LM2576T-ADJ** regulator design for all blocks, the power system stays simple, efficient, and compact while maintaining quiet operation and dependable performance that supports the project's goal of making coding a playful and sensory experience.

Category	**TLV733P-3.3**	**TPS63060**	**LM2576T-ADJ**	**TPS61023**
Regulator Type	Linear (LDO)	Switching DC-DC Converter	Step-Down (Buck)	Step-Up (Boost)
Input Voltage Range	1.4 V - 5.5 V	2.0V - 16V	7V-40V	0.5V - 5.5V
Output Voltage	Fixed 3.3 V	Adjustable	Adjustable	Adjustable / 3.3V or 5V
Output Current	Up to 300 mA	Up to 2A	Up to 3A	Up to 500 mA
Efficiency	40–60% (typical)	Up to 90% (high)	Up to 75% (typical)	Up to 90–95% (high)
Noise / Ripple	Very low - ideal for analog and audio circuits	Moderate - Requires filtering	Low - if properly filtered	Moderate - requires filtering to reduce EMI

Thermal Performance	Moderate - Dissipates heat at higher load; limited efficiency	Good - low heat generation	Good - requires minor heat management	Good – may heat at high duty cycles
Cost	$0.40	$2.30	$2.00	$1.20
Size	2.9 × 1.6 mm	3.0 × 3.0 mm	3.0 × 3.0 mm	2.0 × 2.0 mm
Ideal Application	Audio and sensor rails, noise-sensitive circuits	Main 3.3V	Stable 3.3V and 5V rail	5V boost for LED matrix and audio amplifiers
Use in Blocks o' Code (v3)	Simplifies design, prevents audio distortion	Main 3.3V and 5V supply for ESP32, logic and displays	ALternative buck for stable 3.3V and 5V from 7V input	Provides optional 5V rail for "Disco Mode"effects (LED and amp boost)

Table 3.2 *Regulator Comparison*

3.3 Charging Interfaces

This section investigates different charging interfaces that were considered for implementation in the *Blocks o' Code (v3)*. Charging interfaces refer to the physical and electrical connection systems that allow a device to receive energy from an external power source. They include both the hardware components (such as connectors, pads, or coils) and the communication protocols that regulate how power is transferred [104]. For our types of PCBs in this project, each interface was researched in terms of efficiency, power delivery capability, durability, safety, and overall suitability for use in modular educational electronics.

3.3.1 USB-C: hiBCTR B0FPCL44K7

USB-C is one of the newest and most widely used connectors in electronics today. What makes USB-C stand out is that it can handle both data and power through the same reversible plug. Depending on the version, USB Power Delivery (USB-PD) supports anywhere between 5V and 48V, delivering currents up to 5 A for a maximum of 240W in the Extended Power Range mode [77]. For low power projects like *Blocks o' Code (v3)*, only a few watts are needed, so USB-C easily covers that range without any issues.

What makes USB-C so influential in electronics is its durability. It's rated for roughly ten-thousand plug-in cycles [78], and it includes power negotiation to avoid problems like short circuits or over-voltage. Still, it's not perfect. If the cable is pulled at an angle, it can loosen the port over time, and needing a separate cable for each block could raise the overall cost. Even with those drawbacks, USB-C continues to be one of the most reliable and affordable charging choices in today's consumer and educational products [79], and its small size and seamless appearance would look ideal for *Blocks o' Code (v3).*

3.3.2 Magnetic pogo-pins: Attend 303C-C4115-30-04

Another charging option considered was magnetic pogo-pins. These connectors are small spring-loaded pins that create temporary electrical contact when pressed against a pad or surface. They're already used in products like phones, wearables, and testing equipment because they can transfer both data and power in a compact form [80]. Depending on the type, pogo-pins usually handle between 1A and 3A of current, and their contact resistance is normally between 20 mΩ and 50 mΩ [81][82].

One thing that made pogo-pins a consideration for our project was the idea of reusing the same connection for both charging and communication, and this being one of our functional requirements for our sponsored project, made us consider pogo-pins for both, data transfer and power transfer. This can allow for a simpler PCB design and utilize space within the block. A potential drawback can be that these connectors can wear out after many uses, especially if dust or oxidation builds on the tips [83]. Since the blocks will be handled by children, we are always keeping in mind that the use of the blocks will be in settings like a classroom, and we have to think about sticky hands, crumbs, or small metal pieces that might affect the connection. In short, pogo-pins are practical for modular electronics, but they might need extra protection or coating to stay reliable over time.

The Attend 303C-C4115-30-04 magnetic pogo-pin connector matches well with the idea of using magnetic charging in *Blocks o' Code (v3)*. This connector combines both magnets and spring-loaded pins in a single compact piece, which helps the blocks align automatically when placed together. It has four contact pins, each rated for up to 2A of current, which is more than enough for the system's 5V charging range [154]. The contacts have a low resistance of around 30 mΩ and are gold-plated to prevent corrosion and ensure consistent conductivity over time [155]. The connector is designed for surface-mount installation, making it easy to place on small PCBs without taking up much vertical space or needing through-holes.

For *Blocks o' Code (v3)*, this connector is a strong option because it provides a safe and easy way for children to attach the blocks for charging without worrying about plugging in cables. The built-in magnets guide the pins into the correct position, reducing wear and the risk of reversed polarity. Its durable design supports thousands of connections, which is ideal for a classroom setting where the blocks will be handled frequently. The four-pin layout also leaves room for future features, such as data communication between the blocks [154]. Overall, the Attend magnetic pogo-pin connector fits the project's goals of being simple, reliable, and child-friendly, while maintaining clean power transfer in a compact modular design.

3.3.3 Qi Wireless: Adafruit Qi Wireless Receiver Module

Qi wireless charging has become common in many modern devices, from phones to smartwatches, and we wanted to explore how it could work with *Blocks o' Code (v3).* The Qi standard, developed by the Wireless Power Consortium, allows energy to transfer through electromagnetic induction between two coils, one in the charger and one in the device [84]. Typical Qi systems can deliver up to 5W in the basic power mode and as high

as 15W in the extended mode [85]. Efficiency usually ranges from 60% to 80%, depending on how well the coils are aligned and how far apart they are [86].

In our implementation, the blocks could simply be placed on a charging tray that has the transmitter coil, while each block includes a small receiver coil. That would let the blocks charge without any cables or exposed metal parts, making it safer for children. The main challenge is alignment, if the coils aren't centered, power transfer drops quickly, and more heat is produced [87]. Some designs use small magnets or physical guides to help with positioning, which could be a good solution for us. Even though wireless charging isn't the most efficient, it looks neat, feels modern, and removes the risk of connector damage.

The Adafruit Qi Wireless Receiver Module (5 V, 500 mA) seems to work well with how wireless charging could be added to *Blocks o' Code (v3)*. It follows the Qi 1.2 standard and gives a steady 5V output, which makes it a good fit for small systems that run on lithium-ion batteries [153]. The module is small, lightweight, and easy to connect, and it already has a ferrite shield that helps keep it from heating too much or interfering with nearby parts. It can provide enough current for normal charging speeds, and its built-in regulator keeps the voltage consistent for safe charging [153].

For *Blocks o' Code (v3)*, this type of receiver makes it simple to test wireless charging without using any cables. Each block could just be placed on a pad to charge, which makes it safer for kids and prevents the connectors from wearing out over time. Its 5V output and size fit well with the project's power needs, and it can easily fit inside the small plastic enclosures. Even though wireless charging isn't as efficient as wired methods, the Adafruit module feels like a modern and clean option that would make the project look more polished and easier to use.

3.3.4 Charging Interface selection

USB-C (hiBCTR B0FPCL44K7) was chosen as the charging interface for *Blocks o' Code (v3)* because it is simple, safe, and easy to use. We made this decision based on the comparison in *Table 3.3*. It is a very common connector that can be found in most devices today, which makes it affordable and easy to replace if needed. The reversible plug makes it easy for children to connect without worrying about the direction, and it can carry both power and data through the same port. This helps keep the design clean and saves space on the board. USB-C also has built-in protection features that help prevent short circuits or over-voltage, making it a reliable choice for a project that will be handled often in a classroom setting.

In the future, we could explore adding wireless charging to make the blocks even easier and more fun to use. Each block could have a small Qi receiver coil inside so that it charges when placed on a pad, without any cables or connectors. This would make the design look modern and reduce wear on the parts over time. Even though wireless charging is not as efficient as wired charging, it would be a nice feature for classroom trays or charging docks, making the setup feel more seamless and child friendly.

Category	USB-C (hiBCTR B0FPCL44K7)	Magnetic Pogo-Pins (Attend 303C-C4115-30-04)	Qi Wireless (Adafruit 5V/500mA Receiver)
Power Delivery Range	5V - 48V (up to 5A, 240 W EPR)	Up to 2A per pin (5V nominal)	`Up to 5V @ 500 mA (≈ 2.5W)`
Efficiency	~95%	~85 - 90 % (depending on contact resistance)	~60 - 80 % (depends on coil alignment)
Connection Type	Wired, reversible plug	Magnetic contact pins (spring-loaded)	Wireless (inductive coupling)
Durability/Life Cycle	~10,000 plug cycles	Thousands of connections with proper care	No mechanical wear
Safety Features	Built-in power negotiation, short and over-voltage protection	Magnetic alignment reduces mis-connection risk	Fully isolated, no metal contacts
Cost	$0.50 - $1.50	$2.50 - $4.00	$8.00 - $12.00
Use in *Blocks o' code (v3)*	Best overall choice, affordable, durable, safe	Possible future option for docking systems	Optional concept for future wireless version

Table 3.3 *Charging Interface Comparison*

3.4 Charging IC

Charging a battery through USB-C, wireless coils, or magnetic pogo pins isn't as simple as just connecting power. Doing that directly could easily damage the battery or make it overheat. That's why a charging IC is used; it controls the entire process and makes sure the battery is charged safely. It knows when to start charging, how fast it is to charge, and when to slow down or stop once the battery is full.

In *Blocks o' Code (v3)*, using a charging IC is important because each block has its own small lithium-ion battery inside. The IC helps keep every block safe by preventing overcharging and controlling temperature. It also makes it easier for the system to charge through different inputs like USB-C, magnetic pogo pins, or even wireless charging, while still keeping the process consistent and reliable.

3.4.1 MCP73831T

The MCP73831T is a compact and reliable single-cell lithium-ion charger IC designed to manage the full charging process automatically. It uses the standard constant-current and constant-voltage (CC-CV) method, which ensures the battery is charged safely and efficiently. The IC controls every stage of charging, beginning with pre-charging a low battery, switching to a constant current mode, and finally maintaining a fixed voltage until the battery reaches full capacity. It operates with a 4.4-6 V input range, making it

compatible with a 5V USB-C supply, and supports a charge current of up to 500mA [145]. One of its key features is thermal regulation, which automatically lowers the charging current if the internal temperature rises too high. Since it only requires a resistor to set the current and a few capacitors for stability, it's straightforward to integrate into compact circuit designs [146].

For *Blocks o' Code (v3*), this IC matches the system's requirements for small, low-power charging modules. Each block only needs a few hundred milliamps to recharge, which stays well within the IC's operating range. Its small SOT-23-5 package helps minimize the PCB footprint, allowing for an efficient layout inside each block. The ±1% voltage accuracy ensures consistent and safe charging, and the programmable current control makes it adaptable to batteries of different capacities [145]. Since the system doesn't need to operate while charging, this linear charger provides a reliable balance of safety, efficiency, and simplicity, aligning well with the design goals of the project.

3.4.2 MCP73832T

The MCP73832T is a small linear charging IC designed for single-cell lithium-ion batteries. It follows the constant-current and constant-voltage (CC-CV) charging process, which keeps the charging cycle safe and stable [148]. The chip can operate from 4.4V to 6V, so it works perfectly with a 5V USB-C input. It can provide up to 500mA of current, and that value can be adjusted easily with one programming resistor [147]. It also has built-in thermal protection, automatic recharge, and charge termination, which helps manage heat and prevent the battery from overcharging. One feature that makes it stand out is its open-drain status output, which can be connected to LEDs or a microcontroller to show the charge state or send charging information to the system [147].

For *Blocks o' Code (v3)*, this IC fits the design needs especially well. It's compact, reliable, and gives the team the option to add feedback features later without redesigning the entire charging circuit. The open drain output makes it easy to include a charge indicator light or even let the ESP32 monitor the charging status. Because the MCP73832T has the same size and efficiency as other options but offers more flexibility for data and LED control, it's a strong choice for the final prototype. It balances simplicity, safety, and expandability, making it the most practical option for how the Blocks are designed to work.

3.4.3 TP4056

The TP4056 is a charger IC designed for single-cell lithium-ion batteries that follows the constant-current and constant-voltage (CC-CV) charging method [149]. It works with an input voltage between 4V and 8V and can supply a programmable charge current of up to 1A, depending on the resistor connected to its programming pin [150]. The IC includes several built-in protection features such as thermal regulation, automatic recharge, and charge termination when the charging current falls below a set level. Its voltage regulation accuracy is about ±1.5%, which helps ensure the battery reaches a consistent and safe full charge [150]. The TP4056 is also commonly found on small, ready-to-use charging boards that often include a USB-C or micro-USB port, and sometimes an added protection circuit that prevents overcharging, over-discharging, and short circuits [149].

From a technical point of view, the TP4056 is a linear charger, which means it manages charging by converting extra input power into heat. This makes the circuit simple but less efficient when operating at higher current levels or in tight enclosures with limited ventilation. The IC includes a thermal cutoff that activates around 115 °C, automatically reducing the charge current when the device gets too hot [149]. It comes in a larger SOP-8 package, which helps with mechanical stability and spreading heat across the PCB. Because it combines charge control, protection, and easy modular use, the TP4056 is a popular choice for educational projects, small devices, and prototyping setups that require safe and straightforward single cell battery charging.

3.4.4 ME4057

The ME4057 is a compact linear charger IC made for single-cell lithium-ion batteries. It uses the constant-current and constant-voltage (CC–CV) charging process and can provide up to 800 mA of charge current [151]. The IC comes in a small SOT-23-6 package, which makes it easy to fit into space-limited designs. It includes built-in features such as pre-conditioning for deeply discharged cells, automatic charge termination, thermal protection, and low standby current when not in use. Its voltage regulation accuracy is around ±1%, which helps the battery reach a consistent full charge without overcharging [152]. The ME4057 is designed to work from a standard 5V power source, such as a USB-C or magnetic pogo-pin connection, and only needs a few external components, mainly one resistor to set the current and a couple of capacitors for filtering.

For *Blocks o' Code (v3)*, the ME4057 could be a strong choice for blocks that have limited internal space but still need reliable and moderately fast charging. Its small package and efficient operation at mid-range charge currents (400–600 mA) help minimize heat buildup, which is important since each module is enclosed in plastic [152]. The IC's low idle power consumption is also beneficial for a system where the blocks might stay plugged in or docked between uses. Because it combines compact size, solid thermal performance, and simple external circuitry, the ME4057 fits well with the project's focus on safe, compact, and user-friendly power management inside each educational block.

3.4.5 TP5100

The TP5100 is a small and efficient charging chip made for lithium-ion batteries, and it can work with either one cell (4.2 V) or two cells (8.4V). It uses a constant-current and constant-voltage charging method, which means it gives the battery a steady current at first and then carefully lowers it as the battery gets full. Unlike simpler linear chargers, the TP5100 is a buck converter, so it runs cooler even when charging faster, up to 2 A. It also includes safety features like temperature protection, short-circuit protection, and automatic recharge when the battery drops below a certain level. The chip has built-in LED indicators that show when the battery is charging or done [76].

For *Blocks o' Code (v3)*, the TP5100 works well because it can charge the two 18650 batteries in parallel as if they were one single cell. When set to 1S mode, it regulates the voltage to 4.2 V from a regular 5V USB-C input, so it connects easily to our system. Since it's a switch-mode charger, it stays much cooler and can charge the batteries faster than

smaller linear chips like the MCP73831. This helps the blocks recharge quickly and safely without overheating inside the plastic case. The TP5100 module is also beginner-friendly, it only needs a few extra parts and has header pads that make it easy to connect during prototyping, which fits perfectly with our goal of keeping the design simple, safe, and easy to assemble.

3.4.6 Charging IC Selection

The **TP5100** was chosen as the charging IC for *Blocks o' Code* because it can safely charge the two 18650 batteries used in each block while staying efficient and cool during operation. It can handle higher charging currents without overheating, which helps the blocks recharge faster and makes them ready for use in less time. The chip also includes built-in protection features that keep the batteries safe from overcharging and temperature issues, which is important since each block is enclosed in a small plastic case. We made this decision based on *Table 3.4.*

Another reason we chose the TP5100 is that it works perfectly with a regular 5V USB-C input, so it fits easily into our design. It only needs a few extra parts to work, and its small module is simple to connect during prototyping, which makes assembly much easier. Since it can manage charging automatically and efficiently, it keeps the system safe, reliable, and easy to use for both testing and classroom settings.

Category	**MCP73831T**	**MCP73832T**	**TP4056**	**ME4057**	**TP5100**
Charging Method	Linear, CC-CV	Linear, CC-CV	Linear, CC-CV	Linear, CC-CV	Buck (switch-mode), CC-CV
Input Voltage Range	4.4 - 6.0 V	4.4 - 6.0 V	4.0 - 8.0 V	4.0 - 6.5 V	5.0 - 18 V
Max Charge Current	500 mA (adjustable)	500 mA (adjustable)	1000 mA (adjustable)	800 mA (adjustable)	2000 mA (selectable 1A / 2A)
Voltage Accuracy	±1 %	±1 %	±1.5 %	±1 %	±1%
Thermal Regulation	Yes	Yes	Yes (cutback ≈115 °C)	Yes	Yes (thermal & short-circuit protection)
Typical Efficiency	70 – 80 %	70 – 80 %	60 – 75 %	75 – 80 %	90 – 95 % (switch-mode)
Cost	$0.80	$0.85	$0.25 IC	$0.60	$0.70

Use in *Blocks o' Code (v3)*	Safe, simple charger for main and child blocks	Ideal for final design with LED/MCU status	Useful for quick breadboard testing	Optional for compact, mid-current modules	Best for fast, cool charging from USB-C

Table 3.4 *Charging IC Comparison*

3.5 Display

Displays are a core part of the *Blocks o' Code (v3)* system because they give each block a visual way to communicate information, current state, and logic flow to the user. Since this system is designed for children, visual feedback needs to be quick, clear, and engaging to help reinforce the connection between coding actions and program results. When a child activates a block, the display can respond with symbols, text, or lighting effects that confirm what is happening inside the system. This visual interaction strengthens understanding by connecting abstract programming concepts to something tangible and easy to observe.

Each display technology considered for *Blocks o' Code (v3)* offers different benefits in terms of readability, power efficiency, and visual impact. Some are optimized for text and clarity, like OLED and segment LCDs, while others focus on color, motion, and animation, such as LED matrices and TTL TFT displays. Choosing the right display for each block depends on its role in the overall system, whether it needs to show logic symbols, visual patterns, or interactive color feedback.

3.5.1 OLED Display

Organic Light Emitting Diode (OLED) displays are self-emissive panels where each pixel produces its own light rather than relying on a backlight like traditional LCDs. The SSD1306 is one of the most commonly used OLED driver controllers, often used in compact 128×64 pixel modules. It supports both I²C and SPI communication interfaces, which makes it flexible for embedded applications. In *Blocks o' Code (v3)*, the OLED can connect to the ESP32-WROOM microcontroller through an SPI interface to achieve faster refresh rates and smoother visual updates compared to I²C [65][28].

The SPI interface uses four main signals: MOSI (data line from the ESP32-WROOM to the OLED), SCLK (serial clock), CS (chip select), and D/C (data or command). Since SPI is clock-driven, it allows the display to update its buffer quickly and consistently. This is important when rendering animations or icons that react to user interactions. The OLED runs at 3.3 V logic, so it can connect directly to the ESP32-WROOM without level shifting. Its average power consumption is approximately 20 mA, which makes it suitable for portable, battery-powered systems [65].

For *Blocks o' Code (v3)*, the OLED can serve as a small visual interface for logic and sensor-based blocks. Its 128×64 resolution provides enough space to show intuitive graphics, icons, and short animations that help children understand how their block is

behaving. For example, a loop block could display a rotating arrow, while a sound block could show animated musical notes when activated. The fast SPI communication allows these animations to update in real time, helping children see immediate feedback and understand cause-and-effect relationships in their code.

OLED displays are bright and provide sharp contrast even in classroom lighting conditions. To improve durability, the display can be mounted behind a clear acrylic window or block's housing to prevent damage during frequent handling. To extend the lifespan, brightness can be automatically reduced or the display can be dimmed when idle to conserve power and minimize pixel wear.

The SSD1306 is also easy to program using well-established display libraries such as Adafruit GFX and U8g2. These libraries allow developers to draw text, shapes, and bitmaps without needing complex graphics code. Each OLED can be updated locally by the ESP32-WROOM within the Child Block, while the Brain Block coordinates overall logic and sends relevant commands through the I²C network. This combination of speed, readability, and low power consumption makes the SSD1306 OLED an effective choice for clear and interactive visual feedback in *Blocks o' Code (v3)*.

3.5.2 Segment LCDs

Segment liquid crystal displays (LCDs) are one of the simplest and most power-efficient display technologies used in embedded systems. Instead of controlling individual pixels, a segment LCD divides the screen into predefined shapes or symbols such as digits, bars, or icons. Each segment acts as an independent area that can turn opaque or transparent when voltage is applied. Because of this fixed layout, segment LCDs are ideal for showing basic information such as numbers, timers, or limited status indicators [66][68].

A segment LCD works by manipulating liquid crystal molecules between two glass layers with transparent electrodes. When voltage is applied, the crystals block or allow light to pass through. Most small segment LCDs operate in reflective or transflective mode, which means they rely on room lighting rather than internal illumination.This reflective design makes them extremely power-efficient, often consuming only microamps of current during operation. Their clarity and low energy use make them popular in watches, calculators, and portable instruments [66][68].

In *Blocks o' Code (v3)*, segment LCDs were initially considered for logic or counter-based Child Blocks, such as showing numeric values for loop counts or timing delays. Their sharp, high-contrast digits would make data easy to read, even in bright classroom environments. However, these displays are limited to static layouts. Since each segment is etched into the glass, the display layout cannot be reconfigured or used for icons or animations. This fixed structure limits how creative and interactive each block can be, which is an important part of keeping children engaged, but it still can be used to communicate status updates.

In comparison, OLEDs and LED matrices can show motion, shapes, and expressive feedback that make the coding process more fun and understandable for kids. Segment

LCDs are durable, inexpensive, and extremely efficient. The project emphasizes motion, animation, and immediate feedback to help children connect physical actions with program logic. Because of this, more flexible display types such as OLEDs or LED matrices are a better match for visual engagement and user experience, but we can rely on Segment LCDs for simpler functionality Child Blocks.

3.5.3 LED Matrix Display

An LED matrix display is made up of a grid of light-emitting diodes arranged in rows and columns that can be individually controlled to create images, patterns, or animations. Each LED acts as a pixel that can be turned on, off, or adjusted in brightness to form dynamic visuals. Typical small modules, such as an 8×8 RGB matrix, contain 64 addressable LEDs capable of displaying vivid colors and smooth motion effects [69].

The LED matrix is appropriate for real-time feedback applications because it can update extremely fast and remain visible in a variety of lighting conditions. Each pixel responds within microseconds, allowing smooth transitions for effects like scrolling text, flashing icons, or sequential animations. In *Blocks o' Code (v3)*, these visual responses are key to keeping children engaged. When a block is pressed or a command runs, the matrix can light up with flowing arrows, pulsing waves, or rhythmic colors that reflect program activity.

From a hardware standpoint, each Child Block's matrix can be driven locally by the ESP32-WROOM microcontroller using pulse-width modulation (PWM) to vary LED brightness. This ensures precise color blending and animation control. To simplify timing and current regulation, an external LED driver such as the PCA9685 can provide 12-bit PWM resolution across 16 channels, maintaining consistent brightness without placing a heavy load on the microcontroller [70]. Local control keeps the I²C bus between blocks clear for logic communication, while the matrix updates occur independently through high-speed SPI or GPIO lines.

Color and lighting directly influence how children focus and interact with educational tools. Research shows that bright, warm colors like red, orange, and yellow naturally attract attention and spark excitement, while cooler colors such as blue and green help sustain concentration and calm [71][72][74]. Dynamic and colorful lighting patterns have been shown to capture children's attention for longer periods and help reinforce the connection between visual feedback and their actions with interactive learning environments [73]. By using programmable RGB matrices, *Blocks o' Code (v3)* utilizes these principles to create meaningful feedback loops. For example, a "success" animation might glow green, a "processing" state could pulse yellow, and a "reset" signal might fade to blue, allowing children to interpret system states instantly through color and motion.

LED matrices are also durable and adaptable for classroom environments. The LEDs are encapsulated, making them resistant to vibration and handling, and their brightness can be adjusted for indoor visibility. Since the matrix hardware is modular, different block types can use customized patterns without changing the overall circuit design. Power

consumption remains manageable by limiting brightness and enabling sleep states when blocks are inactive.

Overall, the LED matrix display combines responsiveness, durability, and visual engagement. It supports the educational goals of *Blocks o' Code (v3)* by turning logic flow into an immediate, colorful experience that helps children connect their physical interactions to the program's output. The result is a balance of simplicity, interaction, and expressiveness that makes coding both understandable and exciting.

3.5.4 TTL TFT Display

Thin Film Transistor (TFT) displays are full-color active-matrix screens that use thin-film transistors to control each individual pixel on the display. In a TFT panel, every pixel has its own transistor and capacitor, which allows for fast switching and precise brightness control. These displays typically use a Transistor-Transistor Logic (TTL) interface to communicate pixel data and synchronization signals between the microcontroller and display driver [75][76][77].

TFT technology is known for its high resolution, wide color range, and strong contrast, which make it ideal for interactive and vibrant applications. Each pixel on the display contains three subpixels red, green, and blue that blend to create millions of colors. The result is a vibrant, detailed image that can show icons, animations, or full graphical interfaces. For *Blocks o' Code (v3)*, this type of display could be used in the Brain Block or advanced Child Blocks to visualize logic flow, real-time data, or programming progress in a way that is bright, colorful, and easy for children to understand.

Unlike segment or monochrome displays, TFTs are backlit, allowing them to remain clearly visible in any lighting environment. The backlight uses LEDs to provide uniform illumination, while the transistors ensure each pixel can switch rapidly, enabling smooth transitions and animations [75][77]. This makes the display suitable for presenting animated icons, game-like visuals, or even simple touch interfaces that could expand the educational experience.

From a hardware standpoint, TTL TFT displays use multiple data lines (often 18 or 24 bits for RGB color) along with control signals such as VSYNC, HSYNC, and CLK to synchronize the display with the controller [76]. While this requires more wiring than SPI or I²C, it allows for significantly faster frame rates and higher resolutions. The ESP32-WROOM is capable of driving such displays through a parallel RGB or SPI interface, making it a strong candidate for future *Blocks o' Code (v3)* iterations that may include advanced visuals or touch-based control [78].

Color theory also supports the educational use of full-color TFT displays. Research shows that color directly impacts attention, motivation, and comprehension among children. Warm colors such as red and yellow capture focus and create excitement, while cool colors like blue and green promote calmness and sustained engagement [71][72][74]. By leveraging these effects through carefully chosen visual cues and consistent color schemes, TFT displays can help guide children's attention and make programming concepts more

intuitive. For example, a block could display a glowing blue circle to represent an idle state, a bright green animation to indicate success, or a flashing red border to signal an error.

Although TFT displays consume more power and are more fragile than OLEDs or LED matrices, they offer the greatest potential for visual depth and interactivity. They can display detailed shapes, animations, and full-color feedback that could turn *Blocks o' Code (v3)* into an even more immersive educational tool. With proper mounting, brightness control, and protective housing, a TTL TFT display can provide durable, high-impact visuals that elevate the system from simple lighting feedback to a rich, interactive interface.

3.5.5 Display Selection

After comparing all the display options referenced by Table 3.5, our team decided that *Blocks o' Code (v3)* would benefit most from using different types of displays for different block functions instead of relying on just one. This approach gives us the flexibility to balance power, durability, and visual engagement while making sure every block supports its specific role in the system.

For the Brain Block, we're using a TFT display because it gives the best visual quality and interactivity. It's full of color, refreshes quickly, and is bright enough to show animations, program states, or changes clearly. This display helps the Brain Block feel like the main hub, where children can watch their programs in action. It also works well with the companion app and could be upgraded later to support touch menus or show progress.

For Child Blocks that primarily output light or animation, we're using RGB LED matrices. These connect through SPI, giving us fast frame updates with smooth color transitions. The LED matrices make the learning experience more engaging by lighting up, flashing, or changing color based on what's happening in the code. For example, a logic condition can trigger a pattern or color shift that gives instant visual feedback. The bright colors and animations help children connect their physical actions with the digital logic in a fun and understandable way [71][72][74].

Some simpler input or sensor-based Child Blocks will use Segment LCDs to show quick, low-power feedback like numbers, icons, or ON/OFF states. They're extremely durable and easy to read, which makes them perfect for classroom environments where the blocks are handled often. These displays focus on clarity over flashiness which are ideal for communicating sensor readings or states without distraction [68].

We'll use OLED displays in child blocks that need small, high-contrast text or icons. Their sharp images and low power use make them useful for modules that display values like music notes or even loop configurations. Because OLEDs are more delicate and usually only show one color, we plan to put them inside or in protected encasing, not on parts that get touched a lot [65][66].

By using different displays, *Blocks o' Code (v3)* makes each module unique but keeps the whole system working together. TFT displays show complex visuals, LED matrices give bright, interactive feedback, OLEDs show small bits of information, and Segment LCDs

act as simple, tough indicators for daily classroom use. We pick each display based on what the block needs to do and where it will be used, so the system stays efficient, strong, and exciting for children.

This mix of displays also fits well with our communication setup. TFT and OLED displays use SPI for fast updates, LED matrices also use SPI for smooth animations, and simple LCDs use I²C to keep wiring simple. These choices make *Blocks o' Code (v3)* responsive and easy to use, combining clear visuals with hands-on learning.

Display Type	**Color Capability**	**Interface Type**	**Resolution / Pixel Control**	**Power Consumption**	**Durability & Cost**	**Advantages**	**Limitations**	**Use in *Blocks o' Code (v3)***
OLED	Monochrome (white or blue typical)	SPI / I²C	128×64, individual pixel control	Moderate	Moderate durability, higher cost per inch	High contrast, thin form factor, wide viewing angles	More fragile, burn-in possible, not ideal for heavy handling	Could display icons or logic symbols, but less durable for frequent use
Segment LCD	Limited (no color, segment-based)	Direct / Multiplexed Drive	Predefined segments	Very low	Very durable, inexpensive	Extremely low power, great visibility, simple driver circuit	Limited flexibility, cannot display animations or free-form images	Suitable for basic numeric or icon indicators in simple logic blocks
LED Matrix	Full RGB	SPI / GPIO	8×8, 16×16, or scalable	Moderate to high	Highly durable, modular	Bright, colorful, visually engaging, customizable patterns	Lower resolution, higher power draw for large arrays	Ideal for block feedback and color-coded educational engagement

TTL TFT Display	Full RGB (16-bit or 24-bit color)	Parallel RGB / TTL	240×240 up to 480×480	Higher	Moderate durability, higher cost	Full color, fast refresh rate, rich graphics, backlit	More wiring, higher power, less rugged	Ideal for Brain Block or advanced modules to show animations, program states, and touch interaction

Table 3.5 *Display Comparison*

3.5.6 Senior Design 2 Selection

The final display implementation standardizes three visual elements across all blocks: the 2.8" TFT SPI touch display, the 4×4 WS2812B RGB LED matrix, and the WS2812B addressable LED strip. Rather than varying the display type per block role as initially planned, using the same three-element configuration on every block simplified PCB design, unified the firmware display driver, and ensured consistent visual feedback system-wide.

The TFT renders a block-type-specific interface on each block. For example, a numpad for note and input blocks and animation feedback for LED output blocks. This way children can immediately understand each block's function. The 4×4 LED matrix provides grid-based color output during program execution, and the addressable LED strip runs along the block edges to give continuous status and identification feedback visible from any angle.

Segment LCDs and OLED displays were considered in the research phase but were not implemented in SD2, as the combined TFT and LED outputs fully covered the visual feedback requirements of all block types.

3.6 LEDs and Lighting

Lighting plays a big role in how children interact with *Blocks o' Code (v3)*. The visual feedback from LEDs helps children connect physical actions to digital outcomes, reinforcing core logic concepts like sequencing, cause and effect, and iteration. Different lighting types, such as status indicators, edge illumination, and matrix displays, are used across the system to make programming both engaging and intuitive.

3.6.1 Discrete LEDs

Discrete light-emitting diodes (LEDs) are among the most fundamental electronic components used for illumination and status indication. They function by emitting light when current passes through a forward-biased semiconductor junction. Because LEDs are

current-driven devices, each one requires a series current-monitoring resistor to regulate forward current and prevent damage to the diode. Typical forward-voltage values range from approximately 1.8V for red LEDs to 3.2V for blue and white variants, with recommended operating currents between 10 mA and 20 mA depending on color and brightness requirements [164]. These parameters allow for direct interfacing with 3.3V or 5V logic-level microcontrollers using simple resistor-based control.

In most embedded applications, discrete LEDs are controlled using pulse-width modulation (PWM) to adjust brightness without reducing efficiency. This is often achieved through low-side MOSFET switching, where each LED channel is connected to a logic-level transistor that enables precise current control and dimming at frequencies between 300 Hz and 1 kHz [163]. At these frequencies, the LEDs appear continuously illuminated to the human eye, minimizing visible flicker while maintaining low power loss. For designs that require consistent brightness under varying voltage conditions, constant-current driver circuits can be used to maintain a stable LED current and extend the component's lifetime [163].

In *Blocks o' Code (v3),* discrete LEDs can be used as a status indicator for both the Brain and Child Blocks, such as power availability, inter-block data transmission, and Wi-Fi activity. Their low cost, reliability, and minimal circuitry make them well-suited for providing real-time visual feedback without significantly increasing power consumption or affecting system complexity. This simplicity supports the project's educational goals by helping children quickly identify system states through clear, intuitive visual cues.

3.6.2 Addressable RGB LEDs

Addressable RGB LEDs integrate individual red, green, and blue emitters with a built-in driver integrated circuit (IC), allowing each LED to be digitally controlled for precise color and brightness output. Unlike discrete LEDs, which require separate wiring and external drivers for each color channel, addressable LEDs use a serial data interface that enables the controls of hundreds of LEDs through a single microcontroller pin. This greatly simplifies wiring and allows complex lighting patterns to be generated programmatically.

A common family of addressable LEDs is the WS281x series, including WS2812B, WS2813, and WS2815. These LEDs operate on a single wire data protocol that transmits a continuous stream of 24-bit color data, 8 bits per channel for red, green, and blue. Each LED passes its data to the next in the chain, creating a daisy-chain communication system that minimizes wiring complexity. The WS2813B runs on 5 V power and uses pulse-width modulation (PWM) at roughly 400 Hz. Newer variants such as WS2813 and WS2815 include a redundant data line that allows the strip to keep functioning even if one LED fails, and they increase the PWM frequency to approximately 2 kHz for smoother visual output [166].

An alternative design is the APA102, which uses a two-wire SPI-compatible interface consisting of a clock and data line. This architecture is less timing-sensitive than the WS2812's one-wire protocol and supports data rates up to several megahertz, making it easier for microcontrollers to maintain consistent frame updates. The APA102 also

operates at a much higher PWM frequency of around 19 kHz, virtually eliminating visible flicker even in camera recordings or rapid motion [165]. Each APA102 pixel also includes a 5-bit global brightness register in addition to 8-bit color channels, allowing finer control over overall intensity without changing color balance [165].

Addressable LED strips typically draw significant current when fully illuminated, with each RGB consuming up to 60 mA at maximum brightness, about 20 mA per color channel. In *Blocks o' Code (v3),* these LEDs can be integrated along the edges of the Child Blocks, providing both functional and engaging visual feedback while maintaining a compact layout. Because only a single data line is needed for communication, the wiring remains simple even as the number of LEDs increases. The system uses software-based brightness control to reduce current draw and manage heat, while still delivering smooth, vibrant animations. The individually addressable nature of these LEDs enables the blocks to display dynamic lighting patterns that represent programming logic such as loops, conditionals, or communication states, making coding concepts more tangible and visually appealing for young learners [166].

3.6.3 Lighting Selection

Both **Discrete LEDs** and **Addressable RGB LEDs** are implemented in *Blocks o' Code (v3)*, each serving distinct purposes within the system.

Discrete LEDs are used across all block types as **status indicators** to communicate essential information such as power state, active communication, or successful program execution. Their low power consumption, simple control method, and long lifespan make them ideal for continuous operation. Since these indicators are easily recognizable and consistent across all blocks, they provide children with immediate visual feedback on the block's functionality without requiring complex color interpretation [163][164].

In contrast, **Addressable RGB LEDs** are incorporated into the **conditional** and **logic-oriented Child Blocks**, lining the internal edges of the block enclosures. These LEDs create dynamic lighting effects such as color transitions, pattern animations, and conditional cues that visually represent logic flow (e.g., *if*, *then*, or *loop* operations). Because each LED can be individually controlled, this allows the system to visually "execute" code, reinforcing cause-and-effect learning for children. The use of APA102 and WS2813 variants provides higher PWM frequencies and redundant data lines, ensuring consistent color performance, smooth animations, and reduced flicker during interactive play [165][166].

By combining both lighting systems, *Blocks o' Code (v3)* achieves a balance between simplicity and interactivity. Discrete LEDs ensure reliable, low-power status communication across all modules, while addressable LEDs enhance engagement through color and motion, transforming abstract coding logic into tangible, visual experiences for young learners. This dual-lighting approach supports the project's educational goals by merging technical reliability with creative expression. The comparison of these lighting systems can be referenced in *Table 3.6.*

Feature	Discrete LEDs	Addressable RGB LEDs
Control Method	Each LED requires its own GPIO pin or external driver	All LEDs controlled through one data line
Color Capability	Single color per LED	Full 24-bit RGB color control per LED
Circuit Complexity	Requires resistors, transistors, or MOSFET drivers for each LED	Integrated driver IC built into each LED
Power Consumption	10-20 mA per LED	Up to 60 mA per RGB LED at full brightness
Scalability	Limited, each LED increases wiring complexity	Easily scalable, hundreds of LEDs on one data linePAM8302A schematics open source
PWM Frequency	Typically 300-1000 Hz	WS2812B ~400 Hz; APA102 ~19 kHz for flicker free display
Cost per Unit	Very low	Higher but offset by simplified wiring and control
Durability	Ideal for simple indicators or status feedback	Best for dynamic visual displays, color effects, and interactive lighting

Table 3.6 *Lighting Comparison*

3.6.4 Senior Design 2 Selection

Both discrete LEDs and WS2812B addressable LEDs were implemented across all blocks in the final design. Discrete LEDs serve as status indicators for power, connectivity, and execution state. WS2812B addressable LEDs are used in two forms: a 4×4 LED matrix for grid-based visual output during program execution, and a chainable LED strip running along the block edges for identification and real-time status feedback. The APA102 and WS2813 variants considered during research were not used; the WS2812B was selected for its wide availability, single-wire protocol compatible with the ESP32, and sufficient PWM performance for the system's animation requirements.

3.7 Speakers

In our project *Blocks O' Code (v3)*, speakers are essential for a musically themed idea. The speaker will turn tiny amplifier power into audible feedback that includes beeps, tones, and musical phrases that children will be able to hear clearly without harshness. Since our blocks are designed to be on the smaller side, we must consider compact, 8Ω, efficient enough to meet the .3 W range from a 3.3 V to 3.7 V rail, while also being mechanically

simple to mount. We analyzed 3 common “micro speaker” classes that balance size, loudness - Sound Pressure Level (SPL), and bandwidth for voice/melody cues.

3.7.1 Same Sky (CUI) CMS-2580-05SP

The CMS-2850-058SP is a compact round micro-speaker intended for small devices where shallow depth and modest power are required. It is specified at 8Ω nominal impedance and .5 W rated input power with 1 W max power for short duty. This speaker is also built around a Neodymium (Nd-F-B) magnet, PET cone, and a stamped SPCC frame. The overall dimensions are 28 mm × 5 mm, which will fit comfortably behind a small front grille in our cube enclosure. Another important specification to analyze is resonant frequency and frequency response which is approximately 500 Hz to 20 kHz (Fo ≈ 500 Hz). These ranges make it suitable to hear clear beeps, tones, and music without demanding large amplifier headroom.

Integration with this device is relatively simple. The speakers' flat round and solder-pads terminals make the driver straightforward to mount a shallow, gasketed foam recess to suppress the speaker's “buzz/rattle”. Placing the speaker as a front-baffled radiator with venting to avoid cancellation of low mids inside the PETG cavity, and keeping the Class-D output leads short and paired will minimize the EMI-induced hiss. As far as a reliability and thermal standpoint, our audio target of .3 W into 8Ω sits well below the parts .5W continuous rating and the 5% THD at 1 kHz, .5 W test point. This allows us to meet the loudness goals we want to achieve without stressing the speaker's diaphragm or raising the temperature in the 3D enclosure. [62]

3.7.2 PUI audio AS03608MR-10-R

The AS03608MR-10-R is a low profile 36 mm micro-speaker intended for compact devices that require more midrange output and headroom than a 28mm class. This speaker at 8Ω is specified with 1 W rated and 1.5 W max input power. The sensitivity for this speaker is around 91 dB with a resonant frequency of around 500 Hz. The speaker's diaphragm is Mylar (PET) with an NdFeB magnet and metal frame with a thin package of only 4.6mm in overall height. These features make it easy to front-mount behind a shallow grille in our 50 mm enclosure. In practice, the larger diaphragm area will output a more noticeable loudness and fullness bump at the same electrical power. This makes it an incredibly ideal speaker for the music-focused block that plays melodies.

The integration for this device in our project is straightforward. We would seat the driver in a shallow, foam-gasketed recess and front baffle it with a slightly larger grill compared to the 28mm option to avoid chuffing. We will also pair this with short, twisted speaker leads from the Class-D amplifier to minimize buzz. The datasheet shows that within our .3 W into 8Ω target sits well below the 1 W rated level, which in turn reduces the thermal and mechanical stress. These conditions align well with our PETG enclosure and sub-watt amplifiers, giving us a comfortable acoustic margin without the amplifier overheating. [63]

3.7.3 Soberton SP-2305L

The SP-2350L is a 23mm × 5 mm ultra-thin round micro-speaker for spaces where the front baffle and stack height are tightly constrained. This specific speaker is specified at 8Ω, .5 W rated with .8 W short-term max, and 89dB at .5 W. The resonant frequency is around 750 Hz at 1 V. This means that clear beeps and User Interface (UI) tones are acceptable with melodies above the low-mid band, with less bass than larger 28-36 mm drivers. This is to be an expected tradeoff since the diaphragm is smaller.

For Blocks O' Code (v3) the integration is also straightforward, like the other two options and stay within our comfortable margins of .3 W into 8Ω target. With this speaker we would seat the flat frame in a shallow, foam-gasketed recess to suppress the "buzz/rattle" as well as wiring short, paired speaker leads from the Class-D amp to minimize the EMI. The datasheet shows that we can comfortably say that this speaker would be reliable and safe for classroom use. The THC is <10% at 2 kHz with a 96-hour .5 W white-noise load. The only condition is that the SPL must remain within ± 3 dB of the initial value. Other than this, the conditions align with PETG enclosures, frequent handling, and our sub-watt operation will allow for our system to keep thermal and mechanical stress below the rated limits. [64]

3.7.4 Speaker Selection

For *Blocks O' Code (v3)*, we decided on the Same Sky (CUI) CMS-2580-05SP as the speaker for all blocks based on *Table 3.7*. This 28mm round driver will fit cleanly behind the front grille of our 50 mm enclosure. This speaker also meets all electrical targets of .3 W into 8Ω and stays within comfortable thermal and mechanical limits while also having a frequency range that supports clear beeps, tones, and music for classroom feedback with no demanding large amplifier headroom.

Choosing a single speaker reduces the complexity of our BOM, purchasing risk, and acoustic variability across blocks. The CMS-2850-058SP's shallow profile simplifies the assembly and works with our gasketed, front-baffeled mounting to minimize "buzz/rattle", keeping Class-D wiring short. We also maintain reliability and heat in our PETG due to the operating point being below the .5 W continuous rating and the distortion test point is 5% at 1 kHz. To summarize, this driver will deliver consistent, child-safe loudness, and clarity in a compact form, making it the most practical system-wide choice.

Attribute	**Same Sky (CUI) CMS-2850-058SP**	**PUI Audio AS03608MR-10-R**	**Soberton SP-2305L**
Cost (USD, 1-pc)	$1.92 (Mouser)	$2.14–$3.89 (Mouser, family variants AS03608MR-LW100-R / AS03608MR-SC-R; same 36 mm, 8 Ω, 1 W spec)	$1.75 (Digi-Key)

Rated / Max Power	0.5 W rated / 1.0 W max.	1.0 W rated / 1.5 W max.	0.5 W rated / 0.8 W max.
Resonant Frequency (Fo)	400–600 Hz (typ. ≈ 500 Hz).	≈ 500 Hz (±20%).	750 Hz (±20%) @ 1.0 V.
Frequency Range (vendor)	Fo to 20 kHz.	Fo to 4.5 kHz.	Fo to 20 kHz.
Sensitivity / SPL (ref. condition)	(per datasheet spec section; Fo–20 kHz band; see sheet).	≈ 91 dB @ 0.1 W / 0.1 m (avg 0.8–1.5 kHz).	89 ±3 dB @ 0.5 W / 10 cm (avg 0.8–1.5 kHz, on baffle).
Overall Size / Height	Ø 28 mm × 5.0 mm (±0.3 mm).	Ø 36 mm × 4.6 mm height.	Ø 23.0 mm × 5.0 mm; 70 mm leads.
Construction notes	Nd-Fe-B magnet; PET cone; SPCC frame.	PET (Mylar) diaphragm; NdFeB magnet; metal frame (per datasheet).	Nd-Fe-B magnet; PET diaphragm; SPCC frame; lead wires included.

Table 3.7 *Speaker Comparison*

3.8 Amplifiers

Clear, safe, and audible sound is essential for *Blocks o' Code (v3)* to provide immediate feedback such as beeps, tones and short musical phrases that help kids connect actions to outcomes. The audio power amplifier sits between the microcontroller's audio source and the speaker, converting small audio signals (analog/PWM or I^2S/TDM) into enough power to drive 4-8Ω transducers. In our system, amplifiers must work reliably from a primary 3.3V rail, fit on compact PCBs, maintain child-safe sound levels, and minimize heat to better preserve battery life and enclosure integrity.

3.8.1 PAM 8302A: Class-D (Analog/PWM input)

The PAM8032A is a mono, filterless Class-D amplifier that is intended for low-voltage, battery-powered products. This specific amplifier provides a filterless speaker drive stage and accepts either an analog signal from a DAC or filtered PWM output from a microcontroller. At 5V into 4Ω it is rated up to 2.5 watts at 10% total harmonic distortion (THD) and has a peak efficiency at around 88%. This will help minimize heat and extend runtime in compact enclosures [40].

Integration with this amplifier is relatively simple. The recommended input network from the datasheet states to use a series capacitor to create a high-pass corner with input resistance, allowing for a low-frequency response. Local decoupling should combine a

small ceramic capacitor close to VDD for high-frequency transients paired with a larger bulk capacitor for low-frequency supply noise. The shutdown ($\overline{SD}$) pin will provide a logic-level shutdown which reduces the current to mA range, enabling pop-free power sequencing when asserted before power on and off. There is also an internal under-voltage lockout at around 2.1V. This will protect the start-up behavior along with short-circuit and thermal-shutdown protections. This is an extremely important goal to achieve for classroom use and student handling.

For our 3.3V system with our 8Ω speaker target, the datasheet curves shows about .65W at 3.6V and 1% THD [40]. This comfortably exceeds our target of approximately 0.3W into 8Ω, allowing us to meet loudness goals while keeping junction temperatures low and avoiding large copper pours for heat spreading.

3.8.2 MAX98357A: Class-D (I^2S + integrated DAC)

The MAX98357A is a mono Class-D speaker amplifier designed to be a small power amplifier that takes digital audio directly and turns it into speaker power. This amplifier accepts a standard Inter-IC Sound (I^2S) stream, utilizing 3 wires: DIN (data), BCLK (bit clock), and LRCLK (left/right clock). This allows us to send audio straight from the ESP32-S3 without first converting it to analog. Another nice simplification is that this amplifier doesn't require a separate MCLK (master clock) which is required for many audio chips. This will reduce the wiring and potential noise on the PCB, allowing for simplification while keeping audio quality high.

Integration on this device aligns with our compact, battery-powered blocks. This amplifier offers 5 hardware-selectable gain steps in I^2S or left-justified mode (3, 6, 9, 12, or 15 dB) , a fixed 12 dB gain in Time-Division Multiplexing (TDM) mode, and a channel selection via SD_MODE, which provides shut-down control for low-power idle [41] supporting battery life when no audio is playing.

For the output, the datasheet shows that at 5V the device delivers up to 3.2W into 4Ω, and at 3.7V the device delivers up to .77-.93W into 8Ω depending on THD, with peak efficiency reaching around 92% [41]. This is far above our intended .3W into 8Ω target at 3.3V to 3.7V range, giving us comfortable headroom for clean playback in a small speaker block.

3.8.3 TPA2028D1 (TI): Class-D (AGC/DRC over I^2C)

The TPA2028D1 (TI) is a mono, filter-free class D speaker amplifier for small, battery-powered projects. It runs directly from a single 2.5V to 5.5V supply rail and adds a digitally controlled preamp with Automatic Gain Control (AGC) and Dynamic Range Compression (DRC) over I^2C. The function of the AGC and DRC is to help keep quiet sounds easier to hear and prevent loud peaks, so a tiny 4-8Ω speaker stays clear and protected without constant tuning in the firmware.

Integration in this device is straightforward since it uses a filterless Bridged-Tied-Load (BTL) which uses two amplifier channels to drive a single speaker, placing the speaker between the two amplifier outputs instead of connecting it to ground [43]. Over I^2C you

can set the gain from -28 to +30 dB in 1 dB steps, limiter threshold, compression ratio, and attack/release/hold times. The shutdown current is also very low at .2 µA and includes short-circuit/thermal protection for additional safety in the classroom.

As far as our output, the capability meets our needs within margin. The datasheet lists up to 3W into 4Ω with a 10% THD. Then around 3.6 V into 8Ω it achieves around .71 W at 1% THD and noise, with efficiency above 90% in the sub-watt region. With this information, we can confidently say that this will keep the heat low for prolonged battery life seeing that the amplifier comfortably exceeds our .3 W into 8Ω target at 3.3 V to 3.7 V.

3.8.4 TAS2562 (TI): Class-D with integrated boost (digital I²S TDM input)

The TAS2562 is a digital-input, mono Class-D speaker amplifier with a built-in DC-DC boost as well as an integrated speaker voltage/current sensing ("IV sense"). This amplifier is designed to deliver high peak power into small loudspeakers while staying efficient and cool, which is exactly what our battery-powered blocks need.

Integration is relatively easy, especially for a compact PCB. This device exposes standard I^2S/ TDM pins along with I^2C control, an active-low shutdown (SDZ), and boost nodes (VBAT, SW, VBST, PVDD). The audio outputs are also filterless, BTL to the speaker, and the IV-sense pins (VSNS_P/VSNS_N) tap the outputs after any ferrite beads. Because of these features, the chip can "see" the true voltage/current, enable real-time protection and tune without additional analog parts. [45]

For this amplifier, the performance does well; the efficiency is around 83.5% at 1 W into 8Ω speaker allowing for low idle noise and strong power-supply rejection. This is incredibly useful in tight enclosures with other digital electronics surrounding it. Since our target is around .3 W into 8Ω, the TAS2562 provides good headroom for clean beeps and short musical tones while keeping heat and battery drain low [44]

3.8.5 Amplifier selection

For *Blocks o' Code (v3)*, two amplifiers were selected to match the different audio needs across blocks based on *Table 3.8*. The first amplifier we decided to go with is the PAM8302A, which will be designated for most child blocks due to it meeting the .3 W into 8Ω speaker target from a 3.3 V rail. It also has high efficiency while keeping the cost for the BOM low. This amplifier integrates cleanly with PWM or DAC outputs and keeps the heat output low, allowing for a prolonged battery life. For the music-focused block, we will be utilizing the MAX98357A amplifier. This was selected because it accepts I^2S directly from the ESP32-Wroom, keeping the signal digital on the PCB, and provides simple, stable volume control in the firmware for clear playback.

Using two amplifier solutions allows for each block to be optimized for its designated role. The PAM8302A keeps the cost down and allows for simple assembly where only beeps/tones are needed, while the MAX98357A delivers clean audio quality for short musical phrases without adding analog routing or extra parts. Both devices are expected to

operate reliably from our primary 3.3 V rail, fit on compact PCBs, and help maintain child-safe sound levels with minimal thermal impact.

After prototype discussions with Dr.Weeks, we have decided to shift to a single, simpler solution. We have selected the LM386 low-voltage audio power amplifier. The LM386 achieves our functional goals with fewer parts, lower integration effort, and easier debugging. The device accepts the ESP32 on-chip DAC output directly, meaning we don't need to use I^2S interface. This amplifier operates comfortably from the 5V rail present on the Child Block while delivering sub-watt power into 8Ω, 0.5 W micro-speaker. Overall, the LM386 selection streamlines manufacturing and testing while meeting the acoustic, thermal, and reliability requirements for *Blocks o' Code (v3).*

Attribute	**PAM8302A**	**MAX98357A**	**TPA2028D1**	**TAS2562**	**LM386**
Cost (estimated)	~$1	~$3	~$2-3	~$4-6	~0.50-$1
Technology	Class-D, filterless	Class-D with integrated DAC	Class-D with I2C AGC/DRC	Class-D with integrated boost and IV sense	Class-AB linear amplifier
Input type	Analog or filtered PWM	Digital I2S (no MCLK)	Analog, I2C control	Digital I2S or TDM, I2C control	Analog
Supply range	2.5-5.5 V	2.5-5.5 V	2.5-5.5 V	2.7-5.5 V (VBAT), 1.8 V logic	4-12 V
Headline power	Up to ~2.5 W into 4 Ω at 5 V	Up to ~3.2 W into 4 Ω at 5 V	Up to ~3 W into 4 Ω at 5 V	Up to ~6 W into 4 Ω with boost (conditions per datasheet)	Up to ~0.5 W into 8 Ω (depending on supply voltage)
Output at ~3.6 V into 8 Ω	~0.65 W at 1% THD	~0.77-0.93 W depending on THD	~0.71 W at 1% THD, ~0.88 W at 10% THD	Easily exceeds 0.3 W target with headroom	~0.2–0.3 W at moderate distortion
Efficiency (peak)	~88%	~92%	~90%+ in sub-watt region	~90%+ with boost	~50–60% (lower than Class-D)
Volume control	Fixed internal gain, set in design or by PWM depth	Firmware digital volume over I2S, hardware gain steps	I2C programmable gain -28 to +30 dB, limiter and compression	Firmware digital volume, limiter and boost control via I2C	External components (gain set with resistor/capacitor)

Special features	Simple BOM, tiny footprint	No MCLK required, click and pop suppression, spread spectrum	AGC and DRC over I2C for consistent loudness	Integrated DC-DC boost, IV sense for speaker protection and tuning	Simple design, minimal external components
Protections	Thermal, overcurrent, UVLO	Click and pop suppression, thermal	Thermal, short-circuit, very low shutdown current	Brown-out prevention, limiter, thermal and short-circuit	Thermal shutdown, short-circuit protection
Idle or shutdown current	Low idle, microamp shutdown via /SD	Low standby and shutdown via SD_MODE	Very low shutdown (~0.2 µA)	Sub-µA shutdown typical	Low idle current
PCB complexity	Low	Low to medium	Low to medium	Medium to high due to boost and sensing pins	Low
Best fit in this project	Default beeper and melody blocks at lowest cost	"Music" block with clean digital path and simple wiring	Analog-input block needing automatic loudness control and protection	Premium block where maximum loudness and protection are required	Simple audio output blocks where low cost and easy integration are prioritized

Table 3.8 *Amplifier Comparison*

3.8.6 Senior Design 2 Selection

The final audio design uses the LM386 amplifier for all blocks.

Earlier in the design process, Class-D amplifiers such as the PAM8302A and MAX98357A were considered. While these options are more efficient and support digital audio, they required additional components and more complex integration with the ESP32. The LM386 was chosen because it is simple to implement and works directly with the analog signals generated by the ESP32. This made it easier to design and debug, especially during PCB development.

Testing showed that the LM386 provided enough gain to drive an 8Ω speaker at an acceptable volume level for the system. Although it is less efficient than a Class-D amplifier, it still meets all the project requirements.

Overall, the LM386 was selected because it reduced design complexity and made the system easier to implement while still providing reliable audio output.

3.9 Filters

Clean, safe audio also relies on simple filtering, so kids can hear clear beeps, tones, and musical phrases without hiss or harshness. In *Blocks O' Code (v3)*, they will gently remove rumble we don't, smooth out buzzy edges from PWM, and keep 8Ω speakers from working harder than they should. We will only implement a few simple parts including tiny capacitors and resistors in the right spots to ensure our sound is clear, keep the PCB compact, and keep everything safe and reliable for classroom use.

3.9.1 Input High-Pass (AC-Coupling)

With our small 8Ω micro-speakers, they can't reproduce deep bass and with a DC on an amplifier input it can waste headroom or cause clicks. Using an input high-pass (AC-Coupling) network will solve both of these issues by blocking the DC and rolls off the sub-bass so that the driver isn't pushed to the point where it can't perform. For *Blocks O' Code (v3)*, we set the corner near the speaker's resonant frequency (≈ 500 Hz for our 28mm drier) so beeps, tones, and music remain clear while cone excursion at very low frequencies is trimmed.

Integration will just be along the analog/PWM path (PAM8302A or TPA2028D1), using a series coupling capacitor into a defined shunt resistor at the amplifier. This makes the cutoff predictable, not leaving it to the IC's internal bias while keeping efficiency high and THD low. Our baseline will be C = 3.3 nF and R = 100 kΩ to ground, giving $f_c = \frac{1}{2\pi RC} \approx 480Hz$, making it right around the speakers' f_0. The plan will be to place the capacitor and resistor close to the input pin, and use a stable dielectric if that is available, while keeping the return short. When the source is PWM from the MCU, the PWM smoothing RC comes first, then the AC-coupler. For power-up/down pops, we will assert the amplifier's shutdown while the rails ramp, then release after the input has settled across the coupling capacitor.

The expected result on our system with .3 W into 8Ω drive, the network will protect the 28 mm speaker from unnecessary excursion below a few hundred hertz, while leaving the core band unchanged. At 480 Hz the response is -3 dB, and by 120 Hz it is down about -12 dB, which reduces low-frequency stress in the small PETG enclosure. No external high-pass will be needed on the MAX98357A digital/I2S path, so this network is applied solely to the analog/PWM blocks.

3.9.2 PWM smoothing Low-Pass

When we generate audio with an MCU's PWM, the waveform shows a fast square wave where the duty cycle carries the audio. That "carrier" is far above the audible band, but without a small low pass ahead of the amplifier, the switching energy can leak through as a hiss or edge noise. A simple RC filter fixes this issue by passing the audio band and attenuating the high frequency remnants so that the amplifier sees a cleaner signal. This will keep our tiny 8Ω speaker from working overtime and reduce the chance of "buzz" inside of the enclosure.

Integration with the amplifier is straightforward. On the analog/PWM path (PAM8302A or TPA2028D1), we will use a compact 2 pole RC before the AC-coupling high-pass (3.8.1). With 2 stages, you get a gentle pass-through for voice/melody while stacking attenuation (around 40 dB/dec) on a typical PWM carrier in the tens of kHz. We would place these parts tight to the amplifier input, keep grounds short, then follow with the AC-coupling capacitor and shunt resistor. This sets the 500 Hz high pass corner for our 28 mm speaker. This flow matches the input recommendations in both amplifiers' datasheets for AC-coupled, filterless Class-D use. On the digital I^2S path (MAX98357A), no analog smoothing will be required since the audio will be transferred digitally and reconstructed on the inside of the device.

The ideal result for our system would mean that with the two-pole RC in place, the audible band is preserved, while the PWM edges are substantially reduced before they reach the Class-D input. That will then lower idle “fizz”, prevent audible artifacts in the PETG enclosure, and avoid unnecessary dissipation in the speaker at ultrasonic frequencies. If we find that listening tests show too much roll-off of bright tones, our capacitance can be reduced, or we can increase our capacitance if we find that extra smoothing will be desired for a lower PWM frequency. This allows us to keep our project small, low-cost, and easy to place on our PCB.

3.9.3 Output EMI Tidy for Filterless Class-D

Our amplifiers that we selected drive the speaker in filterless Class-D (BTL) mode, which allows us to keep our BOM cost low and high efficiency. In this mode, the high frequency switching stays mostly confined to the short loop between the two output pins and the speaker, so that way you wouldn't need a large LC “speaker filter”. Instead, you keep the speaker loop tight, route the two outputs as a close pair, and if needed to add EMI cleanup parts. This approach matches the “filterless” application guidance that is given in the PAM8302A, TPA2028D1, and MAX98357A datasheets.

Integrating this means we would just treat the PCB traces to the speaker as a differential pair while also keeping them short and symmetric to avoid running them under sensitive LEDs or connectors. The default build is no output filter populated, so if the enclosure testing ever shows audible buzz, we will have footprints ready for a light ferrite-bead and capacitor network. We would place one bead in series on each speaker lead and a small differential capacitor across the speaker pad. These parts would then damp the high-frequency edges without touching the audio band.

As a result, that we would expect on our system is that with the short, paired leads and decoupling at the amplifier, the filterless output would remain quiet in theory. The extra footprints allow us to have a low-cost “plan B” if an enclosure or cable run needs it. This will keep our blocks efficient, compact, and consistent with vendor-recommended Class-D layouts. We just need to implement good routing and a small RC/FB network for edge-case cleanup.

3.9.4 Filter Selection

For the analog/PWM audio path (PAM8302A or TPA2028D1), we will adopt a small passive network that combines a two-pole PWM smoothing low-pass with an AC-coupling high-pass at the amplifier input. The low-pass uses two identical RC stages (R = 2.2 kΩ, C = 10 nF per stage), each with a corner near 7.2 kHz, so audio-band content passes while high-frequency PWM energy is attenuated before it reaches the amplifier. Immediately after this, a series coupling capacitor into a defined shunt sets the input high-pass with C = 3.3 nF in series with R = 100 kΩ to ground. This places the −3 dB corner around 480 Hz, close to the 28 mm speaker's resonant frequency. This arrangement blocks DC offsets, trims sub-bass that small drivers cannot reproduce, and aligns with the vendors' AC-coupled, filterless Class-D guidance and pop-free power sequencing $\overline{SD}$ recommendations.

For the digital music path (MAX98357A), no analog input filter is populated because audio is transferred over I^2S and reconstructed inside the device. We keep only optional "EMI tidy" footprints at the speaker terminals which is a ferrite bead in series on each lead and a small differential capacitor (about 1nF) across the pads. This matches typical application usage for a digital-input Class-D with integrated DAC and avoids unnecessary parts on the music block.

Together, these choices keep the bill of materials tiny, protect the 8 Ω micro-speaker from low-frequency stress, and deliver clean, classroom-safe audio from the 3.3 V rail without adding heat or complexity. The comparison of parts to make these choices are referenced in *Table 3.9.*

Filter	**Purpose**	**Where it goes**	**Baseline values (our design)**	**Affects which path**	**Pros**	**Cons / trade-offs**	**BOM / PCB impact**
Input High-Pass	Block DC and roll off sub-bass so the 8 Ω micro-speaker isn't over-driven below its useful band (near Fo).	Series C into a defined R-to-GND at the amp input. Place tight to the input pin.	C_HP = 3.3 nF, R_HP = 100 kΩ → (f_c = 480 Hz) (near the 28 mm speaker Fo).	Analog/PWM (PAM8302A, TPA2028D1)	Protects speaker, reduces pops from DC offsets, predictable cutoff.	Slight low-mid roll-off near cutoff; set too high and tones sound thin.	+2 passives; negligible area.

PWM Smoothing Low-Pass	Attenuate PWM carrier/edges so the amp sees audio, not switching spikes.	Two RC stages before the AC-coupler. Keep returns short.	R1=2.2 kΩ, C1=10 nF and R2=2.2 kΩ, C2=10 nF → each (f_c = 7.2 kHz}); ~40 dB/dec attenuation of HF.	Analog/PWM (PAM8302A, TPA2028D1)	Quiets hiss/fizz from PWM; tiny, cheap; easy to tune (±C).	Too low an (f_c) can dull very bright tones; component tolerance shifts (f_c) slightly.	+4 passives; small area.
Output EMI tidy	Tame very-HF edges/radiated noise in filterless Class-D layouts if enclosure/cabling demands it.	Ferrite bead in series on each speaker lead; 1 nF differential C across speaker pads. Close to the connector.	FB: ~100–120 Ω @ 100 MHz (low DCR). C_DIFF= 1 nF (DNP by default).	Both paths (only if needed)	Keeps BOM minimal by default; "Plan B" if EMI shows up.	Not needed in many layouts; wrong values can shave a touch of extreme treble.	+2 FB +1 C footprints; DNP default.

Table 3.9 *Filter Comparison*

3.10 Microphone

This section investigates microphones suitable for the *Blocks o' Code (v3)* system. The microphone is a key input component that allows the Child and Brain PCBs to detect voice commands, claps, or other audio signals. Selection criteria include signal quality, MCU compatibility, power consumption, and ease of integration.

3.10.1 Adafruit I²S MEMS Microphone Breakout - SPH0645LM4H

The Adafruit I²S MEMS Microphone takes digital data in through the I²S peripheral by detecting audio and converting it to voltage, requiring no analog-to-digital conversion. The issue with taking analog input is that noise can seep in. There are three digital pins: Clock, Data, and Left-Right (Word Select) Clock. The I²S controller drives the clocks at a high frequency to read out the data from the microphone [51]. The SPH0645LM4H offers a frequency response of 50 Hz – 15 kHz and an SNR of 65 dB, suitable for general voice and sound detection applications. Its low operating current and small form factor make it ideal for embedded systems where power efficiency and compactness are priorities [53]. A potential drawback with the SPH0645LM4H is that it requires a microcontroller with I²S support, luckily for our project though both our chosen MCUs have that. It is also only

compatible with 3.3V, not offering a range of voltages for the supply which takes away some flexibility.

3.10.2 Adafruit Electret Microphone Amplifier - MAX4466

The Adafruit Electret Microphone Amplifier uses the MAX4466 op-amp to produce an amplified analog output proportional to sound pressure. It operates on a 3.3 V – 5 V supply (with a typical voltage of 3.3V) and has an adjustable gain potentiometer to fine-tune sensitivity. This microphone has a wide input frequency range of 20 Hz – 20 kHz, making it suitable for full-spectrum audio capture [54]. However, since it produces an analog signal, the microcontroller must include an ADC to process the data, and the signal is more prone to noise interference compared to a digital I²S microphone. Despite these limitations, the electret design remains popular for simple applications and systems without I²S peripherals.

3.10.3 INMP441 I²S Microphone

The INMP441 is a low-power, high-performance digital MEMS microphone that outputs audio data via the I²S interface. It supports both left and right channel configurations through a simple pin setting, making it suitable for stereo setups. The microphone features a frequency response of 60 Hz – 15 kHz and a signal-to-noise ratio of 61dB. Operating from 1.8 V – 3.3 V with a typical current draw of 2.2 mA, it provides good performance for general-purpose voice recognition and acoustic sensing. The INMP441 is widely supported on microcontrollers such as the ESP32 and RP2040 and has a compact footprint, but draws more current than the SPH0645LM4H, making it slightly less power efficient [55].

3.10.4 Microphone Selection

After comparing the available options as referenced in *Table 3.10*, the **Adafruit I²S MEMS Microphone (SPH0645LM4H)** was selected for the *Blocks o' Code (v3)* system. Its digital I²S interface allows direct communication with the microcontroller without requiring an ADC, reducing analog noise and simplifying the signal chain. While the Electret Microphone Amplifier provides a broader frequency response, its analog nature introduces additional complexity and potential noise. Signal integrity and clarity are very important because in our project the microphone output is going to be put through an ML API for voice recognition. We want the microphone output to be clear enough for the ML to accurately understand what the user is inputting, so using analog is too risky. The INMP441 offers similar functionality at a lower cost but consumes more current and has slightly lower signal-to-noise performance. Therefore, the SPH0645LM4H provides the best balance of signal clarity, low power consumption, and integration simplicity for the project's audio input requirements.

	I²S MEMS Microphone Breakout	**Adafruit Electret Microphone Amplifier**	**INMP441 I²S Microphone**

Cost	~$7.00	~$7.00	~$3.00
MCU Compatibility	I²S supported	ADC supported	I²S supported
Output Type	Digital	Analog	Digital
Input Frequency	50Hz - 15kHz	20Hz - 20 kHz	20Hz - 20 kHz
Supply Voltage	3.3V	3.3V	3.3V
Signal-to-Noise Ratio	65dB	60dB	61dB
Supply Current	600µA	25µA	2.2mA
Clock Frequency	~3072 kHz	N/A	2.4MHz

Table 3.10 *Microphone Comparison*

3.10.5 Senior Design 2 Selection

Microphone integration was not implemented in SD2. The Voice Block described in Section 2.5.1.1 was identified as an Advanced Goal and was deferred to future work. While SPH0645LM4H hardware was procured, firmware support for microphone input was not completed within the project timeline. No microphone block type exists in the final system's block type registry. Microphone integration remains a candidate for a future iteration alongside expansion of the block type set.

3.11 Connectors and Interfacing

Connectors and interfaces are an important part of how the blocks in *Blocks o' Code (v3)* work together. Since each block needs to share both data and power with the others, choosing the right type of connector is key to keeping everything reliable, safe, and easy to use. The connectors also affect how the blocks feel when handled by children, so they must be durable and simple to connect without exposing any sharp or fragile parts.

In this project, connectors are what link the Brain Block with the Child Blocks and make communication between them possible. Because the blocks are meant to be used often in classroom settings, the connectors need to stay strong even after many uses and remain safe if touched. To decide on the best option, we compared several types of connectors including magnetic pogo pins, board-to-board headers, and magnetic coupling systems. Each studied how well it handled power, data, alignment, and durability to find the most practical and child-friendly choice for *Blocks o' Code (v3)*.

3.11.1 Magnetic Pogo-Pin Connectors: Adafruit Industries LLC - 5358

While pogo pin connectors were previously discussed for power delivery, they can also be used for data communication in modular electronic systems such as *Blocks o' Code (v3)*. Magnetic pogo pin connectors are spring loaded conductive contacts that create an electrical path between two aligned surfaces when pressed together [124]. Each pogo pin includes a plunger, a barrel, and a small internal spring that keeps pressure on the mating pad, maintaining a steady connection as the modules move or shift [125]. When magnets are placed around the contact area, the connectors pull themselves into alignment automatically and lock in place, which makes connecting and disconnecting the blocks much easier for the user [124]. Since this setup allows direct metal to metal contact, pogo pins can handle common low speed protocols like I²C, SPI, and UART. Because of this, they are often seen in modular systems, wearables, and testing fixtures where compact and repeatable connections are needed [124][131].

From an electrical point of view, pogo pin connectors usually have contact resistances between about 20mΩ and 50mΩ, depending on their geometry and the type of plating used [81]. They tend to work best at low or moderate data rates, but when frequency increases, effects like stray capacitance and inductance can start to distort the signal or cause mismatch issues [127]. Some research shows that by adjusting contact geometry and improving PCB routing, the usable bandwidth can reach faster standards such as USB 3.0 [127]. Gold plating is normally added to the pins to reduce oxidation and keep resistance stable across many connection cycles. Many pogo pin designs are rated for 20,000 or even 500,000 uses, though occasional increases in resistance can still occur if dust or debris gets into the contacts [128][129][130].

For the implementation in *Blocks o' Code (v3)*, the Adafruit Industries LLC - 5358connector was seen as a good option for handling both power and data between the blocks. This connector has four pins with a 2.54 mm spacing and gold-plated contacts that help keep resistance low and signals stable [158]. Each pin can carry up to about 2A of current, which is more than enough for what our system needs [158]. Its small size makes it easy to fit inside the block, and because the pins are spring-loaded, they stay connected even if the blocks move slightly while in use.

Since our project only needs to send low-speed signals between the Brain and Child blocks, this connector works well for data lines like UART or I²C while still providing 5V and ground through the other pins. The gold coating also helps prevent corrosion, which is important in classroom settings where the blocks will be handled a lot. Overall, this connector is simple, durable, and easy to integrate, making it a good choice for combining power and data between the Brain and Child blocks in *Blocks o' Code (v3).*

3.11.2 Board-to-Board and Header Connectors

Board to board and header connectors are often used in electronic systems to make dependable electrical links between multiple printed circuit boards (PCBs) or between a PCB and external modules [126]. These types of connectors come in several forms, such as pin headers, sockets, mezzanine connectors, and card edge designs, depending on the

layout and number of signals that need to be passed through. Overall, their job is to create a stable connection that supports both power and data sharing between different boards that work together in one system [132].

Usually, standard pin headers are made of rows of metal pins that follow a set spacing, most commonly 2.54 mm or 1.27 mm. This spacing works well with through hole and surface mount setups [136]. They are generally used together with female headers or sockets, making it easy to disconnect boards during testing or reassembly. Mezzanine style connectors, meanwhile, are designed for tighter spaces and can handle more pins while keeping good alignment. They can be arranged side by side or stacked, depending on how the boards are placed inside the device [133].

From an electrical standpoint, board to board and header connectors can carry both slow and high-speed signals, though their reliability depends a lot on factors such as pin shape, spacing, and the resistance of each contact [137]. In most cases, contact resistance stays within a few milliohms [134]. To keep connections stable, gold plating is used on the contacts to stop corrosion and keep conductivity steady after many uses. Materials like PBT and nylon are common for outer housing, since they offer both mechanical strength and electrical insulation [138]. Still, when connectors are exposed to vibration or if they are not aligned perfectly, the contact may loosen slightly over time, which can cause signal drops or wear on the pins [135].

In the *Blocks o' Code (v3)* project, these connectors were first thought about being used for testing internal communication between modules and for early prototype work. They are easy to find, inexpensive, and work well for quick testing setups. However, as the project develops, we might face several issues that are clear compared to using magnetic or pogo pin systems. One major design requirement was to support magnetic data transfer, which standard headers cannot achieve. They also need precise alignment and more insertion force, which is difficult for children to use safely. The exposed pins are another problem, since they can bend, cause short circuits, or become unsafe when handled too often. Because of these reasons, while board to board and header connectors can be helpful during testing, they cannot be chosen for the final design, where both safety and child friendly interaction are essential, as well as meeting design requirements.

3.11.3 Magnetic Coupling and Contactless Interfaces: ADuM1201

Magnetic coupling and contactless interfaces let the blocks connect without any visible metal pins. The pieces pull together using magnetic attraction or field coupling, so alignment happens naturally instead of by force. In these "wireless connectors," magnets help the parts line up on their own, and once they are in position, the contacts or coils begin to transfer signals [137]. Some designs even include a small amount of mechanical flexibility that helps fix minor misalignments, letting the connector settle itself through the pull of the magnets [139].

From an electrical point of view, these connectors can work in two main ways: through direct conductive pads or through inductive field coupling. In inductive systems, a changing current in one coil produces a signal in another coil across a small air gap, and

some research setups have reached between 1 and 8.5 Gbps [139]. Conductive magnetic connectors instead rely on small pads that touch after alignment and move digital signals directly between the surfaces [137][140]. The performance of each type depends on how strong the magnets are, how the coils are shaped, and how well everything lines up. For inductive designs, efficiency and data quality depend on how close the coils are placed [141]. Conductive magnetic versions care more about low resistance at the contact points, so power and data stay steady [141].

When balancing design choices, magnetic coupling has both pros and tradeoffs. Stronger magnets make alignment effortless but can make separation harder for children. Weaker ones are easier to pull apart but sometimes miss perfect alignment, which lowers how well power or data move through [142]. Components such as ferrite cores or tuning capacitors can help the signal and reduce noise, but they also add cost and complexity [143]. Even so, magnetic connectors fit our design goals very well. They are safe, closed off from crumbs and dust, and meet the original plan for magnetic data transfer. The main limits are some power loss and signal drop when the blocks do not align perfectly, so our design focuses on magnet strength and coil geometry to keep every connection steady.

We came across one part that matches this idea, ADuM1201 uses magnetic coupling to safely send digital signals across a small gap. Instead of using direct metal contact, it transfers data through changing magnetic fields inside the chip, allowing isolation between circuits. It supports low-speed communication like UART or I²C, which is similar to how the Brain and Child blocks in our project exchange data [159]. Since it works within a few millimeters, it could be used between closely aligned blocks to move data without wires or exposed pins. Using a part like the ADuM1201 could make future versions of *Blocks o' Code (v3)* fully contactless, giving them a cleaner look and reducing the chance of connection wear or damage over time. This magnetic coupling works nicely for our project because it gives that "plug free" experience we wanted, children just bring the blocks near each other, the magnets do the rest, and both power and data start to flow. The sealed connection also keeps the modules clean and strong for classroom use. Our main challenge is making sure the magnets and circuitry handle both power and data efficiently, even when the blocks are not aligned perfectly.

3.11.4 Connector and Interfacing Selection

After comparing different connector options, as referenced in *Table 3.11*, **magnetic pogo-pin connectors** were chosen as the best fit for *Blocks o' Code (v3)*. The specific part we selected is the Adafruit Industries **LLC - 5358**, a four-pin connector with gold-plated contacts and a small 2.54 mm spacing. This connector fits our goal of making the blocks easy to use, safe to handle, and strong enough for classroom use. Each pin can carry both data and power, which helps keep the design simple and clean. The gold plating also keeps the resistance low and protects the contacts from corrosion, so the connection stays reliable even after many uses. The magnets help the blocks align automatically, so users don't have to plug anything in or worry about breaking parts.

Other connector types, like board-to-board headers and magnetic coupling systems, were also researched. Board headers are cheap and great for early testing, but their exposed pins

aren't very safe for kids and can bend or wear out easily. Magnetic coupling, which sends signals through magnetic fields instead of direct contact, looked interesting because it removes all exposed metal, but it's harder to design and less efficient right now. It might be a good option for future versions once the system is more advanced.

Overall, the Adafruit Industries LLC - 5358 connector gives the project the right balance of safety, simplicity, and performance. It supports both power and low-speed data transfer while being compact and durable. The magnetic "snap" makes the blocks easy and fun to connect, which fits perfectly with the playful and hands-on learning style that *Blocks o' Code (v3)* aims to create.

Category	**Adafruit Industries LLC – 5358 (Pogo-pins)**	**Board-to-Board / Header Connectors**	**ADuM1201 (Magnetic Coupling / Contactless Interface)**
Data Capability	Supports low-speed data (UART, I²C, SPI) up to ~1 Mbps	Handles digital signals and moderate current (1-3 A per pin).	Transfers digital data magnetically up to ~1 Mbps across isolation barrier
Durability / Lifespan	20,000-500,000 cycles depending on material and plating quality.	Durable but can wear from repeated mechanical stress or bent pins.	No physical wear; lifespan depends on magnet strength and coil alignment.
Ease of Use	Simple magnetic "snap" alignment; connects quickly.	Requires precise alignment and insertion force.	Very easy "snap-on" connection. Automatic alignment.
Safety & Classroom Suitability	Safe to touch, no exposed parts, resistant to minor debris.	Exposed metal pins can cause shorts therefore it is not child-safe.	Completely enclosed, touch-safe, and resistant to dust and spills.
Advantages	Compact, dual-purpose for data and power, reliable connection.	Inexpensive, widely available, simple for prototyping.	Sealed design, no wear, intuitive to use, safe for children.
Limitations	Can be affected by dust or oxidation. Requires clean contact pads.	Easily bent pins, risk of short circuits, poor for modular use.	Lower efficiency, needs tuning components, higher cost and complexity.
Use in Blocks o' Code (v3)	Highly suitable, meets magnetic alignment and power/data needs.	Useful only for internal testing and prototyping.	Ideal for child-safe plug-free design; supports project goals.

Table 3.11 *Connector Comparison*

3.12 Power Supply Unit

This section explores different power supplies and energy sources that could be used to power *Blocks o' Code (v3)*. The options considered include batteries, wall adapters, and

portable power banks. Each one offers different levels of portability, efficiency, and safety, which are important when designing a system meant for children to use.

3.12.1 Battery: 18650 Li-ion Cell (Protected)

Batteries are one of the most common sources of power for embedded systems and portable devices. They provide direct current output, which can easily supply microcontrollers, sensors, and displays. Among the many battery chemistries, lithium-ion and nickel-metal hydride are the most researched for compact electronics because they offer high energy density, stable voltage, and rechargeability [88]. Lithium-ion cells typically operate at 3.6V with an energy density of 150–250 Wh/kg, while nickel-metal hydride batteries average around 60-120 Wh/kg [89].

In modular projects like *Blocks o' Code (V3)*, the main advantages of batteries include portability and ease of integration. Each block could operate independently without needing to remain connected to an external power source. However, batteries require protection of circuits to prevent overcharging, deep discharge, or short-circuit conditions. Their lifetime also decreases over repeated charge and discharge cycles and swelling or leakage can occur if the cells are physically damaged [90][91]. For safety reasons, the use of rechargeable cells in toys requires additional enclosures and thermal protection mechanisms.

One battery type that fits well with this project is the 18650 lithium-ion cell. This is a cylindrical rechargeable battery that provides a nominal voltage of 3.6V and a maximum charge voltage of 4.2V. Most 18650 cells have capacities between 2600 and 3500 mAh, which gives them enough energy to power low-current systems like LEDs, microcontrollers, and small displays for several hours [156]. They also have a high energy density compared to other cell types, allowing more power to be stored in a compact size. Because of their shape and strong metal casing, 18650 cells are easier to mount and handle compared to flat pouch-style LiPo batteries [157].

For *Blocks o' Code (v3)*, the 18650 cell provides the right balance between capacity, safety, and simplicity. A single cell can supply enough energy for each module to run independently without needing an external power supply. Using a protected version of the 18650 adds built-in safety features like overcharge and short-circuit protection, which is important since the blocks will be used by children. The voltage range of this battery also matches well with 3.3V and 5V regulators, making it easy to integrate with the existing circuits. Overall, the 18650 Li-ion cell is an efficient and practical energy source that supports the modular, rechargeable design of *Blocks o' Code (v3)*.

3.12.2 Wall Outlet

Wall outlet power provides a continuous and stable energy source through an external power supply or adapter. This method converts alternating current from the mains into a regulated direct current that can be used by electronic circuits. One of its main advantages is that it removes the limitations of energy storage. Devices powered by an outlet can operate indefinitely without the need for recharging or battery replacement.

From a research standpoint, wall power systems are efficient for high-current applications, with modern AC to DC converters achieving efficiencies above 85 percent [92]. However, the use of outlet cables introduces safety concerns for a product intended for children. Long cords and exposed connectors may cause tripping or electrical hazards if not properly insulated. In addition, relying on a fixed power source reduces portability, which conflicts with the modular and flexible design intent of the project.

3.12.3 Miniature Power Banks

Power banks are portable energy storage units that can provide USB or DC output to recharge or directly power devices. They combine battery technology with voltage regulation and protection circuits. Most commercial power banks contain lithium-ion cells and can deliver 5 V at currents up to 3 A depending on their internal design [93]. Research indicates that power banks have an average conversion efficiency of 80 to 90 percent, depending on the load and converter architecture, for example, some studies estimate internal losses of 5% to 12% and net usable output around 90% under favorable conditions [94][105].

Using power banks in *Blocks o' Code (v3)* could offer a practical compromise between portability and safety. Instead of embedding batteries inside every block, a shared power bank could charge multiple blocks simultaneously or provide energy to a main charging tray. This reduces the need for individual battery management while still keeping the system mobile. The disadvantages include increased charging time and possible uneven power distribution if several blocks draw current at once. Despite this, power banks are convenient, inexpensive, and widely available, making them a strong candidate for experimental testing.

3.12.4 Power Supply Selection

The 2S 18650 lithium-ion battery pack was chosen for *Blocks o' Code (v3)* because it offers a good balance between capacity, size, and safety. Each cell provides a nominal voltage of 3.7V and a capacity of 2600mAh, and when two are connected in parallel, the total capacity doubles to 5200mAh while keeping the same voltage. This gives the blocks enough energy to power the microcontroller, display, and sensors for several hours of continuous use without frequent recharging. The cells also have built-in protection circuits that help prevent overcharging, over-discharging, and short circuits, which keeps the design safe for children to handle during classroom activities. This decision was made based on *Table 3.12*.

Another reason this battery type was selected is its wide availability and reliability. 18650 cells are commonly used in laptops and portable devices, so they are easy to replace and have well-documented performance characteristics. Their cylindrical shape fits neatly into the block's internal layout, and using two cells in parallel reduces current stress on each one, improving efficiency and extending battery life. Overall, the 2S 18650 configuration provides dependable power while staying simple, cost-effective, and safe for the project's modular design.

Category	Battery (18650 Li-ion Cell (Protected))	Wall Outlet	Power Banks
Power Availability	3.6 V nominal, 4.2 V max, 2600–3500 mAh capacity; requires recharging	Continuous supply while plugged in	5V regulated output; 5000–20 000 mAh typical capacity
Portability	Highly portable and compact	Stationary, not portable	Portable but adds some bulk
Efficiency	>90%	Very high and stable	80%-90%
Durability & Cost	$3	$10	$15
Safety	Safe with proper battery chemistry	Requires supervision and proper insulation	Includes built-in protection circuits
Advantages	Compact, easy to integrate, wireless operation	Reliable continuous power for testing or demos	Convenient, reusable, and easy to recharge
Limitations	Requires charger IC	Tied to wall connection, not child-friendly	Bulkier, limited by internal charge cycles
Use in *Blocks o' Code (v3)*	Ideal for individual child blocks for portability	Useful for classroom setups or fixed stations	Suitable for travel or flexible modular setups without wall access

Table 3.12 *PSU Comparison*

3.13 Communication Protocols

Reliable and efficient communication is fundamental to the operation of *Blocks o' Code (v3)*. Each module must exchange commands, logic states, and sensory feedback with low latency, so children can see cause-and-effect relationships between the physical blocks and the virtual code execution. Because our design uses the ESP32-WROOM-32, we looked into four main communication protocols that would work best for our setup: I²C, SPI, Wi-Fi/BLE, and UART. Each protocol offers different benefits in terms of wiring, speed, scalability, and how well it fits the goals of *Blocks o' Code (v3).*

3.13.1 I²C (Inter-Integrated Circuit)

I²C is a two-wire serial protocol that uses a data line (SDA) and a clock line (SCL) to communicate between one master and multiple slave devices [16]. It's synchronous, meaning all devices share the same clock, which keeps timing consistent. The master sends a start condition; the slave responds with an acknowledgement (ACK), and then data bytes are sent one by one until a stop condition is issued. Each message includes a unique address, so the master knows exactly which device it's talking to.

The bus can operate at different speeds depending on the system's needs, such as standard (100 kHz), fast (400 kHz), or fast-mode plus (1 MHz). Since the *Blocks o' Code (v3)* system only needs to send small packets of information (like brightness, color pattern, or logic signals), a 400 kHz clock rate is fast enough for smooth communication without adding unnecessary power draw.

From a hardware perspective, I²C uses what are called open-drain lines, which means that the devices on the bus can only pull the line low and can't drive it high. Instead, small pull-up resistors (typically 4.7 kΩ) connected to 3.3V keep the lines high when no device is pulling them low [16]. This setup lets multiple devices safely share the same data and clock lines without electrical conflicts. Both the ESP32-S3 and the RP2040 operate at 3.3V logic, which means their voltage levels are already compatible, and no level shifting is required. The RP2040 includes two built-in I²C controllers that can act as either master or slave and run at speeds up to 1 MHz [17], while the ESP32-S3 also integrates two I²C peripherals that can easily serve as the master or slave with flexible pin mapping [25].

In *Blocks o' Code (v3)*, all data and power connections between the Brain Block and the Child Blocks can pass through magnetic pogo-pin connectors. These connectors carry the SDA and SCL lines along with power (VCC) and ground, forming a clean, modular, and detachable interface between blocks. This design allows I²C to transfer both control data and sensor feedback efficiently through the same compact connector. The pogo pins keep the blocks aligned and ensure reliable electrical connection while still letting children snap them together or pull them apart easily. Because the pins are spring-loaded and gold-plated, they maintain solid contact even with repeated use which is an important detail for a classroom environment where the block will be handled frequently.

This makes I²C an especially beneficial choice for *Blocks o' Code (v3)* because of the protocol's low pin count and shared-bus structure with just two lines, enough to communicate with every block and keeping the design simple and scalable [16]. Each Child Block can have its own address stored in EEPROM, allowing the Brain Block to automatically detect and communicate with each one on startup. The protocol also supports clock stretching, meaning a slower device can hold the clock line until it's ready. This is useful for making sure every block responds reliably even when performing heavy tasks like refreshing LED animations or reading from sensors [16][17].

3.13.2 SPI (Serial Peripheral Interface)

SPI, or Serial Peripheral Interface, is a high-speed, full-duplex communication protocol commonly used for transferring large amounts of data quickly between a master and one or more slave devices [18][22]. It operates using four main signal lines: a Serial Clock (SCLK) generated by the master, Master-Out-Slave-In (MOSI) for data transmission from master to slave, Master-In-Slave-Out (MISO) for data transmission from slave to master, and a Chip Select (CS) line that enables a specific slave device for communication. Unlike I2C, which shares a two-wire bus with unique addresses, SPI transmits and receives data simultaneously, one bit per clock edge allowing for much better throughput and lower latency [18].

Because every signal has its own line, SPI is ideal for applications that need continuous or high-bandwidth data exchange, such as LED drivers, OLED displays, and sensors. With no addressing or acknowledgment cycles, it moves data efficiently and with very little timing overhead [22]. The trade-off is that every additional device requires its own chip-select line, which increases wiring complexity. For a small, contained system, this is manageable, but in larger modular designs the number of pins and traces can grow quickly.

The RP2040 includes two hardware SPI controllers that support all four SPI modes and can operate at clock speeds up to 133 MHz, depending on the peripheral [17]. This provides more than enough performance for *Blocks o' Code (v3).* In practice, running SPI between 4 MHz and 8 MHz keeps signals stable while still delivering real-time responsiveness. At these speeds, an RP2040 can refresh a 64-pixel LED matrix in under a millisecond, fast enough for smooth animations and quick reactions to children inputs. The ESP32-S3 also supports SPI peripherals such as flash memory or sensors [19].

Electrically, SPI uses push-pull drivers, meaning each line is actively driven both high and low rather than relying on pull-up resistors as in I²C [103]. This enables faster edge transitions and higher data rates, but signal quality becomes more sensitive to PCB layout. If traces are too long or routed too close together, signals can interfere with each other (crosstalk) or create small voltage fluctuations in the ground plane (ground bounce). To prevent this, the traces are kept short and well-spaced and decoupled capacitors are placed near each SPI device to stabilize the power line. In *Blocks o' Code (v3)*, this layout is easy to achieve since each Child Block's SPI lines stay contained on a single board, typically connected to the RP2040 directly to its corresponding peripheral.

SPI is especially useful for intra-block communication where high speed updates are needed for LEDs, displays, or sensor data [46]. Each Child Block can handle its visual output locally using SPI while staying responsive to the Brain Block's command over I²C. This division of work keeps the shared I²C bus dedicated to coordination and logic flow, while SPI manages time-critical updates without delay and flicker.

Although SPI excels at speed, it isn't ideal for connecting many external modules. If SPI were used between the Brain and Child Blocks, each module would need its own CS line and ground reference, leading to more pogo-pin connections and greater wiring complexity [18]. This would go against the simple, modular design that *Blocks o' Code (v3)* aims to maintain. In comparison, I²C allows all modules to share the same two communication lines, making it far more scalable for inter-block communication.

In summary, SPI delivers low-latency, high-bandwidth performance that perfectly fits the needs of each Child Block's local hardware. It can complement I²C by handling fast internal updates like LED animations or sensor reads while I²C manages overall system coordination. By combining these two protocols, *Blocks o' Code (v3)* achieves both speed and scalability, ensuring smooth operation, real-time responsiveness, and deterministic performance across every module [17][18][19][22][46].

3.13.3 Wi-Fi/BLE

Before exploring how these wireless protocols fit into *Blocks o' Code (v3)*, it is important to understand what Wi-Fi and Bluetooth Low Energy (BLE) are and how they differ.

Wi-Fi is a wireless networking technology defined by the IEEE 802.11 family of standards. It allows devices to exchange data using radio waves in the 2.4 GHz and 5 GHz frequency bands [48]. Wi-Fi was designed for high-throughput local-area communication, allowing multiple devices to share a network through an access point or router. Its data rates and range make it ideal for transferring large amounts of information, such as real-time visual feedback between embedded systems and companion applications.

Bluetooth Low Energy (BLE), introduced in the Bluetooth 4.0 specification and expanded in Bluetooth 5.0, was developed for low-power, short-range communication [47]. It operates in the same 2.4 GHz ISM band as Wi-Fi but sends short packets of data only when needed instead of maintaining a constant connection. This design reduces power consumption and is commonly used in devices like wearables, sensors, and IoT modules that need lightweight communication over extended periods. While BLE is slower and shorter in range than Wi-Fi, it is much more efficient and reliable for small, quick data exchanges.

From a technical standpoint, Wi-Fi (IEEE 802.11 b/g/n) uses Orthogonal Frequency Division Multiplexing (OFDM), which splits a channel into multiple subcarriers that send data streams in parallel [49]. This allows for higher throughput and improved efficiency, with typical data rates ranging from 72 to 150 Mbps under the 802.11n standard supported by the ESP32-S3 [26][48][47]. BLE, on the other hand, uses Gaussian Frequency-Shift Keying (GFSK) modulation at 1 or 2 Mbps and applies adaptive frequency hopping across 40 channels to reduce interference with Wi-Fi signals [47]. Both protocols share the 2.4 GHz spectrum, but BLE's short, low-energy transmissions make it less likely to conflict with other wireless traffic.

In *Blocks o' Code (v3)*, the ESP32-S3 acts as a wireless communication hub, handling both Wi-Fi and BLE connections. Wi-Fi is used mainly for Brain-to-App communication, creating a fast and stable link between the Brain Block and the companion mobile or web app. This connection allows users to edit programs, trigger block actions, and view program flow in real time. The Brain Block can either create its own network (Access Point mode) or connect to an existing Wi-Fi network, giving flexibility in both classroom and home environments [26].

BLE complements Wi-Fi by handling lightweight, short-range communication. It is useful for tasks like pairing, block discovery, or sending small configuration updates. Since BLE only transmits when necessary, it helps conserve power and avoids network congestion. In classroom use, BLE can help with quick setup between the Brain Block and tablets before switching over to Wi-Fi for higher-speed data transfer.

The ESP32-S3 supports running both Wi-Fi and BLE at the same time through Espressif's coexistence scheduler [26]. It also includes hardware encryption for both protocols, using

WPA2-PSK for Wi-Fi and AES-CCM for BLE, which ensures secure data transfer. This combination allows the systems to communicate with the companion app and nearby devices at the same time without interference while keeping user data safe.

Overall, Wi-Fi provides the high-bandwidth connection needed for real-time control and data exchange with the companion app, while BLE offers an efficient and low-power option for short-term range communication and device setup. Together, these wireless features can help *Blocks o' Code (v3)* connect the digital and physical sides of programming, creating a reliable and interactive learning experience [26][47][48][49].

3.13.4 UART (Universal Asynchronous Receiver-Transmitter)

UART, or Universal Asynchronous Receiver-Transmitter, is one of the simplest and most widely used serial communication protocols. Unlike synchronous protocols such as I²C and SPI, UART does not require a shared clock signal. Instead, both devices agree on a common baud rate (bits per second), and data is sent asynchronously over two lines: one for transmit (TX) and one for receive (RX) [49]. Because of its simplicity and point-to-point nature, UART is often used for debugging, console output, and device configuration.

At a basic level, UART communication works by converting parallel data from the processor into serial bits that are transmitted one after another. Each data frame typically includes a start bit, a defined number of data bits (usually 8), an optional parity bit for error checking, and one or more stop bits to mark the end of transmission [50]. Both devices must use the same configuration so the receiver can correctly reconstruct each byte. Common baud rates range from 9,600 to 115,200 bits per second, though modern microcontrollers like the ESP32-S3 and RP2040 can achieve much higher speeds when needed [17][25].

Because UART transmits asynchronously, it does not rely on a clock line for timing. Instead, each device's internal clock generates timing based on the agreed baud rate, which can lead to small timing mismatches over long transmissions. However, for short data bursts, it remains highly reliable and requires minimal hardware. With the simplicity of UART wiring just being two signal lines and a shared ground, it makes it easy to implement and test in embedded systems [49][50].

The RP2040 includes two independent UART controllers that support up to 32-bit data transfer, FIFO buffering, and flexible baud rate configuration [17]. Similarly, the ESP32-S3 features three UART peripherals that can be mapped to various pins, allowing for debugging, firmware upload, and real-time serial output [25]. This makes UART ideal for initial firmware flashing, system monitoring, and direct communication with development tools such as Arduino IDE or Espressif IDF serial console.

In *Blocks o' Code (v3)*, UART primarily can serve during development, debugging, and testing phases. It provides a simple way to print diagnostic data from the Brain Block or Child Blocks, monitor communication traffic, and verify I²C or SPI performance. For example, UART can log when a block connects or when commands are sent between the Brain Block and a Child Block, helping the team quickly detect and troubleshoot issues.

During prototyping, UART serves as the main interface for uploading firmware or reading sensor data before finalizing the I²C based block-to-block network.

While UART is reliable and easy to use, it is not designed for multi-device communication. Each connection requires a dedicated TX and RX pair, so scaling to many modules would require additional wiring and microcontroller pins [49][50]. For that reason, UART is reserved for maintenance and debugging tasks rather than the main communication path between blocks.

Overall, UART remains an essential protocol within *Blocks o' Code (v3)* for setup and testing. It complements the other communication interfaces by offering a direct, low-level method for programming, debugging, and verifying signals without interfering with normal system operation.

3.13.5 Protocol Selection

After evaluating each protocol's performance, scalability, and compatibility with the selected hardware, *Blocks o' Code (v3)* adopts a layered hybrid communication structure that strategically combines I²C, SPI, Wi-Fi, and UART. The comparison of these protocols can be found in *Table 3.13.* This approach takes advantage of each protocol's strengths while keeping the system simple, modular, and responsive to real-time changes.

The I²C bus acts as the main coordination layer between the Brain Block and all Child Blocks. Using magnetic pogo-pin connectors, the SDA and SCL lines transfer both control and feedback data through a compact, detachable interface. The simplicity of I²C wiring using just two communication lines shared across all modules makes it ideal for a modular system that may have up to fifteen active blocks connected at once. Each Child Block stores a unique address in EEPROM, which allows the ESP32-S3 in the Brain Block to automatically detect and identify them during startup. Since I²C supports multi-slave addressing and clock stretching, the Brain can manage multiple Child Blocks simultaneously without losing synchronization.

This setup also supports Child to Child communication, which is essential for how the programming logic flows across physical blocks. When a user stacks or connects blocks representing logic operations (like "if," "then," or "loop"), data signals are passed over the shared I²C bus. The Brain Block coordinates this traffic by broadcasting or routing messages between specific addresses, but the actual data exchange between Child Blocks, such as sensor input activating a nearby LED block can happen directly across the I²C line. This design keeps all blocks responsive and allows children to see cause-and-effect feedback instantly as they modify their physical code sequence [16][17][25].

Within each Child Block, SPI handles high-speed, localized communication between the RP2040 microcontroller and internal components such as LED matrices, OLED displays, or sensors. SPI's full-duplex, high-frequency capability (up to several MHz) ensures animations and visual feedback updates almost instantly. By keeping these high-bandwidth operations local to each Child Block, SPI reduces traffic on the main I²C bus, ensuring

smooth operation even as more modules are added. This separation of system-level coordination (I²C) and local updates (SPI) maintains fast response times and a consistent user experience [17][18][22][46].

At the top layer, Wi-Fi and Bluetooth Low Energy (BLE) extend connectivity beyond the physical blocks. The ESP32-S3 uses Wi-Fi to communicate with the companion app, allowing users to run, pause, and visualize their code sequences in real time. BLE is also available for short-range and low-power interactions, such as pairing with a mobile device or classroom tablet. Together, these wireless protocols make it possible for children to interact with *Blocks o' Code (v3)* through an app interface while the system continues executing commands in real time. Both Wi-Fi and BLE use modern security standards like WPA2-PSK and AES-CCM to protect wireless data exchanges [24][25][47][48].

Finally, UART remains a valuable support interface during development and testing. It provides a direct serial connection for programming, debugging, and logging messages without interfering with the active I²C and SPI communication. Both the ESP32-S3 and RP2040 have built-in UART peripherals that make it easy to monitor performance, test message timing, and troubleshoot communication issues between blocks [17][25][49][50].

Protocol	**Physical Layer**	**Clocking Type**	**Typical Speed**	**Wiring Complexity**	**Scalability**	**Primary Role in System**	**Implemented On**
I²C	Two-wire (SDA, SCL)	Synchronous	100 kHz – 1 MHz	Low	High (multi-slave)	Brain ↔ Child communication bus Child ↔ Child communication bus	ESP32-S3 (master), RP2040 (slave)
SPI	Four-wire (MOSI, MISO, SCLK, CS)	Synchronous	1 - 10 MHz	Moderate	Low (point-to-point)	High-speed LED and display control (within Child Blocks)	RP2040
Wi-Fi / BLE	2.4 GHz RF wireless	Asynchronous	Up to 150 Mbps (Wi-Fi 4)	None (wireless)	High	App ↔ Brain connectivity	ESP32-S3
UART	Two-wire (TX, RX)	Asynchronous	≤ 3 Mbps	High (one-to-one)	Low	Debugging, testing, firmware upload	ESP32-S3 / RP2040

Table 3.13 *Communication Protocol Comparison*

3.13.6 Senior Design 2 Selection

In the final system, I²C runs at 100 kHz between the ESP32-WROOM-32 (Brain, I²C master) and all Child Blocks (I²C slaves). SPI handles local peripheral communication on each block independently. Wi-Fi carries the Brain-to-app link over TCP. EEPROM-based address assignment was not implemented; each Child Block's I²C address is statically defined in firmware (0x08–0x15). The "ESP32-S3 / RP2040" split described in the research phase was not used. All blocks run the ESP32-WROOM-32 under a single ESP-IDF toolchain.

3.14 Memory Storage

Memory architecture in *Blocks o' Code (v3)* underpins the system's runtime responsiveness, non-volatile data integrity, and firmware update reliability. The system requires high-speed volatile memory for real-time task execution and deterministic peripheral servicing, complemented by non-volatile storage for persistent configuration and identity data. All selected memory technologies must operate from a 3.3 V supply rail, occupy minimal board area, and maintain integrity under repeated educational use and frequent firmware reprogramming cycles.

3.14.1 SRAM

Static Random-Access Memory (SRAM) serves as a high-speed, volatile memory for when a system is active in execution. Unlike Dynamic Random-Access Memory (DRAM), SRAM maintains data without refresh cycles, storing each bit in a bistable latch. Data will only persist when power is applied, providing low-latency, and quick access. This makes SRAM ideal for stacks, buffers, and time-critical tasks.

In *Blocks o' Code (v3)*, the SRAM functions as the MCU's "workbench" that we will use for RTOS task stacks, driver queues (*I²C*/ /SPI/UART), audio/display buffers, and temporary state. Due to the SRAM being volatile, anything stored here is assumed to survive a reboot, meaning the persistent IDs and settings live elsewhere. SRAM's "no-refresh" behavior keeps the latency predictable for PWM timing, *I²C* servicing, and User-Interface (UI) updates. This is why SRAM primarily is used for active, time-sensitive data rather than long-term storage. [95][96].

Faster access and deterministic timing will allow us to meet real-time deadlines without large safety margins. While SRAM offers fast access, its limited density requires careful buffer management and reuse of ring buffers, analogous to broader computing practice where SRAM serves as cache or registers for time-critical operations. Conceptually, this mirrors broader computing practice where SRAM backs the most time-critical work such as cache and register files, while larger, slower memories hold bulk data. [97]

3.14.2 Flash ROM

Flash-ROM is non-volatile storage and electrically erasable, enabling firmware and static configuration storage across power cycles. Flash ROM is organized into sectors, which can be individually erased and rewritten. In modern devices, this sector-erasable architecture allows updating a single sector without needing to erase the entire chip. This is useful in splitting program and settings/log regions, while updates will happen using standard embedded workflows. One possible workflow is In-System Programming (ISP), which is when the app is halted and placed in a boot/program mode. The other is In-Application Programming (IAP), which is when the device supports flash while the application continues running. Several MCUs include a bootloader in ROM, so the system is able to recover and accept new firmware even if the application image is erased. To keep this reliable, we must ensure that we pair flash updates with verification while testing variable data carefully so limited write/erase endurance isn't exhausted. [98]

Flash ROM is widely used in embedded systems to store firmware and data. Several MCUs include a ROM bootloader to ensure that firmware is able to be received over a port for recovery. These are the behaviors and constraints we would follow while partitioning the program versus data regions to plan safe update paths. [99]

3.14.3 EEPROM

Electrically Erasable Programmable Read-Only Memory (EEPROM) is a non-volatile memory that retains data without power and supports multiple read, write, and erase cycles [101]. A key advantage to EEPROM is the byte-level addressability allowing individual bytes to be updated without erasing large blocks. This characteristic makes EEPROM well-suited for storing small, frequently updated data such as device IDs, calibration constants, and configuration parameters [100]. Its write endurance for small updates typically exceeds that of block-erasable flash, enhancing reliability in applications with repeated minor writes.

In *Blocks o' Code (v3)*, each Child PCB utilizes a small I^2C EEPROM to store per-block identity and calibration data so that the values persist through resets and firmware updates. This method is the typical practice used in embedded systems where EEPROM stores calibration data and configuration settings that must survive power-off and field reflashing. Data updates are intentionally small and infrequent to maximize device longevity, and integrity checks are applied to maintain consistency. This approach mirrors standard embedded system practices for maintaining persistent configuration and calibration data while minimizing flash wear [100][101] [102].

3.14.4 Memory Storage Selection

For *Blocks o' Code (v3)*, we will be using a small serial EEPROM as the sole non-volatile store for per-block data, which was decided based on *Table 3.14*. The reasons for choosing this memory storage is simple, EEPROM is non-volatile and electrically reprogrammable while also being byte-addressable. This allows us to update tiny records such as IDs, calibration values, or settings without erasing large sectors or risking data loss EEPROMs also offers higher write endurance for small, frequent updates compared

to block-erased flash. This suits classroom use where devices may be reconfigured or calibrated many times throughout the school year. To conclude, EEPROMs match our need for a dependable “notepad” that survives power cycles and firmware reflashes without complicated wear-management.

Attribute	SRAM	Flash ROM	EEPROM
Volatility	Volatile	Non-volatile	Non-volatile
Erase / Write Granularity	Random-access read/write (byte/word); no erase requirement	Sector/page level erase; program by word or page after erase	Byte-level addressable; no erase requirement
Access speed	Very fast, deterministic	Slower than SRAM, dependent on type of Flash	Slower than SRAM, faster than flash for small writes (higher endurance)
Typical Embedded Use	RTOS stacks, DMA buffers, queues; caches in broader systems.	Firmware, static assets/config; field updateable logs.	IDs, settings, calibration data; small records that may change occasionally.
Selection Rationale	Deterministic timing, low-latency.	Non-volatile program storage with structured update path	Non-volatile, byte-addressable, high endurance for frequent small writes.

Table 3.14 *Memory Storage Comparison*

3.14.5 Senior Design 2 Selection

In the final implementation, SRAM and Flash ROM are used as expected through the ESP-IDF and FreeRTOS framework. SRAM holds runtime task stacks, communication buffers, and block state data, while Flash stores the firmware binary for each block type. EEPROM was not included in the final PCB design. Rather than storing a unique address in EEPROM at startup, each Child Block's I²C address is statically assigned at compile time within its firmware template, spanning the range 0x08–0x16. This approach simplified bring-up, eliminated the need for a pre-programming step to write unique addresses to each unit, and removed a hardware component that was not necessary given the fixed block type assignments in the current system.

3.15 Version Control Systems

Version Control Systems are great tools to leverage in projects that include lots of software development, especially in collaboration. Version control tracks every single change to the

project files, which allows for collaboration with conflict resolution. It will maintain a record of who made changes, when they made the changes, and why (commit messages). This also allows for developers to create new features and ideas without affecting the main source code and risking breaking the project functionality. Taking advantage of version control for *Blocks o' Code (v3)* will allow for consistent code integration, easier debugging, and traceability of changes.

3.15.1 Git

Git is a distributed version control system that allows developers to maintain local repositories with full history. Key features include branching, merging, and offline work capability. Git tracks changes at a granular level, which helps in isolating features, bug fixes, and experiments. An advantage of Git is that it allows developers to work offline, supports powerful branching and merging, and is fast and lightweight. A disadvantage is that it can have a steep learning curve for beginners, and managing complex merges or resolving conflicts can be challenging [58].

3.15.2 Github

GitHub is a cloud-based platform that hosts Git repositories and provides collaboration tools such as pull requests, code reviews, issue tracking, and project management features. An advantage of GitHub is that it enables easy team collaboration, integrates with numerous third-party tools and CI/CD pipelines, and has strong community support with extensive documentation. A disadvantage is that free plans limit private repositories, some advanced features require paid subscriptions, and cloud-based operations depend on internet access [57].

3.15.3 BitBucket

BitBucket is a cloud-based Git hosting service that offers free private repositories for small teams and strong integration with Atlassian products like Jira and Confluence. An advantage of BitBucket is that it is ideal for private or enterprise-level projects and includes built-in CI/CD pipelines. A disadvantage is that it has a smaller user community than GitHub, its documentation and interface can be less intuitive, and it offers limited third-party integrations outside the Atlassian ecosystem [59].

3.15.4 Version Control System Selection

For the *Blocks o' Code* project, **GitHub** was selected as the primary version control platform. This decision was made based on the comparison in *Table 3.15*. GitHub was chosen because it provides robust collaboration features, such as pull requests, code reviews, and issue tracking, which are essential for coordinating work among multiple team members. Its widespread use in both academia and industry ensures strong community support and extensive documentation, making troubleshooting and learning easier. Additionally, GitHub integrates seamlessly with other development tools and CI/CD pipelines, which streamlines the build, test, and deployment process. While alternatives like BitBucket offer advantages for private enterprise projects, GitHub's balance of

accessibility, collaboration capabilities, and community resources made it the most suitable choice for this multi-member, educational project.

	Git (standalone)	**Github**	**BitBucket**
Hosting	Local-only	Cloud-based hosting	Cloud-based hosting
Collaboration	Limited	High, pull requests, issues	High, pull requests, issues
Privacy	N/A	Limited free private repositories	Free for small teams
Best Use	Local development	Open-source or team projects	Private projects

Table 3.15 *Version Control System Comparison*

3.16 Embedded Framework

The embedded framework for *Blocks o' Code (v3)* defines the software environment for the microcontrollers, managing peripheral devices, sensors, actuators, and communication both between blocks and with the companion application. Selecting an appropriate framework is critical because it determines how efficiently the firmware can handle real-time operations, manage resources, and interact with hardware. A well-chosen framework also simplifies development, debugging, and future maintenance. For this project, compatibility with the selected microcontrollers, the ESP32-S3 and RP2040, is essential to ensure reliable performance and seamless block-to-block and block-to-application interactions.

3.16.1 ESP-IDF

ESP-IDF is Espressif's IoT Development Framework built for the ESP32 series. It is an open-source framework under the Apache 2.0 license. A big advantage to ESP-IDF is the ability to leverage the microcontroller because of its many software components such as peripheral drivers, RTOS, and wireless connectivity libraries for Wi-Fi and Bluetooth [37]. A major drawback to ESP-IDF is that it is only compatible with ESP32 devices, and in our project we are also using a Raspberry Pi MCU, so we would need another framework along with ESP-IDF if we were to choose this framework.

3.16.2 Arduino

Arduino is an open-source electronics platform that provides a simplified programming environment and a large library ecosystem for microcontroller development [38]. It supports a wide range of boards, including the ESP32 and RP2040, making it highly portable across hardware platforms. Arduino's main advantage is its ease of use and rapid prototyping capabilities, which allow developers to quickly test hardware interactions and implement basic functionalities. However, Arduino abstracts many low-level details, which can limit performance and fine-grained control over hardware. For applications

requiring precise timing, real-time task management, or advanced peripheral usage, Arduino may not provide sufficient capabilities.

3.16.3 PicoSDK

PicoSDK is the official development framework for the Raspberry Pi Pico and other RP2040-based boards. It is designed for programming in C and C++ and provides low-level access to the microcontroller's dual cores, I/O, timers, and peripherals [39]. A major advantage of PicoSDK is the ability to fully utilize the RP2040's hardware capabilities, including precise timing and multicore processing. A drawback of PicoSDK is that it requires more embedded systems knowledge and effort to implement higher-level features, as it provides minimal abstractions compared to frameworks like Arduino. It is also limited to RP2040-based boards, so it cannot be used for ESP32 devices.

3.16.4 Embedded Framework Selection

For *Blocks o' Code (v3)*, **ESP-IDF** was selected for the **ESP32-WROOM-32** because it provides full access to wireless connectivity, RTOS support, and peripheral drivers necessary for reliable block-to-block and block-to-application communication.

ESP-IDF was selected as the primary build framework, with the Arduino-esp32 library integrated as a component to leverage its ecosystem where needed. Arduino, while easier to use as a standalone option, was not chosen as the root framework because it lacks the low-level RTOS control and peripheral management that ESP-IDF provides. Even though using two frameworks requires separate toolchains and learning two different ones, we think the tradeoff is worth it to fully leverage each MCU's abilities. To see the comparison of each framework, reference *Table 3.16*.

	ESP-IDF	**Arduino**	**PicoSDK**
Compatible with MCUs	ESP32-S3 only	Both	RP2040 only
Real-time Performance	Built-in FreeRTOS	Limited	Libraries and features for precise timing
Peripheral Access	Extensive Drivers	Has libraries	Full low-level access

Table 3.16 *Embedded Framework Comparison*

3.16.5 Senior Design 2 Selection

ESP-IDF served as the primary development framework for all blocks (Brain and Child), running on the ESP32-WROOM-32. Arduino was not used as a standalone framework; instead, the Arduino-esp32 library was integrated as a component within ESP-IDF, allowing use of Arduino-compatible libraries while retaining ESP-IDF's RTOS, peripheral drivers, and build system as the foundation. PicoSDK was not used, as the RP2040 was not

included in the final design. This ESP-IDF-first approach with Arduino component support simplified the build system, reduced toolchain fragmentation, and allowed a consistent firmware structure across all block types.

3.17 Companion Application Framework

This section explores the frameworks considered for developing the companion application for *Blocks o' Code (v3)*. The companion app allows users to interface with the physical blocks, monitor status, see block configuration, and have real-time visual feedback. Choosing the right framework is essential to ensure cross-platform compatibility, high performance, maintainability, and ease of development, particularly favoring options with a shorter ramp time.

3.17.1 React Native

React Native, developed by Meta, is a UI framework for iOS and Android, with cross-platform compatibility, allowing for one codebase [33]. This framework uses JavaScript and React's declarative UI model. The declarative approach allows developers to describe what the user interface should look like in components such as "Thumbnail", "Video", etc. [31] Its whole purpose is to make mobile app development simpler. One significant advantage of React Native is its large community, which provides extensive libraries, tools, and support that accelerate development and troubleshooting. Another advantage is quick development through features like live and hot reloading, which allow developers to see changes instantly without rebuilding the app. Some drawbacks include performance limitations compared to fully native apps, particularly for graphics-intensive or resource-heavy tasks, because of the JavaScript bridge [32].

3.17.2 Unity

Unity is an app development framework focused on real-time 3D and 2D simulations, particularly for gaming. This framework uses C# as a programming language, and is cross-platform across Android, iOS, Windows, macOS, web, and also console. This broad platform support makes Unity a versatile choice. It also has very visually rich graphics abilities, great for an application that wants to have engaging visualizations. Additionally, Unity offers an extensive asset store and a large developer community, providing numerous prebuilt resources and documentation that streamline development. However, the framework can be excessive for simpler mobile applications, as it introduces more overhead and complexity than lightweight mobile frameworks [35]. It also is a harder ramp given that it is built for 3D real-time models, making it more complex than a typical mobile UI.

3.17.3 Flutter

Flutter is Google's mobile UI development framework that enables the creation of natively compiled applications from a single codebase using the Dart programming language. Flutter's declarative UI approach allows developers to define the appearance of widgets for a given state, which the framework then renders automatically. One advantage of Flutter is its high-performance rendering engine, which produces smooth animations and

responsive interfaces. Another advantage is cross-platform portability, with a single codebase supporting iOS, Android, web, and desktop applications [36]. Drawbacks include a smaller developer ecosystem compared to JavaScript-based frameworks, potential complexity when integrating platform-specific features through platform channels, and the requirement to learn Dart, which is less commonly used than other languages.

3.17.4 Companion App Framework Selection

After evaluating React Native, Unity, and Flutter, **Flutter** was selected as the framework for the *Blocks o' Code (v3)* companion app. Flutter provides strong cross-platform performance and compiles directly to native code, resulting in smooth and responsive interfaces.

While React Native has a large community, Flutter offers more consistent performance and a modern UI toolkit suited for real-time feedback and configuration. Unity was considered too heavy for this project, as it is mainly designed for complex 2D/3D simulations and games, which is currently out of the scope for our project.

Flutter also has extensive documentation and community support, and Dart's C-style syntax makes it easier for the team to learn, given their familiarity with C-like languages. Overall, Flutter strikes a good balance between performance, maintainability, and ease of development. This decision was made based on *Table 3.17*.

	React Native	**Unity**	**Flutter**
Language	JavaScript	C#	Dart
Cross-Platform	Yes (mobile)	Yes (mobile, desktop, web, console)	Yes (mobile, desktop, web)
Performance	Limited (compared to native)	High (for graphics-heavy)	High
Typical Use Case	Mobile Apps	Games	Mobile Apps
Complexity	Medium	High	Medium

Table 3.17 *Companion App Framework Comparison*

Chapter 4 – Standards and Design Constraints

4.1 Industrial Standards

4.1.1 Electrical Standards

Electrical standards basically describe how voltage, current, and safety are handled in a circuit. Following these rules helps every part of the system work properly together and keeps users safe. In *Blocks o' Code (v3)*, these standards were used to keep all blocks running at safe, low voltages that are easy to manage in a classroom setting. The main idea is that every block follows the same power limits so they can connect and share power or data without causing damage or safety issues.

All circuits in *Blocks o' Code (v3)* follow low-voltage operation under 5V, which stays within the Safety Extra Low Voltage (SELV) range [160]. Most of the logic components, such as the ESP32-WROOM, and electronic components, such as the speaker and displays, run at 3.3V. The 3.3V logic level was chosen because it works well with the ESP32-WROOM and makes communication between the Brain and Child blocks easier to manage. The 5V input from the USB-C port is mainly used for charging, but there's also a small 5V boost rail that turns on only when brighter LEDs or louder sound are needed. This saves power while still letting the blocks show stronger light or audio effects when active. All voltage and current values stay within the safe limits listed in each part's datasheet.

Each block also follows standard protection and power safety practices. Voltage regulators include built-in overcurrent and short-circuit protection, and the battery circuit prevents overcharging and reverse connection [120][121][123]. Decoupling capacitors are placed near key power pins to smooth out any voltage dips or noise when the blocks are connected or when wireless functions are active [161]. These protection steps are based on common consumer-electronics safety principles, helping the system stay stable and safe even after repeated classroom use.

For connectivity, the USB-C port follows the Type-C specification and accepts a standard 5V input from certified chargers. The connector's control pins handle current detection automatically, allowing safe charging without special cables. The selected USB-C connector is designed for many plug-in cycles, supporting long-term durability. Between the blocks, the magnetic pogo-pin connectors use gold-plated contacts that resist corrosion and meet contact-current ratings recommended by the manufacturer [158]. This ensures a steady connection for both power and data while keeping the design easy and child-friendly.

Wireless and electromagnetic safety standards were also considered. The ESP32-WROOM module used in both PCBs is pre-certified under FCC and CE standards, meaning it already meets wireless and emission rules [160]. The PCB design follows the manufacturer's guidelines for antenna clearance and grounding to maintain strong wireless performance. ESD (electrostatic discharge) protection and solid ground planes were added following IPC-2221 layout spacing rules to reduce static or interference during classroom use [162].

Together, these electrical standards define how each block manages its power, connections, and safety. By following low voltage, well protected, and standardized design practices,

Blocks o' Code (v3) ensures reliable performance, user safety, and durability for educational environments.

4.1.2 Communication Standards

4.1.2.1 I²C Communication Standard

The Inter-Integrated Circuit (I²C) protocol, originally developed by Philips in 1982, is a standardized serial interface widely used in embedded systems for communication between low-speed peripherals and microcontrollers. I²C operates on two bidirectional lines, the SDA (Serial Data) and the SCL (Serial Clock), both pulled up to a defined logic level (common 3.3 V in our design) through resistors to maintain signal integrity.

I²C follows the NXP I²C Bus Specification, which defines modes including Standard (100kHz), Fast (400 kHz), Fast (400 kHz), and Fast Mode Plus (1 MHz) operation [16]. Each device on the bus is assigned a unique 7-bit or 10-bit address, allowing multiple peripherals to share the same two wires.

In *Blocks o' Code (v3),* I²C is used for inter-block communication between the Brain Block and Child Blocks, following these standardized voltage and speed limits to ensure reliable data transfer. Sticking to the I²C standard also helps maintain compatibility with future hardware updates or third-party modules using the same electrical conventions.

4.1.4 Mechanical and Manufacturing Standards

In our project, our mechanical rules are simple and tied to open, vendor-published guidance that matches how we'll actually build the Blocks O' Code (v3) enclosures. We will be utilizing PETG FDM standards to ensure our parts print consistently and assemble cleanly without rework. We use Prusa Knowledge Base to set our baseline: heated bed around 80-85 °C, tuned retraction/cooling to limit stringing, and the expectation that overhangs are less forgiving than PLA. These printer behaviors drives our printing orientation choices and our decision to avoid long, unsupported bridges in the 50mm enclosure. [60]

For external interfaces and materials, we anchor our requirements to public specifications that we are able to access online. The face mounted USB-C receptacle must meet a 10,000-cycle durability rating and survive repeated handling. With this in mind we are selecting connectors whose datasheets specify the lifetime and design out panel cutout to the vendor's drawing. [61]

4.2 Design Constraints

4.2.1 Power Constraints

The power design of *Blocks o' Code (v3)* is built around a single rechargeable 18650 lithium-ion battery. This battery gives a steady voltage of about 3.6V and enough capacity to run the blocks for a few hours, but it also limits how much power we can use at one time. Because every block works on its own, each one needs to manage power carefully so that performance stays stable and safe while students use it in the classroom.

One of the main limits comes from how the battery voltage changes as it discharges, from around 4.2V when full down to about 3.0V when nearly empty. To keep the system running normally through that range, a buck-boost regulator (TPS63070) is used to hold the output steady at 3.3V for the microcontroller and other components. This keeps the blocks consistent, but since regulators lose a little energy as heat, it slightly shortens battery life. To help with that, low power modes on the ESP32 and efficient regulators are used so the blocks don't waste extra power when idle.

Charging also creates some limits. The MCP73832T charger controls how much current goes into the battery, usually around 500mA, to keep it safe and cool. Because the charging process follows constant current and constant voltage steps, it takes a few hours to fully recharge a block. That means the blocks need to be used with realistic activity times between charges. Protection parts, like reverse polarity diodes and thermal limits, make charging safer but also add small voltage drops and take up a bit of space on the PCB.

The power demand inside each block changes depending on what it's doing. The ESP32 draws more current when Wi-Fi is active, LEDs and displays need extra current when bright, and the speaker uses more power when sound plays loudly. To handle this, each block includes decoupling capacitors and a small 5V boost converter that turns on only when brighter lights or louder audio are needed. This keeps performance strong without draining the battery all the time.

Finally, the system must stay within safe power and temperature limits since everything fits inside a cube. All voltage rails stay under 5V to meet safety standards, and current levels are kept low to avoid heat buildup inside the enclosure. These limits guide how long the blocks can run, how fast they charge, and how reliably they perform during classroom use.

4.2.2 Size and Layout Constraints

Blocks O' Code (v3) uses a standardized 5 × 5 × 5 inch enclosure to keep our system compact, durable, and easy to assemble. Inside of the constructed 3D cube defines the usable footprint for the PCB, speaker, battery, and wiring which constrains our board outlines, component heights, and connector placement. The most critical part is that the PCB must fit with space leftover for screw bosses and wall thickness. Our layouts prioritize a single board with corner mounting, short cable runs, and unobstructed access to test pads. Due to each enclosure being 3D-printed the mating features, cutouts, and button/light windows are designed to have clearances to avoid binding or misalignment in the assembly process.

User interaction and inter-block connectivity further constrains the shape of our layout. The features for each block whether that be USB-C cutouts, status LEDs, buttons, and the magnetic/pogo interface must register consistently to the enclosure. The pogo contact edge demands a straight, reinforced section of the PCB with copper keep-backs to prevent chipping, and trace escape near this edge is limited to preserve the mechanical strength. Across all blocks the shared footprints, connector orientations, and fastener patterns reduce manufacturing risk and speed up the assembly process. These size and layout constraints

combined allows for us to ensure every block fits the enclosure, can be built repeatably, and operates in safe conditions without acoustic, thermal, or electrical interference.

4.2.3 Memory Constraints

The *Blocks o' Code* system has limited onboard memory, which impacts both program storage and runtime data handling. Each microcontroller has a finite amount of Flash memory for firmware and EEPROM or RAM for temporary storage of sensor readings, user input, and communication buffers. Because the system logs sensor data, processes user commands, and manages inter-PCB communication, memory must be carefully allocated and managed to prevent overflows or data corruption. Efficient memory usage is critical, especially for features like data logging, real-time processing, and multitasking between inputs and outputs.

The advantage of working within these memory constraints is that it encourages highly efficient coding practices, such as optimizing algorithms, reusing buffers, and prioritizing essential features. However, the drawbacks are significant: memory limitations can restrict the complexity of the system, force compromises in data resolution or logging frequency, and limit future expansion or feature upgrades. Developers must carefully consider trade-offs between functionality, responsiveness, and memory consumption. In addition, debugging and testing are more challenging when memory is constrained, as improper allocation or leaks can cause unexpected behavior or system crashes.

4.2.4 Communication Constraints

Communication is one of the most important parts of *Blocks o' Code (v3)* because every block needs to talk to the others quickly and reliably for the system to work in real time. Our setup uses several communication methods – I²C, SPI, Wi-Fi/BLE, and UART. Each has its own benefits and limitations. These protocols shape how we design the hardware layout, manage data, and handle timing between the Brain Block and the Child Blocks.

The biggest limitation comes from I²C, which is the main wired bus connecting all the blocks together. Since I²C only uses two shared lines (SDA and SCL), we have to be careful with how many blocks we connect before the signal starts to weaken or get delayed. The more devices we add, the higher total capacitance on the bus, which can affect timing and reliability. To keep communication stable, the pull-up resistors have to be sized correctly to keep the signal clean without wasting power. I²C's speed, usually around 400 kHz, also limits how often the Brain Block can update every Child Block. That means we have to plan how data is sent. That involves prioritizing quick updates for LED patterns, less frequent messages for status checks, and coordinating interrupts for tasks.

Inside each Child Block, SPI is used for faster, short-distance communication with components like LED drivers or displays. SPI runs much faster than I²C but needs more wires (MOSI, MISO, SCLK, and CS). Since each PCB is small, routing all these traces while keeping them short and properly spaced is a challenge. If the lines are too long or too close, signals can interfere with each other, which can cause display flicker or incorrect data flow. We have to balance space and performance so the blocks stay compact but still responsive.

Wi-Fi and Bluetooth Low Energy add another layer of complexity. Each ESP32-WROOM module can connect wirelessly, but classroom environments are often noisy, with many devices and routers that can interfere with signals, and cause small delays or disconnects. For this reason, we can keep time-critical communication, such as logic between blocks, wired through I²C, while Wi-Fi is mainly used for connecting to the companion app or sending data back for visualization.

The magnetic pogo-pin connectors that link the blocks also come with their own communication constraints. Because these pins carry both data and power, alignment and wear matter a lot. Even small shifts or dirt on the contacts can cause the connection to drop or introduce noise on the line. To handle this, our firmware includes built-in error checking and automatic reconnection routiness so that the system can recover on its own if a block disconnects momentarily.

Lastly, UART is mainly used for programming and debugging during development, but it still uses pins on the microcontroller. Since only one UART channel can be used at a time, we need to plan its use, so it does not interfere with normal operation.

All of these factors, from the bus speed and signal integrity to connector alignment, define the communication limits of *Blocks o' Code (v3).* Finding the right balance between speed, simplicity, and reliability has been one of the most important and challenging parts of designing how these blocks interact.

4.3 Practical Constraints

4.3.1 Time

Time is a major constraint for this project because we are only given a fixed academic year (Fall and Spring) to complete all phases. This limitation heavily restricts the scope of what can be accomplished, as deadlines cannot be moved, and the project must progress at the pace dictated by the senior design schedule. Any delays can have significant ripple effects, particularly on critical milestones such as PCB fabrication, testing, and final integration.

To mitigate this constraint, the team has taken proactive measures to work ahead whenever possible. Implementation began in parallel with the planning phase of SD1, and we prioritized early group formation and project decision-making to maximize productive time. This forward momentum has allowed us to address potential risks early, identify dependencies, and allocate resources effectively.

A key challenge within this limited timeframe is ensuring thorough and accurate testing. Rushing tests could compromise results or lead to design errors, so careful planning and scheduling of test procedures are essential. We anticipate setbacks, such as design revisions, part delivery delays, or unexpected bugs, and have strategies in place to adapt without falling behind.

PCB design in particular is a time-intensive process, compounded by fabrication and shipping lead times. Early design iterations, rigorous review, and planning for shipping delays are critical to staying on track.

The team addresses the time constraint through disciplined time management and consistent collaboration. We have a team rule to never struggle in silence: all members actively communicate challenges and progress. Each member is responsible for completing their tasks on time, attending meetings, and contributing to group problem-solving. By setting the goal to stay ahead of deadlines rather than merely meeting them, we create a buffer for unexpected delays while maintaining project quality.

Additionally, we use project management tools to track tasks, dependencies, and milestones. This helps prioritize work, allocate effort efficiently, and ensure that high-risk or high-impact tasks receive attention first. Regular reflection and adjustment of our timeline allow us to stay realistic yet ambitious in completing the project on schedule.

4.3.2 Budget

Budget is one of the most central practical constraints because it determines which features make it into the prototype, how many units we can build for testing, and whether schools can adopt the kit at scale. To keep the per-cost block predictable, the design we chose favors widely available, low-variance components and reuse the same enclosure, fasteners, and PCB footprints across blocks. This approach reduces the complexity of purchasing items, shortens assembly time, and limits the number of items that can experience price spikes.

Cost tradeoffs are handled at the subsystem level. For audio, most blocks use a low-cost Class-D path (PAM8302A with PWM/analog input) that meets the .3 W into 8Ω target with minimal parts and very low heat. A single "music" block justifies a higher-cost digital path (MAX98357A with I^2S) to deliver cleaner playback without extra analog routing. Similar choices pop up over the course of this project such as standardizing connectors and passives, selecting dual-sourced regulators, and preferred parts that are stocked by multiple distributors. These decisions keep function where it matters while keeping our BOM in check.

Procurement and build planning also reflects the budget constraint. As a team we have decided to do staged buys to avoid over-committing before designs stabilize, and maintain a modest contingency to absorb price swings or replacements. We also have open-source toolchains (ESP-IDF, PicoSDK) and shared test rigs to minimize non-recurring engineering costs. Together, these practices help keep our project financially realistic without compromising safety, reliability, or classroom usability.

4.3.3 Environmental

Blocks o' Code (v3) is made to be used in classrooms, so the design has to handle everyday conditions like temperature changes, humidity, and frequent handling by students. The blocks need to stay safe, durable, and reliable even if they are dropped or used often.

Since each block has a battery and small electronic parts inside, heat and moisture are important to control. The enclosure is designed to keep dust out and protect the circuit, while the parts inside are chosen to stay cool and work efficiently. Charging current is kept

low so the battery doesn't overheat, and all materials used are safe and non-toxic for classroom use.

Humidity and static electricity can also affect performance, so the pogo-pin connectors are gold-plated to prevent rust, and the PCBs have ESD protection to avoid damage. The plastic shells are 3D printed from materials like PLA or PETG, which are lightweight and easy to replace if broken.

Overall, these environmental limits guide how the blocks are built and protected so they can last through daily classroom use while staying safe and reliable for students.

4.3.4 Manufacturability

Manufacturing is an important constraint for *Blocks o' Code (v3)* since the system is a functional prototype rather than a mass-produced classroom product. Our main focus has been on demonstrating reliable functionality, durability, and modular interaction between blocks, but several manufacturability factors influence how easily the design could be replicated or scaled for real-world classroom use.

The first constraint involves custom PCB fabrication. Each block contains an ESP32-WROOM, LEDs, sensors, amplifiers, filters, and pogo-pin connectors that require precise trace routing and alignment within the small enclosure. The limited space makes it difficult to maintain proper clearances for power and signal lines, and even small layout or alignment errors can cause instability or fit issues. Minor manufacturing tolerances in both the PCB and 3D-printed housings can also impact how blocks connect and snap together.

Another major manufacturability constraint is component availability and sourcing. The project relies on components such as the ESP32-WROOM, pogo-pin connectors, LEDs and displays that have varying stock levels depending on distributor supply chains. During research, several recommended parts from online sources were found to be out of stock or discontinued, forcing the team to locate substitutes with matching specifications. This reinforces the need to standardize around well-supported, widely available parts for long-term production feasibility.

The 3D-printed enclosures also present scalability and consistency challenges. While 3D printing makes rapid prototyping easy, it introduces small dimensional variances that affect fit and finish between units. A manufacturable version would likely require injection molding to achieve the precision, strength, and surface quality needed for classroom use. Similarly, the magnets and pogo pins must be embedded or press fit accurately to maintain strong mechanical and electrical connections. These tolerances are achievable by hand for prototypes, but mass production would require dedicated precise tooling.

Durability and assembly time are additional considerations. Each block must withstand repeated snapping and detaching during normal classroom play. This requires choosing materials that balance toughness and weight while preventing internal strain on the pogo pins and connectors. The wiring and solder joints must also be robust enough to handle mechanical stress without detachment. From an assembly standpoint, aligning each

connector and securing components within such a small enclosure increases labor time, which would affect manufacturing cost if scaled.

To address these challenges, the design emphasizes modularity and standardization. Using consistent PCB footprints, connector placements, and enclosure dimensions across all block types streamlines assembly and reduces cost. While the current version of *Blocks o' Code (v3)* is not yet optimized for high-volume manufacturing, identifying these manufacturability constraints ensures that the design can evolve toward a scalable, reliable educational product. Recognizing the limitations of small-scale fabrication helps us plan for future improvements such as simplified assembly, alternative materials, and automated production techniques.

4.4 Other Constraints

4.4.1 Sustainability

Sustainability plays an important role in the design of *Blocks o' Code (v3)*, especially in terms of power consumption, component durability, and material selection. While the main focus of the project is creating an engaging and educational experience for young students, the design also considers how each decision affects long-term environmental impact and overall system longevity.

One major sustainability factor is energy efficiency. Each block contains an ESP32-WROOM microcontroller, LEDs, and a display, which are powered through magnetic pogo-pin connectors and inductive charging systems. Although the system operates at low voltage, optimizing how often LEDs refresh and how frequently the displays update can significantly reduce power draw. Implementing PWM brightness control and placing the ESP32 modules in deep-sleep mode during idle periods are key strategies for minimizing current consumption without affecting responsiveness.

Component lifespan and recyclability are also important considerations. *Blocks o' Code (v3)* uses widely available and well-documented components such as the ESP32-WROOM, which is known for its long-term manufacturer support. This makes replacement and maintenance easier, reducing waste and preventing full system disposal if a single block fails. The modular structure also allows individual blocks to be repaired, upgraded, or recycled rather than replaced entirely, which helps extend the product's usable life.

PCB manufacturing introduced additional sustainability challenges. Even though our system is medium-scale, PCB fabrication typically involves materials and processes that generate waste. If scaled for classroom production, sustainable practices such as using lead-free solder, smaller board footprints, and recyclable material would be prioritized to reduce the project's ecological footprint.

Digital sustainability was also part of the design process. Tools like ChatGPT were used to assist in research and documentation, but all design and sourcing decisions were verified using datasheets and manufacturing documentation to ensure reliability. This hybrid approach balances efficiency and accuracy while reducing unnecessary rework.

By emphasizing energy efficiency, modularity, and responsible material use, *Blocks o' Code (v3)* acknowledges the sustainability constraints that come with the classroom electronics while outlining clear, achievable steps to make future iterations even more environmentally conscious.

4.4.2 Political

Political factors come into play whether classroom technology is welcomed, funded, or blocked on the district level. Adoptions of technology decisions in public schools often reflect the priorities of the board, community, and current debates about technology in early education. As a result of this, Blocks O' Code (v3) must be positioned as a learning tool that supports existing curriculum goals such as literacy, numeracy, or problem-solving rather than a novelty. This means that certain features or messaging that could be interpreted as partisan, sensitive, or culturally polarizing must be taken into consideration.

Procurement policies are also another factor in politics. Districts may possibly prefer vendors that meet local "buy-local" goals, sustainability targets, or approved-vendor lists shaped by initiatives set by the board. To reduce this friction, our bill of materials favors widely available parts that are obtained from multiple sources. Our documentation also highlights classroom outcomes such as aligning with standards and measurable skills that board administrators and board members can support publicly. Clear, practical guidance for teachers on limiting screen time and tailoring activities to each age group helps address common concerns raised during board meetings.

Finally, supply-chain politics such as tariffs or export restrictions can impact the cost and availability of components. While these factors are out of our control, the design mitigates risk by selecting parts with second-source options and keeping PCB revisions modular to allow for substitutions without redesigning the entire system. Keeping this all in mind, this has helped keep our project neutral, adoption-friendly, and resilient to shifts in local policy or broader geopolitical conditions.

4.4.3 Ethical

Ethical considerations are an important constraint for this project. One aspect is ensuring that all parts and materials are sourced responsibly, avoiding suppliers with poor labor practices or environmentally harmful manufacturing processes. Wherever possible, components should be chosen with sustainability and social responsibility in mind.

The project should also promote inclusivity and accessibility. As the target users include young children and children with disabilities, all design choices should consider fairness, accessibility, and non-discrimination, ensuring that the product can be safely and effectively used by all intended users.

The project must also consider its impact on child development. Design choices should avoid negative influences, such as over-reliance on technology or exposure to inappropriate content, and should instead promote learning, creativity, and safe exploration.

Finally, intellectual property must be respected. All software, designs, and resources used should comply with copyright and licensing laws, and any third-party materials should be properly credited and legally used.

4.4.4 Health and Safety

Health and safety are critical constraints for this project, particularly because the end users are very young children, including those with disabilities. These users may not fully understand the risks associated with interacting with electronic devices and may lack the skills for self-preservation. As a result, all aspects of the project must be designed to minimize any potential for injury. This includes preventing access to sharp edges, pinch points, or small parts that could be a choking hazard, as well as ensuring that the device cannot generate excessive heat or other dangerous conditions.

Another important consideration is the safety of the components themselves during development and testing. All electronic and mechanical components must be operated within their specified limits to avoid failures such as explosions, overheating, or melting. Proper insulation, fusing, and protective circuitry must be included to safeguard both users and developers during testing and operation.

4.4.5 Scalability

Scalability is an important part of *Blocks o' Code (v3)* because the system is made to grow by adding more blocks and features over time. However, this also creates some limits. Each new block added to the system increases power use, communication traffic, and the amount of data the Brain block needs to handle. If too many blocks are connected, the system can slow down, lose messages, or drain the batteries faster.

The wireless connection and I^2C communication between blocks also have limits on how many devices can be active at once. To keep things stable, the system has to balance how much data is shared and how often updates happen. This means the project must stay small enough to work smoothly during classroom activities without needing expensive hardware or complicated software changes.

Another scalability limit comes from cost and production. Every block uses similar parts like batteries, PCBs, and connectors, so adding more units quickly increases material and assembly time. For now, the design focuses on a small set of core blocks that show the main functions while keeping the project affordable and easy to maintain.

Overall, scalability is limited by power, communication speed, and cost. Future versions can expand with better networking or shared power systems, but the current version focuses on being simple, reliable, and easy to use for small classroom sets.

Chapter 5 – Application of ChatGPT and Other Similar Platforms

5.1 Introduction

Throughout the design and implementation of *Blocks o' Code (v3)*, ChatGPT and similar AI platforms were used to aid in design, research, and documentation as supplemental tools. These tools provided support in brainstorming, technical explanation, and document outlining/drafting. Rather than replacing technical judgement, these tools were used to maximize efficiency as accessible research and planning assistants.

The motivation towards using AI tools for this particular project is mainly this project's wide interdisciplinary scope. This project touches many different engineering domains such as:

- Embedded Firmware and Real-Time Systems
- PCB Design
- Wireless Power and Data Transfer
- Full-Stack Software Development
- Mechanical Design

Given the full extent of this project's technical breadth, our team does not have the full expertise of skills for this project, and to help with the ramp up of all these different subsystems, we used conversational AI.

The following sections of this chapter evaluate in detail the benefits, limitations, and educational impacts of incorporating AI-based tools into a senior design environment. Specific examples from the *Blocks o' Code (v3)* workflow demonstrate how ChatGPT accelerated research, streamlined documentation, and improved learning outcomes when combined with traditional engineering methods and critical review.

5.2 Benefits of Using Generative AI

The integration of ChatGPT and similar AI systems into the Blocks of Code (v3) workflow provided several measurable advantages throughout design, testing, and documentation. These benefits were especially valuable given the project's wide technical scope and the team's limited time to master new tools. The most significant areas of benefit included time efficiency, expanded research scope, accessibility, and enhanced brainstorming and creativity.

5.2.1 Time Efficiency

The most immediate benefit was the dramatic reduction in time required to obtain preliminary information and draft technical material. Instead of searching multiple datasheets or documentation sources, the team could request concise explanations directly through ChatGPT and then verify accuracy afterward.

For example, while evaluating microcontroller options, ChatGPT summarized key specifications such as core type, operating frequency, memory, and wireless support into a comparison table within minutes. This accelerated the research phase and allowed the team to move more quickly into simulation and prototype development.

Similarly, when writing formal sections of the report, ChatGPT was used to generate initial templates for chapters such as Hardware Design and Software Design based on previous Senior Design reports. These templates established consistent section ordering and formatting, which minimized the time spent aligning writing styles across the team.

Ultimately, this time efficiency enables the group to make decisions quicker and spend more time on the hands-on implementation of the project. Doing research and documentation completely by hand would take much longer and would significantly slow down the design cycle, especially given the number of unfamiliar technologies involved in Blocks of Code (v3). Without AI assistance, the team would have to individually search through dozens of datasheets, reference manuals, and forum discussions to gather information. ChatGPT and similar AI platforms can find and gather this information in minutes. For instance, compiling specifications for multiple microcontroller families or comparing wireless communication protocols would typically require manually cross-referencing technical documentation from different manufacturers. By using ChatGPT to generate initial summaries and organize data into clear comparison tables, the team was able to bypass this repetitive groundwork and move directly into analysis, validation, and system integration.

5.2.2 Expanded Research Scope

Another key advantage was the ability to broaden the team's research scope without becoming overwhelmed or feeling fire hosed by the volume of information. An AI tool can summarize large amounts of technical documentation down to the small bits of details that the team needs to get potential solutions that might have otherwise been overlooked because of how massive technical documentation can be, making it difficult and taxing to pick through.

In the initial phases of design, before the team even knew which microcontroller families, sensors, or communication modules to investigate, ChatGPT provided an efficient starting point. Instead of beginning with a blank slate, the model could suggest a list of potential device categories, technologies, or part families relevant to the system's requirements and how we envisioned the project to look.

For example, when the team first started brainstorming hardware architectures, ChatGPT was used to explore what types of microcontrollers were best suited for modular, low-cost, and networked designs. Within a single conversation, it identified a range of viable candidates such as the ESP32, RP2040, and ATmega series, summarizing their architectures, wireless capabilities, and approximate cost per unit. This not only saved hours of initial searching but also exposed the team to new options—like the RISC-V-based ESP32-C3—that might not have been considered otherwise.

Beyond listing options, AI tools also helped check basic compatibility between components, such as confirming that logic voltages matched between the ESP32 and LED drivers or suggesting proper communication interfaces for displays. While every suggestion was later verified through datasheets, it gave the team an early sense of which parts could work together.

Overall, ChatGPT expanded the team's research scope by quickly connecting hardware, firmware, and power design concepts that would normally take days to identify. It helped turn the early "where do we start?" uncertainty into a focused set of actionable design paths.

5.2.3 Accessibility and Learning Support

One of the most valuable benefits was accessibility. ChatGPT functioned as an always-available tutor capable of explaining difficult concepts in simpler terms without judgment or delay. This was particularly useful for ramping up on unfamiliar software frameworks like Flutter and FreeRTOS or getting a refresher on hardware protocols such as SPI, I²C, and UART.

When the team encountered a confusing register configuration or communication error, ChatGPT could quickly provide step-by-step troubleshooting guidance or suggest potential causes. This reduced dependence on long forum threads or vendor technical support, making the learning process more direct and approachable.

Moreover, the conversational format allowed team members to learn iteratively by asking follow-up questions and refining understanding over multiple prompts, mirroring the experience of discussing problems with a teacher.

5.2.4 Brainstorming and Design Exploration

ChatGPT also served as a creative collaborator during the conceptual design stages. By generating alternative approaches to subsystem problems, it encouraged broader exploration before decisions were finalized.

For instance, when designing the communication architecture, the model helped outline trade-offs between a centralized "Brain" controller to connect to the companion app and distributed microcontrollers in each block. The AI's structured pros-and-cons analysis prompted the team to further investigate cost, latency, and synchronization requirements, leading to a more informed final design.

Similarly, during a lot of the creative planning, ChatGPT suggested potential ideas like block functionality, how a companion app could potentially supplement the programming blocks, and what context we could use for the blocks. While not all ideas were used, this brainstorming phase expanded the team's creative possibilities, and ChatGPT served as a useful sounding board for exploring ideas even without a physical collaborator present.

5.2.5 Writing and Communication Clarity

Beyond technical research, ChatGPT improved clarity in written communication. Draft paragraphs were refined for readability, grammar, and consistency with academic style. It also helped ensure terminology remained uniform across different chapters of the report.

This assistance was particularly helpful for ensuring that each section followed the same professional tone and structure, which simplified the integration of contributions from multiple team members.

5.3 Limitations and Challenges

While ChatGPT and similar AI tools greatly accelerated research and documentation, their use also introduced a number of limitations and challenges. These constraints primarily centered on technical accuracy, context retention, ethical use, and the inability of AI to validate physical or system-level results. Recognizing these limitations was essential to ensure that the project's final outcomes were grounded in verified engineering practice rather than over-reliance on generative outputs.

5.3.1 Lack of Physical Validation

A major limitation of generative AI tools is their inability to interact with or verify real-world hardware. ChatGPT cannot test circuits, measure current draw, or confirm that a PCB layout meets electrical and thermal constraints.

During *Blocks o' Code (v3)*, this was most evident when reviewing suggested pin configurations and power-supply topologies. The AI could outline general design principles but could not account for part-specific tolerances or noise coupling between analog and digital sections. As a result, all AI-generated hardware suggestions were treated as conceptual references only and were later validated through datasheets, simulation, or bench testing.

5.3.2 Inaccuracy and Fabricated Information

ChatGPT and other generative AI tools have the possibility of not always producing 100% correct and accurate information. It is possible that ChatGPT could recommend discontinued components or not fully understand if parts would be compatible. This limitation requires the team to cross-check every specification gained from AI with the primary sources like datasheets. Generative AI responses should only be used as a starting point, and must always be verified before using information for the actual project.

5.3.3 Limited Project Context Retention

Another challenge was the model's tendency to lose context over long or complex discussions. When conversations spanned multiple sessions, earlier design decisions were sometimes forgotten, leading to repeated or inconsistent recommendations. A way of mitigating this is updating ChatGPT with the current status and documentation of our project regularly, so that AI suggestions could be more accurate and integrated with the current project workflow.

5.3.4 Confirmation Bias in AI Responses

Another limitation observed while using ChatGPT is its tendency to agree with user assumptions or suggestions, effectively acting as a "yes man." In many cases, this behavior can be beneficial. By aligning with the user's input, the AI keeps responses focused on the team's perspective, allowing conversations to remain direct, concise, and centered on the user's intended direction. This reduces the risk of being distracted by tangential suggestions or overly speculative ideas, helping the team maintain clarity and efficiency in discussions.

However, this tendency can also present challenges. If a user is mistaken or holds an incorrect assumption, the AI is likely to reinforce that error rather than challenge it, potentially leading the team astray. Over-reliance on this agreement can also limit creative exploration, as the AI may not propose alternative approaches or solutions unless explicitly prompted. To mitigate this risk, team members needed to remain critical of AI responses, actively questioning outputs, and verifying suggestions through datasheets, simulations, or independent reasoning.

5.3.5 Over-Reliance

A subtler challenge was the risk of leaning too heavily on AI for decision-making. While ChatGPT is effective for exploring possibilities, excessive reliance could discourage deeper investigation or independent reasoning. To counter this, the team plans to make sure that: every AI-generated suggestion must be replicated, verified, or improved upon by a team member before being incorporated. This approach maintained a healthy balance between leveraging AI efficiency and exercising engineering judgment.

5.4 Case Studies

5.4.1 Case Study 1 - Using ChatGPT for Deep Research and Component Sourcing

When using ChatGPT for my research contributions, particularly during the early stages of the project, I noticed a recurring limitation that affected both writing efficiency and sourcing accuracy. While ChatGPT was helpful for structuring sections like *Communication Protocols* and *Displays*, many of the sources it generated were not actually accessible or verifiable. For example, when I asked for manufacturer application notes or datasheets, the links provided would often lead to placeholder pages or broken URLs. This made it difficult to directly access the materials I needed, forcing me to manually search on vendor websites such as Texas Instruments, Espressif, or Adafruit to confirm the information.

In some cases, ChatGPT would describe a legitimate datasheet or technical document correctly, but the link provided didn't match the actual source. The inconsistency slowed down my workflow since I had to spend extra time tracking official documentation to verify specifications. For example, while researching LED driver ICs, the model accurately summarized how PWM dimming worked, but the datasheet link it gave for a specific TI driver was invalid. The same happened when I requested SPI component recommendations. Several suggested parts were listed as "commonly available," but were out of stock or discontinued when checked through Digi-Key and Mouser.

These issues highlight one of the major limitations of relying on ChatGPT for detailed hardware research. The model is excellent at explaining what a part or concept does, but not as reliable for determining where or how to source it. Because ChatGPT doesn't have live access to distributor databases, it can't confirm real-time stock, pricing, or lead times. This creates a gap between conceptual understanding and practical design implementation.

To overcome this, I learned to use ChatGPT primarily for early-phase research, such as defining terminology and comparing protocols, and then validating all component data directly from official sources. This hybrid workflow was more efficient because it kept the clarity and structure of ChatGPT's summaries while grounding the technical details in real, verifiable data.

While ChatGPT was helpful for learning and organization, its limited source accessibility and lack of live component verification showed that it shouldn't be used as a standalone engineering research tool. Instead, it's best treated as a supplementary writing and explanation assistant.

5.4.2 Case Study 2 - Using ChatGPT to Decide the Audio Path (Amplifier/Speaker/Filters)

Early on in SD1, we knew as a group that we needed a clear, low-heat audio path that would fit a 50 mm enclosure and deliver .3 W into 8Ω for clear beeps, tones, and musical phrases. Originally the team was split between a simple analog/PWM path and a digital I^2S path, we also needed to match the speaker and determine whether analog filters were necessary.

To figure this out, we utilized Chat-GPT as a "first-pass researcher" and drafting aid. In a single thread we asked for "amplifiers, speakers, and filters to give me lists of them to research for my project" in a project folder tab, so it knew what we were researching specifically. ChatGPT returned a decent list and from there we scoured through datasheets and determined whether the ideas of ChatGPT would be doable.

With the AI's help we were able to determine two clean and complimentary paths:

- a low-cost analog/PWM path (PAM8302A) for most blocks.
- a digital I²S path (MAX98357A) for a music-focused block.

This split solved recurring debates about audio quality vs cost/complexity. ChatGPT also suggested setting the analog input high-pass near the 28 mm speaker's resonance and adding a small two-pole RC to smooth PWM. After verifying the math and the datasheet input recommendations, we fixed the networks at 480 Hz (AC-coupling) and 7.2 kHz (two-pole RC).

Because we used ChatGPT to our advantage, it saved us time and quality. After five days of researching and writing we went from scattered opinions to a documented audio architecture:

- PAM8302A and 28 mm 8 Ω speaker with defined RC filters for most blocks
- MAX98357A and the larger 36 mm 8 Ω driver for the music block.

The AI-drafted tables and section templates cut at least a week of manual collation and formatting. More importantly, the back-and-forth made us articulate acceptance criteria (child-safe loudness, 3.3 V rail, PETG fit, efficiency) and lock values we could build and test immediately.

Overall, ChatGPT is stellar at broadening options, structuring the possible trade-offs, and accelerating the decision process, but it had to be challenged in order to effectively get the accurate information needed. Treating ChatGPTs outputs as a starting point, then searching through datasheets and calculations, we were able to pick specific parts quicker rather than not utilizing it to its fullest potential. [67]

5.4.3 Case Study 3 - ChatGPT Confirmation Bias When Deciding Charging Interfaces

When deciding the charging interface for our project, my initial assumption was that we could use pogo pins for both charging and data transfer. This came from my limited understanding at the time of how pogo pins worked and how our low-power system operated. I didn't yet know about concepts such as maximum charge voltage, battery protection circuitry, or the challenges of designing stable contact points for multiple modules.

At one point, I asked ChatGPT whether it agreed that pogo pins were better than USB-C for charging our system. It immediately agreed with my statement, reinforcing my initial assumption without challenging it. At the time, ChatGPT already had access to most of our project details and constraints, yet it still validated my bias instead of questioning the feasibility.

If I had not gone back and performed proper research, this agreement could have been misleading. After learning more, I discovered that using pogo pins as our main charging interface would require a custom dock, precise alignment, and additional circuitry such as eFuses, reverse-FET protection, and temperature monitoring. Implementing all that would have effectively turned into another senior design project on its own, requiring an extra PCB and full documentation.

This experience taught me an important lesson: while ChatGPT is a useful tool for brainstorming and generating ideas quickly, it can also mirror a user's confirmation bias if the initial question is framed with assumptions. It's a reminder that engineers must always verify information through technical research and critical analysis instead of relying solely on AI validation.

5.5 Ethical and Academic Considerations

While ChatGPT and similar AI platforms offered significant support throughout the *Blocks o' Code (v3)* project, their use required careful attention to ethical and academic responsibilities to maintain integrity and professional standards.

5.5.1 Accuracy and Verification

AI-generated content must always be verified with reputable sources. Technical recommendations were cross-checked with the components' datasheets, reference manuals, and other technical documentation. This could relate to anything in the project, such as microcontroller specifications and capabilities, compatibility of peripherals and microcontrollers, and proper communication protocols. This process helps reduce the risk of accidentally using incorrect or outdated information and reinforces the importance of relying on AI too much in engineering design.

5.5.2 Academic Integrity

Proper acknowledgment of AI assistance in reporting and presentations was essential. The team documented the use of ChatGPT in the appendices of this report, ensuring that all contributions were accurately represented. This practice aligns with academic policies on the responsible use of external assistance and clarifies the distinction between AI-generated drafts and team-originated decisions.

5.5.3 Ethical Engineering Practice

AI recommendations were never treated as the final decision on safety-critical and performance-sensitive decisions. For example, guidance related to power management, wireless communication reliability, and PCB design tolerances was always evaluated and validated by the team. Ethical engineering requires accountability for design outcomes, and AI cannot assume responsibility for these decisions.

5.5.4 Long-term Learning

ChatGPT was used as a learning tool rather than a replacement for technical understanding. Iterative questioning, follow-up research, and independent verification allowed the team to deepen knowledge across embedded firmware, PCB design, wireless communication, and software development, while still benefiting from AI-assisted efficiency.

In summary, the responsible integration of AI into the *Blocks o' Code (v3)* project reinforced accuracy, integrity, ethical practice, and long-term learning, ensuring that AI served as a supportive tool rather than a replacement for human judgment.

Chapter 6 – Hardware Design

6.1 System Hardware Overview

6.1.1 Brain PCB

The Brain Block is the main controller of our project. It brings everything together by managing power, logic, and communication between all the connected blocks. It is also where most of the user interaction happens because when you touch the screen, see a display, or hear a sound that indicates starting of a program, it all comes from this board. The Brain Block was designed to be simple and reliable, combining all the main features into one small board that controls how the other blocks work.

As shown in *Figure 6.1*, the block diagram gives a simple overview of how each part of the Brain Block connects and communicates. It shows how power flows from the USB-C port into the TP5100 charging circuit and the 2S 18650 lithium ion battery pack, which provides energy to the rest of the board. Two regulators create the 3.3V and 5V rails that supply power to all the components, including the ESP32, display, and audio circuits. The diagram also highlights the signal connections between the microcontroller, the ILI9341 touch display, the WS2812B LEDs, and the UM66T melody IC with the amplifier. This layout helps visualize how all sections of the Brain Block work together to manage power, control, and user interaction.

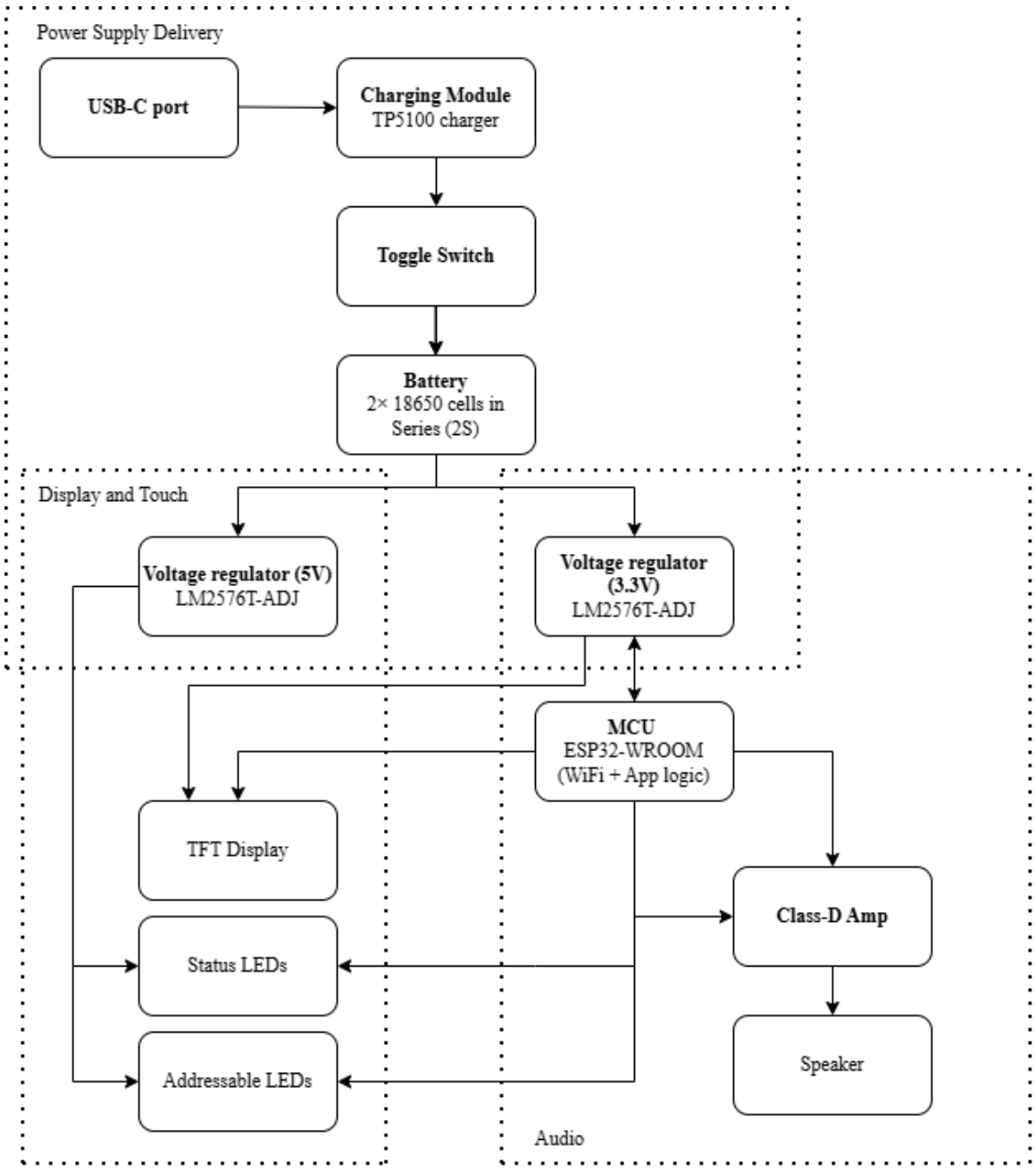


Figure 6.1 *Brain Block Hardware Diagram*

The schematic of the Brain Block, shown in *Figure 6.2*, presents a detailed view of how all the components are connected electrically. It shows the complete power path from the USB-C input and charger to the battery and both voltage regulators. The schematic also includes every supporting element, such as capacitors, resistors, and control lines, that help stabilize the board and keep power consistent. It organizes each subsystem clearly using dotted lines, showing where the ESP32 connects to the display, LEDs, and audio components. By following the schematic, it becomes easier to understand how the Brain PCB is wired internally and how power and signals are shared across the different circuits.

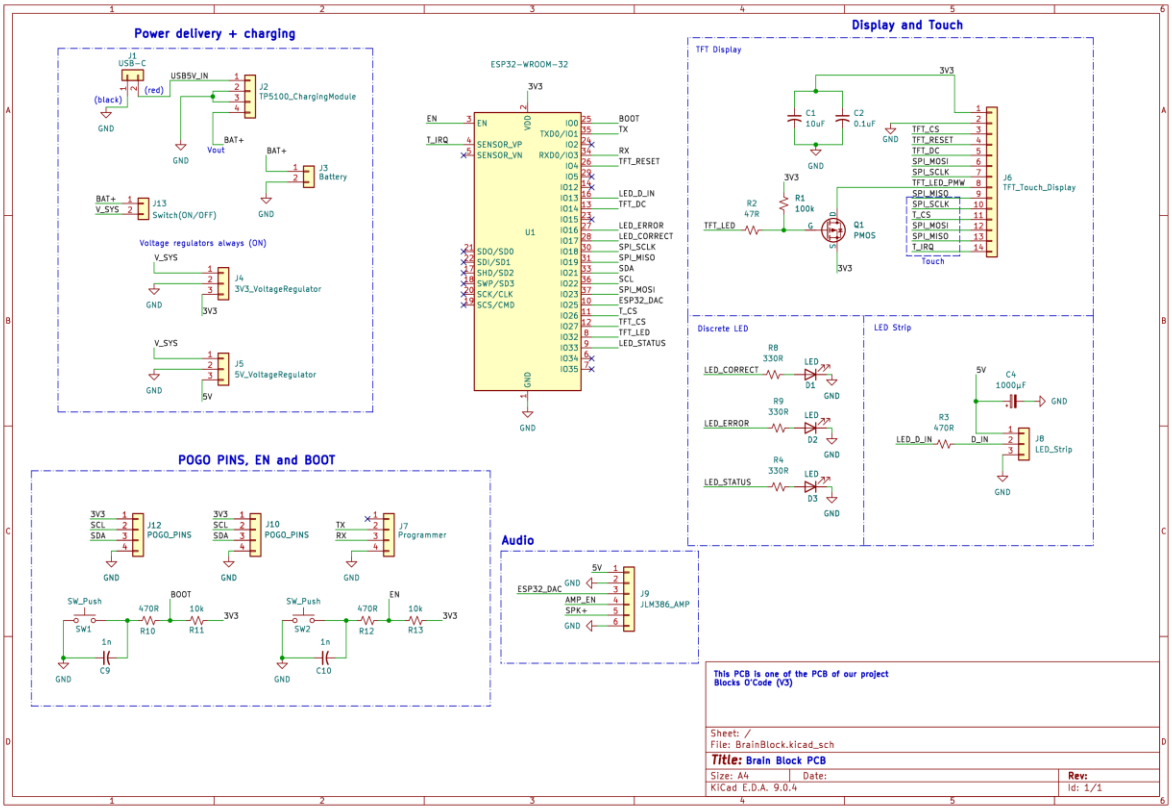


Figure 6.2 *Brain PCB Schematic*

The pinout chart of the Brain Block, shown in *Figure 6.3*, lists the functions of each GPIO pin used on the ESP32-WROOM-32 module. It identifies which pins control the display, which ones handle the LEDs, audio, and regulator enable lines, and which are reserved for communication or debugging. Having this chart keeps the design organized and makes it easier to connect everything correctly during assembly and programming. It also helps ensure that signals do not interfere with one another and so that the display, sound, and sensors can all work together without issues.

Signal Name	GPIO - MCU Pin	Net Name	Description
TFT LCD Display			
TFT_SCK (SCK)	GPIO 18 - Pin 30	SCK	Shared SPI clock (TFT + Touch)
TFT_MOSI (SDI)	GPIO 23 - Pin 37	SDI(MOSI)	Shared SPI data to peripherals
TFT_MISO (SDO)	GPIO 19 - Pin 31	SDO(MISO)	Shared SPI data from peripherals
TFT_CS	GPIO 27 - Pin 12	TFT_CS	Chip select for TFT on SPI bus
TFT_DC	GPIO 14 - Pin 13	TFT_DC	Data/Command control line
TFT_RESET	GPIO 33 - Pin 9	TFT_RESET	Hardware reset for display
TFT_LED (Backlight)	GPIO 32 - Pin 8	TFT_LED	PWM dimming control via low-side MOSFET
T_CS	GPIO 5 - Pin 29	T_CS	Chip select for XPT2046 touch IC
T_IRQ	GPIO 36 - Pin 4	T_IRQ	Interrupt (LOW when touched)
T_CLK	GPIO 18 - Pin 30	SCK	Shared SPI clock (TFT + Touch)
T_DIN	GPIO 23 - Pin 37	SDI(MOSI)	Shared SPI data to peripherals
T_DO	GPIO 19 - Pin 31	SDO(MISO)	Shared SPI data from peripherals
VCC		3V3	Power Source
GND	Pin 1	GND	Common system ground
LEDs			
Discrete LED (error)	GPIO 16 - Pin 27	LED_ERROR	Status LED
Discrete LED (correct)	GPIO 17 - Pin 28	LED_CORRECT	Status LED
5V		5V	Power Source
GND		GND	Common system ground
Addressable RGB			
5V		5V	Power Source
GND		GND	Common system ground
D_IN	GPIO 13 - Pin 16	GPIO13	LED Strip
Audio			
AMP_SD	GPIO 26 - Pin 11	GPIO26	Turns amplifier ON/OFF (shutdown control)
UM66_vdd	GPIO 25 - Pin 10	GPIO14	Turns UM66T power ON/OFF
Programming/Boot/EEPROM/POGO			
UO_TXD	GPIO 1 - PIN 35	TX	transmit data
UO_RXD	GPIO 3 - PIN 34	RX	receive data
EEPROM/POGO	GPIO 21 - Pin 33	SDA	bi-directional line that carries the actual data bits.
EEPROM/POGO	GPIO 22 - Pin 36	SCL	timing signal
EN	EN - Pin 3	EN	
BOOT	GPIO 0 - PIN 25	BOOT	

Figure 6.3 *GPIO and Net Reference Table*

Overall, the Brain Block was made to be the heart of the system. Its schematic and PCB layout show how the power and signal lines are neatly organized, and the pinout helps keep everything consistent between the hardware and the software. Together, these make the Brain Block a stable and easy to understand design that keeps the whole Blocks o' Code system running smoothly.

6.1.2 Child PCB

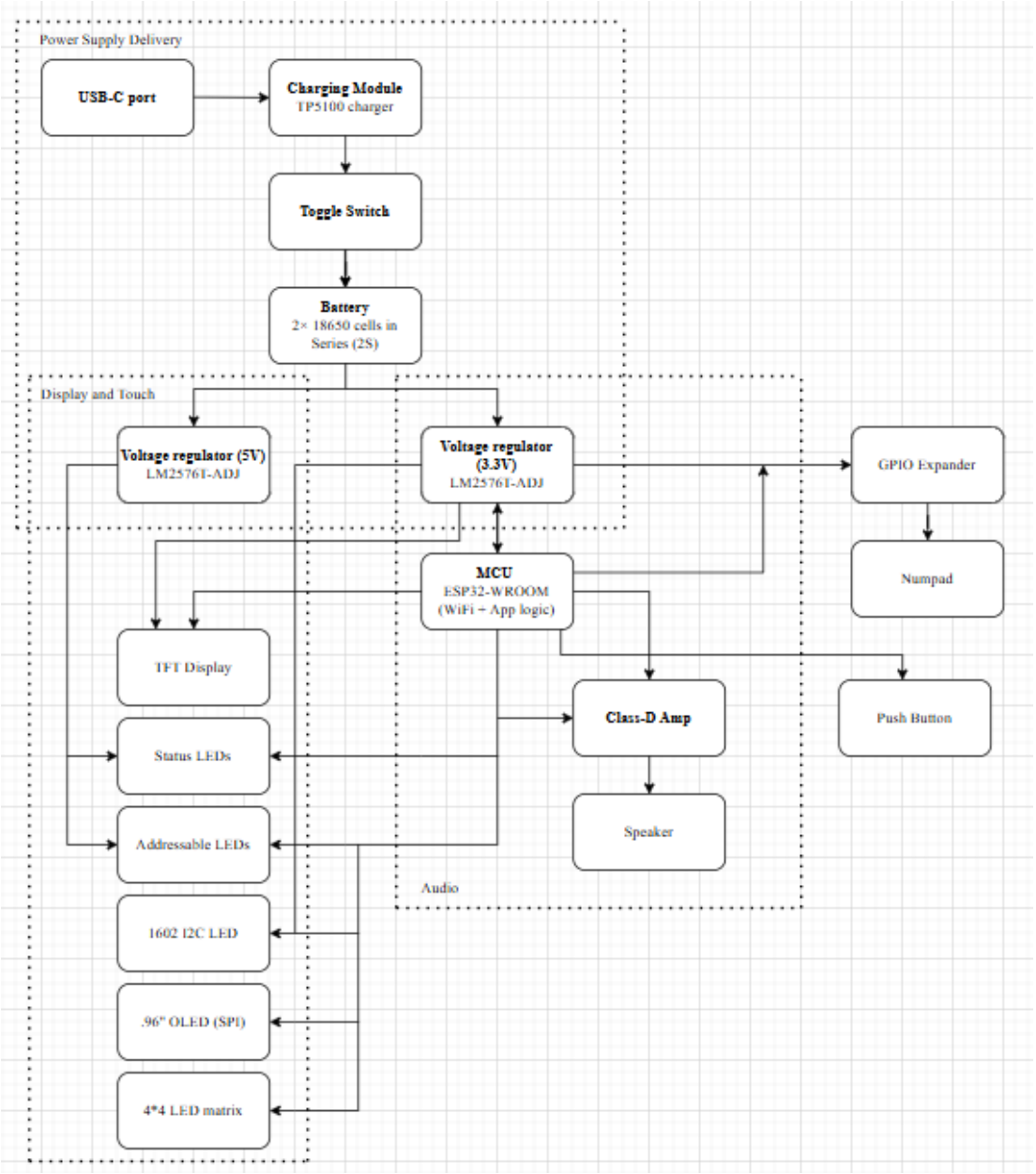


Figure 6.4 *Child Block Hardware Diagram*

The Child block centers on an ESP32-WROOM-32, which provides Wi-Fi/BLE and all primary I/O. The design uses two rails: 3.3 V for logic and sensors and 5 V for the TFT backlight and addressable LEDs. Local decoupling is placed at each connector and device to keep both rails stable. A MCP23017 I²C GPIO expander offloads the 4×3 numpad; address pins A0–A2 are strapped low (0x20), and the expander's interrupt is routed to the ESP32 so firmware can wake and scan on demand instead of polling. The SPI bus is shared by the main TFT display, and a 0.96″ SSD1306 OLED status panel; SCLK, MOSI, DC,

and RESET are common, and each display has its own chip-select to avoid contention. A 1602-character LCD connects on the I²C bus, reusing the board pull-ups and keeping the interface to four wires. Visual feedback is provided by two discrete LEDs ("correct"/ "error") and a 4×4 WS2812 LED matrix driven from a single data line; the matrix header includes a bulk capacitor and 330 Ω series data resistor for power and signal integrity, and a D_OUT pin for daisy-chaining. A single push button uses a 10 kΩ pull-up to 3.3 V for an active-low input that is easy to debounce in software. For audio, a 4-pin header exposes 5 V, GND, the ESP32 DAC output, and an optional AMP_EN line to a plug-in LM386 amplifier module used on both the Child and Brain blocks.

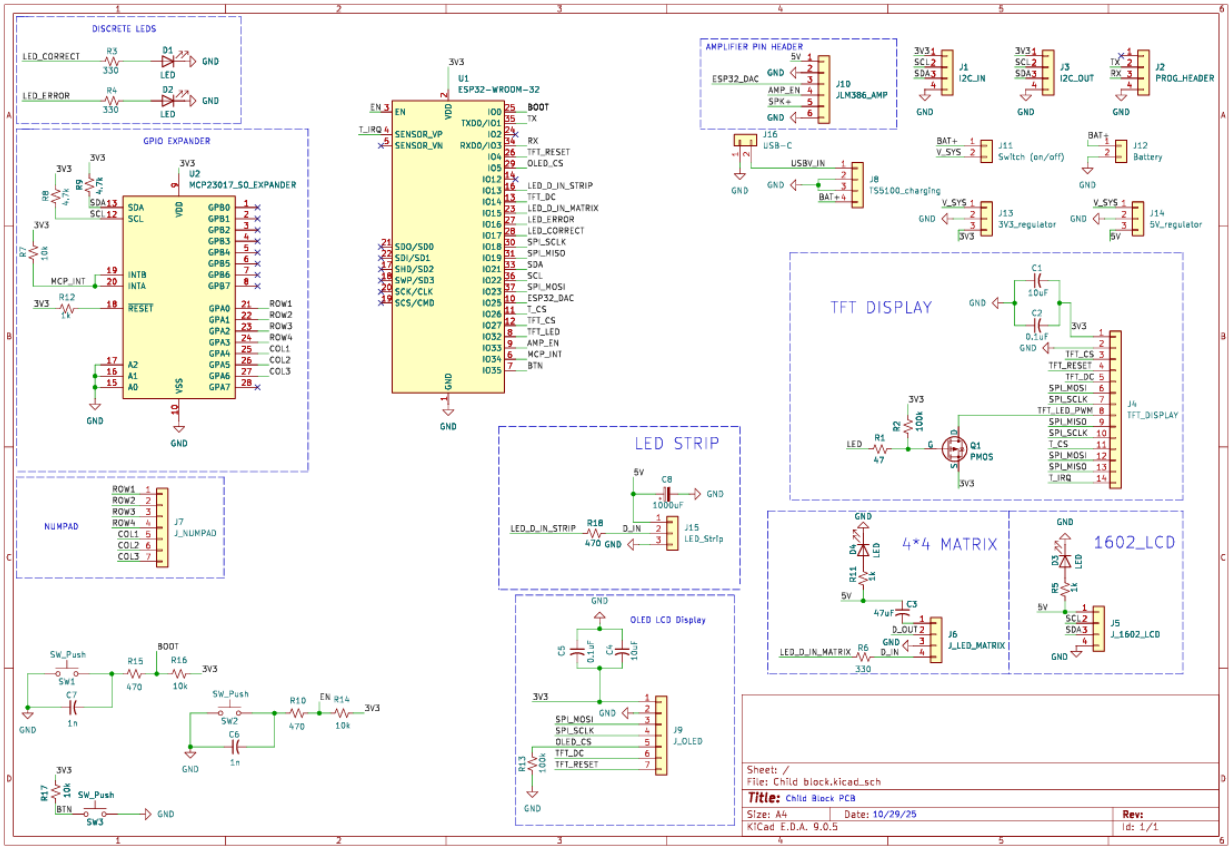


Figure 6.5 *Child Block Schematics*

The Child PCB meets its goals of simplicity and modularity while providing all required I/O for the activity flow. Shared buses reduce pin count and wiring; the LM386 module keeps audio serviceable, and conservative routing with clear labeling supports straightforward bring-up and classroom use. The design balances cost and robustness and leaves clean expansion points without re-spinning the core layout.

6.1.3 Senior Design 2 selection

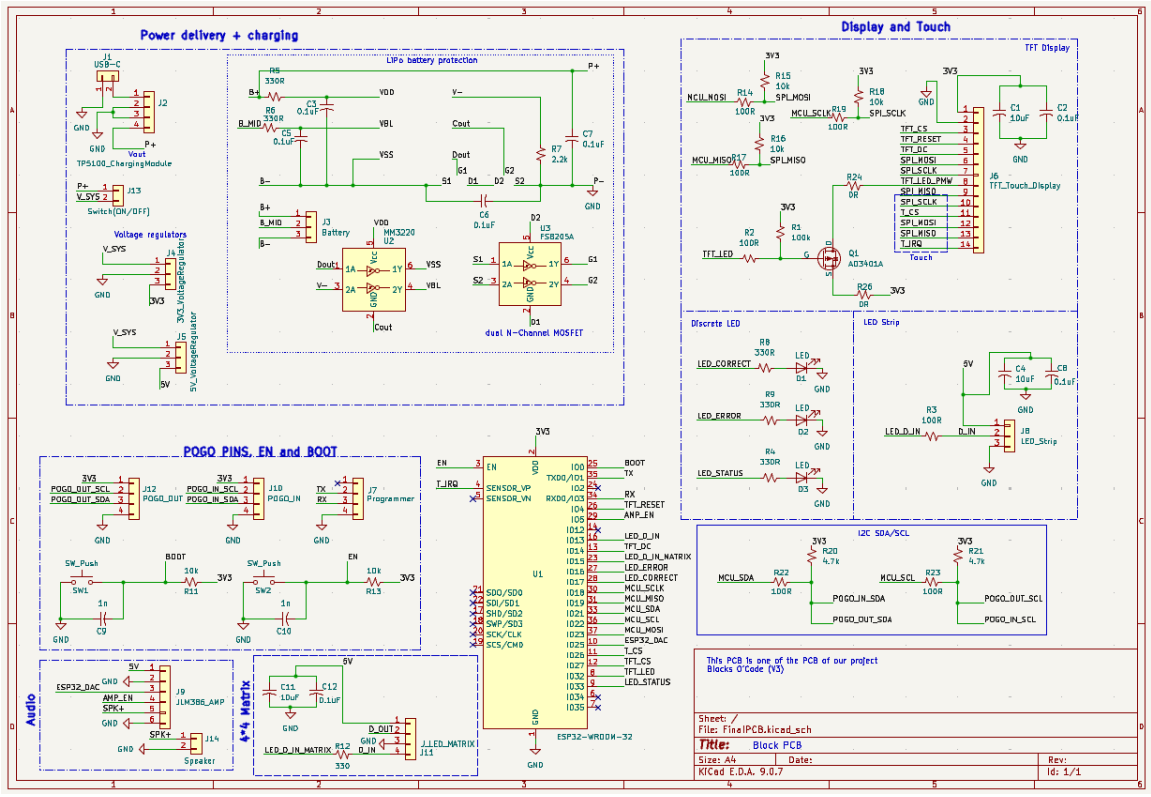


Figure 6.6 *Final Block Schematic*

The final block schematic shown in Figure 6.6 represents the completed modular PCB design used for both the Brain Block and Child Blocks. Instead of maintaining separate board designs, the hardware was consolidated into one flexible PCB platform that could support multiple block functions depending on the components installed and firmware loaded onto the ESP32-WROOM-32. This approach reduced the need to design and manufacture different boards for each block type while creating a more consistent hardware platform across the system.

The schematic includes the major subsystems required for operation, such as power delivery and charging circuitry, voltage regulation, controller connections, display and touch interfaces, audio support, programming headers, and peripheral expansion points. These sections were organized by function to improve readability, simplify debugging, and make future revisions easier.

Using one modular PCB also made assembly and testing more efficient, since the same base board could be reused for multiple blocks with only minor hardware or firmware changes. Overall, this design improved scalability, reduced manufacturing complexity, and created a cleaner and more adaptable final hardware solution for the project.

6.2 Subsystem Hardware Design

6.2.1 Power Delivery and Charging

The power delivery and charging section of the Blocks, shown in *Figure 6.6*, shows how power is handled throughout the system. It begins with the USB-C connector, which provides the main 5V input and ground. These connections are clearly labeled so it's easy to follow how power moves through the schematic. The 5V line goes into the TP5100 charging module, which is in charge of recharging the 2S 18650 lithium ion battery pack. The header for the charger includes simple pin labels like Vout, GND, and BAT+, which show how the module links to the rest of the circuit.

After the charging module, the BAT+ line connects to the battery header and then goes to a switch to turn on or off, when on BAT+ connects to V_SYS then splits into two voltage regulator modules that produce 3.3V and 5V. Each one has labeled pins for V_SYS, GND, and the output voltage, keeping everything neat and clear. The net names used in the schematic, like USB_IN, BAT+, V_SYS, 3V3, 5V, and GND, make it easy to trace where each connection goes without drawing extra wires. Overall, this part of the schematic gives a straightforward view of how the USB input, charger, battery, and regulators are all connected to keep the Brain PCB powered safely and consistently.

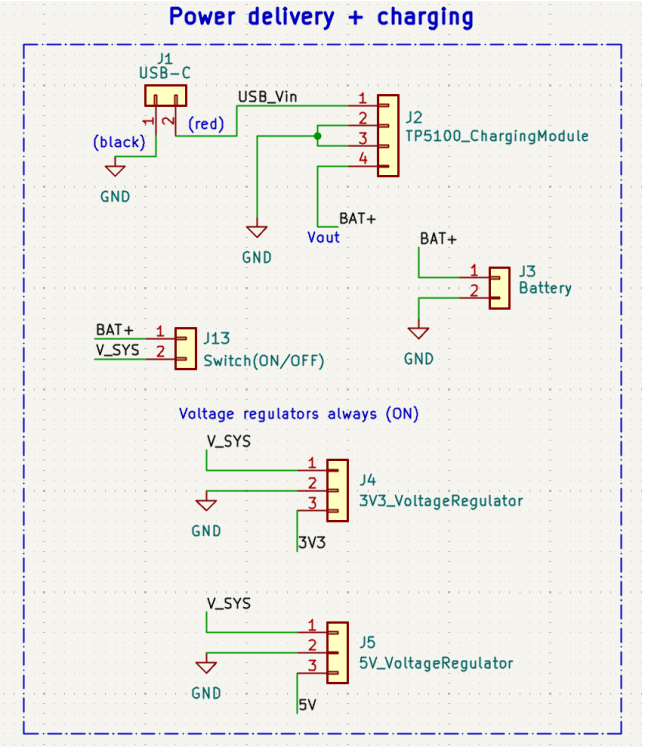


Figure 6.7 *Power Delivery and Charging*

6.2.2 Display and Touch

The TFT display section, shown in *Figure 6.8,* shows how the screen and touch input are connected to the ESP32. The TFT touch display uses the SPI interface, which means it shares the same lines for data transfer like SCK, MISO, and MOSI. Some extra pins are used just for the display, like RESET, DC, and CS, while the touch part uses its own pins for T_CS, T_CLK, T_DIN, T_DO, and T_IRQ. To control the screen brightness, a PMOS transistor is used to switch the LED backlight on and off from the 3V line. The ESP32 sends a PWM signal to this transistor's gate so the brightness can be adjusted in software. There's also a pair of resistors that make sure the transistor stays off when the signal isn't active. Finally, two decoupling capacitors (10 µF and 0.1 µF) are added close to the display to help keep the voltage steady and reduce any flickering. Overall, this section makes it possible for the Blocks to show images and detect touch smoothly.

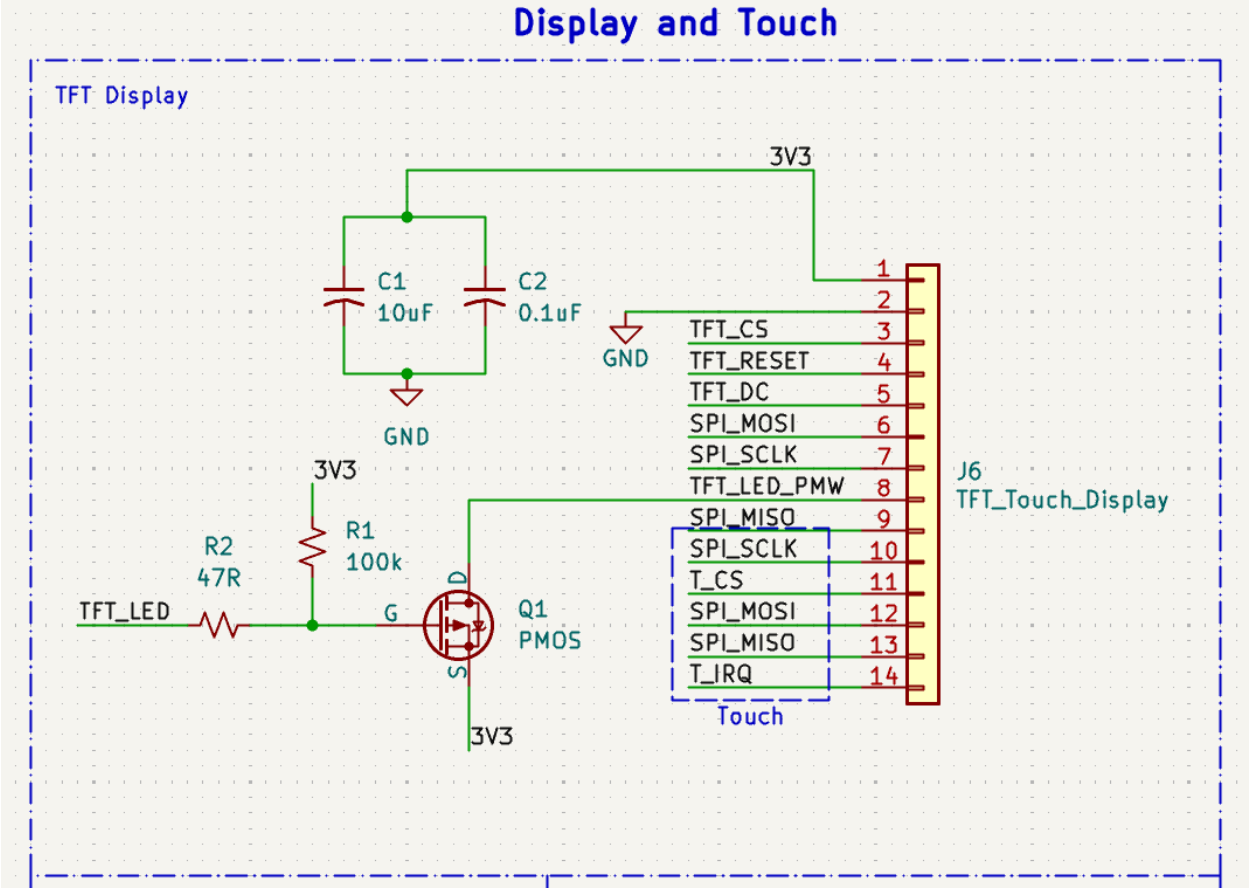


Figure 6.8 *TFT Display*

The LED section, shown in *Figure 6.8*, is used to control the colorful lights on the Blocks. It uses WS2812B addressable LEDs, which all connect in a chain and share a single data line controlled by GPIO13 on the ESP32. A 330 Ω resistor is placed on the data line to protect the first LED from voltage spikes, and a 5µF capacitor is added between 5V and ground to prevent sudden power drops when the LEDs turn on. The strip runs on 5V power, and even though the data line is 3.3V, it still works fine for short distances. The Discrete LEDs are for status, they will light on or be off depending on the block configuration, and

connectivity. This simple setup makes it easy to light up the Brain Block with different colors or effects, which can be used for feedback, interaction, or just fun visual designs.

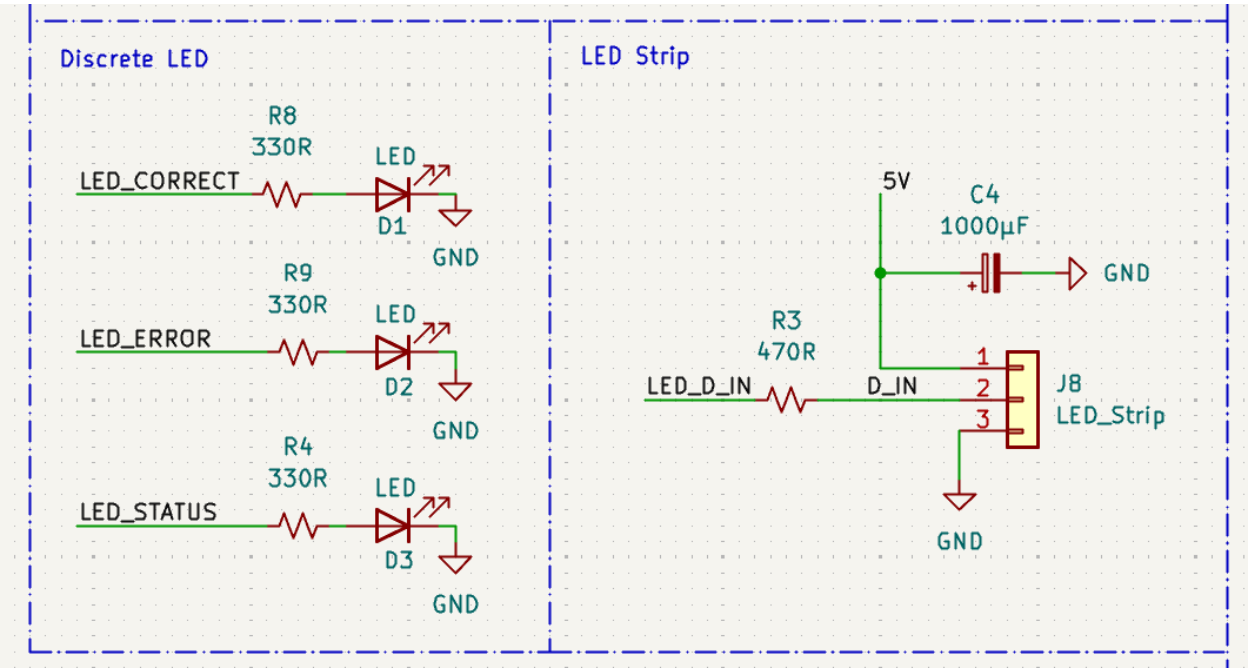


Figure 6.9 *LED Displays*

6.2.3 Audio

The audio section of the Brain Block and Child Block, shown in *Figure 6.9*, is designed to keep things simple while still adding a fun interactive touch for the child. The ESP32 sends its audio signal through GPIO25, which is labeled as ESP32_DAC in the schematic. This signal goes straight into a small JLM386 amplifier module through a header, so the amplifier can easily be swapped or replaced without changing the whole board. The amplifier is powered by the 5V rail and also has an enable pin, called AMP_EN, so the system can turn the sound on only when it is needed. The last two pins on the header, SPK+ and ground, go out to the speaker through a simple wire connection.

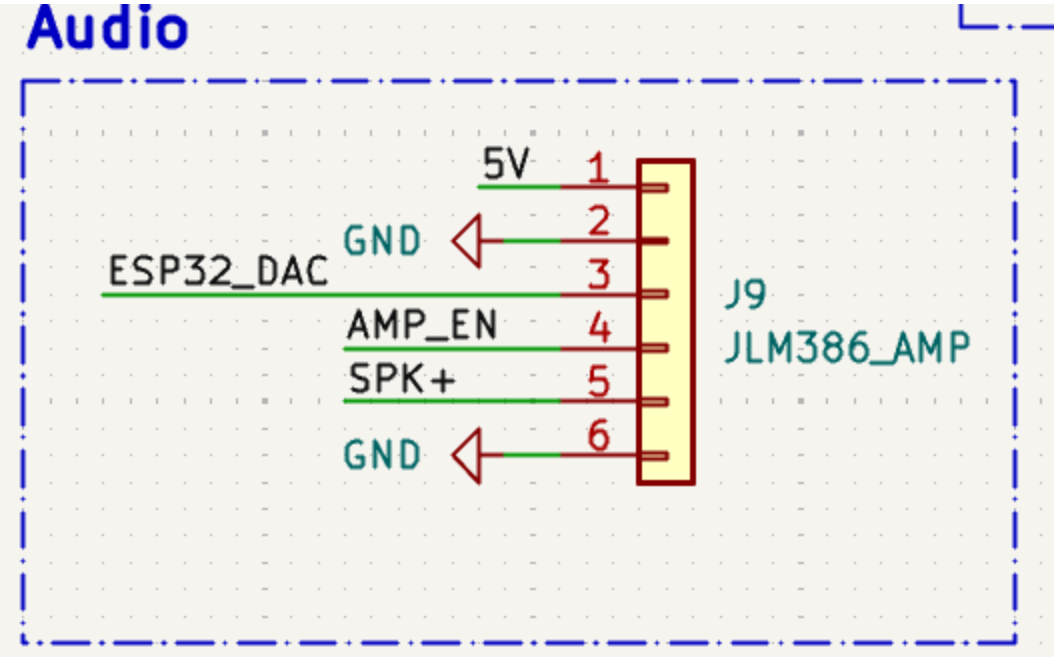


Figure 6.10 *Audio*

Overall, this section makes sure that every time the Brain Block starts running, it plays a short tone, so children know their program has started successfully, and it is the way we will be incorporating music and tone sound in the Child Block.

6.2.4 Programmer, POGO pins, EN and BOOT

The programmer section, also shown in *Figure 6.10*, is used to upload code directly to the ESP32 microcontroller. Instead of using a USB port on the board, the programming connection is made through another ESP32 development board. The TX and RX pins from the dev board are connected to the Brain Block's TX0 and RX0 pins, along with a shared ground. This makes it easy to upload new code to the bare ESP32 without adding extra hardware or components, keeping the design simple and easy to test.

The same figure also shows the pogo pins, which let the Brain Block share power and data with the Child Blocks when they are connected together. Each pogo pin header has 3.3V, ground, and I²C communication lines (SCL and SDA). These pins help the blocks talk to each other and pass information like saved values. This connection is small, simple, and easy to use, which makes it perfect for letting the blocks work together just by snapping them in place.

The same figure, the EN and BOOT buttons are there to help the ESP32 start up and go into programming mode. The EN one is kind of like a reset button when you press it, the ESP32 turns off for a moment and then starts again when you let go. The BOOT button is used when you want to upload new code. You normally keep it high so the ESP just runs your program, but if you hold BOOT down while pressing EN, it goes into a special mode

that lets you flash new code to it. It's just an easy way to control when the chip restarts or when you want to program it.

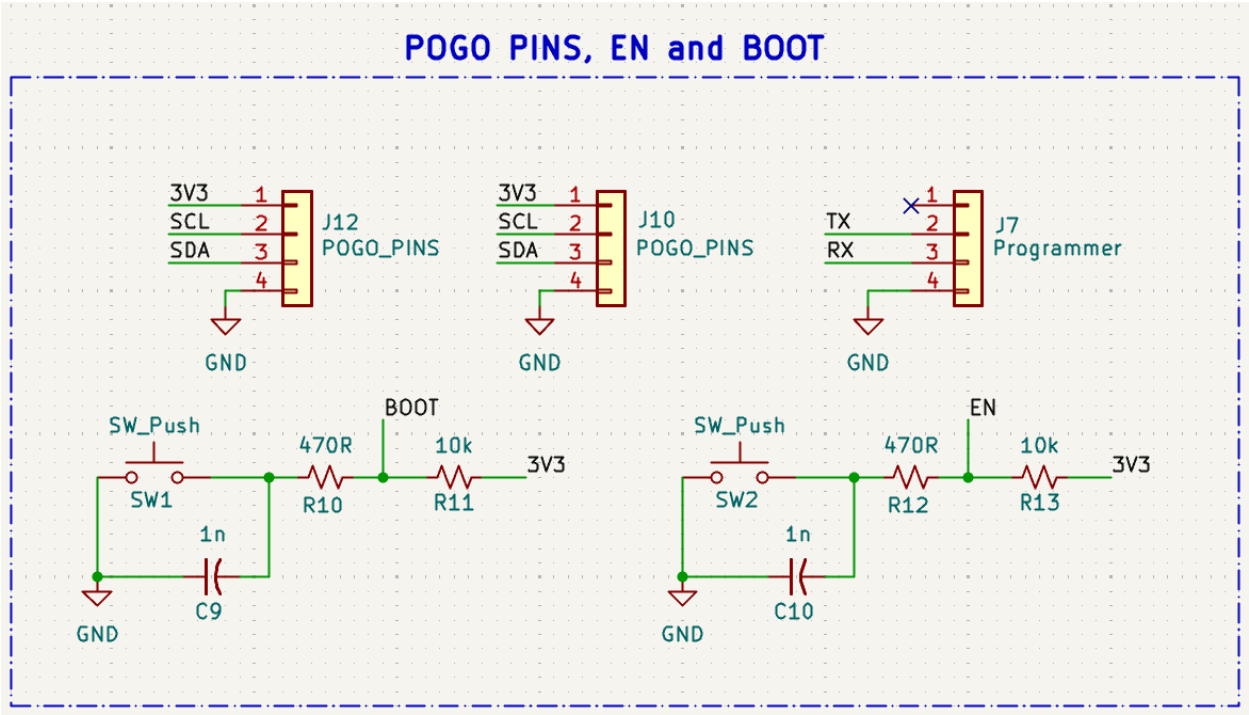


Figure 6.11 *Pins, BOOT and EN Schematics*

6.2.5 Amplifier

The audio path is a small plug-in LM386 board used on both the Child and Brain blocks. It mates to each host via a 1×4 header carrying 5V, ground, the ESP32 DAC signal, and an optional enable line. The LM386 runs at its default gain, receives the DAC through an AC-coupling capacitor with a 10 kΩ bias to ground, and uses local supply/bypass capacitors to keep noise low. Output is capacitively coupled to the 28 mm, 8 Ω speaker, with a standard series of RC Zobel network for stability. The speaker connects to SPK+ and GND terminals. This modular approach delivers clear, safe audio while letting the amplifier be assembled, tested, or swapped independently on either block.

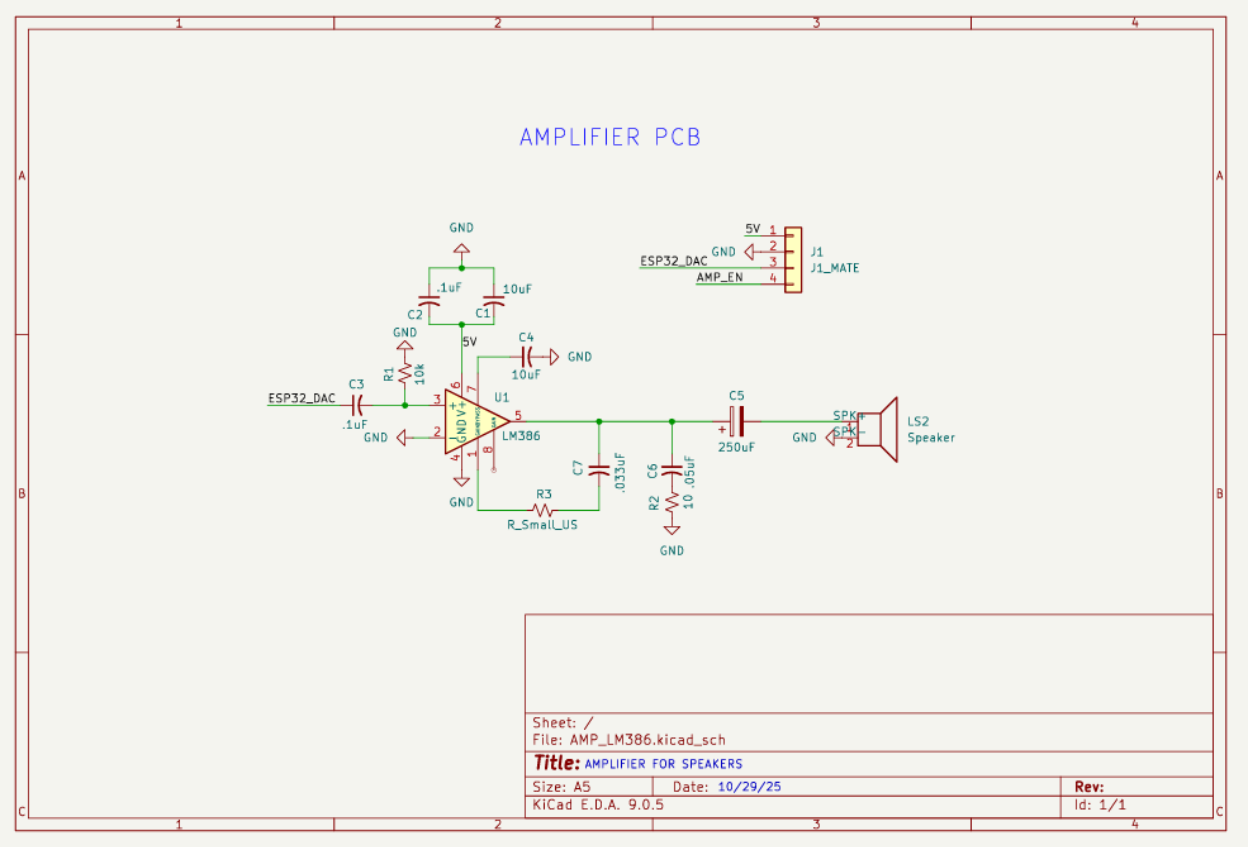


Figure 6.12 *Amplifier Schematic*

6.2.6 OLED LCD Display

The 0.96″ SSD1306 OLED is implemented as a Child-block status display. It rides on the existing SPI bus to avoid extra routing and pin usage, sharing SCLK, MOSI, DC, and RESET with the TFT while using its own chip-select (OLED_CS), so only one display is active at a time. The module is powered from 3.3 V with local decoupling at the header. In firmware, both chip-selects are held high during startup; the SPI bus is initialized, and then the OLED is brought up and refreshed periodically from cached state to present icons and short text without blocking other tasks.

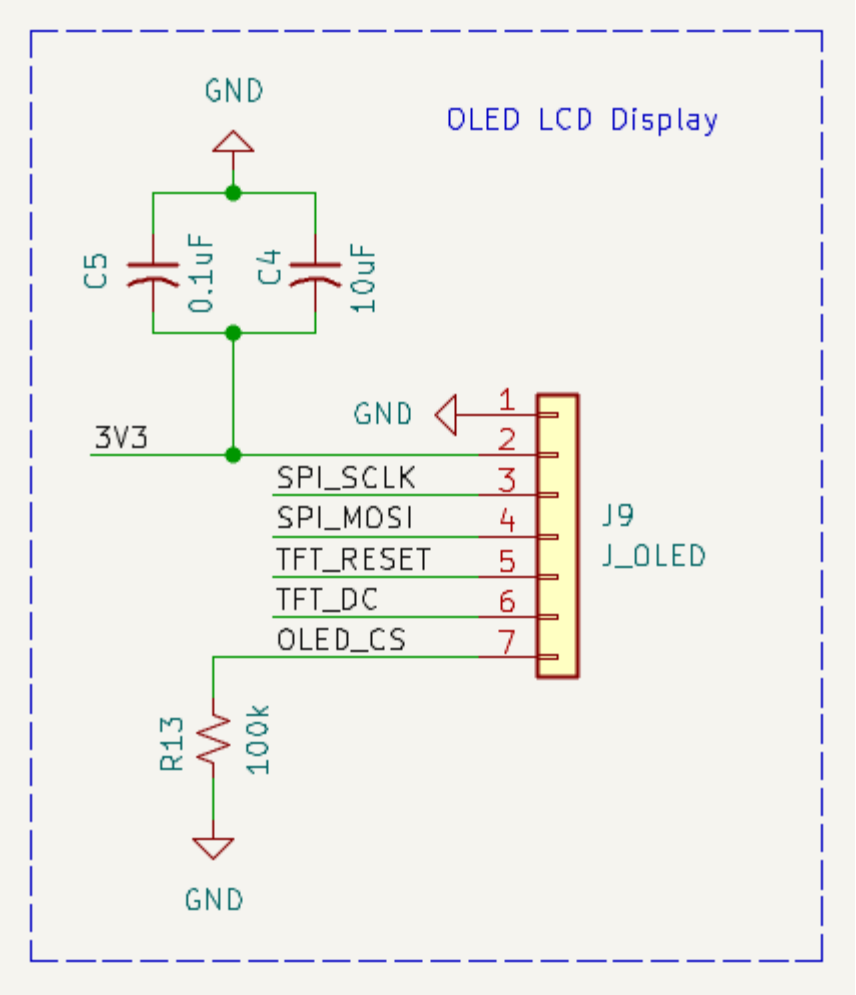


Figure 6.13 *OLED LCD Display Schematic*

6.2.7 Push Button

The push button provides a simple, reliable user input for the Child PCB using an active-low scheme. A 10 kΩ pull-up resistor biases the BTN net to 3.3 V, so the line reads logic HIGH when the switch is open; pressing SW1 connects BTN directly to ground, producing a clean LOW that the ESP32 can detect without ambiguity. This topology prevents the input from floating, minimizes parts, and avoids unnecessary current draw. The net is routed short and referenced to ground to reduce noise pickup, and the design leaves room for optional hardware to debounce (e.g., a small RC at the MCU pin) if field testing indicates it is needed. In firmware, the pin is configured as a digital input with interruptions on the falling edge, with 10–20 ms software debounce to filter switch chatter. The convention used throughout the codebase is released = HIGH, pressed = LOW, which keeps the logic consistent with other active-low controls in the system and simplifies test and bring-up procedures.

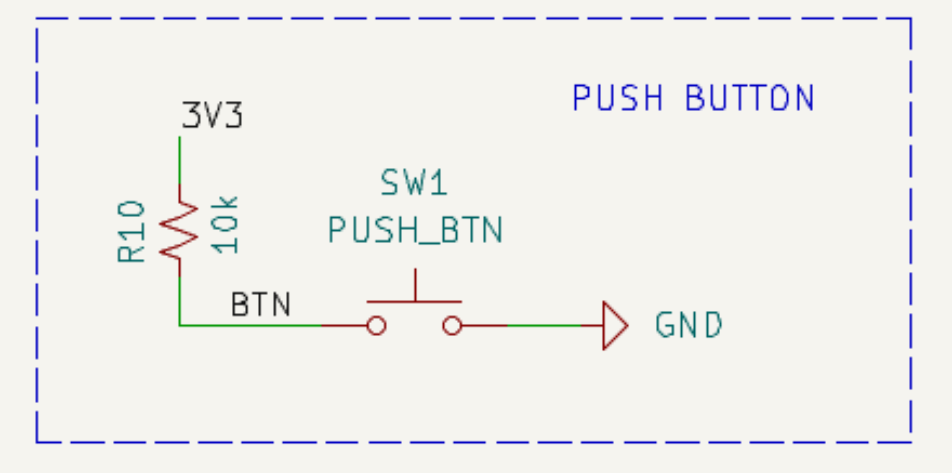


Figure 6.14 *Push Button Schematic*

6.2.8 Numpad

The numpad is a passive 4×3 matrix keypad and does not require power; it is wired only by its row and column nets. A seven-pin header (ROW1–ROW4, COL1–COL3) brings the matrix into the design, where the lines are assigned to the MCP23017 I²C expander. Columns are configured as inputs with the expander internal pull-ups enabled, and rows are driving low one at a time to scan. When a key is pressed, it connects the active row to a pulled-up column, producing a clean logic transition. The expander interrupt output is connected back to the MCU, so a change on any column can wake up the firmware to perform a quick scan; this avoids constant polling and saves power. All routing is short and kept as a bundle to reduce crosstalk, and no diodes are required because only one row is asserted at a time. In software, a simple state machine debounces each detection for 10–20 ms and maps the (row, column) pair to the desired digit or function, providing a reliable, low-overhead keypad interface for the Child block.

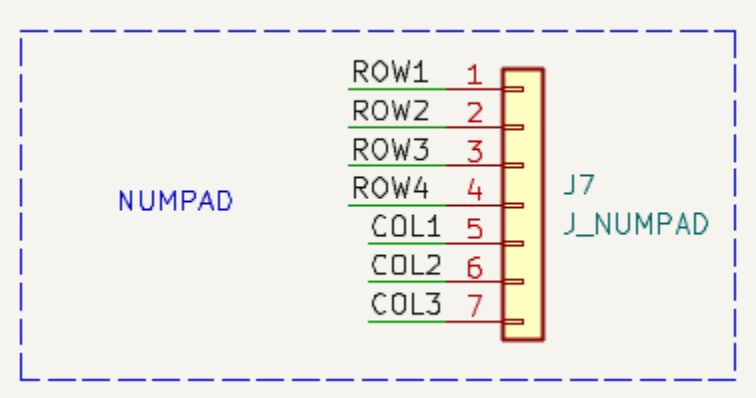


Figure 6.15 *Numpad Schematic*

6.2.9 GPIO Expander

A MCP23017 I²C GPIO expander is used on the Child block to handle the 4×3 numpad and free core ESP32 pins. The device is powered from 3.3 V with local pull-ups on SDA/SCL (4.7 kΩ each) to ensure a clean I²C bus. Address pins A0–A2 are strapped low,

selecting device address 0x20 for firmware. The RESET pin is held high with a pull-up, so the expander comes up enabled after power-on, and the open-drain INT output is pulled up to 3.3 V and routed to the MCU as MCP_INT for wake/scan events. The keypad matrix connects to the expander's GPA port: GPA0–GPA3 drive ROW1–ROW4, and GPA4–GPA6 read COL1–COL3. In software, the columns are configured as inputs with internal pull-ups; the firmware asserts one row low at a time and reads the columns to detect closures. Any change in a column generates an interruption, allowing the MCU to sleep between keypresses instead of polling. This arrangement keeps wiring simple, provides reliable key detection with light software debounce, and preserves ESP32 pins for other peripherals while maintaining a single, well-defined I²C address for future expansion.

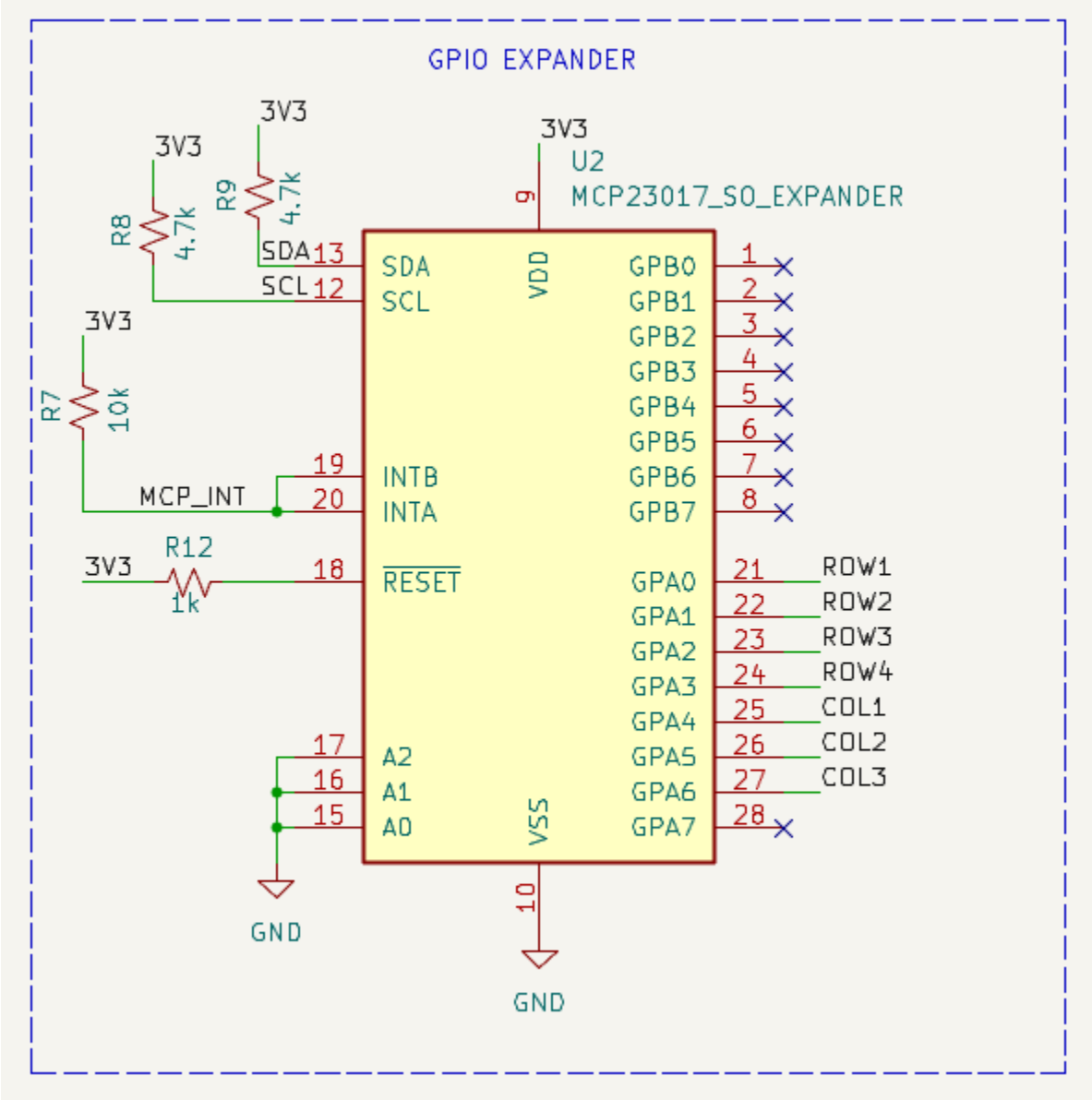


Figure 6.15 *GPIO Expander Schematic*

6.2.10 4X4 Matrix Display

The 4×4 LED matrix is an addressable WS2812-type module driven from a single data line. A four-pin header (5 V, D_OUT, GND, D_IN) brings power and signal to the panel and allows chaining to additional matrices via D_OUT. A 47 µF bulk capacitor is placed across the 5 V rail at the connector to absorb inrush and prevent brownouts when multiple LEDs change state, and a 330 Ω series resistor on D_IN limits ringing and protects the first LED. A small power indicator LED with a 1 kΩ resistor confirms that the 5 V rail is present. The matrix is powered at 5 V while the control signal originates from a 3.3 V ESP32 pin; the short, series-resisted trace typically works reliably, but if field tests show intermittent data at cold starts or long runs, a tiny 74AHCT1G125/74HCT125 level shifter can be added at the header to translate 3.3 V to a full 5 V logic high. In firmware, the LEDs are updated using the ESP32's RMT/Neo Pixel driver at 800 kHz. Timing-critical updates are performed in short bursts, and global brightness is limited to keeping current and temperature within safe limits. This arrangement provides a simple, robust lighting block that can be daisy-chained while keeping power integrity and signal quality under control.

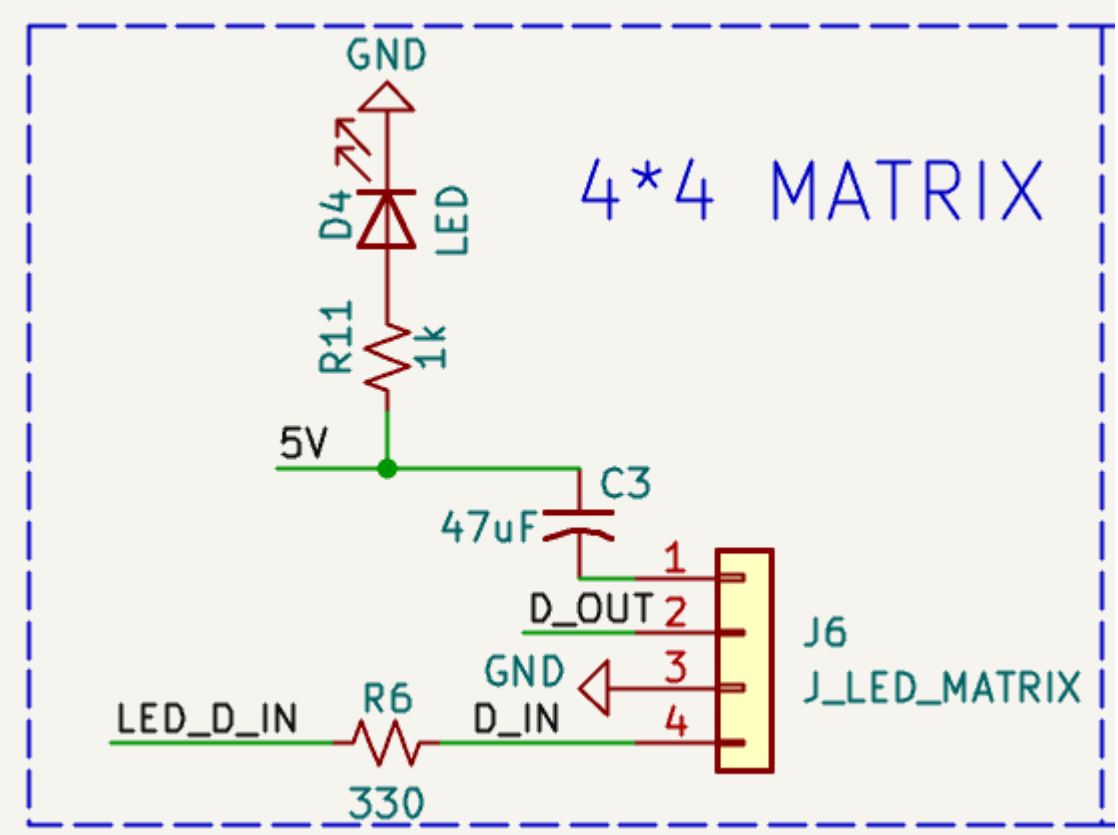


Figure 6.16 *4X4 Matrix Schematic*

6.2.11 1602 LCD Display

The 16×2-character display is the common "1602 + I²C backpack" module and connects through a 4-pin header carrying 5 V, SCL, SDA, and GND. The module is powered from the 5 V rail; a small power LED and 1 kΩ series resistor provide a visual power indicator

at the header. I²C signals share the board's main bus and are pulled up to 3.3 V on the Child PCB, so the open-drain lines remain 3.3 V-safe even though the display runs at 5 V. Typical backpack addresses are 0x27 or 0x3F (selectable by on-module jumpers); the firmware scans for the active address during initialization and configures the display for 16×2 characters. Updates consist of short text writes (status, prompts) and are rate-limited to avoid I²C bus contention with other peripherals. The arrangement keeps wiring simple—one 4-wire cable—and leverages the existing I²C infrastructure while delivering a readable auxiliary text display for the Child block.

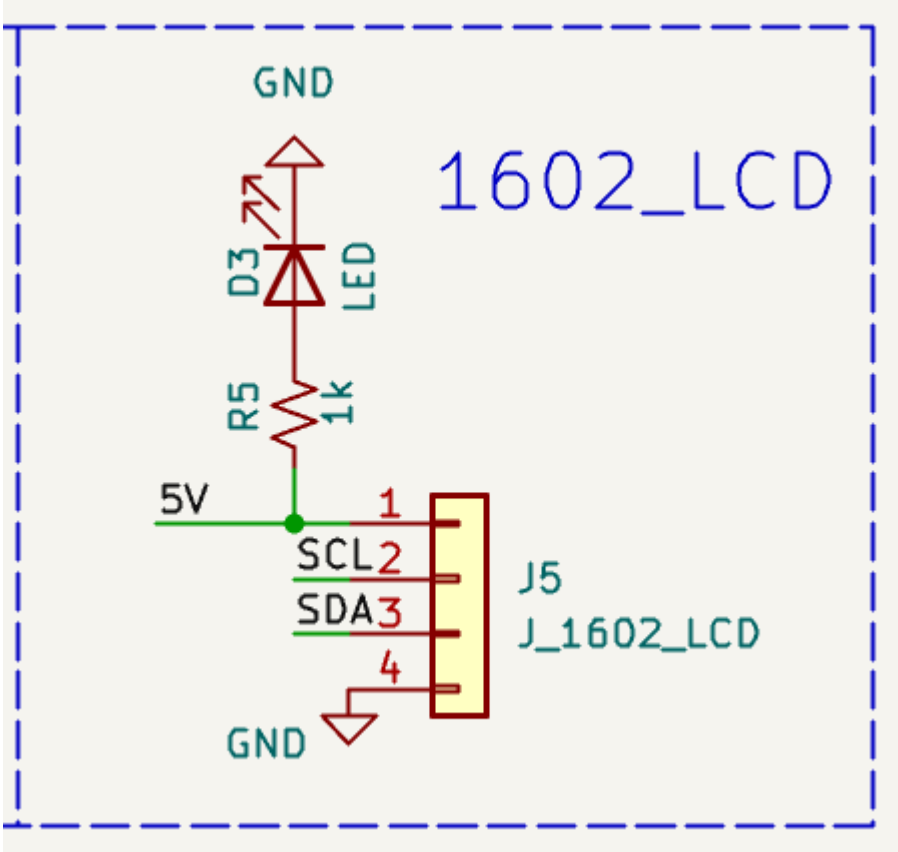


***Figure 6.17** 1602 LCD Display Schematic*

6.2.12 Amplifier Pin Header

The 16×2-character display is the common "1602 + I²C backpack" module and connects through a 4-pin header carrying 5 V, SCL, SDA, and GND. The module is powered from the 5 V rail; a small power LED and 1 kΩ series resistor provide a visual power indicator at the header. I²C signals share the board's main bus and are pulled up to 3.3 V on the Child PCB, so the open-drain lines remain 3.3 V-safe even though the display runs at 5 V. Typical backpack addresses are 0x27 or 0x3F (selectable by on-module jumpers); the firmware scans for the active address during initialization and configures the display for 16×2 characters. Updates consist of short text writes (status, prompts) and are rate-limited to avoid I²C bus contention with other peripherals. The arrangement keeps wiring simple: one

4-wire cable and leverages the existing I²C infrastructure while delivering a readable auxiliary text display for the Child block.

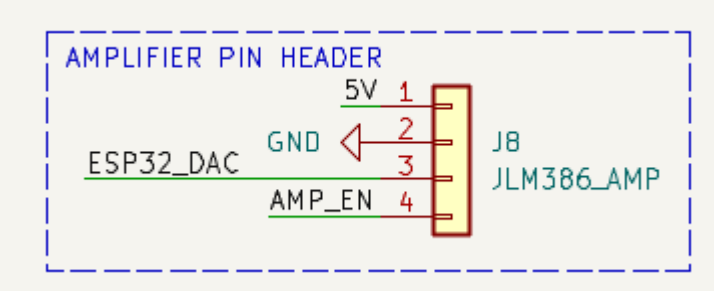


Figure 6.18 *Amplifier Pin Headed Schematics*

6.2.7 Power Distribution System

For both the Brain Block and the Child Block, power starts in the same way. It comes from the USB-C connector and goes into the TP5100 charging module. This charger takes care of safely recharging the two 18650 batteries that are connected in series. Once the batteries are full, they become the main power source for everything else on board. From the battery, the system needs two voltage rails, one at 3.3V and one at 5V, so that each component gets the right amount of power.

To start with the charging module, the TP5100 schematic was followed based and as it is specified in the datasheet. To make sure we can prevent future troubleshooting, the TP5100 CS and Vreg pins are connected with a 0R resistor, shown in *Figure 6.19*, this is in order to be able to decide whether to include 1S battery or 2S batteries depending on the necessities of our project in the future.

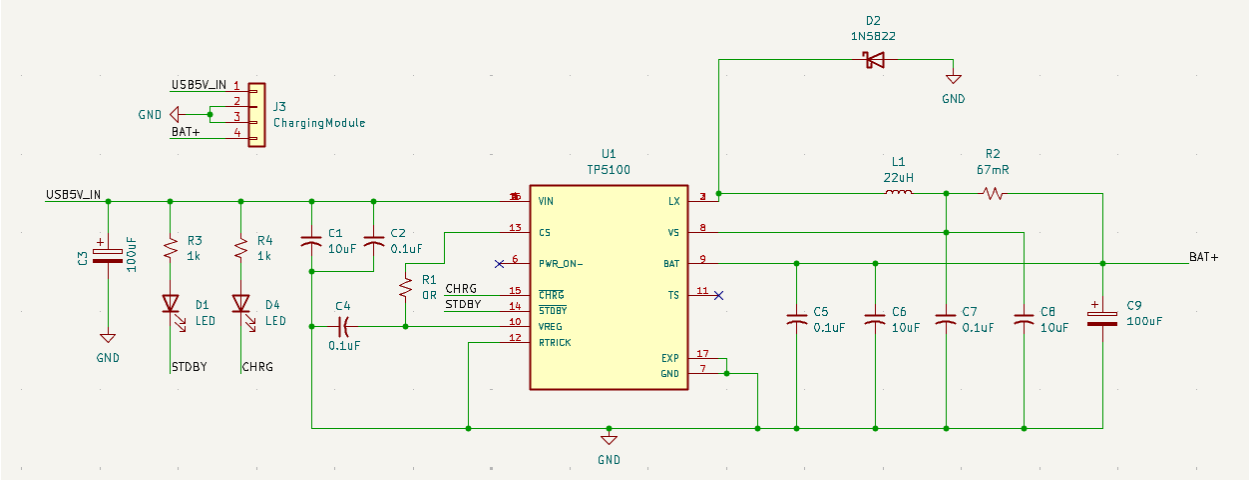


Figure 6.19 *Charging Module*

To make this easier, I decided to design a small voltage regulator board that could be used for both voltages instead of making two different ones. The regulator uses the LM2576T-

ADJ chip, which can step down the voltage depending on the battery level as long as it is equal or higher than 7V. Since the batteries drop from around 4.2 volts when fully charged down to about 3 volts, for one cell, when they start to run low, using a buck converter keeps the system stable. This way, even when the battery gets weaker, both the 3.3V and 5V lines stay steady, and the blocks keep working normally.

The schematic for this circuit is based on the recommended setup for the LM2576T-ADJ. It has an inductor, capacitors for filtering, and a few resistors that set the output voltage, I have also decided to add an LED to be able to troubleshoot easily and make sure the regulators are working, and to be able to be easily swappable between Child PCB and Brain Block PCB I decided to add a 4 pin header.

For the 3.3V regulator, shown in *Figure 6.20,* the main difference is the R1 Feedback resistor. This one being 1.7k.

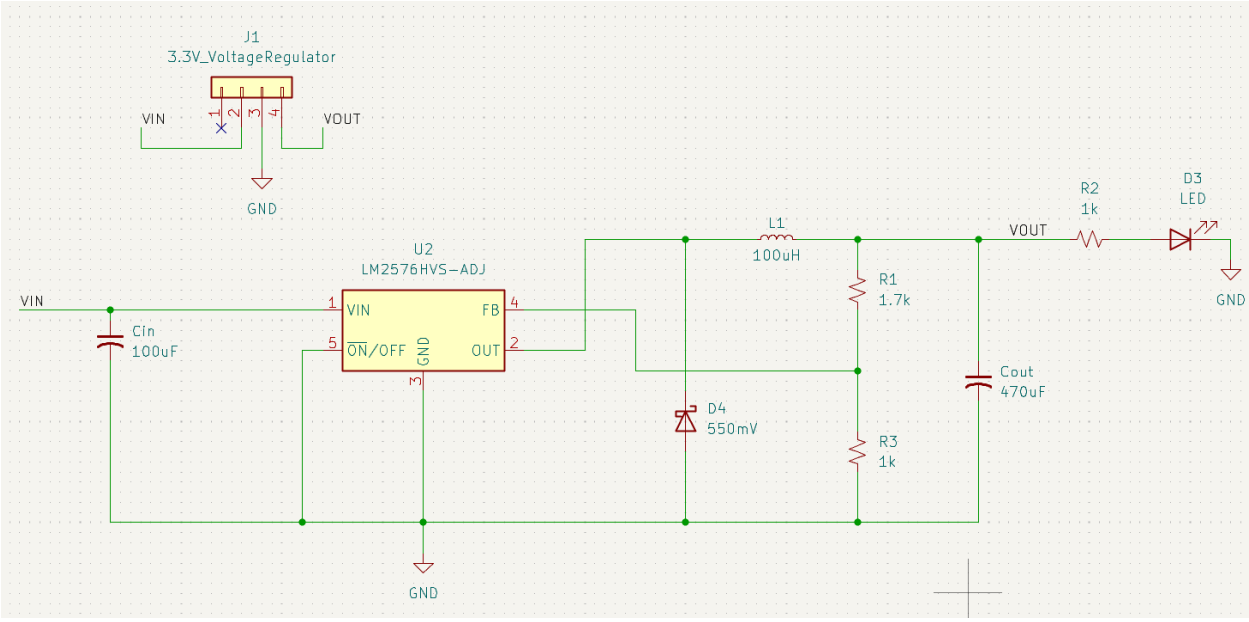


Figure 6.20 *3.3V Voltage Regulator*

For the 5V regulator, shown in *Figure 6.21,* the main difference is the R1 Feedback resistor. This one being 3k.

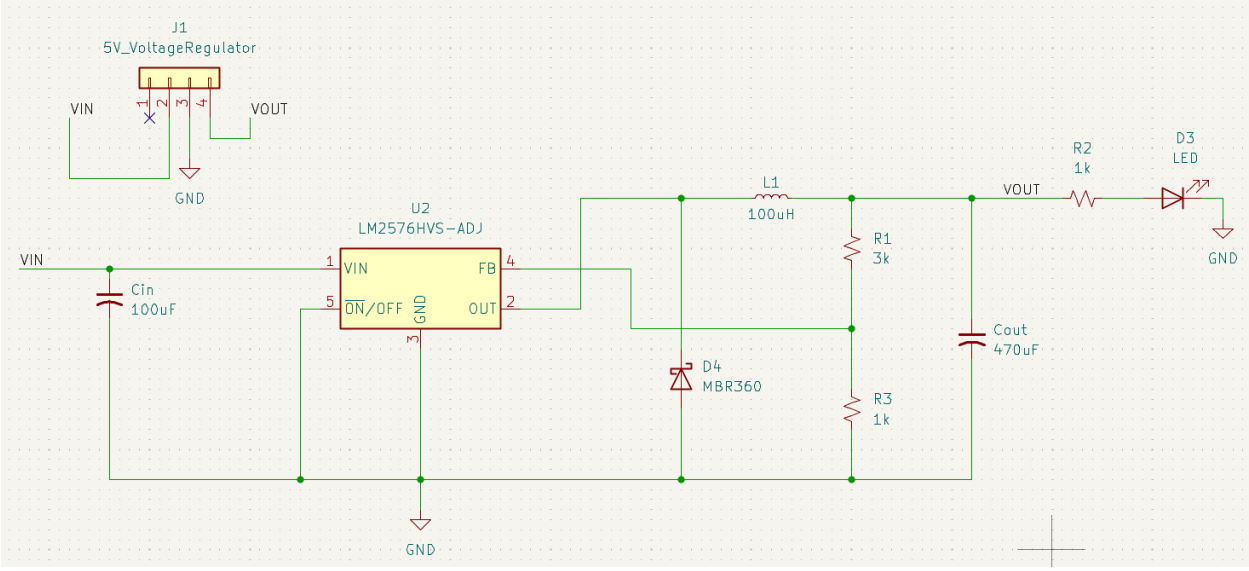


Figure 6.21 *5V Voltage Regulator*

This setup keeps things really simple. Both the Brain Block and Child Block use the same power path, USB-C to the charger module, to the battery, and then to the two regulator boards. It also makes the design easier to assemble, test, and replace if needed. Having just one regulator design that can do both voltages saves time and makes sure the power system works the same way across the whole project.

6.3 Power Distribution Table

When it comes to power delivery, our project *Blocks o' Code (v3)* faces a few limitations. Each block has its own setup, with different parts and functions, so the amount of power each one needs will not always be the same. Because of this, we had to look closely at how power is managed to make sure everything runs smoothly and safely.

Our main goal is to design two circuit boards, one for the **Brain Block** and one for the **Child Block**. These two boards use different components, which means they also use different amounts of power. Some blocks may have displays; others may have speakers, so their power needs will change depending on what they do.

In this section, we take a closer look at how power is distributed through each block. We include the voltage rails, regulators, and the typical peak current for each main component. This helps us understand how much power each block needs to work properly and how long it can run on its battery. The next tables show the results for both the Brain Block and the Child Block.

Brain Block

For our brain block, please refer to *Table 6.2*.

Category	Battery (3.7 V nom)	3.3 V Rail (TPS63060 #1)	5 V Rail (TPS63060 #2)

Regulator IC	N/A	Texas Instruments TPS63060 Buck-Boost	Texas Instruments TPS63060 Buck-Boost
Nominal Output Voltage	3.7V nom (2P 18650)	3.3V	5.0V
Main Loads / Components	Input to both rails	ESP32-WROOM-32, ILI9341 TFT logic + touch controller, EEPROM, UM66T, PAM8302A audio amp	TFT backlight, 3× WS2812B LEDs
Typical Current Draw	N/A	≈ 170 mA (ESP32 active + logic/ICs + PAM8302A idle few mA)	≈ 80–100 mA (BL ~50–60 mA + LEDs @ ~20% ≈30–40 mA)
Peak Current Draw	N/A	≈ 520 mA (ESP32 RF burst ~300 mA + audio burst from PAM8302A on 3.3 V ≈200 mA)	≈ 280 mA (3× WS2812B full-white ≈180 mA + BL up to ~100 mA)
Typical Power (mW)	N/A	≈ 560 mW (3.3 V × 0.17 A)	≈ 500 mW (5 V × 0.10 A)
Peak Power (mW)	N/A	≈ 1 720 mW (3.3 V × 0.52 A)	≈ 1 400 mW (5 V × 0.28 A)
Estimated Regulator Efficiency (%)	N/A	~ 92 %	~90 %
Approx. Power From Battery (mW)	N/A	≈ 610 mW (560/0.92)	≈ 555 mW (500/0.90)

Table 6.2 *Brain Block Power Distribution*

Child Block

For all of the configurations of our child block please refer to *Table 6.3*.

Category	**Battery (3.7 V nom)**	**3.3 V Rail (TPS63060 #1)**	**5 V Rail (TPS63060 #2)**
Regulator IC	N/A	Texas Instruments TPS63060 Buck-Boost	Texas Instruments TPS63060 Buck-Boost
Nominal Output Voltage	3.7 V nom (2P 18650 pack)	3.3V	5.0V
Main Loads / Components	Input to both rails	ESP32-WROOM-32 (Wi-Fi off), PAM8302A audio amp (3.3 V), SPH0645LM4H-B I²S MEMS mic	1602 I²C LCD module (logic + backlight)
Typical Current Draw	N/A	≈ 55–65 mA (ESP32 no-radio ~40–45 mA + PAM8302A idle ~2–4 mA	≈ 30–35 mA (LCD logic few mA + backlight ~25–30 mA)

		+ SPH0645 ~0.6 mA + misc)	
Peak Current Draw	N/A	≈ 200–250 mA (ESP32 CPU burst + short audio burst at 3.3 V)	≈ 120–130 mA (LCD backlight at max; module-dependent)
Typical Power (mW)	N/A	~ 3.3 V × 0.06 A ≈ 200 mW	~ 5 V × 0.033 A ≈ 165 mW
Peak Power (mW)	N/A	~ 3.3 V × 0.25 A ≈ 825 mW	~ 5 V × 0.13 A ≈ 650 mW
Estimated Regulator Efficiency (%)	N/A	~ 92 %	~90 %
Approx. Power From Battery (mW)	N/A	≈ 200 mW / 0.92 ≈ 217 mW	≈ 165 mW / 0.90 ≈ 183 mW

Table 6.3 *Child Block Power Distribution*

In addition to the components that receive regulated power directly, other components such as LED indicators have been powered via Pulse Width Modulation (PWM) this in order to reduce the power consumption.

6.3.1 Senior Design 2 Power Distribution System

Category	**Battery (7.4V nom, 2S 18650)**	**3.3V Rail (LM2576T-ADJ)**	**5V Rail (LM2576T-ADJ)**
Regulator IC	N/A	Texas Instruments LM2576T-ADJ Buck	Texas Instruments LM2576T-ADJ Buck
Nominal Output Voltage	7.4V nom (8.4V max)	3.3V	5.0V
Main Loads / Components	Input to both rails	ESP32-WROOM-32, 2.2" TFT controller, I²C bus / misc	30× WS2812B LEDs (status strip), 4×4 LED matrix (16 LEDs), LM386 amplifier + speaker, 2.2" TFT backlight
Typical Current Draw	N/A	110 mA (ESP32 ~100 mA + TFT ~8 mA + I²C ~2 mA)	705 mA (LEDs ~600 mA + matrix ~50 mA + amp ~30 mA + backlight ~25 mA)
Peak Current Draw	N/A	275 mA (ESP32 Wi-Fi burst ~260 mA + misc)	2,100 mA (LEDs full white ~1,800 mA + matrix ~160 mA + amp ~100 mA + backlight ~40 mA)
Peak Power (mW)	N/A	908 mW (3.3V × 0.275 A)	10,500 mW (5V × 2.10 A)
Estimated Regulator Efficiency	N/A	~82%	~82%

Approx. Power From Battery (mW)	N/A	442 mW (363 / 0.82)	4,299 mW (3,525 / 0.82)

Table 6.3.1 *Block Power Distribution (Brain and Child, per block)*

Scenario	3.3V Load	5V Load	Total Load Power	Battery Current (at 7.4V)	Estimated Runtime
Idle / Standby	40 mA	55 mA	0.41 W	68 mA	~38 hrs
Typical Demo	110 mA	705 mA	3.89 W	641 mA	~4.1 hrs
Peak / Worst Case	275 mA	2,100 mA	11.4 W	1,880 mA	~1.4 hrs

Table 6.3.2 *System Summary*

The WS2812B status strip dominates the 5V rail at full white (9W peak). LED brightness is capped at 10% in firmware, yielding the ~4 hour typical demo runtime. A 470 µF bulk capacitor near the ESP32 VCC absorbs Wi-Fi burst current, and a heatsink on the 5V LM2576T-ADJ manages the 1.85W regulator dissipation. The TP5100 charging IC provides over-discharge protection with a low-voltage cutoff at 6.0V.

Chapter 7 – Software Design

7.1 System Software Overview

The *Blocks o' Code (v3)* system software is designed around a unified microcontroller platform, where both the Brain and Child PCBs are powered by the ESP32-WROOM-32 module. This consistency simplifies firmware development, communication protocols, and system maintenance. The software coordinates real-time control, wireless communication, and user interaction through modular and scalable design principles.

We will be organizing the software architecture into 3 layers:

1. **Firmware Layer:** Runs directly on the ESP32-WROOM-32 to manage peripherals, device logic, and control tasks.
2. **Communication Layer:** Handles data exchange between the Brain and Child PCBs, as well as the companion app, via UART, I2C, or Wi-Fi.
3. **Application Layer:** Implements the Companion App interface for user configuration, visualization, and firmware management.

Functionally, the **Brain PCB** serves as the system coordinator, managing display output, synchronization, and communication with connected Child boards. Each **Child PCB** performs local operations such as LED control, audio output, or sensor data collection, and transmits updates to the Brain. The **Companion App** provides an accessible wireless interface for the user to monitor, configure, and visualize the system.

7.1.1 Firmware Overview

The firmware for *Blocks o' Code (v3)* is built on a unified architecture, with both the Brain and Child PCBs utilizing the ESP32-WROOM-32 module. This standardization simplifies the development process, allowing for a consistent codebase and streamlined communication protocols across the entire system. The firmware is organized into two primary components: the Brain firmware and the Child firmware, both developed within the ESP-IDF framework to leverage its real-time operating system (RTOS) and extensive driver support.

The **Brain block's firmware** acts as the central coordinator. Upon initialization, it scans the shared I²C bus to build a registry of all connected Child blocks. It then enters an idle state, waiting for interrupts from Child blocks or commands from the companion app. Its primary responsibilities include processing logic flow, managing synchronization between blocks, and relaying status updates to the companion app via Wi-Fi and JSON interface.

The **Child block's firmware** is responsible for local operations. Each Child block initializes its specific peripherals, such as LEDs, sensors, or speakers, and configures itself as a slave on the I²C bus. It continuously checks its peripherals for events (e.g., a button press) and asserts an interrupt to notify the Brain when an action occurs. It also responds to commands from the Brain to execute actions, such as playing a tone or displaying a light pattern. High-speed peripheral updates, like refreshing an LED matrix, are handled locally using the SPI protocol to avoid congesting the main I²C bus.

7.1.2 Companion App Overview

The companion application serves as the primary user interface for visualizing and interacting with the physical block configurations in real time. Developed using the **Flutter framework**, the app offers cross-platform compatibility and a high-performance, responsive user interface from a single codebase.

The app's core function is to mirror the physical arrangement of the blocks on-screen. It establishes a Wi-Fi connection with the Brain block and communicates using JSON data packets. As blocks are physically connected, rearranged, or removed, the Brain block sends status updates to the app, which instantly refreshes its visual display to reflect the new configuration.

Upon launching, the app initializes its UI and attempts to connect to the Brain block. Once connected, it enters an idle state, listening for data. When new data arrives, the app parses the JSON and updates the display. A key feature is the "disco visualization," which generates animated lights and effects that mirror the outputs of the physical blocks once

the program is executed. The app also allows for user input, which are then sent back to the Brain block as JSON objects.

7.2 Subsystem and Components

7.2.1 Software Block Diagram

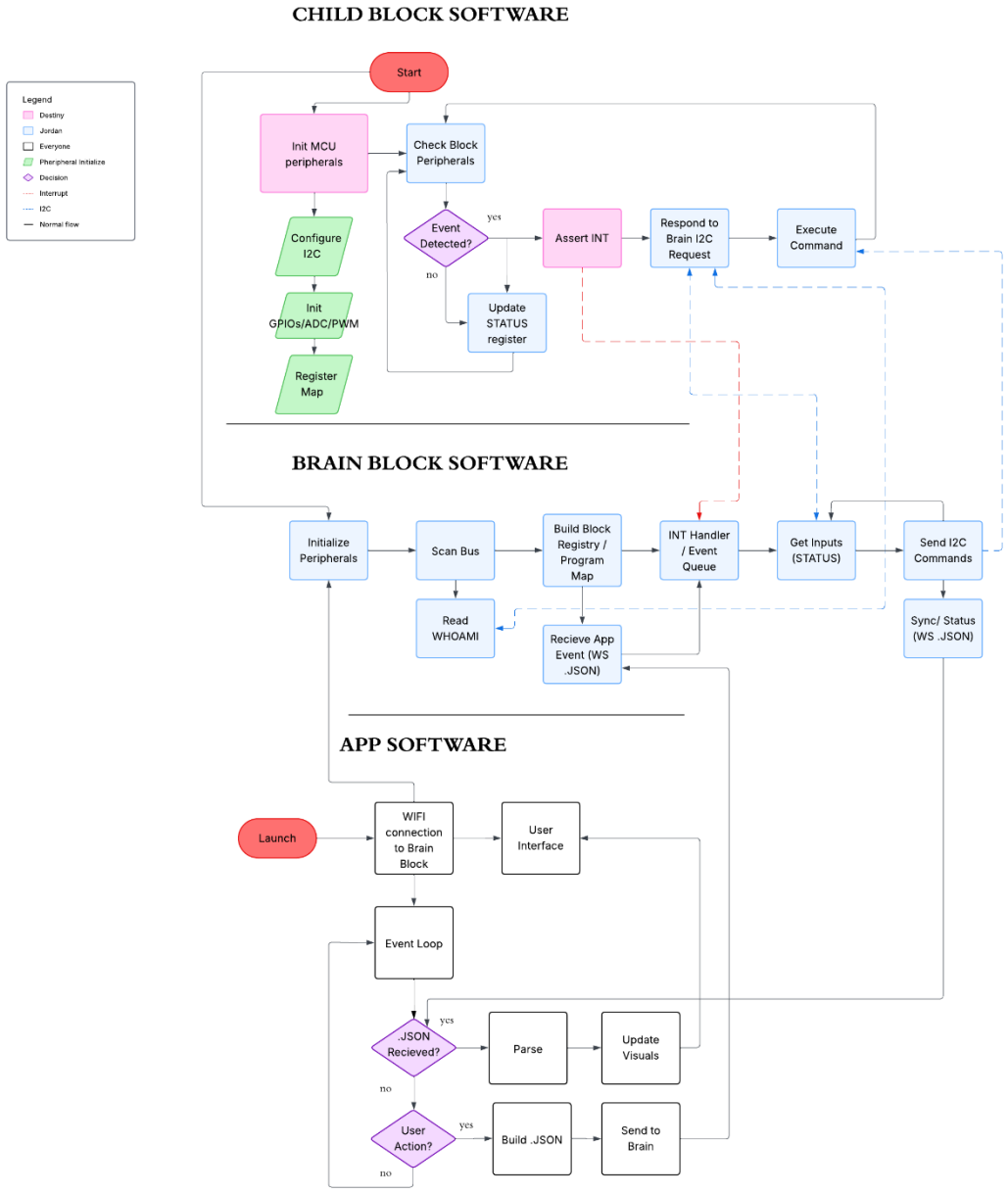


Figure 7.1 *Software Block Diagram*

7.2.2 Component Interaction Description

The *Blocks o' Code (v3)* system relies on continuous interaction between the three main components: the Brain Block, the Child Blocks, and the Companion App. The Brain Block, powered by an ESP32-WROOM module, acts as the central controller that manages logic,

synchronization, and wireless communication. Each Child Block, also using an ESP32-WROOM, performs localized functions, such as lighting LEDs, taking input, or generating sound and reports its status back to the Brain through the I²C communication bus.

When the system is powered on, the Brain Block scans the I²C bus to detect all connected Child Blocks and builds a device registry that maps each block's unique address, type, and function. The Brain then continuously polls these devices for status updates and waits for interrupts from them when an event is detected, such as a button press or sensor reading. Commands and configuration data are sent back over I²C to control outputs or trigger programmed behaviors.

The Companion App connects to the Brain Block over Wi-Fi, serving as the visual and interface for users. It receives live JSON status packets from the Brain to display the current program layout and block activity in real time.

Together, these three components form a closed-loop system where physical actions, logic processing, and digital visualization remain tightly integrated. The Brain coordinates hardware-level communication, the Child Blocks execute functions at the edge, and the Companion App provides a high-level user interface ensuring a synchronized and intuitive learning experience.

7.3 Communication and Data Transfer

7.3.1 I2C Communication

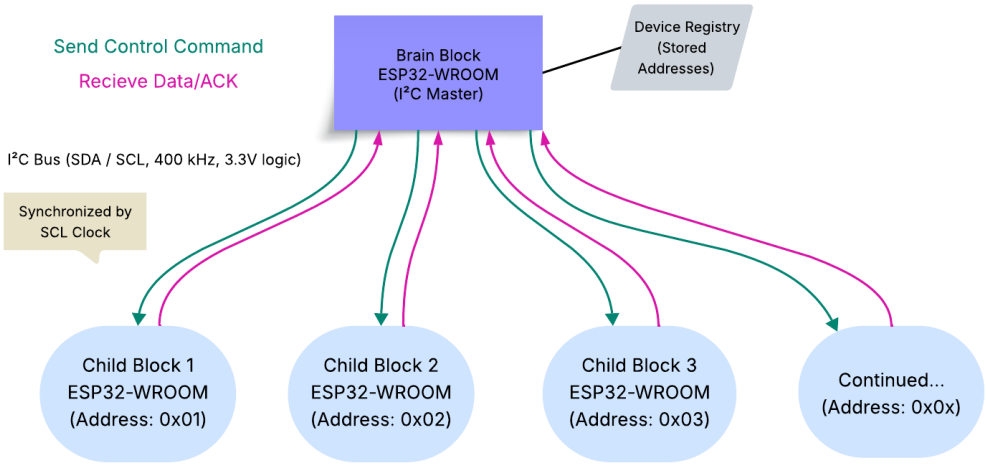


Figure 7.2 *I²C Communication*

7.3.2 SPI Communication

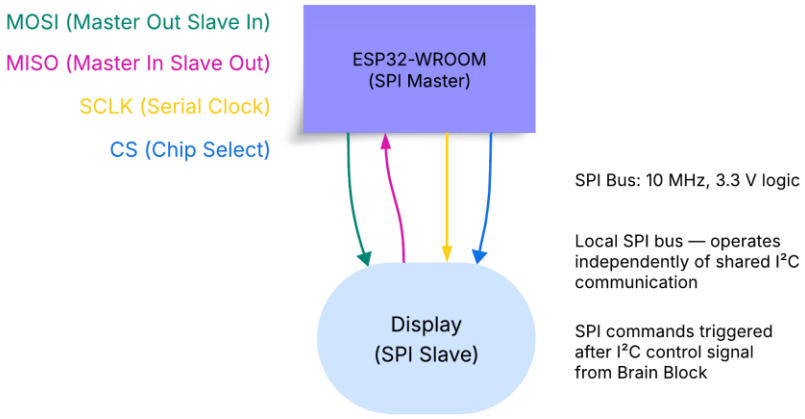


***Figure 7.3** SPI Communication*

7.3.3 Wi-Fi Communication

The Wi-Fi communication system in *Blocks o' Code (v3)* enables real-time data exchange between the Brain Block and the Companion App. Both the Brain and Child Blocks use ESP32-WROOM modules, which include an integrated Wi-Fi transceiver that can operate as either an access point or station. In this setup, the Brain Block acts as the central Wi-Fi server while the Companion App connects to it as a client over a local network or direct access-point mode.
Once a connection is established, communication occurs using JSON packets transmitted over TCP or WebSocket. The Brain Block periodically sends structured JSON messages containing the system status, block registry data, and sensor updates. The Companion App listens to these packets and parses the incoming information to update its on-screen visualization in real-time.

When the user interacts with the app, such as pressing a button or changing a parameter, the app builds a new JSON command packet and sends it back to the Brain Block. The Brain then decodes the JSON payload, updates its logic or LED patterns, and synchronizes the changes across all connected Child Blocks through I²C.

This Wi-Fi link provides a reliable two-way communication channel that supports continuous updates and low-latency feedback. It allows the physical blocks and digital interface to remain synced so that children can instantly see the effects of their interactions on both the hardware and the app display.

This Wi-Fi link provides a reliable two-way communication channel that supports continuous update and low-latency feedback. It allows the physical blocks and digital interface to remain in sync so that children can instantly see the effects of their interactions on both the hardware and the app display.

7.3.4 Timing and Synchronization

Timing and synchronization ensure that the Brain and Child Blocks in the *Blocks o' Code (v3)* system remain coordinated across all communication layers whether exchanging data through I²C, SPI, or Wi-Fi. Each ESP32-WROOM module operates under the FreeRTOS environment, which uses a hardware timer-based system to tick as the global clock reference. This global clock controls the task scheduling, software timers, and event sequencing throughout the system.

At the hardware level, the I²C bus uses the Serial Clock Line (SCL) generated by the Brain Block to synchronize all Child Blocks. Because the Brain acts as the single master, it defines the timing for all read and write transactions, ensuring deterministic communication. Each message is timestamped in the Brain's registry, and if a Child fails to acknowledge within the specified timeout window, the firmware marks it as inactive until the next scan cycle.

Within each block, the SPI interface relies on its own clock (SCLK) for synchronized data transfer between the microcontroller and its connected peripheral, such as an OLED or TFT display. These SPI clocks are derived from the same global system clock on the ESP32, maintaining consistent timing across devices even though SPI and I²C operate independently.

At the network layer, Wi-Fi communication runs asynchronously but remains synchronized through FreeRTOS software timers. The Brain Block periodically sends JSON updates to the Companion App. The app listens using a non-blocking event loop that parses incoming packets and updates the display in real time.

By combining the FreeRTOS global clock, hardware bus clocks, and software timers, the system achieves unified timing across all communication domains. This layered synchronization ensures smooth coordination between data transfers, consistent animation timing, and reliable user feedback.

7.4 State Diagrams

7.4.1 Brain Block State Machine

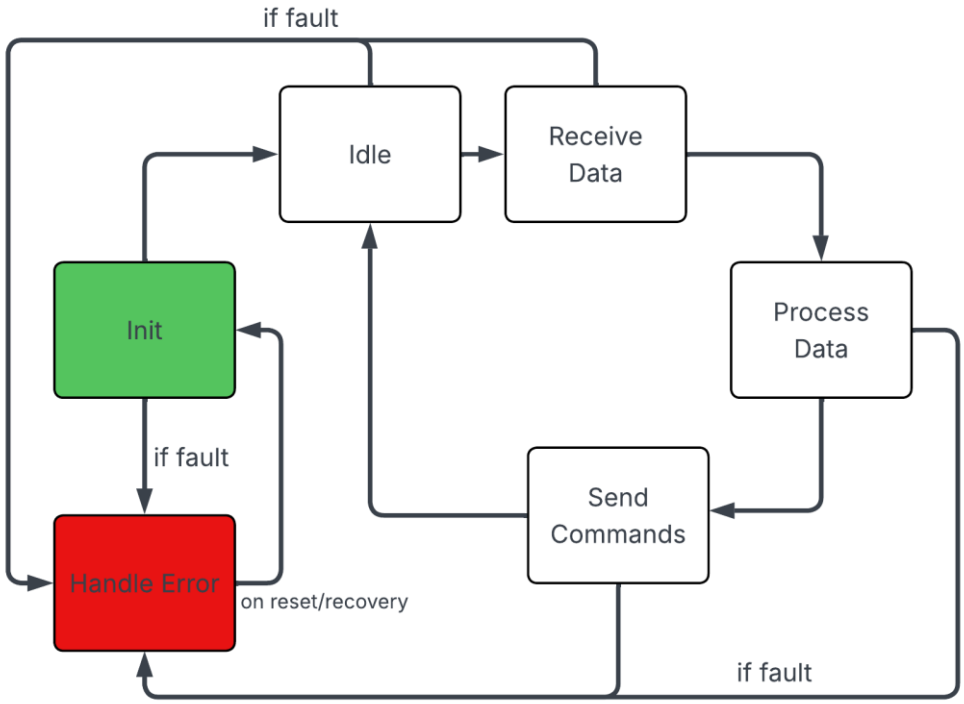


Figure 7.4 *Brain Block State Machine*

The **Brain MCU state machine** begins in the **Init** state, where the ESP32-S3 initializes all communication interfaces, including UART, I²C, and SPI, along with its connected peripherals such as the EEPROM, sound IC, and display. Once initialization is complete, the MCU transitions to the **Idle** state, where it awaits data packets or commands from the Child MCUs or the Companion App.

When a message or sensor update is received from a Child MCU, the system transitions to **Receive Data**, parsing and validating the incoming information. It then enters the **Process Data** state to interpret sensor readings, manage synchronization between Child blocks, and determine the appropriate response or system update. In the **Send Commands** state, the Brain MCU transmits configuration data, control signals, or acknowledgments back to the Child MCUs and updates the Companion App through wireless communication.

If a communication timeout, invalid data packet, or peripheral fault occurs at any point, the system transitions into the **Error Handling** state, where the MCU attempts recovery actions such as reinitializing interfaces or performing a controlled reset before returning to normal operation.

7.4.2 Child Block State Machine

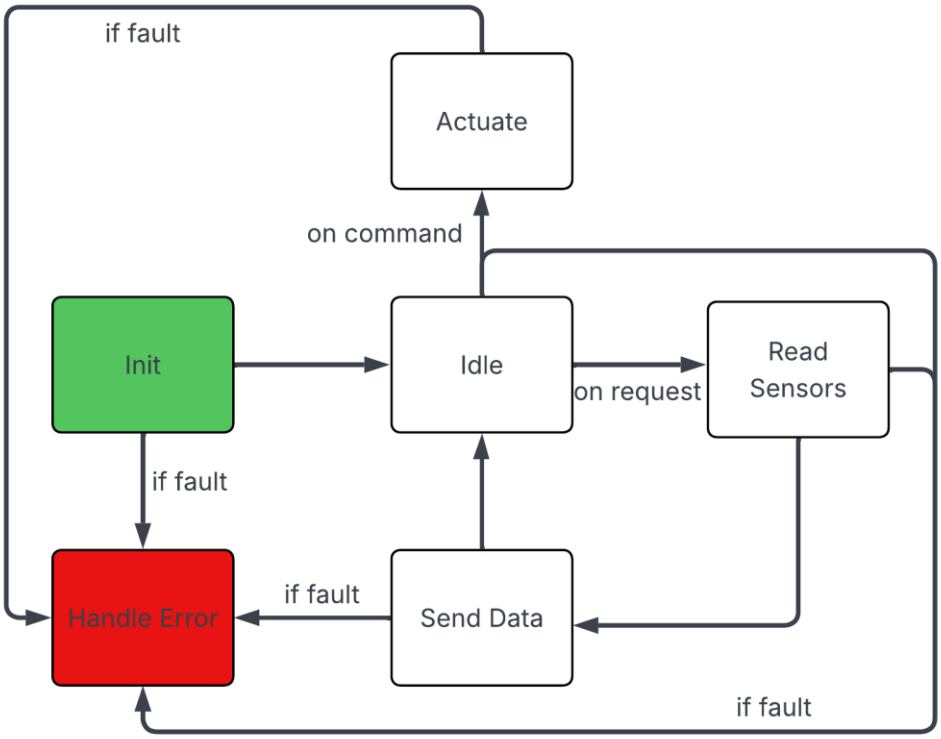


Figure 7.5 *Child Block State Machine*

The **Child MCU state machine** begins in the **Init** state, where the ESP32-WROOM-32 initializes its GPIO pins, communication interfaces (UART, I²C, or SPI), and connected peripherals such as sensors, LEDs, or actuators. Once setup is complete, the system transitions to the **Idle** state, where it waits for a request or command from the Brain MCU.

When the Brain MCU issues a data request, the Child MCU enters the **Read Sensors** state, where it samples inputs from its peripherals. This may include capturing audio levels from the microphone or reading user input from switches, buttons, or rotary dials. After collecting and formatting the data, the MCU transitions to the **Send Data** state to transmit the results back to the Brain MCU via I2C communication.

If the Brain MCU instead issues a control or actuation command, the Child MCU transitions from **Idle** to the **Actuate** state, where it drives connected components such as LEDs, motors, or sound devices according to the received parameters. After completing the action, the MCU returns to **Idle**, ready for the next instruction.

At any point, if a communication timeout, sensor read error, or hardware malfunction occurs, the system enters the **Handle Error** state. Here, the Child MCU may attempt recovery operations, such as resetting peripherals or reinitializing interfaces, before returning to **Init** or **Idle** for normal operation.

7.4.3 Companion App State Machine

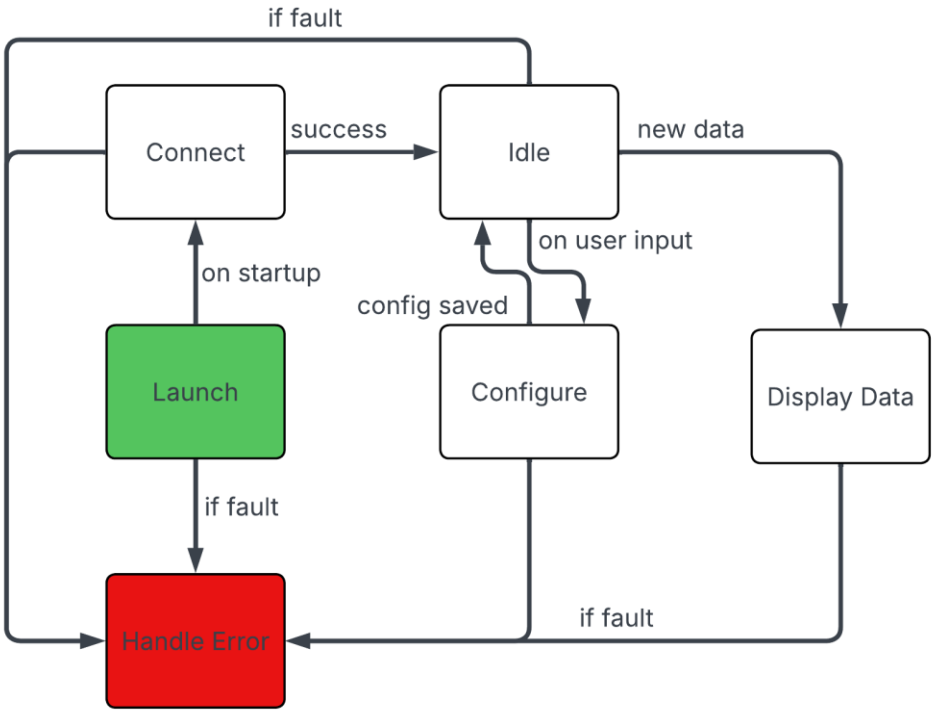


Figure 7.6 *Companion App State Machine*

The **Companion App state machine** begins in the **Launch** state, where the mobile application initializes its user interface, loads stored preferences, and prepares wireless communication such as Bluetooth Low Energy (BLE) or Wi-Fi to connect with the Brain MCU. After startup, the app transitions to the **Connect** state, where it attempts to establish communication with the Brain module. Once the connection is successful, the app moves to the **Idle** state.

In the **Idle** state, the Companion App maintains the connection and waits for new data or user interaction. When the Brain MCU sends updated block data, such as sensor readings, configuration values, or system status, the app transitions to the **Display Data** state to refresh the user interface with the latest information.

If the user modifies settings or sends control inputs, the app enters the **Configure** state. In this state, it updates configuration parameters such as LED brightness, sound playback options, or sensor thresholds, then sends the new settings to the Brain MCU. Once the configuration is saved and confirmed, the app returns to the **Idle** state.

If a connection failure, synchronization problem, or invalid input occurs, the system moves to the **Handle Error** state. Here, the app notifies the user or retries the connection. Once recovery is complete, it returns to the **Connect** state to re-establish normal operation.

7.4.4 Senior Design 2 Brain Block State Machine

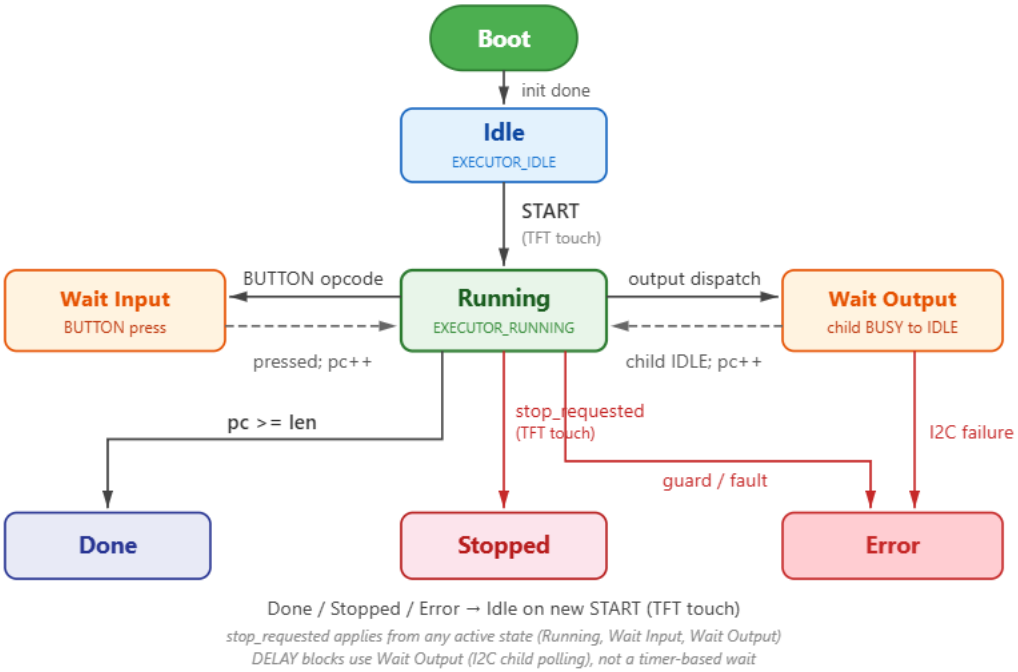


Figure 7.7 *Updated Brain Block State Machine*

The Brain Block state machine, shown in *Figure 7.7*, begins in the **Boot** state, where app_main() initializes all peripherals and creates the necessary FreeRTOS tasks. Once initialization completes, the system transitions to the **Idle** state (EXECUTOR_IDLE), where it awaits a START command from the user via the TFT touchscreen.

When the user presses Start and the block configuration has been validated, the executor transitions to the **Running** state (EXECUTOR_RUNNING), where it steps through the user's block program using a program counter, dispatching opcodes to Child Blocks over I²C. From the Running state, the executor may branch into one of two wait states depending on the opcode. When a Button Press opcode is reached, the executor enters **Wait Input**, pausing until the physical button block reports a press event via I²C polling, after which the program counter is incremented and execution returns to Running. When any output action is dispatched such as LED Flash, Note, Music Sequence, or Delay, the executor enters **Wait Output**, polling the child block's I²C status register until it transitions from STATUS_BUSY to STATUS_IDLE before incrementing the program counter and returning to Running. Delay blocks are handled through this same Wait Output path using I²C child polling rather than a timer-based wait.

When the program counter reaches the end of the block sequence, the executor transitions to the **Done** state. If the user presses Stop at any point during Running, Wait Input, or Wait Output, the executor transitions to **Stopped**. If an I²C failure or guard fault occurs, the executor transitions to the **Error** state. From Done, Stopped, or Error, the system returns to Idle when the user initiates a new START command from the TFT touchscreen.

7.4.5 Senior Design 2 Child Block State Machine

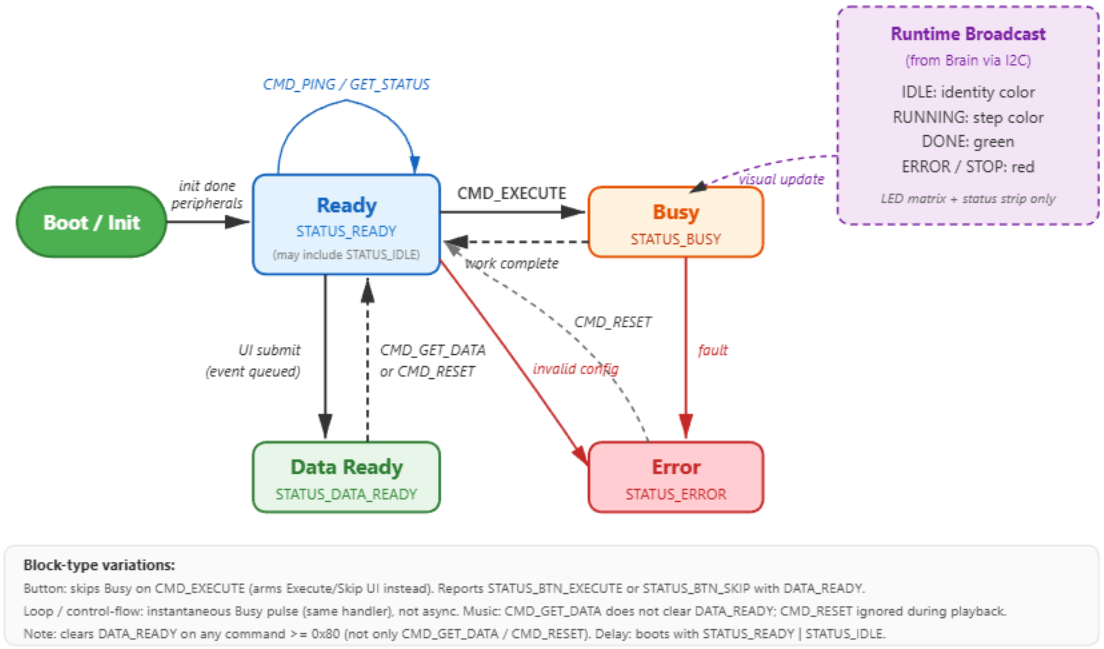


Figure 7.8 *Updated Child Block State Machine*

Child blocks do not use a single state enum. Instead, each child maintains a status bitmask composed of five flags defined in i2c_protocol.h: STATUS_READY (0x01), STATUS_BUSY (0x02), STATUS_ERROR (0x04), STATUS_DATA_READY (0x08), and STATUS_IDLE (0x10). Button blocks additionally use STATUS_BTN_EXECUTE (0x20) and STATUS_BTN_SKIP (0x40). These flags can be combined, providing a composable status model suited to the variety of child block behaviors.

The child block state machine begins in the **Boot** phase, where the microcontroller initializes its peripherals, I²C slave interface, LED matrix, status strip, and status registers including REG_WHOAMI for block-type identification. Once initialization completes, the block enters the **Ready** state, indicated by STATUS_READY. Some blocks, such as the Delay block, also set STATUS_IDLE at boot to indicate they are between execution steps.
When the Brain sends a CMD_EXECUTE command, output blocks including LED Flash, Note, Music Sequence, and Delay transition to the **Busy** state by setting STATUS_BUSY. Each output block processes its action asynchronously through a FreeRTOS worker task: LED Flash runs a color animation, Note plays audio via the speaker, Music Sequence plays a full song, and Delay runs a timer task. The Brain polls the child's status register during this period, waiting for STATUS_BUSY to clear before continuing program execution. Once the work completes, the block returns to **Ready**. Control-flow blocks including If, Then, End If, Loop, and End Loop set STATUS_BUSY only briefly within the same command handler for visual feedback before immediately

returning to STATUS_READY — they do not perform asynchronous work. Button blocks intentionally skip the **Busy** state entirely on CMD_EXECUTE, instead arming their touchscreen UI to present an Execute or Skip choice to the user.

Blocks with user-facing configuration can enter the **Data Ready** state when the user submits a selection via the block's local UI. This sets the STATUS_DATA_READY flag and populates REG_DATA_LEN, signaling the Brain to retrieve the event payload. For most blocks, the Brain issues CMD_GET_DATA or CMD_RESET to clear the flag and return to **Ready**. However, behavior varies by block type: Note blocks clear STATUS_DATA_READY on any received command with opcode 0x80 or higher rather than only on CMD_GET_DATA or CMD_RESET, while Music Sequence blocks do not clear STATUS_DATA_READY on CMD_GET_DATA at all. Their selection state is driven entirely by the block's local UI.

Additionally, Music Sequence blocks ignore CMD_RESET while playback is active.
If an invalid configuration is detected during execution or a hardware fault occurs such as a speaker initialization failure or queue overflow, the block transitions to the **Error** state by setting STATUS_ERROR. A CMD_RESET command from the Brain clears the error and returns the block to **Ready** for most block types, with the Music Sequence exception noted above.
Independently of the command and status cycle, each child block receives **runtime broadcast** messages from the Brain via CMD_RUNTIME_BROADCAST. These 3-byte payloads carry the current execution state, active program counter, and step block type. The shared status strip module uses this information to update the block's LED matrix and WS2812 status strip, displaying the block's identity color during Idle, the active step color during Running, green on Done, or red on Error and Stop. This broadcast-driven visual update operates as a parallel overlay and does not modify the block's REG_STATUS flags. LED Flash blocks additionally suppress runtime broadcast visuals while an effect is actively playing.

7.4.6 Senior Design 2 Companion App State Machine

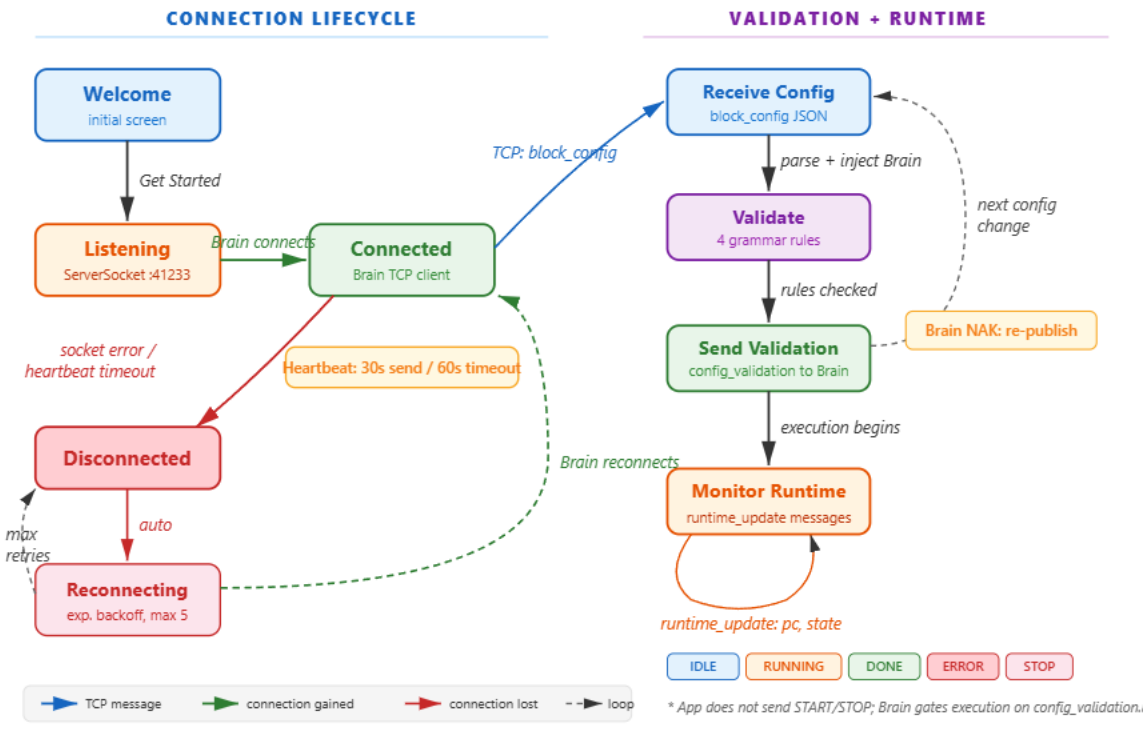


Figure 7.9 *Updated Companion App State Machine*

The Companion App state machine encompasses two interleaved concerns: the TCP connection lifecycle and the validation and runtime monitoring pipeline.

Connection Lifecycle

The app begins on the **Welcome** screen. When the user taps "Get Started," the app binds a ServerSocket on port 41233 and enters the **Listening** state, awaiting an inbound TCP connection. The app acts as a TCP server while the Brain Block, acting as a TCP client, connects to the app over Wi-Fi. Upon accepting the connection, the app transitions to the **Connected** state, starts a heartbeat timer that sends periodic JSON keep-alive messages every 30 seconds, navigates the UI to the Block Configuration screen, and immediately publishes any cached validation state to the newly connected Brain.

If the TCP socket encounters an error or no qualifying activity, including heartbeat acknowledgment, block configuration, runtime update, or telemetry message is received within the 60-second timeout window, the app transitions to the **Disconnected** state. From here, it automatically enters the **Reconnecting** state, employing exponential backoff with delays of 1, 2, 4, 8, and 16 seconds capped at 30 seconds for up to 5 attempts. During reconnection, the TCP server remains bound and listening while the Brain must re-establish the TCP connection. If the Brain reconnects within the retry window, the app returns to **Connected**. If all retry attempts are exhausted, the reconnection counter resets and the app returns to **Disconnected**, requiring user intervention.

Validation and Runtime Pipeline
While connected, the app processes newline-delimited JSON messages from the Brain. When a block_config message arrives, the app enters the **Receive Config** phase, parsing the JSON payload into a BlockConfiguration model via BlockConfigParser. If the firmware reports zero blocks, a synthetic Brain block is injected at index 0 for UI consistency.

The app then enters the **Validate** phase, where the ConfigurationRules engine applies four structural grammar rules: the Brain block must exist at index 0; If sequences must follow the pattern If, Button Press, Then, one or more outputs, End If; Loop sequences must follow the pattern Loop, one or more outputs, End Loop; and interleaved or malformed nesting produces a warning. The validator uses a stack-based approach to track open control-flow frames and produces a list of RuleViolation entries with error or warning severity.

After validation, the app enters the **Send Validation** phase, transmitting a config_validation JSON message back to the Brain containing the validity boolean, error count, and timestamp. The Brain's executor uses this validation result to gate whether it will accept a START command from the TFT touchscreen. The app itself does not send START or STOP commands. Those are triggered locally on the Brain's TFT display. If the Brain sends a NAK:NEED_VALIDATION or NAK:INVALID_CONFIG message, the app re-publishes its most recent validation result. This validation cycle repeats each time a new block_config message arrives as blocks are physically added or removed.

Once execution begins on the Brain, the app enters the **Monitor Runtime** phase, processing runtime_update messages that carry the executor's current state such as idle, running, step, done, error, or stop, along with the active program counter and the current step's block type. The app merges this runtime snapshot into the current configuration model and updates the UI to highlight the actively executing block in the chain visualizer.

7.5 Data Structures and Storage

Data handling in the *Blocks o' Code (v3)* system is structured around FreeRTOS, which is implemented on the ESP32 within the ESP-IDF framework. FreeRTOS provides the foundation for task scheduling, inter-task communication, and resource synchronization. To achieve this, it uses several core data structures that keep the firmware organized and predictable during real-time operation.

Task Control Blocks (TCBS):
Every task created by the system, such as LED control, sensor monitoring, or Wi-Fi communication is represented by a TCB. This structure stores essential information including the task's state (running, ready, or blocked), priority, stack pointer, and name. The TCB allows FreeRTOS to track and switch tasks efficiently, ensuring deterministic multitasking on the ESP32.

Queues:
Queues are used for inter-task communication, allowing different software modules to exchange data safely. In *Blocks o' Code (v3)*, queues handle messages such as sensor readings, LED updates, or block status reports between the Brain and Child tasks. Each queue acts as a circular buffer, ensuring data integrity when multiple tasks access shared information.

Semaphores and Mutexes:
Semaphores are used to manage access to shared resources like I²C and SPI buses. Binary semaphores ensure only one task communicates at a time, while mutexes protect shared memory regions or global data structures from race conditions.

Event Groups:
Event groups provide synchronization across multiple tasks. For instance, a Brain Block task may wait until all Child Blocks confirm readiness before beginning program execution. Each event group uses bitmasks to represent different task states, enabling flexible coordination without heavy polling.

Beyond task management, the firmware organizes project-specific data into compact C structures and higher-level JSON objects:

- **Local Structures:** Each Child Block maintains a small structure containing its unique ID, type, and live sensor or output data (e.g., RGB values or sound levels).
- **Brain Registry:** The Brain maintains an array of these structures indexed by I²C address, forming a live map of all connected blocks.
- **JSON Packets:** For Wi-Fi communication with the Companion App, the Brain serializes this data into JSON format, transmitting updates in real time.

Storage in the system uses both **volatile** and **non-volatile memory**.
Volatile RAM holds runtime variables, buffers, and queue data, while the ESP32's **NVS (Non-Volatile Storage)** retains Wi-Fi credentials, calibration values, and user configurations between sessions.

Together, these FreeRTOS and application-level data structures ensure that *Blocks o' Code (v3)* remain synchronized, efficient, and scalable. This combination of low-level real-time control and structured data representation allows the system to coordinate communication, logic execution, and user interaction smoothly across all components.

7.6 User Interface Design

The software user interface for *Blocks o' Code (v3)* is a **digital companion application** designed to run on mobile devices. The companion app complements the physical blocks with a real-time digital interface. The app's primary role is to **visualize the physical block arrangement** and its corresponding logic. As a user adds or removes blocks, the app's display updates instantly to mirror the physical sequence.

The app's core functionality is to establish and maintain a **wireless connection** with the Brain Block via Wi-Fi. This connection serves as the **main data link**, sending and receiving information through JSON packets.

The main features of the software UI include:

- **Real-Time Visualization:** The app aims to mirror the physical arrangement of the coding blocks. As a child adds, removes, or rearranges the tangible blocks, the Brain Block sends status updates to the app, which refreshes its display to reflect the new program's sequence with determinism.
- **Virtual "Disco" Experience:** A key feature is the app's ability to generate a visual representation of the program's output. When the user activates the "Start" block, the app simultaneously creates a "virtual party." This includes animated lights, music cues, and effects that directly mirror the outputs of the physical LED and Speaker blocks, creating a fun and engaging multi-sensory experience.
- **Configuration and Control:** The app allows users to interact with the system, not just observe it. Users can potentially modify settings, trigger commands, or manage firmware through the interface.

7.6.1 Senior Design 2 Implementation

The companion app developed in senior design 2 serves as the primary interface for real-time block configuration visualization and validation. The app runs as a TCP server on port 41233, waiting for the Brain Block to connect as a TCP client over Wi-Fi. Once connected, the Brain Block streams newline-delimited JSON configuration snapshots to the app whenever the physical block arrangement changes.

The main features as implemented are:

- **3D Block Chain Visualization:** The app renders the connected block sequence as an interactive 3D chain of labeled blocks, updating in real time as blocks are added, removed, or rearranged.
- **Configuration Validation:** The app parses each JSON update against the system's rule set and displays validation results — errors (e.g., missing End If, invalid sequence) and warnings — so users can correct configurations before running a program.
- **Config-Change Latency Display:** The app measures and displays the end-to-end latency from physical block rearrangement to UI update, which measured 50 ms under full 15-block chain testing.
- **Guided Playthrough:** On launch, the app prompts users to try a guided playthrough. Once started, an overlay tracks live block configuration changes and walks the user through assembling a complete If/Then sequence step by step, advancing automatically as each block is connected. A restart option and completion screen are included.
- **Tutorial Screen:** A dedicated reference screen accessible from the navigation bar provides descriptions of each block type and valid configuration patterns for users to consult at any time.

The virtual disco visualization described in Section 7.6 was not implemented in senior design 2. The app focuses on configuration guidance and real-time feedback rather than output animation. Disco-style outputs are produced by the physical blocks themselves through LED Color Flash and Music Sequence block combinations.

7.7 Error Handling

Robust error handling is essential for maintaining system reliability, especially in a classroom environment. The firmware implements real-time monitoring and recovery mechanisms to detect, report, and correct errors during operation. This function is primarily managed by the Brain Block, which continuously validates the system's state and responds to communication faults, hardware faults, and logical errors.

7.7.1 Communication Faults

If the Brain Block fails to receive an acknowledgment (ACK) from a Child Block during an I²C transaction, the firmware will log the fault and attempt to retry the communication a set number of times. If the block remains unresponsive, it will be flagged as disconnected, and the system's logical map will be updated. Similarly, the companion app is designed to handle Wi-Fi connection timeouts and JSON parsing errors, notifying the user if the connection to the Brain is lost.

7.7.2 Hardware Faults:

The firmware for both Brain and Child blocks includes "if fault" transitions within their state machines. If a peripheral fails to initialize, or a communication fault persists, the firmware transitions to its 'Handle Error' state. This state's primary function is to perform a **programmatic soft reset**. It does this by logging the fault, illuminating the 'Error' LED, and then forcing the state machine to transition back to the **init state**. This re-initialization routine effectively resets all peripherals and communication buses from a clean slate without requiring a full power cycle, allowing the block to recover from a temporary fault. If recovery fails, the block will report a fault status to the Brain.

7.7.3 Invalid Block Configurations

This is the most critical error check for user interaction. The Brain Block's firmware is responsible for validating the sequence of connected blocks. When the user assembles a program, the Brain scans the I²C bus to build a registry of all blocks and their order. It then parses this sequence to check for logical errors (e.g., an "IF" block without a "THEN" or "ENDIF" block).

- If an invalid configuration is detected, the Brain Block's software immediately **pauses all execution tasks**.
- It sends a JSON status message to the companion app, notifying it that the current configuration is invalid.

- The companion app's user interface will then update to reflect this error state and will **disable the "Start" functionality**, preventing the user from running the faulty program.
- The Brain Block will not process any further program logic and will remain in this paused state until the user physically corrects the block configuration.
- Once a valid configuration is detected on a subsequent scan, the Brain will send a new status message to the app, which will then show the correct block configuration. The user will also be able to use the start button on the brain block.

7.7.4 Watchdog Timer (WDT)

Beyond the state machine's fault handling, the firmware will also implement a Watchdog Timer (WDT) as a final fail-safe mechanism. The WDT is an independent hardware timer within the microcontroller that must be periodically 'fed' (or reset) by the main software loop. If the software crashes, freezes, or gets stuck in an infinite loop, it will fail to 'feed' the watchdog. When the timer expires, it forces a **hardware hard reset**, rebooting the entire microcontroller. This ensures that a block can recover from an unexpected catastrophic software failure and never remain 'bricked' or unresponsive in a classroom environment.

Chapter 8 – System Fabrication/Prototype Construction

8.1 Brain Block PCB

The layout of the Brain Block PCB, shown in *Figure 8.1*, was designed with clarity and usability in mind. One of the first decisions I made was to place all the components that need wire connections around the outer edges of the board. This way, we can easily solder in modules like the charger, regulators, speaker, and battery without having to reach over other parts of the PCB.

The ESP32 microcontroller was placed near the top center so its antenna has open space around it and does not get blocked by nearby components. This helps keep the wireless performance strong and consistent.

I also added the pogo pin connectors on both the left and right sides of the board. Placing them in these positions lets the Brain Block attach directly to the Child Blocks with a matching pin configuration, so everything lines up perfectly without needing extra cables. It makes the whole system feel more like a set of puzzle pieces that fit together in an intuitive way.

Another small but important detail is the placement of the three status LEDs at the top right. They are grouped together inside a labeled box, and each one clearly shows its purpose, like correct, incorrect, or status for connectivity. This makes it really easy for anyone using the board to understand what the lights mean while the program is running. Overall, the layout focuses on making the Brain Block look organized, easy to follow, and beginner friendly while keeping all the electrical connections clean and efficient.

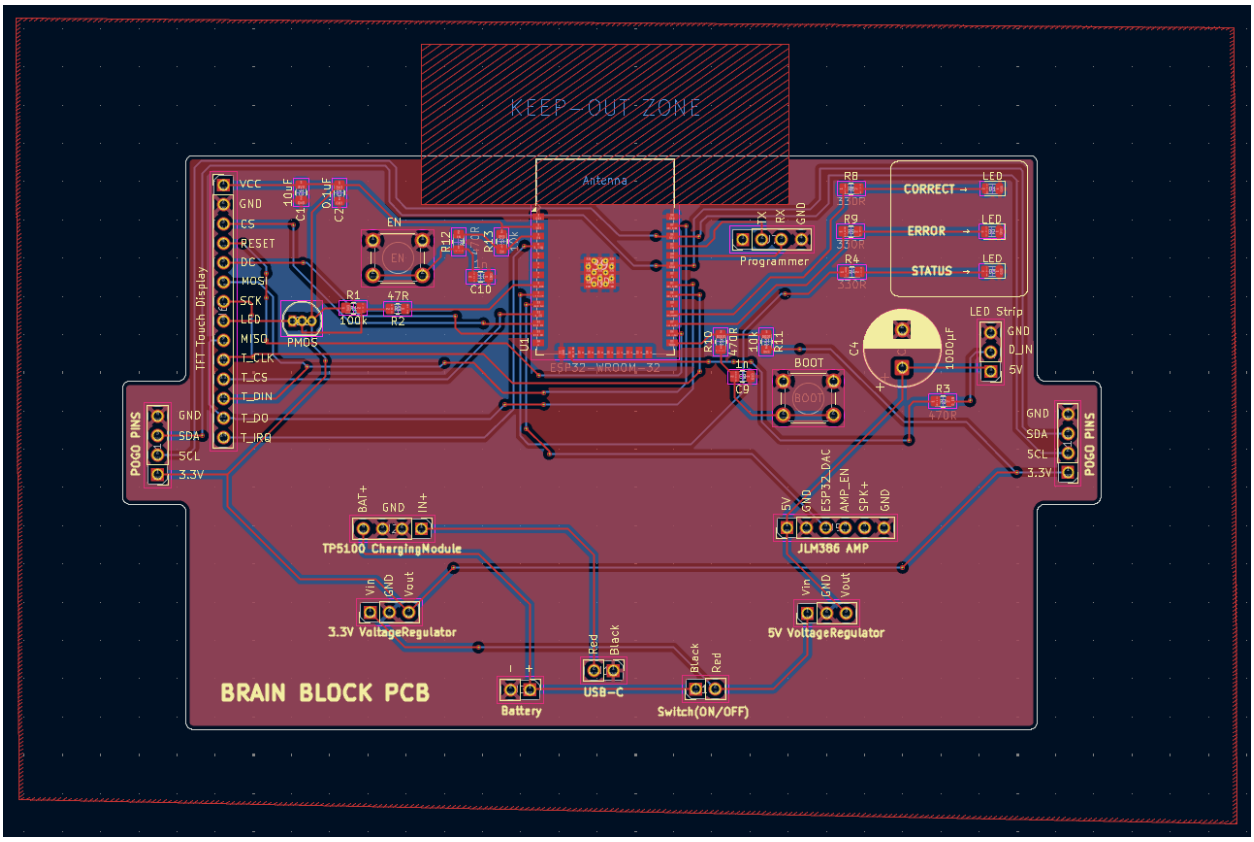


Figure 8.1 *Brain Block PCB Layout*

Figures 8.2 and 8.3 show the 3D model of the Brain Block. In *Figure 8.2*, you can clearly see how the ESP32 module sits above the board and how the buttons, LEDs, and connectors look once they are actually placed on the PCB. The 3D view also makes it easy to notice the shape of the board and how the pogo pin extensions stand out on both sides. *Figure 8.3* shows the back side, where the 3D model lets you see the through hole pads and how the traces run across the board. Together, these views give a nice visual of what the final Brain Block will look like once it is fully assembled.

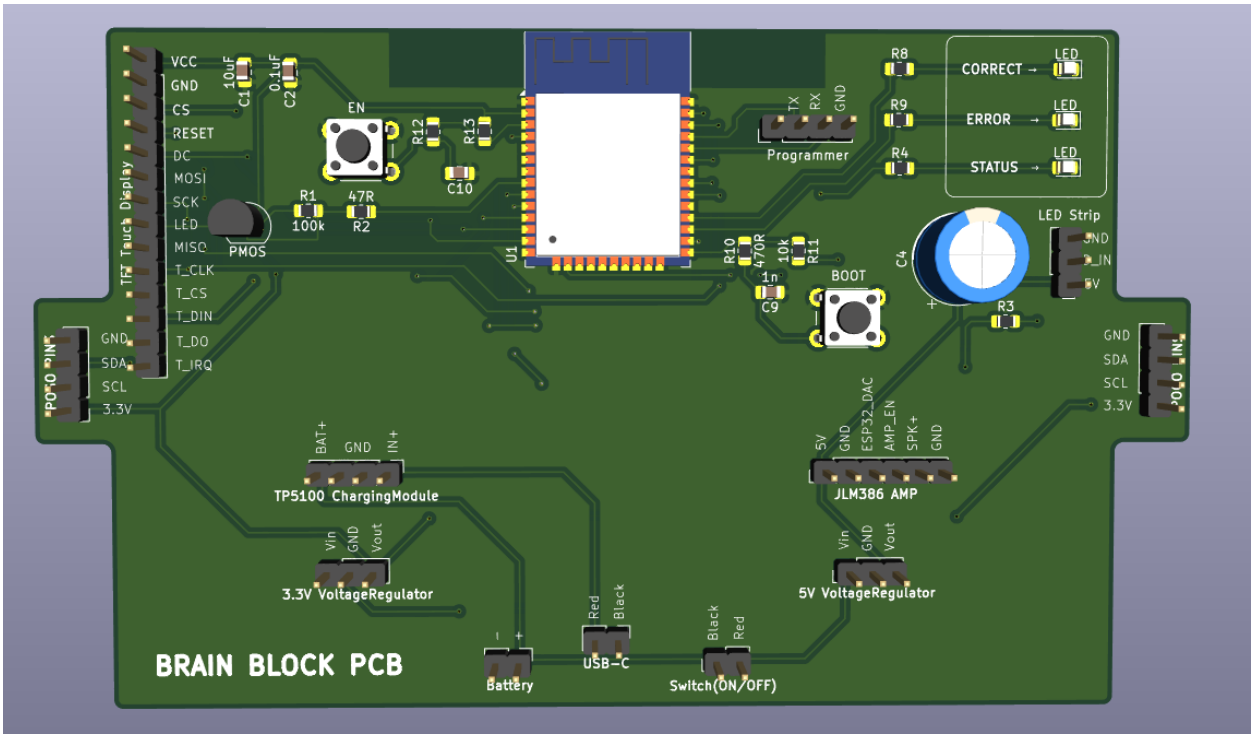


Figure 8.2 *Brain Block PCB 3D Model (Front)*

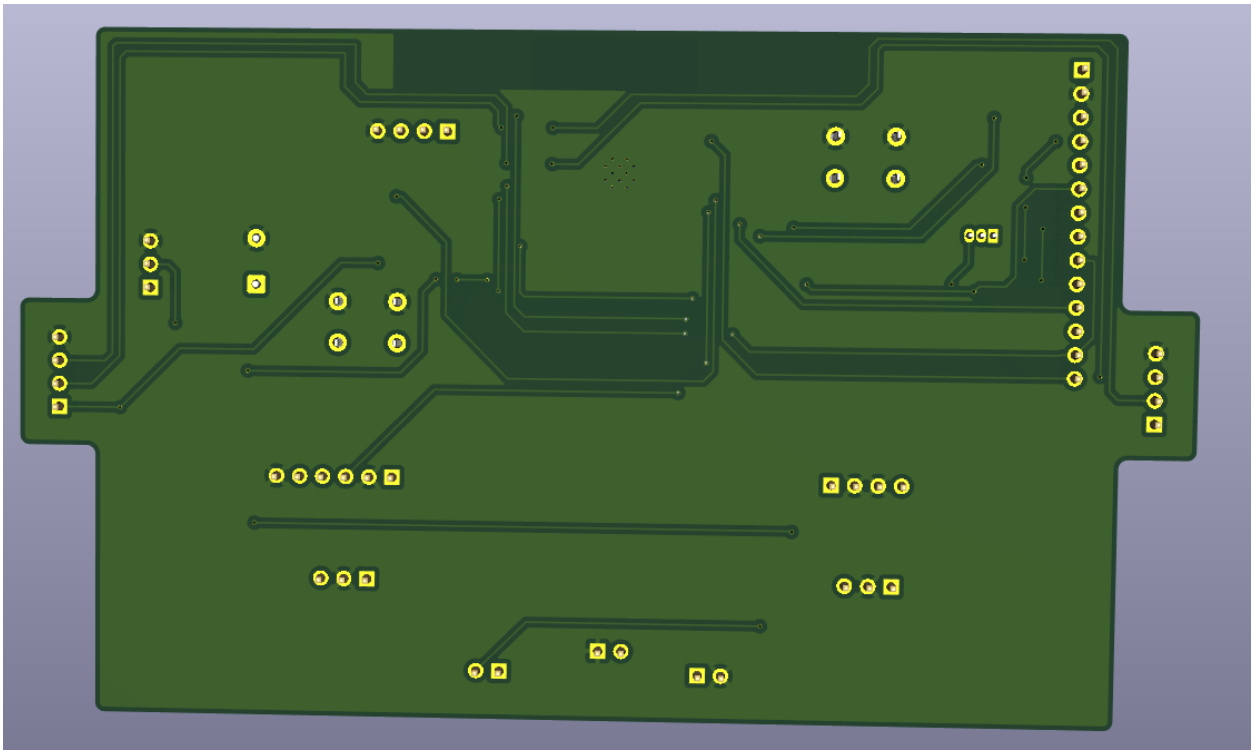


Figure 8.3 *Brain Block PCB 3D Model (Back)*

8.2 Child Block PCB

The layout of the Child Block PCB, shown in *Figure 8.4*, was designed around the idea that it should feel like a “peripheral hub” for the system while still fitting cleanly inside the standard 5 inch enclosure. Just like the Brain Block, all of the user-facing connectors are pushed toward the board's edges so that cables and modules can plug in without crossing over other components. The TFT display header is placed along the left edge, the I²C IN and I²C OUT headers sit on the left and right wings, and the LED-matrix, OLED, 1602 LCD, and LED strip headers are grouped along the bottom and right edges. Keeping these connectors on the perimeter makes assembly in the cube easier and keeps wiring short and predictable.

The ESP32-WROOM-32 sits near the top center of the board, aligned so that its PCB antenna has a clear “keep-out” window directly above it. No copper pour or tall components are allowed in this region, which helps preserve Wi-Fi and BLE performance. The three push buttons (BTN, EN, and BOOT) and the “correct”/“error” status LEDs are arranged around the ESP32, so they are easy to access once the block is inside the enclosure. This cluster also keeps all the critical reset and programming signals in one area, routed with short traces to reduce noise and make debugging easier.

On the upper-right side of the PCB, the MCP23017 I²C GPIO expander and its numpad header are placed together so that all keypad row and column lines can travel as a tight bundle. The LM386 amplifier header is also kept nearby, which shortens the analog path between the ESP32 DAC output and the audio module. Along the bottom center of the board, the USB-C connector, TP5100 charging interface, battery pads, and on/off switch are arranged in a straight path so that power flows logically from input to charger, then to the battery and finally to the 3.3 V and 5 V regulator headers in the middle of the board. Wider traces are used on the LED strip connector and the main 5 V rail to safely carry the higher current required by multiple addressable LEDs, and a large 1000 µF capacitor is placed right next to the LED strip header to handle current surges and keep the supply stable when LED animations switch.

The entire board uses a solid ground pour on both layers, with plenty of vias tying the top and bottom planes together shown in *Figure 8.5* and *Figure 8.6*. Most of the signal traces run on the bottom layer, so the top can remain clean around the antenna and user-visible components. This also keeps the back side, shown in *Figure 8.6*, mostly dedicated to routing and through-hole pads, which simplifies soldering and rework. Overall, the Child Block PCB balances routing density with clarity: each group of peripherals (displays, LEDs, numpad, audio, and power) is clearly separated, while shared buses like SPI and I²C are routed as short, organized bundles to keep the design reliable and easy to debug.

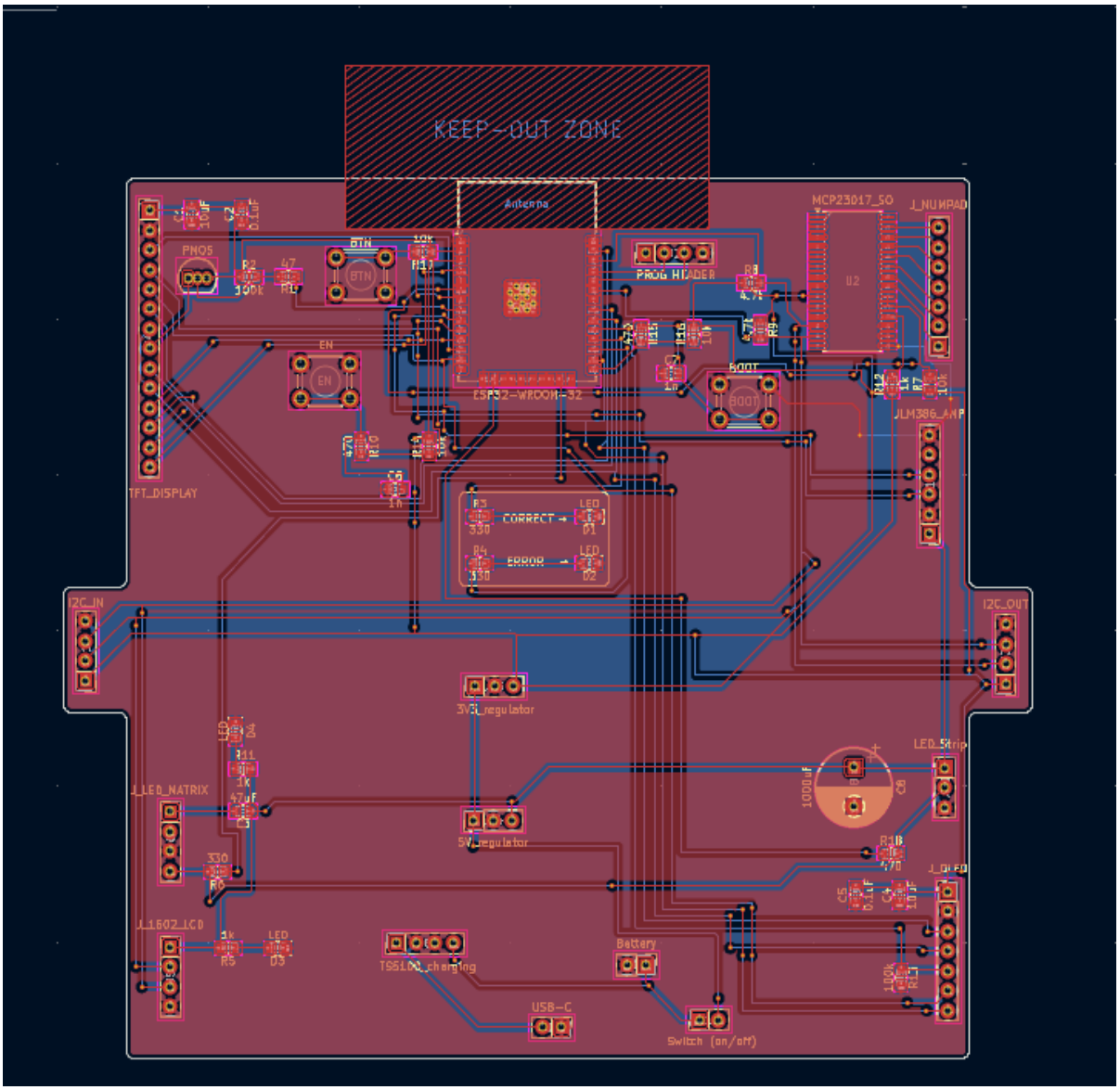


Figure 8.4 *Child Block PCB Layout*

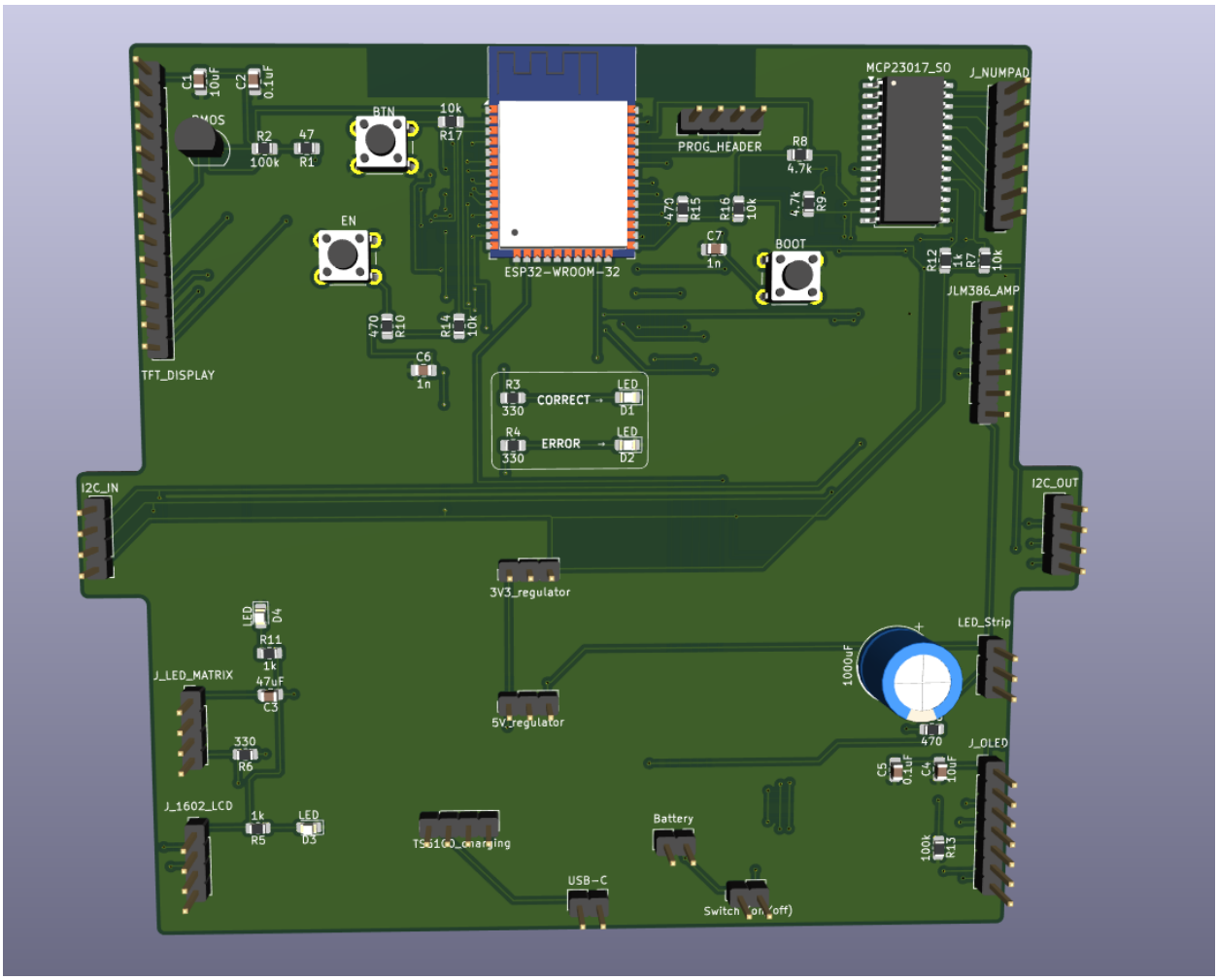


Figure 8.5 *Child Block PCB 3D Model (Front)*

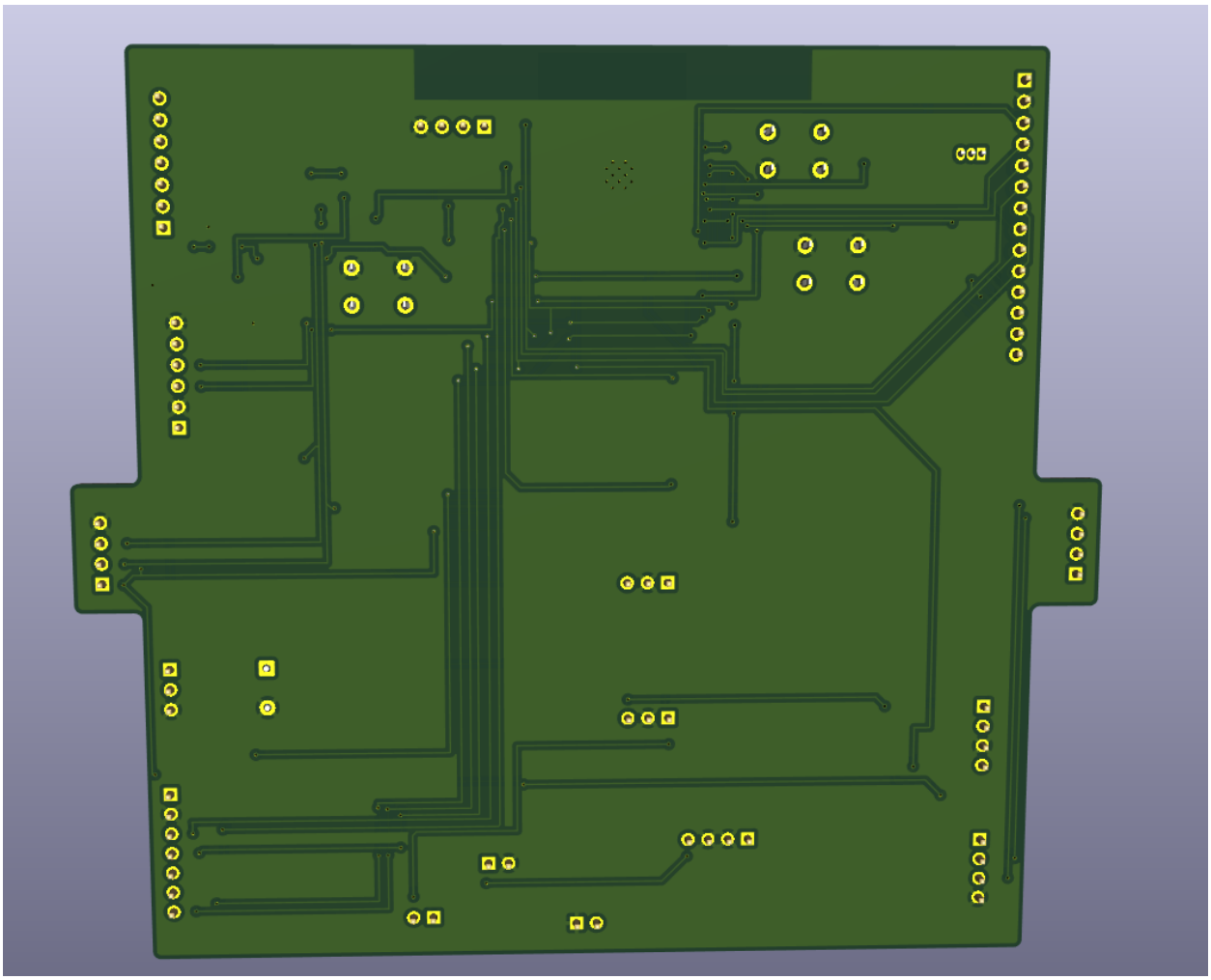


Figure 8.6 *Child Block PCB 3D Model (Back)*

8.2.1 Senior Design 2 Implementation

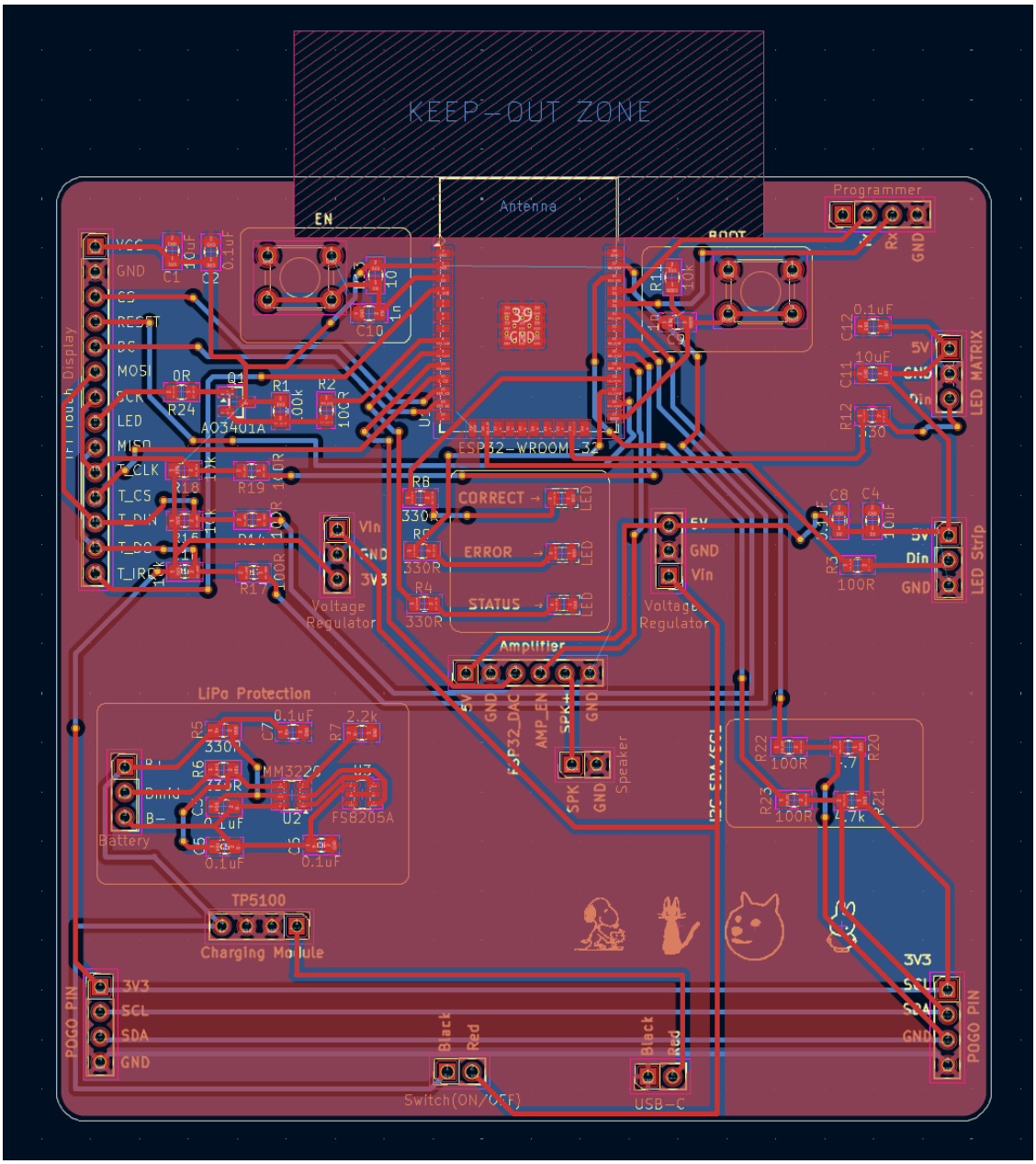


Figure 8.7 *Final Brain/Child PCB Layout*

The final Brain Block PCB was redesigned into a single main board that combines the controller, charging circuit, power regulation, and core system connections into one layout. Integrating the primary subsystems onto one PCB reduced the amount of internal wiring between separate control boards and removed unnecessary connection points that could create reliability issues during operation. While some external modules were still connected by wire for peripherals such as displays, speakers, and other outputs, the main system hardware was centralized onto one board for a cleaner overall design. This also allowed for better use of the enclosure space and improved organization of the internal components. The board was designed with a modular layout by including headers and dedicated connection points for peripherals and future expansion features, making it easier

to test components, replace modules, or add new functionality without redesigning the full board. Overall, moving to a single main PCB simplified assembly, improved reliability, and resulted in a cleaner and more flexible final design.

8.3 Amplifier

To support audio playback within the Child Block, we designed a compact LM386-based amplifier module, shown in *Figures 8.7-8.9*. The goal was to create a small, reliable board that could be mounted inside the enclosure while providing clean, adequate gain for simple sound effects generated by the ESP32's DAC output.

The module is built around the LM386 low-voltage audio amplifier, which is well-suited for battery-powered systems and requires only a handful of passive components. The layout follows the recommended TI reference design, including decoupling capacitors, a gain-stabilizing network, and an output shaping capacitor. A 250 µF electrolytic capacitor sits directly on the output to the speaker to filter DC and protect the driver, while additional 10 µF and 0.1 µF capacitors reduce noise on the supply rails and help the amplifier remain stable under varying load conditions.

The input, output, and power connections are broken into a single-row pin header along the left side of the PCB, allowing the module to plug directly into the Child Block. All analog routing is kept away from digital switching nodes to reduce interference. The bottom layer contains only a few wide power and output traces, while the top layer holds most of the components to simplify assembly and rework.

Overall, this amplifier module provides a clean, modular audio interface for the Child Block. By separating it onto its own PCB, the design minimizes noise coupling into the main board, simplifies debugging, and makes future replacements or revisions easy without needing to modify the primary Child Block PCB.

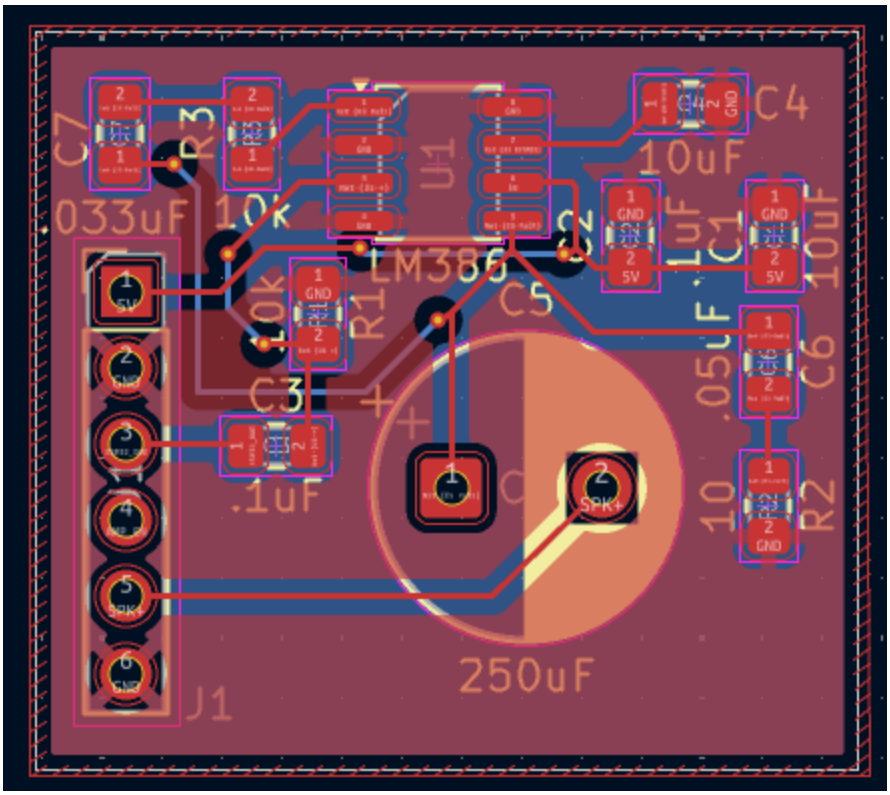


Figure 8.8 *Amplifier PCB Layout*

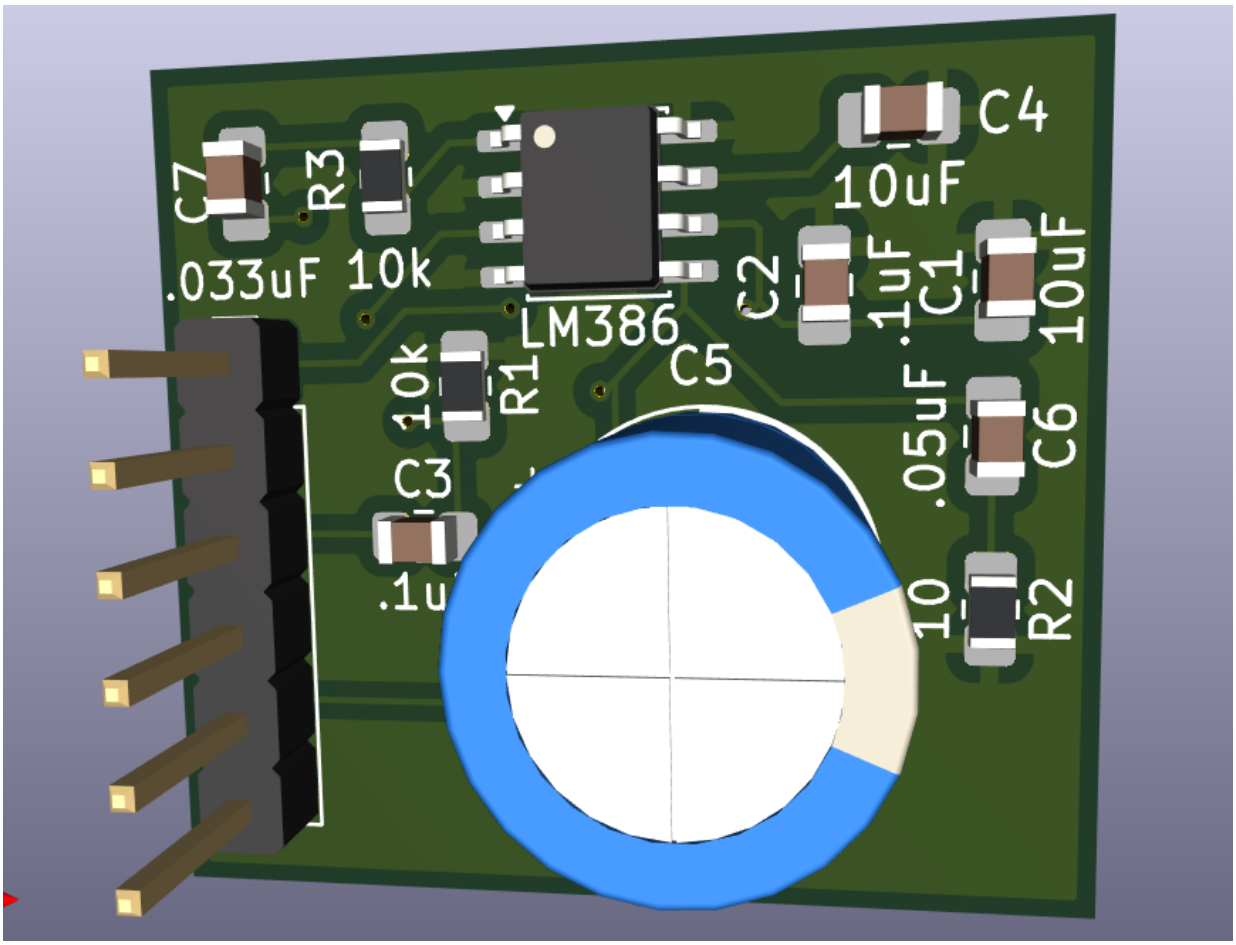


Figure 8.9 *Amplifier PCB 3D Model (Front)*

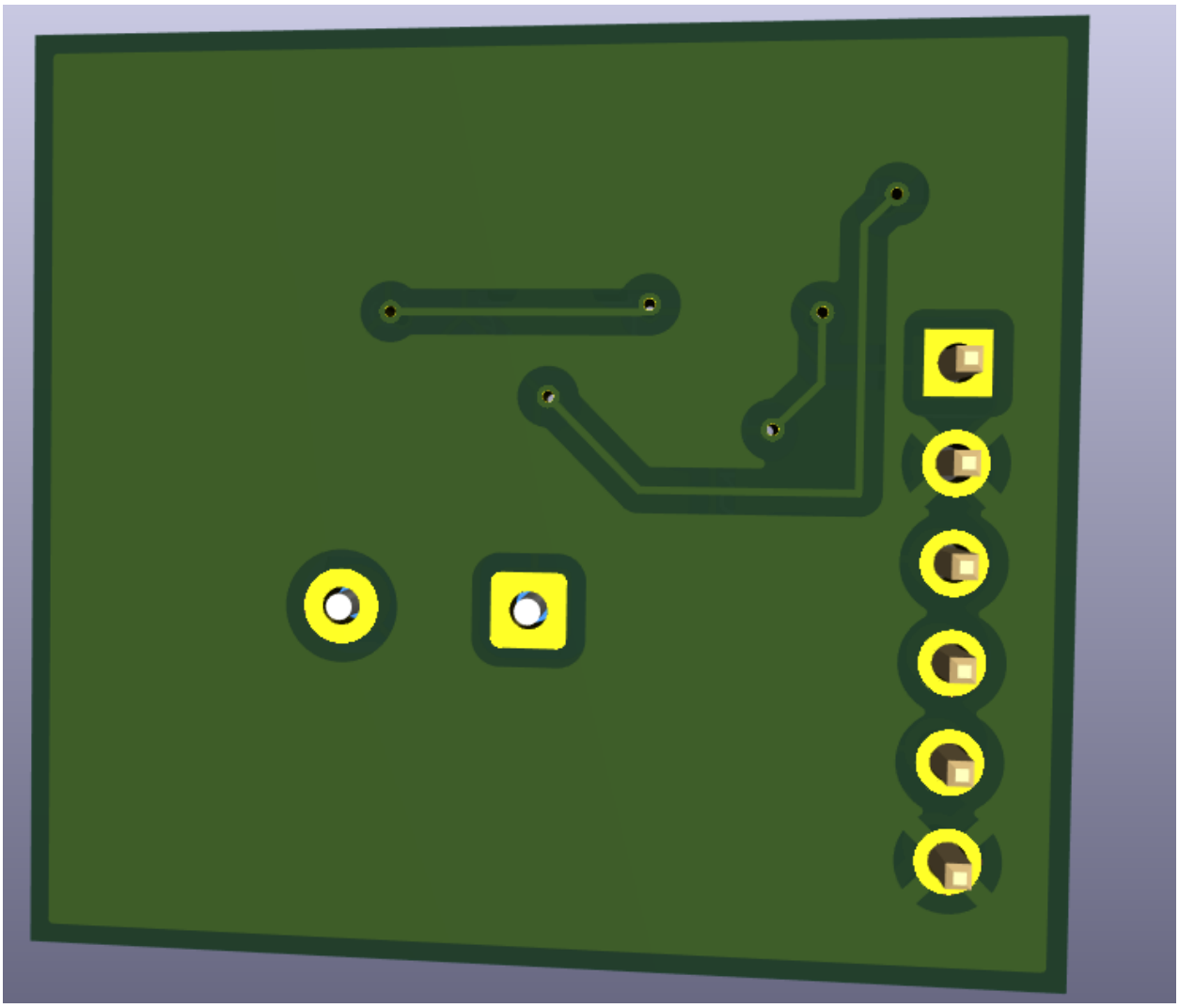


Figure 8.10 *Amplifier PCB 3D Model (Back)*

8.4 Power Distribution System

The charging module layout, shown in *Figure 8.10*, is built around the TP5100 charger and focuses on keeping the power routing clean and safe. One of the main design choices was to make the high current paths as short and wide as possible. The BAT+ and ground traces use a thicker width so they can handle the charging current without unnecessary voltage drop. Most of the components are placed very close to the TP5100 to keep the switching loop tight and stable. The large 100 microfarad capacitors are positioned right at the input and output to help smooth out the voltage during charging, while the smaller 0.1 microfarad capacitors sit right next to the TP5100 pins to filter out high frequency noise.

The routing follows a very straightforward flow. The USB input enters from the left and immediately passes through input filtering before feeding the TP5100. The BAT+ output then travels toward the right side of the board where the battery header is placed, making it easy to connect the battery wires directly. The entire board uses a copper ground fill on the top and bottom layers so all the components share a solid and low impedance ground.

This also helps spread heat while the charger is running. The charging LEDs and their resistors are placed neatly along the lower left so the user can quickly see the charging or full status.

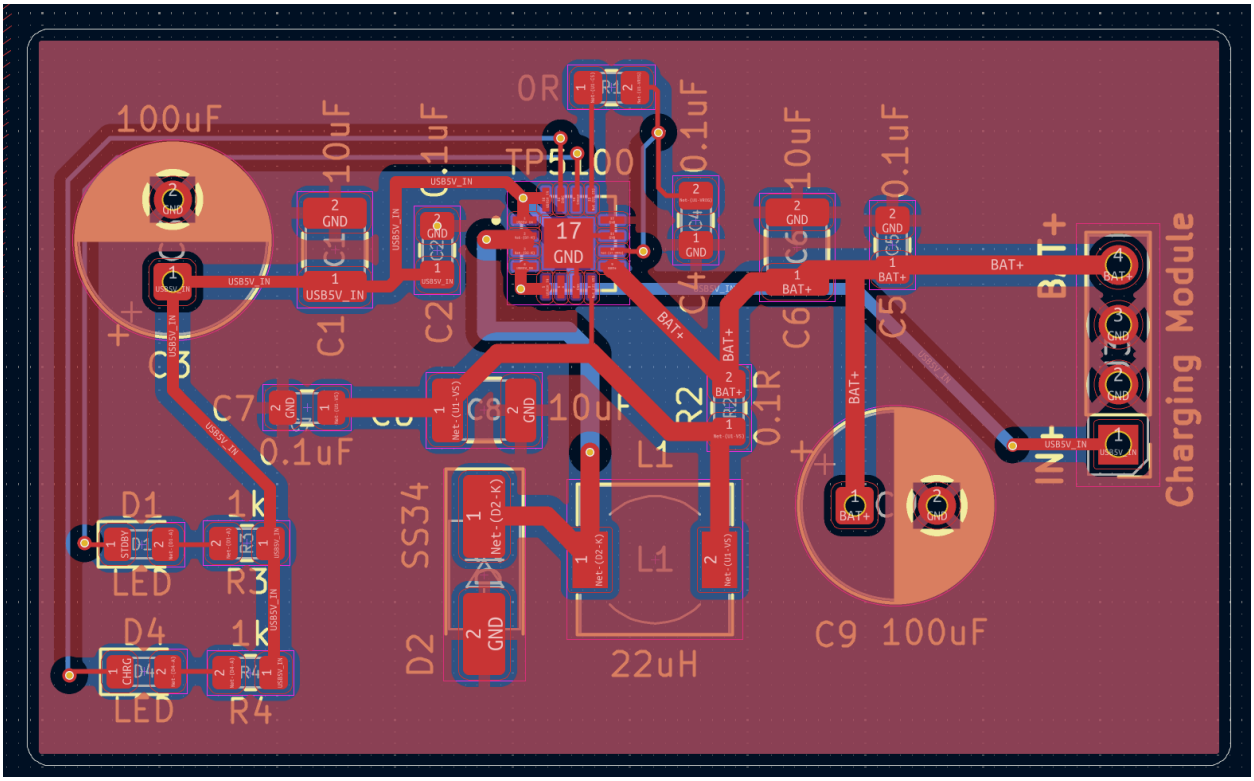


Figure 8.11 *Charging Module PCB Layout*

Figures 8.11 and 8.12 show the 3D model of the charging module. In *Figure 8.11*, you can clearly see the height and spacing of the large capacitors and how the ceramic capacitors sit closely around the TP5100. The layout becomes easier to visualize in 3D, especially the way the battery header extends outward on the right side. *Figure 8.12* shows the back side where the traces run cleanly and the extra copper fill can be seen more clearly. The 3D view highlights how everything fits together physically and how compact the final charging module really is.

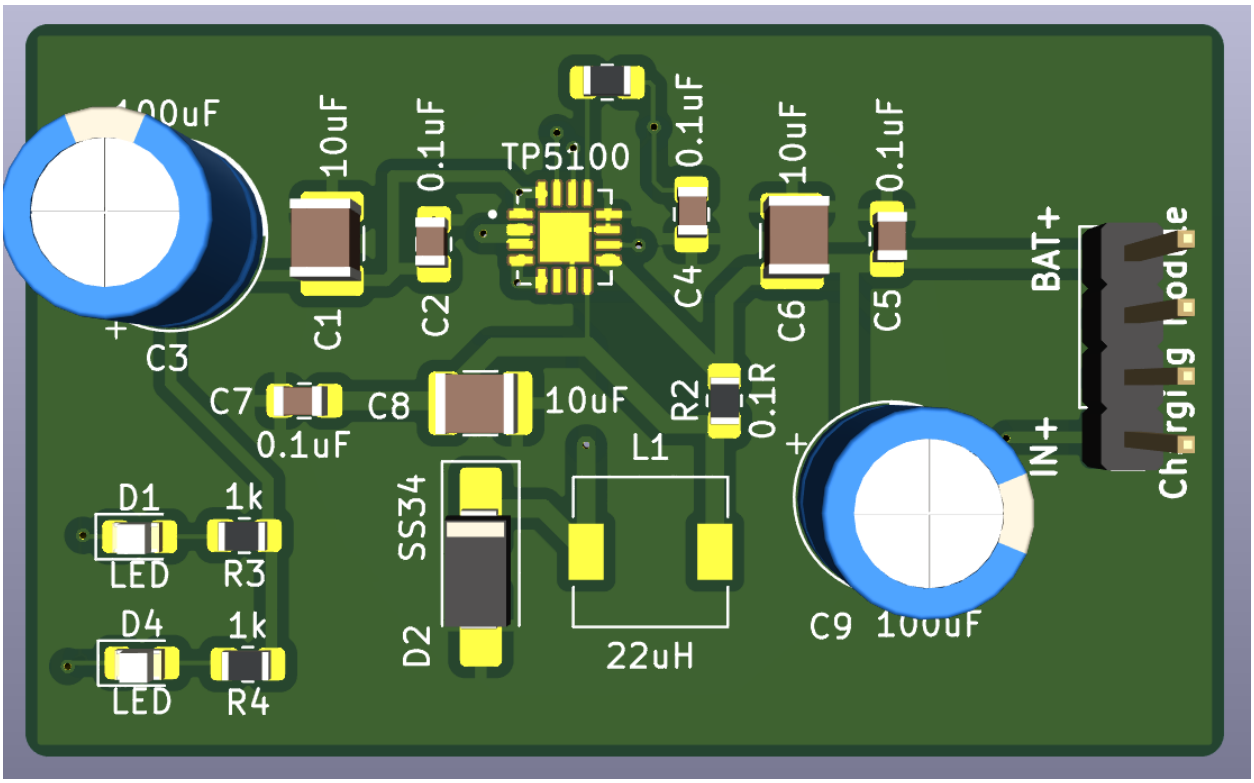


Figure 8.12 *Charging Module PCB 3D Model (Front)*

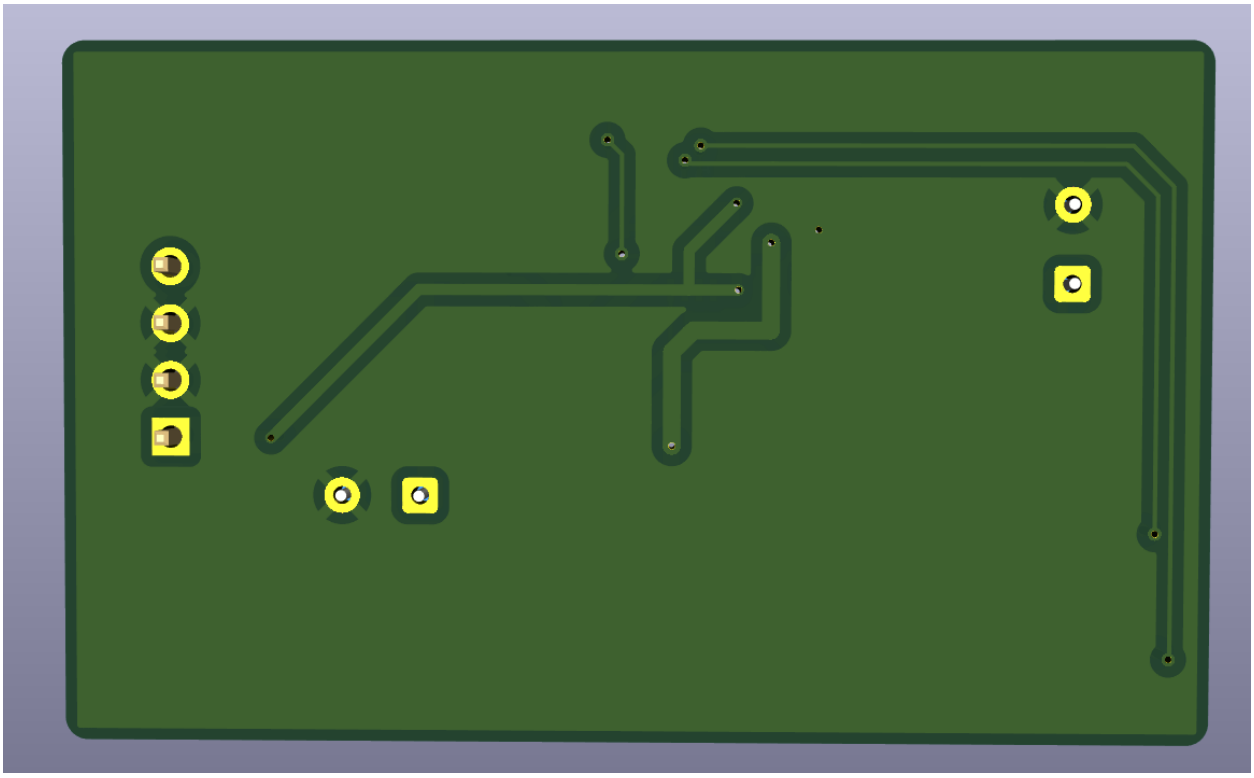


Figure 8.13 *Charging Module PCB 3D Model (Back)*

The voltage regulator board, shown in *Figure 8.13*, was designed to work for both the 3.3V and the 5V versions by only changing one feedback resistor. This made the design really flexible because instead of creating two separate boards, the same layout can be used and the output voltage is set by swapping just R1. The feedback resistor section is grouped inside its own little box on the right side of the board, so it is easy to find and change. The traces that carry Vout were kept wide and short since the regulator can deliver a decent amount of current, and I added extra vias and via stitching around the main power paths to help with thermal regulation. This helps the board spread heat more evenly, especially around the Schottky diode and the inductor.

The placement of the components follows the recommended layout for the LM2576 ADJ, so the input capacitor, diode, inductor, and output capacitor all stay close together to keep the switching loop tight. At the top of the board, I added a little heart shape that marks the voltage output label. The idea was that once the correct feedback resistor is chosen, I could write the final voltage on the board with a Sharpie marker inside the heart. Around the edges, I included tiny drawings that represent everyone in our team, which makes the board feel more personal and fun instead of just a plain technical design.

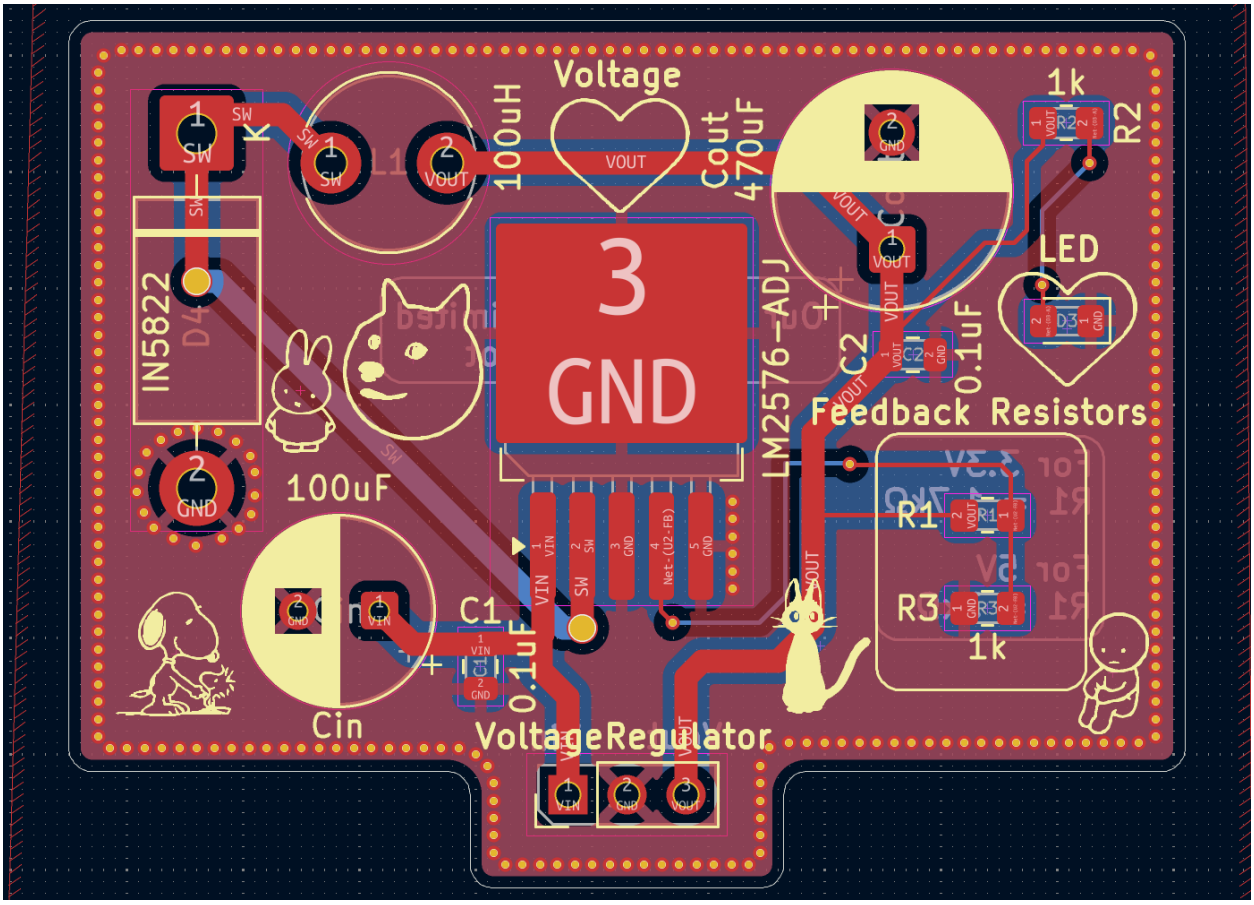


Figure 8.14 *Voltage Regulator PCB Layout*

Figures *8.14 and 8.15* show the 3D models of the regulator. In *Figure 8.14*, you can clearly see the big inductor, the Schottky diode, and the 470 microfarad output capacitor that all shape the power path. The feedback resistors sit neatly in their little marked area, easy to

spot. The 3D model also makes the decorative drawings stand out more, making the board feel very unique. I also included a small reference box that lists the resistor values needed for each voltage option so anyone using the board can quickly remember what R1 value gives 3.3V and what R1 value gives 5V. Overall, this board mixes functionality with a little bit of personality, making it both practical to use and meaningful to look at.

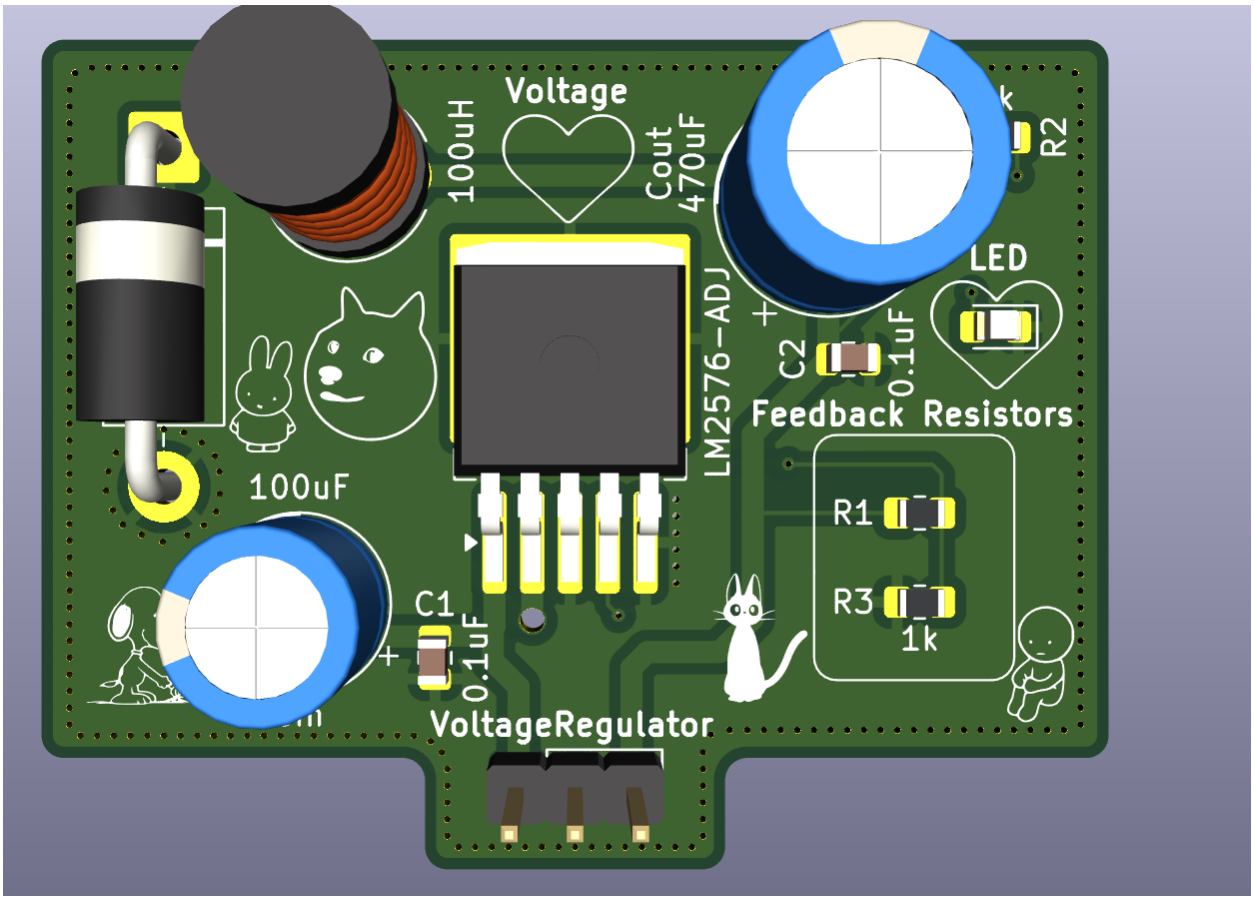


Figure 8.15 *Voltage Regulator PCB 3D Model (Front)*

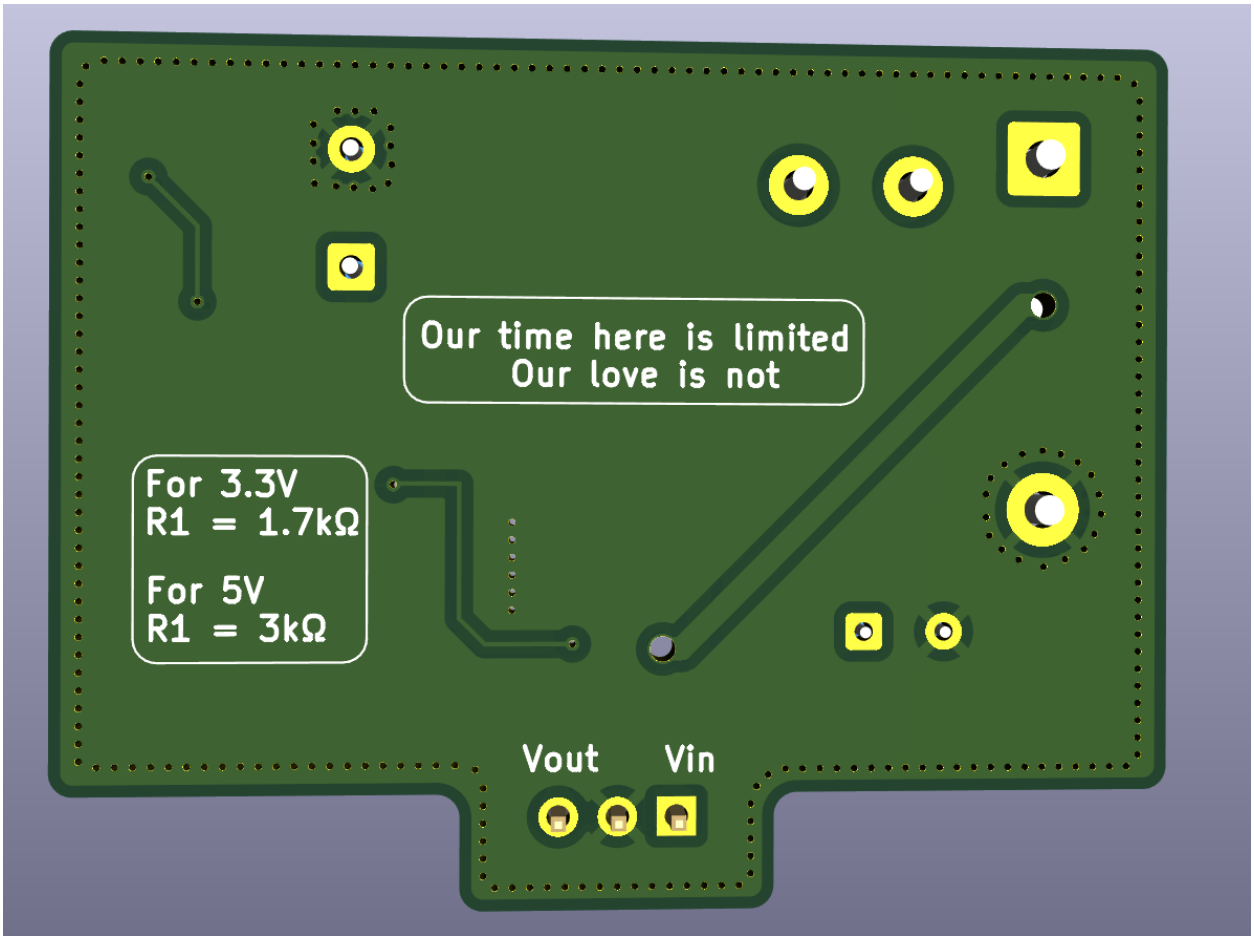


Figure 8.16 *Voltage Regulator PCB 3D Model (Back)*

8.5 Block Enclosure Diagram

This enclosure diagram designed in SolidWorks uses a two-screw hinged top that swings open like a small service door. This allows quick access to the PCB, battery, amplifier module, and wiring without disassembling the entire block.

The hinge mounts are integrated directly into the enclosure body, while the opposite side uses only two screws to keep the lid secured during normal use. Removing these screws allows the top panel to rotate upward on the hinge, giving clear access to all internal components. This significantly reduces assembly time and makes debugging, battery replacement, and wiring adjustments far easier compared to a traditional four-screw enclosure.
Inside the enclosure, standoffs match the Child PCB's mounting hole pattern, so the board remains fixed at the correct height relative to the display, I²C, LED strip, and peripheral connectors along the walls. The internal layout provides open space for the 18650 batteries, TP5100 charging board, and amplifier module, ensuring nothing obstructs the hinge mechanism or interferes with the ESP32 antenna keep-out zone.

Externally, the block retains the standard cube form factor, so it remains fully compatible with the rest of the *Blocks o' Code* ecosystem. The flat front panel allows for future

cosmetic customization, while the hinged lid provides a practical quality-of-life upgrade for testing and repairs.

8.5.1 Exterior Enclosure

Figures 8.16 and 8.17 show the front and back views of the finalized exterior enclosure, including the display cutout, LED matrix window, power switch, USB-C port placement, and magnetic pogo pin connectors.

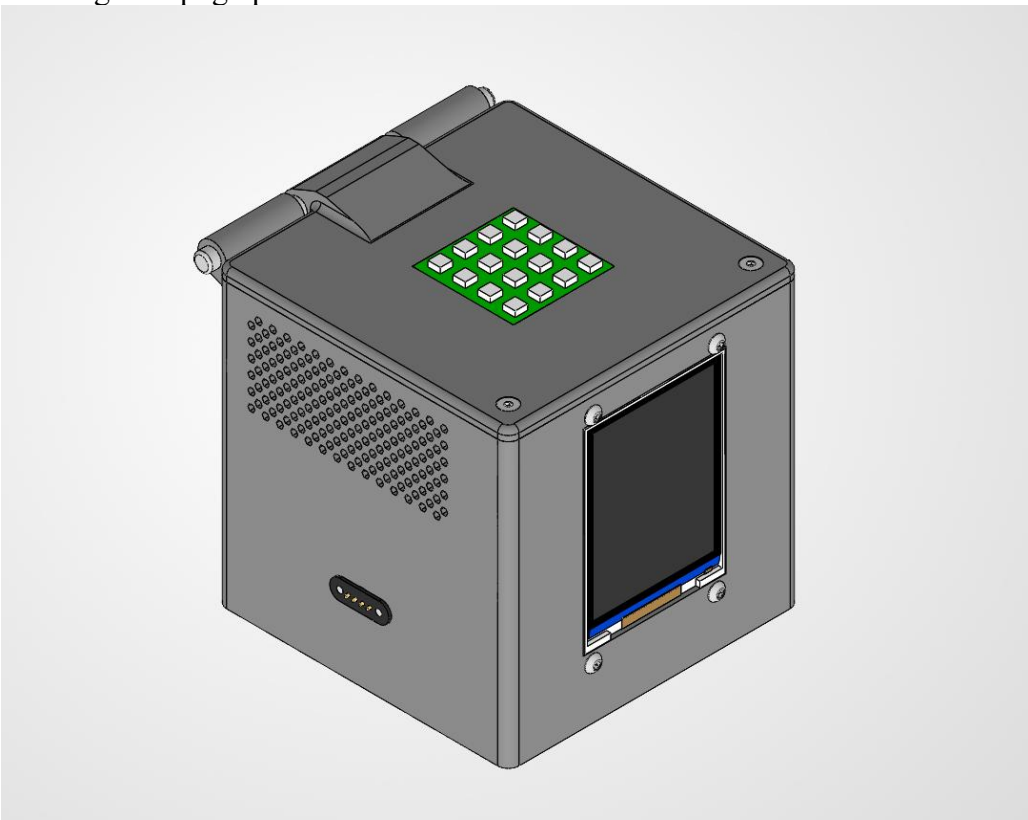


Figure 8.17 *Enclosure Exterior Front View*

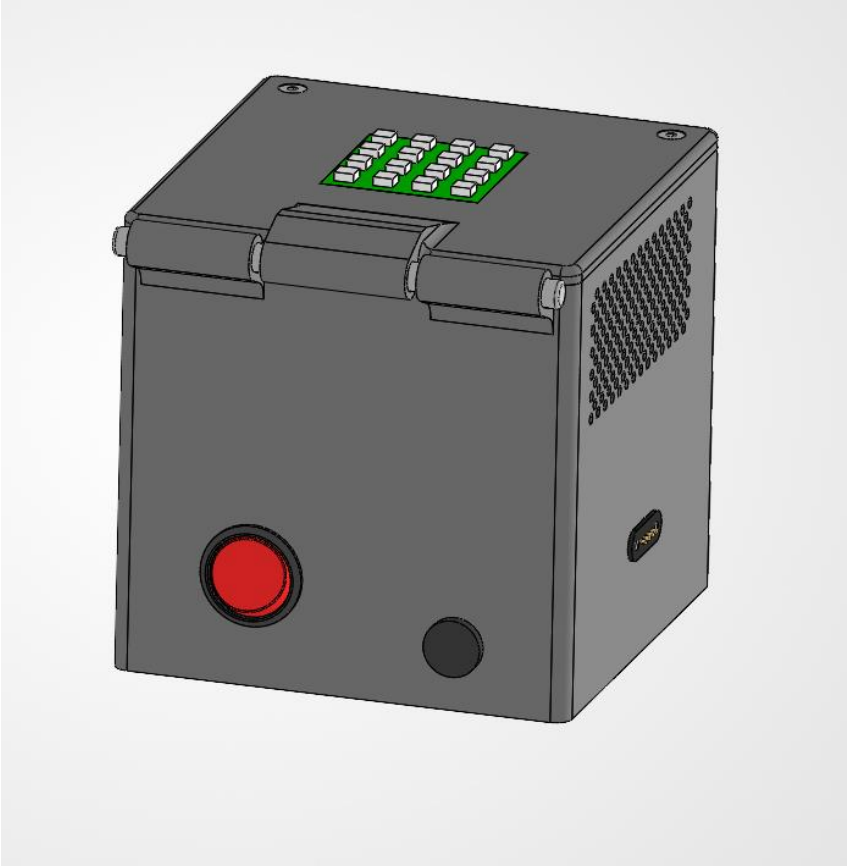


Figure 8.18 *Enclosure Exterior Back View*

8.5.2 Interior Enclosure

Figures 8.18 and 8.19 show the left and right side interior views with the hinged lid open, illustrating the PCB mounting position, internal component layout, and hinge mechanism.

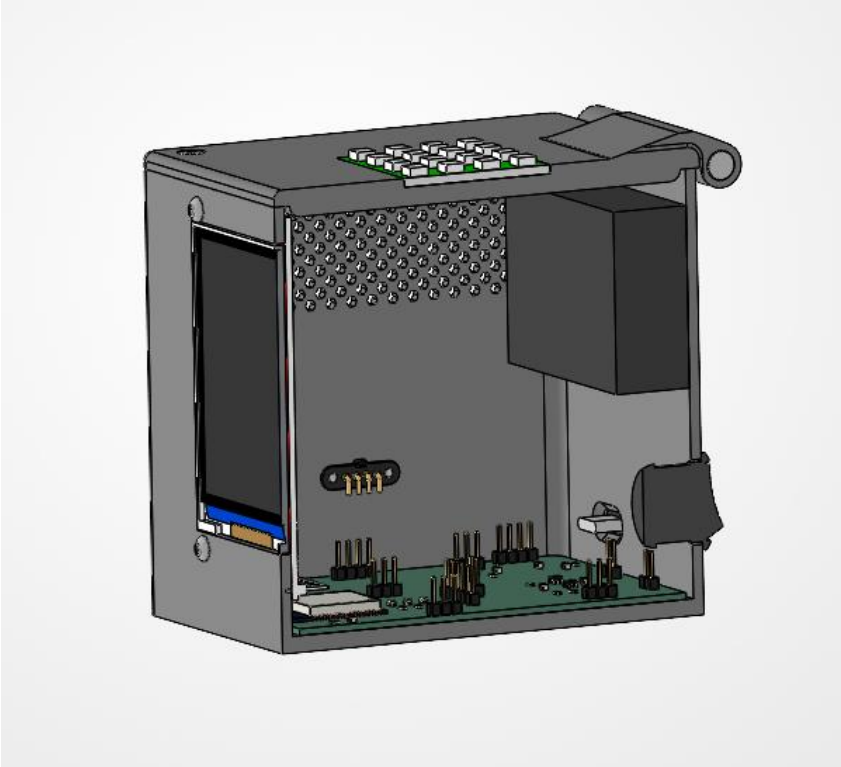


Figure 8.19 *Enclosure Interior Left Side View*

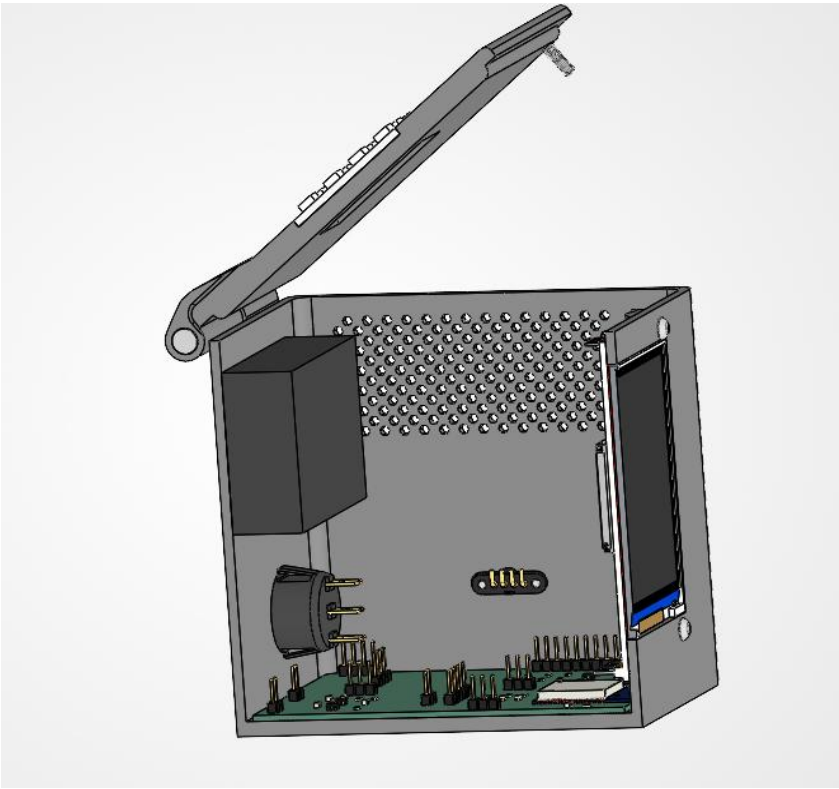


Figure 8.20 *Enclosure Interior Right Side View*

Chapter 9 – System Testing and Evaluation

9.1 Hardware Testing

9.1.1 ESP32-WROOM-32

The ESP32-WROOM-32 microcontroller served as the central processing unit for the system, where functional verification prioritized GPIO responsiveness and timer consistency. Input pins were tested for immediate reaction to physical triggers, while interrupt routines were verified to ensure they were executed without blocking the main program loop. Hardware timers were configured to manage periodic tasks, and a self-test routine was implemented to sample timer duration over ten iterations.

To ensure the reliability of time-sensitive operations, the internal clock accuracy was quantified by measuring a target interval of 1,000,000 µs (1 second). Data analysis of the self-test logs revealed a mean duration of 999,427.9 µs across the samples [170]. The measured deviation ranged from a minimum of 999,347 µs to a maximum of 999,510 µs. These results indicate a negligible drift of approximately 0.057%, confirming that the ESP32's internal clock is sufficiently accurate for the project's timing requirements without the need for an external Real Time Clock (RTC).

9.1.2 3.3V and 5V Regulator

Figure 9.1 shows the testing setup we used to verify both the 3.3V and 5V regulator boards. The top section of the breadboard is where we tested the 5V regulator, and the bottom section is where we tested the 3.3V regulator. Everything was powered using the bench power supply, which we set to 7.2V to match the expected voltage of our two-cell battery pack. Once the system was running, the circuit drew around 1.682A, which you can see on the power supply display.

For the 3.3V regulator, the ideal feedback resistor value for R1 was 1.7k, but the closest part we could find in the lab was 1.8k. Even with that small difference, the regulator still performed very well. In Figure 9.2 you can see that the output voltage stays very close to 3.3V and the current is almost exactly 1 amp under load. For the 5V version, we were able to use the exact resistor values required, and because of that, the output readings are right on target. The electronic load shows a stable 5.07V at 1A, just as expected.

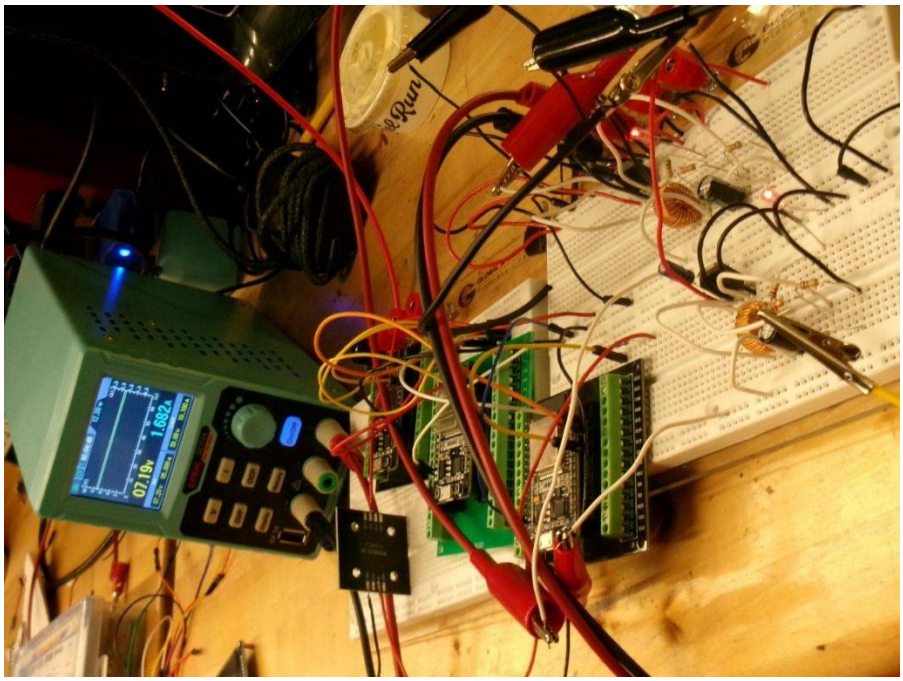


Figure 9.1 *Voltage Regulators Breadboarding*

Figure 9.2 *Electronic Load Displaying Output Voltage and Current from both regulators*

Overall, these measurements showed that both regulators behaved correctly under load, and that the small resistor difference in the 3.3V version did not cause any major deviation from the expected output.

9.1.3 LED Matrix

During the demo, the LED matrices performed really well, especially in terms of brightness and stability. In the code, both matrices were set to 75% brightness, and even with all the animations running, there was absolutely no flickering at all. The brightness stayed completely steady the whole time, which shows that the LED matrices are taking the right amount of current. They draw enough power to light up brightly, but not so much that they cause any instability in the system.

Because the current draw stayed within a safe range, we were able to run two LED matrices at the same time without any dimming or color shifts. Every pixel lit up evenly and the animations looked smooth and clean. This confirmed that the LED matrices were a good choice for our project, since they give strong visual output while still being efficient enough to work alongside the other components like the TFT display and the indicator LEDs. Overall, the matrices proved they can run reliably at the brightness level we need for clear and fun visuals.

9.2 Software Testing

9.2.1 Wi-Fi Connectivity

The integrity of the wireless communication subsystem was evaluated by configuring the ESP32 in Station mode (STA), allowing it to function as a standard network client. To simulate a realistic deployment environment, connectivity tests were conducted using a personal hotspot with the SSID "Jordan." The device successfully executed the authentication handshake to establish a secure WPA2-PSK encrypted link, ensuring unauthorized access is prevented at the link layer. Following successful association, the network interface controller (NIC) initiated the DHCP discovery process. The device successfully negotiated with the gateway to obtain the dynamic IPv4 address 172.20.10.9, confirming that the TCP/IP stack was fully initialized and ready for socket-level communication [169].

9.2.2 TCP Client-Server

To facilitate remote command and control, a custom TCP server was implemented on the ESP32. The server successfully initialized and began listening for incoming connections on port 41233 at address 0.0.0.0. Testing confirmed a successful three-way handshake with the Flutter client application. The protocol flow was validated by sending a START command from the client, which the ESP32 parsed and immediately acknowledged with an ACK:START response [171]. This feedback loop occurred near-instantaneously in the local network environment, validating the efficiency of the TCP socket implementation and the responsiveness of the application layer.

9.2.3 Flutter Application

The Flutter application was tested to evaluate UI responsiveness and socket handling on a mobile device. The interface demonstrated robust state management by dynamically updating to reflect the current network status, transitioning smoothly from "Server

listening" to "Connected" once the socket was established. The control interface was validated through the Start/Stop buttons, which triggered the correct payload transmission. Upon sending a command, the application successfully parsed the incoming UTF-8 stream from the ESP32, displaying Received from ESP32: ACK:START or ACK:STOP [171]. This confirmed that the bidirectional communication link was functional and that the application could maintain a stable UI state while awaiting socket responses.

9.3 Overall Integration

The final phase of Senior Design 1 validation focused on the full system integration, combining the independently tested power, software, and peripheral subsystems into a unified proof-of-concept. The primary goal was to demonstrate a complete communication chain from the user interface (Companion App) through the central controller (Brain Block) to multiple distributed peripherals (Child Blocks).

The integration setup utilized the custom-fabricated 3.3 V and 5 V voltage regulator boards to power the system components.

- **Power Stability:** The 3.3 V regulator successfully powered the ESP32-WROOM-32 (Brain Block), maintaining logic stability during Wi-Fi transmission bursts. The 5 V regulator simultaneously drove two separate LED matrix modules (representing two Child Blocks).

- **Visual Verification:** Status LEDs on both regulator boards provided immediate visual confirmation of active power rails, remaining illuminated and steady throughout the test, indicating no thermal shutdown or over-current events occurred under the combined load

The testing procedure verified the signal transmission integrity and functional response of the entire system hierarchy:

- **Network Connection:** The Flutter application successfully established a TCP connection with the Brain Block via Wi-Fi, transitioning the UI state from "Listening" to "Connected".

- **Command Propagation:** Upon a user input (pressing the "Start" button in the app), a control command was transmitted to the Brain Block.

- **Peripheral Response:** The Brain Block took the command and triggered the corresponding "Disco" animation routine. Both LED matrices responded immediately, displaying synchronized lighting patterns with consistent brightness and no visible flicker.

This successful demonstration verified that the power architecture is capable of supporting multiple active peripherals and that the software stack can reliably bridge high-level app commands with low-level hardware control. The integration of the App, Brain, and

multiple Child-level peripherals confirms the system's readiness for the final PCB fabrication and mechanical assembly planned for Senior Design 2.

9.4 Senior Design II Results

With the research and planning from SD1 complete, SD2 focused entirely on implementation, testing, and integration. Both Brain and Child Block PCBs were fabricated, assembled, and brought up successfully. Power subsystem validation confirmed stable 3.3 V and 5 V outputs under load, and the ESP32-WROOM-32 internal clock was verified to be sufficiently accurate for all timing-sensitive operations. Firmware was implemented under ESP-IDF with FreeRTOS, establishing reliable I²C communication between the Brain Block and all connected Child Blocks. Wi-Fi connectivity and TCP client-server communication were validated, and the Flutter companion app was extended to include real-time 3D block chain visualization and configuration validation with a measured end-to-end latency of 50 ms. LED matrix animations ran smoothly with no flicker under concurrent load. Full system integration was demonstrated with the complete communication chain — from app command through the Brain Block to synchronized LED output — operating reliably across a 15-block configuration.

Chapter 10 – Administrative Content

10.1 Budget

The total projected cost for the *Blocks o' Code (v3)* prototype is **$823.90**, which falls comfortably under the approved **$1,000 development budget cap** supported by our sponsor, Dr. Mike Borowczak. This budget covers all essential components required to construct one Brain Block and multiple Child Blocks, including microcontrollers, displays, lighting, power modules, and interconnection hardware.

A summary of the component breakdown is shown in the following figures.

All listed components are readily available through major distributors such as **Digi-Key**, **Adafruit**, and **Amazon**, ensuring procurement feasibility and replacement availability.

The system's most significant cost driver is the **power subsystem**, particularly the rechargeable 18650 lithium-ion cells and charging hardware, which collectively account for approximately 36% of the total budget. This cost is justified by the project's portability requirements and the need for reliable, safe, and rechargeable power sources in classroom environments.

Because the *Blocks o' Code (v3)* prototype is still in development, additional funding within the $1,000 cap remains reserved for **testing materials, 3D-printed enclosures, and contingency replacements**. This margin ensures flexibility during debugging and integration while keeping total expenses within the sponsor-approved limit.

10.2 Bill of Materials

10.2.1 Prototype

Item	Qty	Manufacturer	MPN	Supplier	Supplier P/N	Unit Price (USD)	Ext. Price
ESP32-WROOM-32 (MCU)	15	Espressif Systems	ESP32-WROOM-32-N4	Digi-key	B0CDRLRYJB	$6.56	$98.04
18650 lithium-ion cell batteries	30	Ultralast	N/A	Ace Hardware	UL1865-26-2P	$19.99	$299.85
TP5100 (charging module)	15	DKARDU	JK-US-386	Amazon	B09TKNFB3W	$9.99	$29.97
Type-C Female Chassis	20	Coliao	687117711298	Amazon	B0D813FX3J	$8.99	$8.99
SPI TFT LCD Display Touch Panel	1	HiLetgo	3-01-1433	Amazon	B073R7BH1B	$16.39	$16.39
4x4 RGB WS2812B LED Matrix (5 Pcs)	2	GODIYMODULES	N/A	Amazon	B0F59HC7PT	$10.99	$21.98
0.96" SSD1306 128X64 OLED SPI LCD Display	10	HiLetgo	3-01-1234-YB-AU-1PC	Amazon	B01N0KIXJ6	$6.99	$69.99
1602 I2C LCD Display	5	Hosyond	N/A	Amazon	B0BWTFN9WF	$11.99	$23.98
WS2812B IC RGB Addressable LEDs (300 ft)	2	BTF-LIGHTING Technology Co., Limited	WS2812B5M60LW67	Amazon	B01CDTEID0	$29.99	$59.98
Discrete LED 0805 (Red)	15	ams-OSRAM USA INC.	LS R976-NR-1-0-20-R18	Digi-Key	475-LSR976-NR-1CT-ND	$0.15	$2.25
Discrete LED 0805 (Blue)	15	Würth Elektronik	150080BS75000	Digi-Key	732-4982-1-ND	$0.19	$2.85
Discrete LED 0805 (Green)	15	ams-OSRAM USA INC.	LG R971-KN-1-0-20-R18	Digi-Key	475-LGR971-KN-1CT-ND	$0.15	$2.25
Same Sky (CUI) CMS-2580-05SP (speakers)	4	Same-Sky	CMS-2850-058SP	Digi-Key	2223-CMS-2850-058SP-ND	$2.13	$8.52
PAM8302A (Amplifier)	5	JESSINIE	PAM8302	Amazon	B0BG2F3LMP	$9.95	$9.95
MAX98357A (Amplifier)	4	GODIYMODULES	MAX98357A	Amazon	B0DPJRLMDJ	$6.88	$13.76
Adafruit I²S MEMS Microphone (SPH0645LM4H)	3-4	Adafruit	SPH0645LM4H	Adafruit	SPH0645LM4H	$6.95	$27.80
Mill-Max 858-22-004-10-011101 (POGO pins)	15	Mill-Max Manufacturing Corp.	858-22-004-10-011101	Digi-Key	ED11072-ND	$8.49	$127.35
						Total:	$823.90

Table 10.2.1 *BOM Prototype*

10.2.2 Manufacturer Level

Item	Qty	Manufacturer	MPN	Supplier	Supplier P/N	Unit Price (USD)	Ext. Price
ESP32-WROOM-32 (MCU)	30	Espressif Systems	ESP32-WROOM-32-N4	Digi-key	B0CDRLRYJB	$6.56	$196.80
18650 lithium-ion cell batteries	60	Ultralast	N/A	Ace Hardware	UL1865-26-2P	$19.99	$1,199.40
TP5100 (charging module)	30	DKARDU	JK-US-386	Amazon	B09TKNFB3W	$9.99	$299.70
Type-C Female Chassis	40	Coliao	687117711298	Amazon	B0D813FX3J	$8.99	$359.60
SPI TFT LCD Display Touch Panel	1	HiLetgo	3-01-1433	Amazon	B073R7BH1B	$16.39	$16.39
4x4 RGB WS2812B LED Matrix (5 Pcs)	10	GODIYMODULES	N/A	Amazon	B0F59HC7PT	$10.99	$109.90
0.96" SSD1306 128X64 OLED SPI LCD Display	20	HiLetgo	3-01-1234-YB-AU-1PC	Amazon	B01N0KIXJ6	$6.99	$139.80
1602 I2C LCD Display	10	Hosyond	N/A	Amazon	B0BWTFN9WF	$11.99	$119.90
WS2812B IC RGB Addressable LEDs (300 ft)	4	BTF-LIGHTING Technology Co., Limited	WS2812B5M60LW67	Amazon	B01CDTEID0	$29.99	$119.96
Discrete LED 0805 (Red)	30	ams-OSRAM USA INC.	LS R976-NR-1-0-20-R18	Digi-Key	475-LSR976-NR-1CT-ND	$0.15	$4.50
Discrete LED 0805 (Blue)	30	Würth Elektronik	150080BS75000	Digi-Key	732-4982-1-ND	$0.19	$5.70
Discrete LED 0805 (Green)	30	ams-OSRAM USA INC.	LG R971-KN-1-0-20-R18	Digi-Key	475-LGR971-KN-1CT-ND	$0.15	$4.50
Same Sky (CUI) CMS-2580-05SP (speakers)	8	Same-Sky	CMS-2850-058SP	Digi-Key	2223-CMS-2850-058SP-ND	$2.13	$17.04
PAM8302A (Amplifier)	10	JESSINIE	PAM8302	Amazon	B0BG2F3LMP	$9.95	$99.50
MAX98357A (Amplifier)	8	GODIYMODULES	MAX98357A	Amazon	B0DPJRLMDJ	$6.88	$55.04
Adafruit I²S MEMS Microphone (SPH0645LM4H)	8	Adafruit	SPH0645LM4H	Adafruit	SPH0645LM4H	$6.95	$55.60
Mill-Max 858-22-004-10-011101 (POGO pins)	30	Mill-Max Manufacturing Corp.	858-22-004-10-011101	Digi-Key	ED11072-ND	$8.49	$254.70
						Total:	$3,058.03

Table 10.2.2 *BOM Manufacturer level*

10.3 Distribution of Worktable

Discipline	Member	Responsibilities
Computer Engineering	**Jordan Merkel**	Companion App development using Flutter; JSON data handling and visualization; Wi-Fi communication interface between Brain Block and mobile app; documentation of software architecture.
Electrical Engineering	**Camilla Torres**	Power subsystem design and implementation, including battery selection, charging circuitry, and power management; PCB layout coordination for Brain Block.
Electrical Engineering	**Annesley Kolb**	Audio subsystem design and implementation, amplifier selection, and output filtering; mechanical CAD design and enclosure modeling for the Blocks; PCB layout coordination for Child Blocks.
Computer Engineering	**Destiny Ellenwood**	Firmware design and implementation for Brain and Child Blocks; FreeRTOS-based task scheduling and communication protocols (I²C, SPI, and Wi-Fi); software documentation, testing, and synchronization of system-level timing.

Table 10.3: *Worktable*

10.4 Milestones

Project Milestones (Senior Design 1)

Milestone number	Milestone Description	Start Date	End Date (estimated)	Advisors and requirements
1	Brainstorming	08/27/2025	09/04/2025	Meeting virtually to pitch ideas
2	Sponsor Meeting	08/27/2025	08/27/2025	Discuss project requirements with Dr. Mike and various ideas

3	Initial Document	08/26/2025	09/05/2025	First 10 pages of Divide and Conquer
4	Advisor Meeting	09/08/2025	09/08/2025	First meeting with Dr. Sundaresh reviewing Divide & Conquer paper and feedback
5	Revised Document	09/08/2025	09/12/2025	Final revision of Divide and Conquer paper
6	Sponsor Meeting	09/22/2025	09/22/2025	Reviewing project progress with Dr. Mike and Dr. Weeks for feedback
7	Reviewer Meeting	10/8/2025	10/8/2025	Meet with Dr. Mike for feedback and order parts
8	60-Page Milestone	10/7/2025	10/31/2025	First half of final document finished
9	Midterm report meeting	11/4/2025	11/4/2025	Meeting with Dr. Sundaresh for midterm report
10	Final Document	08/26/2025	11/25/2025	Final document of SD1 completed
11	Mini Demo Video	11/20/2025	11/25/2025	Demo of prototype

Table 10.4.1: *Milestones for SD1*

Project Milestones (Senior Design 2)

Milestone number	Milestone Description	Start Date	End Date (estimated)	Deliverables and requirements
1	PCB Fabrication & Procurement	01/12/2026	01/26/2026	Submit Gerber files to PCB manufacturer for Brain and Child Blocks; Order remaining BOM components (Digi-Key/Mouser).
2	Hardware Assembly (Phase 1)	01/27/2026	02/03/2026	Solder and assemble 1 Brain Block and 2 Child Blocks for

				initial "smoke testing" and power rail validation.
3	Firmware Implementation: Core Logic	01/12/2026	02/05/2026	Implement full I²C device registry, collision detection, and stable Wi-Fi telemetry with the Companion App.
4	Subsystem Integration: Audio & Visual	02/06/2026	02/20/2026	Integrate synchronized LED Matrix animations and Audio (PWM/DAC) output with the main control loop.
5	Mechanical Integration & Enclosures	02/21/2026	03/05/2026	3D print final PETG enclosures; install magnetic pogo-pin connectors; verify mechanical fit and durability.
6	Full System Scaling (15 Blocks)	03/06/2026	03/15/2026	Assemble remaining Child Blocks; test power stability and signal integrity with the full 15-block chain.
7	Final System Validation	03/22/2026	04/08/2026	Comprehensive latency testing, battery life verification, and "User Experience" testing with the app.
8	Final Demo & Documentation	04/09/2026	04/24/2026	Final presentation, demo video production, and submission of the SD2 Final Report.

Table 10.4.2: *Milestones for SD2*

Chapter 11 – Conclusion

The completion of Senior Design 2 for *Blocks o' Code (v3)* marks the realization of a project that began as a conceptual idea and evolved into a functional, tested, and integrated educational platform. Across both semesters, the team progressed from foundational research and architecture planning in senior design 1 to full hardware fabrication, firmware implementation, and system integration in senior design 2. This chapter summarizes the accomplishments of both semesters, reflects on the technical challenges encountered and resolved along the way, and offers closing remarks on the project's impact and potential.

11.1 Summary of Accomplishments

The primary achievement of SD1 was defining the complete system concept for *Blocks o' Code (v3)* at a detailed engineering level. Early in the semester, the team refined the project vision: creating a modular, magnetically connected block-based programming system capable of generating synchronized audio-visual behaviors. Building upon the lessons from prior versions of the project, Version 3 expands both complexity and capability by incorporating a distributed network of microcontrollers, I²C-based communication, real-time data exchange, and multimedia output.

Subsystem research and selection formed a major portion of SD1's workload. Every core component including but not limited to microcontrollers, power delivery hardware, regulators, displays, sensors, connectors, amplifiers, LEDs, speakers, charging ICs, and storage technologies were evaluated in terms of cost, performance, power consumption, manufacturability, and long-term reliability. These selections represent not only the preferred components, but also the reasoning and tradeoffs behind each decision, ensuring that future design revisions can be made without repeating early-stage research.

Beyond component selection, the team also defined the entire hardware architecture for both Brain and Child Blocks. This includes the communication topology (I²C master-slave structure), power distribution strategy, PCB layouts, subsystem interfaces, boot and programming considerations, and the mechanical integration of 3D-printed enclosures with magnetic connectors. On the software side, the team developed a complete firmware architecture and companion app framework, outlining state machines for both block types, I²C data formats, timing requirements for synchronized audio-light effects, and error-detection mechanisms. SD1 concluded with a fully documented engineering plan that gave the team a unified blueprint entering SD2.

SD2 transformed that blueprint into a working system. The Brain Block and Child Block PCBs were fabricated and assembled, along with dedicated module boards for the LM386 amplifier, TP5100 charging circuit, and LM2576T-ADJ voltage regulators. Both the 3.3 V and 5 V regulator boards were verified under load, producing stable outputs within specification. Firmware was developed under the ESP-IDF framework with FreeRTOS, standardized across all blocks using the ESP32-WROOM-32. I²C communication was established at 100 kHz with statically assigned Child Block addresses (0x08–0x16). Wi-Fi connectivity was validated using WPA2-PSK authentication and DHCP negotiation, and a

TCP server on port 41233 successfully handled bidirectional communication with the Flutter companion app. The app was expanded to include a 3D block chain visualization and configuration validation, with a measured end-to-end config-change latency of 50 ms across a full 15-block chain. The WS2812B LED matrices delivered smooth, flicker-free animations at 10% brightness under concurrent load. A full system integration test confirmed the complete communication chain, from app command to Brain Block to synchronized LED output resulting in it operating reliably.

11.2 Technical Insights and Design Challenges

One of the most valuable outcomes of SD1 was identifying major technical challenges before implementation began. SD2 validated those concerns and demonstrated how early preparation shaped the final design.

Reliable I²C communication across a modular, user-assembled system was the most fundamental challenge. Unlike a fixed PCB with permanent traces, pogo-pin connectors introduce variability, contact wear, and potential noise. In SD2, the team addressed bus addressing by statically assigning I²C addresses at compile time, eliminating the need for EEPROM pre-programming and simplifying bring-up considerably. Firmware-level timeout handling and retry logic ensured the system could recover from momentary connection drops without user intervention.

Power distribution required careful attention across multiple blocks, each with its own microcontroller and peripherals. The LM2576T-ADJ buck regulator design using a single configurable board for both voltage rails proved effective in practice. Testing confirmed stable 3.3 V and 5 V outputs under load, with no thermal or over-current events observed during extended operation. The two-cell 18650 battery pack provided adequate runtime for the system's active load profile.

Audio integration required a mid-project adjustment. The Class-D amplifier paths explored in SD1 were replaced with the LM386 low-voltage amplifier following prototype discussions, prioritizing simplicity and ease of debugging over efficiency. This decision reduced PCB complexity and enabled faster integration with the ESP32's DAC output, while still meeting the system's acoustic requirements.

Mechanical design presented its own layer of complexity. The final enclosure adopted a two-screw hinged lid rather than a traditional four-screw assembly, significantly reducing maintenance time and making internal access practical during testing and iteration. PCB standoffs were aligned to the Child Block mounting hole pattern, ensuring consistent height and connector alignment across all assembled units.

Software reliability was addressed through FreeRTOS task structures, hardware watchdog timers, state machine-based error recovery, and configuration validation logic in both the firmware and the companion app. These mechanisms ensured that invalid block sequences were caught and communicated to the user before execution, and that a faulting block could recover without requiring a full power cycle.

11.3 Closing Remarks

The completion of Senior Design 2 marks the full realization of *Blocks o' Code (v3)*. It was a journey that began with research and planning and culminated in a functional, hands-on educational platform. What started as a documented vision in SD1 has been brought to life through PCB fabrication, firmware development, mechanical assembly, and full system integration. The working prototype now demonstrates tangible block-based programming, synchronized multimedia interactions, and the seamless blend of physical and digital learning that the project set out to achieve.

Blocks o' Code (v3) represents more than a senior capstone. It is a meaningful contribution to the field of educational technology. By giving young learners a tactile, engaging way to explore programming concepts, the platform lowers the barrier to computational thinking and makes coding accessible before a screen ever enters the picture. The embedded systems engineering beneath the surface models the kind of thoughtful, constraint-driven design that defines real-world product development.

The team is proud of what has been built and confident in its potential. With the right support, *Blocks o' Code (v3)* could reach classrooms, makerspaces, and homes. Sparking curiosity and building foundational skills in the next generation of engineers, designers, and problem solvers. The work done across both semesters has not only produced a prototype worth using, but a foundation worth building on.

Appendices

Appendix A - References

- [1] Erebus Labs, “Blocks o’ Code V1 Worksheet.” [Online]. Available: https://github.com/erebus-labs/blocks-o-code/blob/master/Learning_Materials/Generic/HOWTO_WORKSHEET.pdf
- [2] University of Central Florida, “Senior Design Group 13: Blocks o’ Code V2.” [Online]. Available: https://www.ece.ucf.edu/seniordesign/fa2023sp2024/g13/
- [3] MatataLab, “Musician Add-On.” [Online]. Available: https://matatalab.com/en/musician-add-on
- [4] Cubroid, “Cubroid Coding Blocks.” [Online]. Available: https://www.cubroid.com/cubroid-coding-blocks
- [5] Tynker, “Tynker Toolbox: The Sound Blocks.” [Online]. Available: https://www.tynker.com/blog/tynker-toolbox-the-sound-blocks/
- [6] LEGO Education, “SPIKE Prime and SPIKE Essential.” [Online]. Available: https://education.lego.com/en-us/products/lego-education-spike-essential/
- [7] Modular Robotics, “Cubelets Robot Blocks.” [Online]. Available: https://www.modrobotics.com/cubelets/
- [8] Primo Toys, “Cubetto.” [Online]. Available: https://www.primotoys.com/cubetto/
- [9] MIT Media Lab, “Scratch.” [Online]. Available: https://scratch.mit.edu
- [10] Google Developers, “Blockly.” [Online]. Available: https://developers.google.com/blockly
- [11] J. Flynt, “Should You Print with PETG Instead of PLA?,” 3D Insider, Jul. 10, 2022. [Online]. Available: https://3dinsider.com/pla-vs-petg/
- [12] GoEngineer, “How to Make a Cube in SOLIDWORKS,” Goengineer.com, 2019. [Online]. Available: https://www.goengineer.com/blog/how-to-make-a-cube-in-solidworks
- [13] M. S. Horn and R. J. K. Jacob, “Designing Tangible Programming Languages for Classroom Use,” in Proc. Tangible and Embedded Interaction Conf. (TEI ’07), 2007, pp. 159–162.
- [14] P. Wyeth, “How young children learn to program with sensor, action, and logic blocks,” J. Learn. Sci., vol. 17, no. 4, pp. 517–550, 2008.
- [15] D. H. Rose and A. Meyer, Teaching Every Student in the Digital Age: Universal Design for Learning. Alexandria, VA: ASCD, 2002.
- [16] Texas Instruments, “A Basic Guide to I²C”, Application Report SBAA565, Nov. 2022. [Online]. Available: https://www.ti.com/lit/pdf/SBAA565. Accessed: Oct. 8, 2025
- [17] Raspberry Pi Ltd., “RP2040 Datasheet”, Rev. 3.4, 2024. [Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Accessed: Oct. 8, 2025
- [18] Analog Devices, “Introduction to SPI Interface”, Technical Article, 2023. [Online]. Available:https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html. Accessed: Oct. 8, 2025

- [19] Espressif Systems, "ESP32-C3 Datasheet", Rev. 1.3, 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf. Accessed: Oct. 8, 2025
- [20] Atmel (Microchip), ATmega328P 8-bit AVR Microcontroller Datasheet, 7810D–AVR–01/15, 2015. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- [21] Espressif Systems, "ESP Product Selector," *Espressif Products*, [Online]. Available: https://products.espressif.com/#/product-selector?language=en&names=. Accessed: Oct. 8, 2025.
- [22] Texas Instruments, "Introduction to the Serial Peripheral Interface (SPI)", Application Report SBAA565, Nov. 2022. [Online]. Available:https://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf?ts=1759924050684.
- [23] Mouser Electronics, "ESP32-C3 Ultra-Low Power SoCs — Espressif Systems," *Mouser*, [Online]. Available: https://www.mouser.com/ProductDetail/Espressif-Systems/ESP32-C3?qs=iLbezkQI%252BsiMXe7tMG%252Bo%2FA%3D%3D. Accessed: Oct. 8, 2025.
- [24] Espressif Systems, "ESP32-S3 — Wi-Fi & BLE 5 SoC," *Espressif*, [Online]. Available: https://www.espressif.com/en/products/socs/esp32-s3
- [25] Espressif Systems, *ESP32-S3 Series Datasheet*, version 2.0, Jul. 9, 2021. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf
- [26] Espressif Systems, "ESP32-S3," *Digi-Key Electronics*, [Online]. Available: https://www.digikey.com/en/products/detail/espressif-systems/ESP32-S3/15822445. Accessed: Oct. 8, 2025.
- [27] Raspberry Pi, "SC0914(7) – ARM® Cortex®-M0+ Microcontroller IC," *Digi-Key Electronics*, [Online]. Available: https://www.digikey.com/en/products/detail/raspberry-pi/SC0914-7/14306009 Accessed: Oct. 8, 2025.
- [28] Raspberry Pi, "RP2040 microcontroller," Raspberry Pi, [Online]. Available: https://www.raspberrypi.com/products/rp2040/. Accessed: Oct. 8, 2025.
- [29] Microchip Technology, "ATMEGA328PB-AN 8-bit Microcontroller," Mouser Electronics, [Online]. Available: https://www.mouser.com/ProductDetail/Microchip-Technology/ATMEGA328PB-AN?qs=jy4bLUHv09gDoS2J01KCIw%3D%3D. Accessed: Oct. 8, 2025.
- [30] Adafruit Industries, "Adafruit I2S MEMS Microphone Breakout — SPH0645LM4H," *Adafruit Industries*, [Online]. Available: https://www.adafruit.com/product/3421. Accessed: Oct. 9, 2025.
- [31] Meta Platforms, "React Native," React Native, [Online]. Available: https://reactnative.dev/. Accessed: Oct. 8, 2025.

- [32] Meta Platforms, “React Native: A framework for building native applications using React,” GitHub, [Online]. Available: https://github.com/facebook/react-native. Accessed: Oct. 9, 2025.
- [33] Norbert Kamienski, “React Native Pros and Cons For App Development in 2025,” Pagepro, Jul. 4, 2025. [Online]. Available: https://pagepro.co/blog/react-native-pros-and-cons/. Accessed: Oct. 9, 2025.
- [34] Unity Technologies, “Mobile game development software & engine,” Unity, [Online]. Available: https://unity.com/solutions/mobile. Accessed: Oct. 9, 2025.
- [35] Microsoft, “Unity — .NET Real-Time Development Platform,” Microsoft .NET, [Online]. Available: https://dotnet.microsoft.com/en-us/apps/games/unity. Accessed: Oct. 9, 2025.
- [36] Google, “Flutter — Build apps for any screen,” Flutter, [Online]. Available: https://flutter.dev/. Accessed: Oct. 9, 2025.
- [37] Espressif Systems, “ESP IoT Development Framework,” Espressif, [Online]. Available: https://www.espressif.com/en/products/sdks/esp-idf. Accessed: Oct. 9, 2025.
- [38] Arduino, “Arduino IDE,” Arduino, [Online]. Available: https://www.arduino.cc/en/software/#ide. Accessed: Oct. 9, 2025.
- [39] Raspberry Pi Ltd., “Raspberry Pi Pico SDK Doxygen Documentation,” *Raspberry Pi Documentation*, [Online]. Available: https://www.raspberrypi.com/documentation/pico-sdk/index_doxygen.html. Accessed: Oct. 9, 2025.
- [40] PUI Audio, Inc., “AS03608MR-10-R Data Sheet,” [Online]. Available: https://docs.google.com/gview?url=https://api.puiaudio.com/filename/AS03608MR-10-R.pdf&embedded=true [Accessed: Oct. 30 2025].
- [41] Analog Devices, “MAX98357A / MAX98357B — Tiny, Low-Cost, PCM Class D Amplifier with Class AB Performance,” Rev. 13, July 29, 2019. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/max98357a-max98357b.pdf [Accessed: Oct. 30 2025].
- [42] Texas Instruments, “TPA2028D1 3-W Mono Class-D Audio Amplifier With Fast Gain Ramp SmartGain™ AGC/DRC,” Rev. C, Jan. 2010 (revised Oct. 2015). [Online]. Available: https://www.ti.com/lit/ds/symlink/tpa2028d1.pdf [Accessed: Oct. 30 2025].
- [43] Analog Devices, Inc., “Bridge-Tied Load (BTL),” *Resources → Glossary*, Accessed: Oct. 30, 2025. [Online]. Available: https://www.analog.com/en/resources/glossary/bridge-tied_load.html
- [44] Texas Instruments, TAS2562 6.1-W Boosted Class-D Audio Amplifier with IV Sense, datasheet, doc. no. SLASEI7A, Rev. A, Dec. 2018. [Online]. Available: https://www.ti.com/lit/ds/symlink/tas2562.pdf. Accessed: Oct. 30, 2025.
- [45] Texas Instruments, “How TI Smart Amp technology enables high quality audio with built-in advanced speaker protection,” *Video*, Dec. 17, 2020. [Online]. Available: https://www.ti.com/video/6216968196001. Accessed: Oct. 30, 2025.
- [46] GeeksforGeeks, “What is Serial Peripheral Interface (SPI)?,” Electronics Engineering, Jan. 2024. [Online]. Available: https://www.geeksforgeeks.org/electronics-engineering/what-is-serial-peripheral-interface-spi. Accessed: Oct. 9, 2025.

- [47] Bluetooth SIG, “Bluetooth Low Energy Technology Overview,” 2023. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. Accessed: Oct. 9, 2025.
- [48] IEEE Standards Association, “The Evolution of Wi-Fi Technology and Standards,” Beyond Standards, 2023. [Online]. Available: https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/. Accessed: Oct. 9, 2025.
- [49] Texas Instruments, “Introduction to UART,” Application Report SLLA037, 2023. [Online]. Available: https://www.ti.com/lit/an/slla037/slla037.pdf. Accessed: Oct. 10, 2025.
- [50] E. Peña and M. G. Legaspi, “UART: A Hardware Communication Protocol – Understanding Universal Asynchronous Receiver/Transmitter,” Analog Devices Technical Article, Vol. 54, No. 4, Dec. 2020. [Online]. Available: https://www.analog.com/en/resources/technical-articles/uart-a-hardware-communication-protocol.html. Accessed: Oct. 10, 2025.
- [51] Adafruit Industries, “Adafruit I2S MEMS Microphone Breakout — SPH0645LM4H (Product ID 3421),” *Adafruit*, [Online]. Available: https://www.adafruit.com/product/3421. Accessed: Oct. 10, 2025.
- [52] Adafruit Industries, “Adafruit I2S MEMS Microphone Breakout — Learn Guide,” *Adafruit Learning System*, [Online]. Available: https://learn.adafruit.com/adafruit-i2s-mems-microphone-breakout/. Accessed: Oct. 10, 2025.
- [53] Adafruit Industries, *SPH0645LM4H I2S Microphone Datasheet*, Rev. B, [Online]. Available: https://cdn-shop.adafruit.com/product-files/3421/i2S+Datasheet.PDF. Accessed: Oct. 10, 2025.
- [54] Adafruit Industries, “Electret Microphone Amplifier – MAX4466 – Product ID 1063,” *Adafruit*, [Online]. Available: https://www.adafruit.com/product/1063?srsltid=AfmBOor-3mvJ0P8cW1xtfyO8jprSH9uA9CjOwwlwTaCDq42zzSmysYN-. Accessed: Oct. 10, 2025.
- [55] InvenSense (TDK), *INMP441 Omnidirectional MEMS Microphone with I²S Digital Output — Datasheet*, Rev. 1.1, May 21, 2014. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf. Accessed: Oct. 10, 2025.
- [56] GeeksforGeeks, “Difference Between Git and GitHub,” *GeeksforGeeks*, Sep. 9, 2024. [Online]. Available: https://www.geeksforgeeks.org/git/difference-between-git-and-github/. Accessed: Oct. 10, 2025.
- [57] GitHub, Inc., “About GitHub,” *GitHub*, [Online]. Available: https://github.com/about. Accessed: Oct. 10, 2025.
- [58] Git, “Git – Free and Open Source Distributed Version Control System,” *git-scm.com*, [Online]. Available: https://git-scm.com/. Accessed: Oct. 10, 2025.
- [59] B. Li, “Bitbucket vs GitHub: Which Code Repository Is Better?” *Kinsta*, Aug. 28, 2025. [Online]. Available: https://kinsta.com/blog/bitbucket-vs-github/#:~:text=GitHub%20is%20a%20powerful%20open,for%20your%20private%2C%20proprietary%20code. Accessed: Oct. 10, 2025.

- [60]“PETG,” *Prusa Research Help Center*, last updated [date of access]. [Online]. Available: https://help.prusa3d.com/article/petg_2059 [Accessed: Oct. 30, 2025].
- [61] USB Implementers Forum, “Universal Serial Bus Type-C Connectors and Cable Assemblies Compliance Document, Rev. 2.1b,” June 2021. [Online]. Available: https://www.usb.org/sites/default/files/USB%20Type-C_Compliance%20Document_Rev_2_1b_June_2021.pdf [Accessed: Oct. 30, 2025].
- [62] Same Sky (formerly CUI Devices), “CMS-2850-058SP Data Sheet,” Sep. 11, 2024. [Online]. Available: https://www.sameskydevices.com/product/resource/cms-2850-058sp.pdf [Accessed: Oct. 30, 2025].
- [63] PUI Audio, Inc., “Data Sheet: AS03608MR-10-R,” PUI Audio, 2021. [Online]. Available: https://docs.google.com/gview?url=https://api.puiaudio.com/filename/AS03608MR-10-R.pdf&embedded=true [Accessed: Oct. 30 2025].
- [64] Soberton Inc., “SP-2305L Datasheet,” Feb. 8 2021. [Online]. Available: https://www.soberton.com/wp-content/uploads/2021/03/SP-2305L-dated-8-Feb-2021.pdf [Accessed: Oct. 30 2025].
- [65] Adafruit Industries, “Monochrome 0.96-inch 128×64 I²C/SPI OLED Display – SSD1306,” Adafruit Product Guide, 2024. [Online]. Available: https://learn.adafruit.com/monochrome-oled-breakouts/overview Accessed: Oct. 10, 2025.
- [66] “LCD benefits and uses,” ChatGPT, Accessed: Oct. 30, 2025. [Online]. Available: https://chatgpt.com/share/68eee81f-c490-8000-908e-acebb7216105
- [67] “Amplifiers speakers filters lists,” *ChatGPT*, Accessed: Oct. 30, 2025. [Online]. Available: https://chatgpt.com/share/68ebfde1-5a90-800d-98d6-9e5269c98d86
- [68] Texas Instruments, “Using Segmented Displays (LCD),” *MSP430 Design Workshop*, Chapter 12, 2023. [Online]. Available:https://software-dl.ti.com/trainingTTO/trainingTTO_public_sw/MSP430_LaunchPad_Workshop/v4/Chapters/MSP430m12_LCD.pdf Accessed: Oct. 10, 2025.
- [69] Adafruit Industries, “Adafruit 8×8 RGB LED Matrix,” Product Documentation, 2024. [Online]. Available: https://learn.adafruit.com/adafruit-neopixel-8x8-matrix. Accessed: Oct. 10, 2025.
- [70] *16 Channel 12-bit PWM/Servo Driver PCA9685 with I2C Interface*. UrukTech. https://www.uruktech.com/product/servo-driver-pca9685/ Accessed: Oct. 10, 2025.
- [71] M. Kaya and A. Epps, “Relationship Between Color and Emotion: A Study of College Students,” *College Student Journal*, vol. 38, no. 3, pp. 396–405, 2004.
- [72] Y. Kuller, “The Influence of Color on Children’s Behavior and Emotions,” *Early Child Development and Care*, vol. 179, no. 8, pp. 1041–1056, 2009.
- [73] McMillan Pazdan Smith Architecture, “A Palette for Learning: The Role of Color in Early Childhood Education Design,” 2023. [Online]. Available: https://www.mcmillanpazdansmith.com/ideas/a-palette-for-learning-the-role-of-color-in-early-childhood-education-design/. Accessed: Oct. 11, 2025.

- [74] H. Li, “The Influence of Color on Children’s Behavior: A Review of Psychological and Environmental Perspectives,” *Review Paper*, 2023.
- [75] N. Gmerek, K. Nguyen, and U. Serna, *“DC to DC USB-C Charger,”* B.S. senior project, Dept. Electr. Eng., California Polytechnic State Univ., San Luis Obispo, CA, USA, Jun. 2019. [Online]. Available: https://digitalcommons.calpoly.edu/eesp/447/
- [76] "Understanding of TP5100 2A Lithium Battery Charger Module FAQ," *UTmel Electronics*, 2023. [Online]. Available: https://www.utmel.com/components/understanding-of-tp5100-2a-lithium-battery-charger-module-faq?id=1856
- [77] USB Implementers Forum (USB-IF), “USB Charger (USB Power Delivery),” 2021. [Online]. Available:https://www.usb.org/usb-charger-pd. [Accessed: Oct. 13, 2025].
- [78] Molex Inc., *USB Type-C Connectors Product Specification*, Datasheet, 2023. [Online]. Available: https://www.content.molex.com/dxdam/c9/c924179b-a897-4f79-a5ae-cfdc94fc73cb/987651-4081.pdf. [Accessed: Oct. 13, 2025].
- [79] Encore Data Products, “Budget-Friendly USB-C Charging Solutions to Power Your Classroom Tech,” News Article, 2023. [Online]. Available: https://www.encoredataproducts.com/news/budgetfriendly-usbc-charging-solutions-to-power-your-classroom-tech-ae083e/. [Accessed: Oct. 13, 2025].
- [80] EDAC Connectivity, “Magnetic Spring-Loaded Connectors,” Product Overview, 2024. [Online]. Available: https://edac.net/products/magnetic-spring-loaded-connectors/227. [Accessed: Oct. 13, 2025].
- [81] Promax Pogo Pin Co., “Spring-Loaded Pins and Connectors,” Product Specification, 2024. [Online]. Available:https://promaxpogopin.com/spring-loaded-pin. [Accessed: Oct. 13, 2025].
- [82] Harwin plc, *Pogo Pins and Contact Pads Datasheet*, Technical Document, 2024. [Online]. Available:https://cdn.harwin.com/pdfs/Harwin_Datasheet-Pogo-Pins-Contact-Pads.pdf. [Accessed: Oct. 13, 2025].
- [83] Johoty Precision Co., “What is Common Failure Mode of Pogo Pins?,” Technical Blog, 2023. [Online]. Available: https://www.johotypro.com/failure-mode/. [Accessed: Oct. 13, 2025].
- [84] All About Circuits, “How Does the Qi Wireless Charging Standard Work?,” News Article, 2024. [Online]. Available: https://www.allaboutcircuits.com/news/how-does-qi-wireless-charging-standard-work/. [Accessed: Oct. 13, 2025].
- [85] Laird Technologies, “Inductive (Wireless) Charging Technology Overview,” Technical Article, 2024. [Online]. Available: https://www.laird.com/knowledge-center/inductive-wireless-charging. [Accessed: Oct. 13, 2025].
- [86] G. Mou et al., *Energy Efficient and Adaptive Design for Wireless Power Transfer*, Durham University Repository, 2019. [Online]. Available: https://durham-repository.worktribe.com/preview/1361245/21296.pdf. [Accessed: Oct. 13, 2025].
- [87] Texas Instruments, “Adapting Qi-Compliant Wireless Power Solutions to Low-Power Wearables,” Application Note SLYT570, 2013. [Online]. Available: https://www.ti.com/lit/ml/slyt570/slyt570.pdf. [Accessed: Oct. 13, 2025].

- [88] U. Krishnamoorthy, “Efficient Battery Models for Performance Studies — Lithium and NiMH,” *Batteries (MDPI)*, vol. 9, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2313-0105/9/1/52. [Accessed: Oct. 13, 2025].
- [89] S. Kumar et al., “Advancing Energy Storage: The Future Trajectory of Lithium-Ion Battery Systems,” *Energy Reports*, vol. 13, pp. 150–164, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X25012241. [Accessed: Oct. 13, 2025].
- [90] R. Jain et al., “System Protection for Lithium-Ion Battery Management: A Review,” *IEEE Access*, 2019. [Online]. Available:https://www.researchgate.net/publication/332140142_System_protection_for_Lithium-ion_batteries_management_system_A_review. [Accessed: Oct. 13, 2025].
- [91] S. Nishikawa and A. Hirasawa, “Internal Battery Overcharge Protection,” *Coordination Chemistry Reviews*, vol. 514, 2025. [Online]. Available:https://www.sciencedirect.com/science/article/pii/S0010854525005247. [Accessed: Oct. 13, 2025].
- [92] A. A. Fardoun, E. H. Ismail, M. A. Al-Saffar, and A. J. Sabzali, “New ‘Real’ Bridgeless High Efficiency AC-DC Converter,” *IEEE Transactions on Power Electronics*, 2014. [Online]. Available:https://www.researchgate.net/publication/261115580_New_real_bridgeless_high_efficiency_AC-DC_converter. [Accessed: Oct. 13, 2025].
- [93] Anker Innovations, “Anker MagGo 10,000 mAh Qi2 Power Bank (MagSafe Compatible),” Product Specification, 2024. [Online]. Available: https://www.anker.com/products/a1654-maggo-10000mah-qi2-power-bank-magsafe-compatible. [Accessed: Oct. 13, 2025].
- [94] A. Sharma and P. Singh, “Studies on Portable Power Banks for Recharging Electronic Devices,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 3, pp. 1–5, 2018. [Online]. Available:https://www.irjet.net/archives/V5/i3/IRJET-V5I3346.pdf. [Accessed: Oct. 13, 2025].
- [95] "What is SRAM (static random access memory)?," *TechTarget*. https://www.techtarget.com/whatis/definition/SRAM-static-random-access-memory (accessed Oct. 30, 2025).
- [96] "What is SRAM (Static Random Access Memory)?," *phoenixNAP Global IT Services*. https://phoenixnap.com/glossary/sram-static-random-access-memory (accessed Oct. 30, 2025).
- [97] Sanfoundry, “SRAM.” [Online]. Available: https://www.sanfoundry.com/sram. [Accessed: Oct. 30, 2025].
- [98] S. Lee, "Mastering flash memory in embedded systems," *Number Analytics*. https://www.numberanalytics.com/blog/mastering-flash-memory-in-embedded-systems (accessed Oct. 30, 2025).
- [99] O. Pfeiffer and A. Ayre, "Using flash memory in embedded applications," *Embedded Systems Academy*. https://www.esacademy.com/en/library/technical-articles-and-documents/8051-programming/using-flash-memory-in-embedded-applications.html (accessed Oct. 30, 2025).

- [100] P. Dheer, “EEPROM Overview and Its Importance in Embedded Systems,” Vskills Tutorials, 2024. [Online]. Available: https://www.vskills.in/certification/tutorial/eeprom-overview-and-its-importance-in-embedded-systems/. [Accessed: 22-Nov-2025].
- [101] Sarah Lee, “EEPROM in Embedded Systems: A Comprehensive Guide to Understanding and Utilizing EEPROM,” *Number Analytics Blog*, Jun. 11, 2025. [Online]. Available: https://www.numberanalytics.com/blog/ultimate-guide-eeprom-embedded-systems [Accessed: Oct. 30 2025].
- [102] “A Complete Guide to Electrically Erasable Programmable Read-Only Memory (EEPROM),” *RS Online Australia*, 10 Dec. 2024. [Online]. Available: https://au.rs-online.com/web/content/discovery/ideas-and-advice/eeprom-guide [Accessed: Oct. 30 2025].
- [103] Cadence Design Systems, “Can Pull-Up Resistors Be Used on SPI Lines?,” *Cadence PCB Design Blog*, 2023. [Online]. Available: https://resources.pcb.cadence.com/blog/can-pull-up-resistors-be-used-on-spi-lines. Accessed: Oct. 11, 2025.
- [104] V. N. Saraswathi, “A Comprehensive Review on Charger Technologies, Types and Topologies for EVs,” *[Article title]*, vol. [volume], no. [issue], pp. [pages], 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844024149766
- [105] W. Diao, S. Saxena, and M. Pecht, “Analysis of Specified Capacity in Power Banks,” *IEEE Access*, 2020. [Online]. Available: https://www.researchgate.net/publication/338985271_Analysis_of_Specified_Capacity_in_Power_Banks. [Accessed: Oct. 13, 2025].
- [106] Monolithic Power Systems, “Voltage Regulator Types,” *Monolithic Power Systems Learning Resources*. [Online]. Available: https://www.monolithicpower.com/en/learning/resources/voltage-regulator-types
- [107] I. L. Pressman, “Introduction to Power Electronics,” *Switching Power Supply Design*, Elsevier, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/B9780750686266000016
- [108] Linear Technology, “AN32—Linear Regulators: Theory of Operation and Applications,” *Application Note 32*. [Online]. Available: http://listas.exa.unne.edu.ar/.../Linear_an32_lin_reg.pdf
- [109] ROHM Semiconductor, “DC/DC Converter Basics,” *TechWeb ROHM*. [Online]. Available: https://techweb.rohm.com/product/power-ic/dcdc/88
- [110] R. Keim, “Understanding Switch-Mode Regulation: The Buck Converter,” *All About Circuits*, 2020. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/understanding-switch-mode-regulation-the-buck-converter
- [111] Monolithic Power Systems, “Buck Converter Topology Design Guide,” *MPS Scholar*. [Online]. Available: https://www.monolithicpower.com/en/buck-converter-topology-design-guide/introduction/advantages-and-limitations
- [112] R. Keim, “Understanding the Operation of a Boost Converter,” *All About Circuits*, 2020. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/understanding-the-operation-of-a-boost-converter

- [113] S. M. Ayob, “Comparison of Electronic Load Using Linear Regulator and Boost Converter,” *ResearchGate*, Aug. 2021. [Online]. Available: https://www.researchgate.net/.../Comparison-of-electronic-load-using-linear-regulator-and-boost-converter.pdf
- [114] Monolithic Power Systems, “Boost Converters,” *MPS Scholar*. [Online]. Available: https://www.monolithicpower.com/en/learning/mpscholar/power-electronics/dc-dc-converters/boost-converters
- [115] F. Zhang et al., “Recent Advances in DC–DC Converter Technology,” *Frontiers in Energy Research*, vol. 9, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2090447921003828
- [116] J. Kim et al., “Performance of Compact Converter Circuits,” *Nature Scientific Reports*, vol. 12, 2022. [Online]. Available: https://www.nature.com/articles/s41598-022-26660-7
- [117] Analog Devices, “Linear Regulators in Portable Applications,” *Technical Article*. [Online]. Available: https://www.analog.com/en/resources/technical-articles/linear-regulators-in-portable-applications.html
- [118] R. Rincon-Mora, “Efficiency Optimization in Low-Dropout Regulators,” *Journal of Low Power Electronics*, 2009. [Online]. Available: https://rincon-mora.gatech.edu/publicat/jrnls/jolpe09_uw.pdf
- [119] *Power Electronics Europe*, “Industry Developments in Power Regulation,” 2018. [Online]. Available: https://www.power-mag.com/pdf/issuearchive/79.pdf
- [120] Texas Instruments, “TLV733P Low-Dropout Linear Regulator Datasheet,” 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/tlv733p.pdf
- [121] Texas Instruments, *TPS63060 High Efficiency Step-Down/Step-Up Converter with 1.2A Switches*, datasheet, Rev. E, Apr. 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps63060.pdf
- [122] Texas Instruments, *LM2576 SIMPLE SWITCHER® Step-Down Voltage Regulator Datasheet*, Dallas, TX, USA, 2016. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2576.pdf
- [123] Texas Instruments, “TPS61023 Boost Converter Datasheet,” 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps61023.pdf
- [124] Connector Supplier, “What Are Spring Probes?,” *Connector Supplier Articles*. [Online]. Available: https://connectorsupplier.com/what-are-spring-probes
- [125] Mill-Max, “Spring-Loaded Pogo Pins and Connectors,” *Engineering Notebook*. [Online]. Available: https://www.mill-max.com/engineering-notebooks/spring-loaded-pogo-pins-connectors
- [126] Wevolver, “Types of PCB Connectors: An In-Depth Guide,” *Wevolver Engineering Articles*. [Online]. Available: https://www.wevolver.com/article/types-of-pcb-connectors-an-in-depth-guide
- [127] Pogopin Connector Co., “Spring-Loaded Connector Overview,” 2024. [Online]. Available: https://www.pogopinconnector.com/index.php/new/index/id/140.html
- [128] Harwin, “Spring Loaded Contacts,” *Product Technical Manual*. [Online]. Available: https://cdn.harwin.com/pdfs/Spring_Loaded_Contacts_PTM.pdf

- [129] Smiths Interconnect, “Spring Probe Contact Technologies,” *Product Page*. [Online]. Available: https://www.smithsinterconnect.com/products/connectors/contact-technologies/spring-probe
- [130] European Space Components Information Exchange System, “Pogo Pin Connector Standards,” *ESCIES Document*, 2019. [Online]. Available: https://escies.org/download/webDocumentFile?id=64655
- [131] Digi-Key, “The Basics of Pogo Pin Connectors,” *Digi-Key Articles*, 2022. [Online]. Available: https://www.digikey.com/en/articles/the-basics-of-pogo-pin-connectors
- [132] Digi-Key, “Use Tight-Pitch Board-to-Board Connectors,” *Digi-Key Articles*, 2022. [Online]. Available: https://www.digikey.com/en/articles/use-tight-pitch-board-to-board-connectors
- [133] Samtec, “MEG-Array® Connector Datasheet,” *Mouser Electronics*, 2024. [Online]. Available: https://www.mouser.com/ds/2/154/950554-007_meg-array-21286.pdf
- [134] Samtec Blog, “Understanding Low-Level Contact Resistance (LLCR),” *Samtec Technical Blog*, 2024. [Online]. Available: https://blog.samtec.com/post/understanding-low-level-contact-resistance-llcr
- [135] FORCE Technology, “Connector Weaknesses: Halt and Accelerated Failure,” *FORCE Articles*. [Online]. Available: https://forcetechnology.com/en/articles/connectors-weeknes-halt-accelerate-failure-temperature-vibration
- [136] GCT, “Pin Header Board-to-Board Connectors,” *GCT Products*. [Online]. Available: https://gct.co/board-to-board-connector/pin-header
- [137] Connector Supplier, “Self-Aligning Magnetic Connectors for Medical Applications,” *Connector Supplier Articles*. [Online]. Available: https://connectorsupplier.com/self-aligning-magnetic-connectors-provide-optimal-power-and-data-solutions-for-medical-applications
- [138] Smiths Interconnect, “Understanding Fretting Corrosion in Ruggedized Backplane Connectors,” *White Paper*. [Online]. Available: https://www.smithsinterconnect.com/library/technical-library/white-papers/understanding-fretting-corrosion-in-ruggedized-backplane-connectors
- [139] K. Suzuki et al., “Inductively Coupled Connectors and Sockets for Multi-Gb/s Pulse Signaling,” *IEEE Trans. Components and Packaging Technologies*, vol. 33, no. 4, 2011. [Online]. Available: https://www.researchgate.net/publication/224348963_Inductively_Coupled_Connectors_and_Sockets_for_Multi-Gbs_Pulse_Signaling
- [140] Adafruit, “Magnetic Connector 2-Pin Product Page,” *Adafruit Industries*. [Online]. Available: https://www.adafruit.com/product/5360
- [141] EDAC, “Magnetic Spring-Loaded Connectors,” *Product Catalog*, 2024. [Online]. Available: https://edac.net/products/magnetic-spring-loaded-connectors/227
- [142] *Military Aerospace*, “Connectors: Magnetic Power or Signal?,” 2024. [Online]. Available:

https://www.militaryaerospace.com/power/article/14284426/connectors-magnetic-power-or-signal

- [143] Y. Wang et al., "Magnetic Connector Applications for Wearable Electronics," *Journal of Advanced Engineering Materials*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2950160124000123
- [144] Monolithic Power Systems, "Battery Charger Fundamentals," *Learning Resources*. [Online]. Available: https://www.monolithicpower.com/learning/resources/battery-charger-fundamentals
- [145] Microchip Technology, "MCP73831 Li-Ion/Li-Polymer Charge Management Controller Datasheet," 2024. [Online]. Available: https://ww1.microchip.com/downloads/.../MCP73831-Family-Data-Sheet-DS20001984H.pdf
- [146] Cirkit Designer Docs, "MCP73831 Component Page," *Cirkit Designer Documentation*, 2024. [Online]. Available: https://docs.cirkitdesigner.com/component/eb1ee89f-d645-5be8-eba4-3dbb15a1a520/mcp73831
- [147] Microchip Technology, "MCP73831 Family Datasheet," *Microchip*, 2024. [Online]. Available: https://ww1.microchip.com/downloads/.../MCP73831-Family-Data-Sheet-DS20001984H.pdf
- [148] UTMEL, "MCP73832T-2ACI/OT Charge Management Controllers," *UTMEL Blog*, 2024. [Online]. Available: https://www.utmel.com/components/mcp73832t-2aci-ot-li-polymer-charge-management-controllers-diagram-pinout-and-datasheet
- [149] Blikai, "What Is TP4056 Module? All You Need to Know," *Blikai Blog*, 2024. [Online]. Available: https://www.blikai.com/blog/featured-products/what-is-tp4056-module-all-you-need-to-know
- [150] NanJing Top Power ASIC Corp., "TP4056 Linear Li-Ion Charger Datasheet," 2024. [Online]. Available: https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf
- [151] ChipSourceTek, "Charge Chip ME4057ASPG Overview," *Product Page*. [Online]. Available: https://en.chipsourcetek.com/Charge-Chip/2625.html
- [152] Micro One Electronics, "ME4057ASPG Datasheet," *LCSC Electronics*, 2021. [Online]. Available: https://datasheet.lcsc.com/lcsc/2111091230_MICRONE-Nanjing-Micro-One-Elec-ME4057ASPG_C82651.pdf
- [153] Adafruit, "Micro-Lipo Charger for Li-Ion/Li-Poly Batteries," *Product Page*, 2024. [Online]. Available: https://www.adafruit.com/product/1901
- [154] Attend Technology, "USB-C Connector Specification 303C-C4115-30-04," *Product Datasheet*, 2024. [Online]. Available: https://www.attend.com.tw/data/download/file/303C-C4115-30-04_Spec.pdf
- [155] Advanced Plating Tech, "Gold Plating Thickness for Connectors," *APT Technical Resources*, 2024. [Online]. Available: https://advancedplatingtech.com/gold-plating/gold-plating-thickness-connectors

- [156] Fenix Lighting, “The Ultimate Guide to the 18650 Battery,” *Fenix Blog*, 2024. [Online]. Available: https://www.fenixlighting.com/blogs/news/the-ultimate-guide-to-the-18650-battery
- [157] Texas Instruments, “SLUA420A: Lithium-Ion Charger Design Considerations,” *Application Note*, 2024. [Online]. Available: https://www.ti.com/lit/an/slua420a/slua420a.pdf
- [158] Adafruit Industries, “5358 – Pogo Pin with Pointed Tip (Pack of 10),” *Adafruit Industries LLC*, 2024. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/5358/15965218. [Accessed: Oct. 23, 2025].
- [159] Analog Devices, “ADuM1250/ADuM1251 Dual I2C Isolators Datasheet,” *Analog Devices*, 2024. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adum1250_1251.pdf
- [160] *ESP32-WROOM-32 Datasheet Version 3.5*, [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Accessed: Oct. 17, 2025. espressif.com
- [159] “Understanding Electrical Safety Standards and Regulations,” CVE.com, [Online]. Available: https://www.cve.com/news/understanding-electrical-safety-standards-and-regulations/
- [160] “Safety Extra-Low Voltage (SELV): What Does It Mean?,” RS Online / DesignSpark, 10 Aug 2023. [Online]. Available: https://www.rs-online.com/designspark/safety-extra-low-voltage-selv-what-does-it-mean
- [161] “What is the Use of a Decoupling Capacitor?,” Proto Express Blog, 30 Mar 2021. [Online]. Available: https://www.protoexpress.com/blog/decoupling-capacitor-use/
- [162] “Applying IPC-2221 Standards in Circuit Board Design,” Proto Express Blog, 30 Jan 2023. [Online]. Available: https://www.protoexpress.com/blog/ipc-2221-circuit-board-design/
- [163] Electric Fire Design — “LEDs for Light Art, Part 4: LED Driver Circuits.” https://electricfiredesign.com/2021/02/26/leds-for-light-art-part-4-led-driver-circuits/#:~:text=power%20supply%20,operating%20current%2C%20using%20the%20formula
- [164] Electronics Tutorials, “LED Circuit Basics – Resistors, Forward Voltage, and Current,” https://www.electronics-tutorials.ws/diode/diode_8.html
- [165] Suntech LEDs, “WS2812B vs. APA102: What Is the Difference Between These Addressable LED Strips?,” *Suntech LED Technical Articles*, 2023.https://www.suntechleds.com/ws2812b-vs-apa102.html#:~:text=What%20is%20the%20difference%20between,and%20APA%20102%20LED%20strip
- [166] ESPBoards.dev, “Addressable LED Strips with ESP32,” *ESPBoards Technical Blog*, 2024.
- [167] “APA102, WS2812, SK6812 Addressable LED Strips Evolution,” *ESPBoards Blog*, Oct. 10 2023. [Online]. Available: https://www.espboards.dev/blog/addressable-led-strips-esp32 [Accessed: Oct. 30 2025].

- [168] Apichet Garaipoom, “UM66T Melody Generator Circuit – Easy Examples for Beginners,” *ElecCircuit.com*, Oct. 22, 2025. [Online]. Available: https://www.eleccircuit.com/create-music-with-ic-um66t/ [Accessed: Oct. 30, 2025].

Appendix B - Copyright Permissions

Appendix C - Data Sheet

- [17] Raspberry Pi Ltd., “RP2040 Datasheet”, Rev. 3.4, 2024. [Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Accessed: Oct. 8, 2025
- [19] Espressif Systems, “ESP32-C3 Datasheet”, Rev. 1.3, 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf. Accessed: Oct. 8, 2025
- [18] Atmel (Microchip), ATmega328P 8-bit AVR Microcontroller Datasheet, 7810D–AVR–01/15, 2015. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- [25] Espressif Systems, *ESP32-S3 Series Datasheet*, version 2.0, Jul. 9, 2021. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf
- [40] PUI Audio, Inc., “AS03608MR-10-R Data Sheet,” [Online]. Available: https://docs.google.com/gview?url=https://api.puiaudio.com/filename/AS03608MR-10-R.pdf&embedded=true [Accessed: Oct. 30 2025].
- [41] Analog Devices, “MAX98357A / MAX98357B — Tiny, Low-Cost, PCM Class D Amplifier with Class AB Performance,” Rev. 13, July 29, 2019. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/max98357a-max98357b.pdf [Accessed: Oct. 30 2025].
- [53] Adafruit Industries, *SPH0645LM4H I2S Microphone Datasheet*, Rev. B, [Online]. Available: https://cdn-shop.adafruit.com/product-files/3421/i2S+Datasheet.PDF. Accessed: Oct. 10, 2025.
- [55] InvenSense (TDK), *INMP441 Omnidirectional MEMS Microphone with I²S Digital Output — Datasheet*, Rev. 1.1, May 21, 2014. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf. Accessed: Oct. 10, 2025.
- [51] Adafruit Industries, “Adafruit I2S MEMS Microphone Breakout — SPH0645LM4H (Product ID 3421),” *Adafruit*, [Online]. Available: https://www.adafruit.com/product/3421. Accessed: Oct. 10, 2025.
- [52] Adafruit Industries, “Adafruit I2S MEMS Microphone Breakout — Learn Guide,” *Adafruit Learning System*, [Online]. Available: https://learn.adafruit.com/adafruit-i2s-mems-microphone-breakout/. Accessed: Oct. 10, 2025.
- [53] Adafruit Industries, *SPH0645LM4H I2S Microphone Datasheet*, Rev. B, [Online]. Available: https://cdn-shop.adafruit.com/product-files/3421/i2S+Datasheet.PDF. Accessed: Oct. 10, 2025.

- [54] Adafruit Industries, “Electret Microphone Amplifier – MAX4466 – Product ID 1063,” *Adafruit*, [Online]. Available: https://www.adafruit.com/product/1063?srsltid=AfmBOor-3mvJ0P8cW1xtfyO8jprSH9uA9CjOwwlwTaCDq42zzSmysYN-. Accessed: Oct. 10, 2025.
- [55] InvenSense (TDK), *INMP441 Omnidirectional MEMS Microphone with I²S Digital Output — Datasheet*, Rev. 1.1, May 21, 2014. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf. Accessed: Oct. 10, 2025.
- [62] Same Sky (formerly CUI Devices), “CMS-2850-058SP Data Sheet,” Sep. 11, 2024. [Online]. Available: https://www.sameskydevices.com/product/resource/cms-2850-058sp.pdf [Accessed: Oct. 30, 2025].
- [63] PUI Audio, Inc., “Data Sheet: AS03608MR-10-R,” PUI Audio, 2021. [Online]. Available: https://docs.google.com/gview?url=https://api.puiaudio.com/filename/AS03608MR-10-R.pdf&embedded=true [Accessed: Oct. 30 2025].
- [64] Soberton Inc., “SP-2305L Datasheet,” Feb. 8 2021. [Online]. Available: https://www.soberton.com/wp-content/uploads/2021/03/SP-2305L-dated-8-Feb-2021.pdf [Accessed: Oct. 30 2025].
- [78] Molex Inc., *USB Type-C Connectors Product Specification*, Datasheet, 2023. [Online]. Available: https://www.content.molex.com/dxdam/c9/c924179b-a897-4f79-a5ae-cfdc94fc73cb/987651-4081.pdf. [Accessed: Oct. 13, 2025].
- [121] Texas Instruments, *TPS63060 High Efficiency Step-Down/Step-Up Converter with 1.2A Switches*, datasheet, Rev. E, Apr. 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps63060.pdf
- [122] Texas Instruments, *LM2576 SIMPLE SWITCHER® Step-Down Voltage Regulator Datasheet*, Dallas, TX, USA, 2016. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2576.pdf
- [123] Texas Instruments, “TPS61023 Boost Converter Datasheet,” 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps61023.pdf
- [147] Microchip Technology, “MCP73831 Family Datasheet,” *Microchip*, 2024. [Online]. Available: https://ww1.microchip.com/downloads/.../MCP73831-Family-Data-Sheet-DS20001984H.pdf

Appendix D - Software Code

- [169] **ESP32 Execution Logs:** The following transcript captures the serial output during the system boot sequence. It confirms the successful initialization of the Wi-Fi stack, DHCP negotiation, and the startup of the TCP server task.

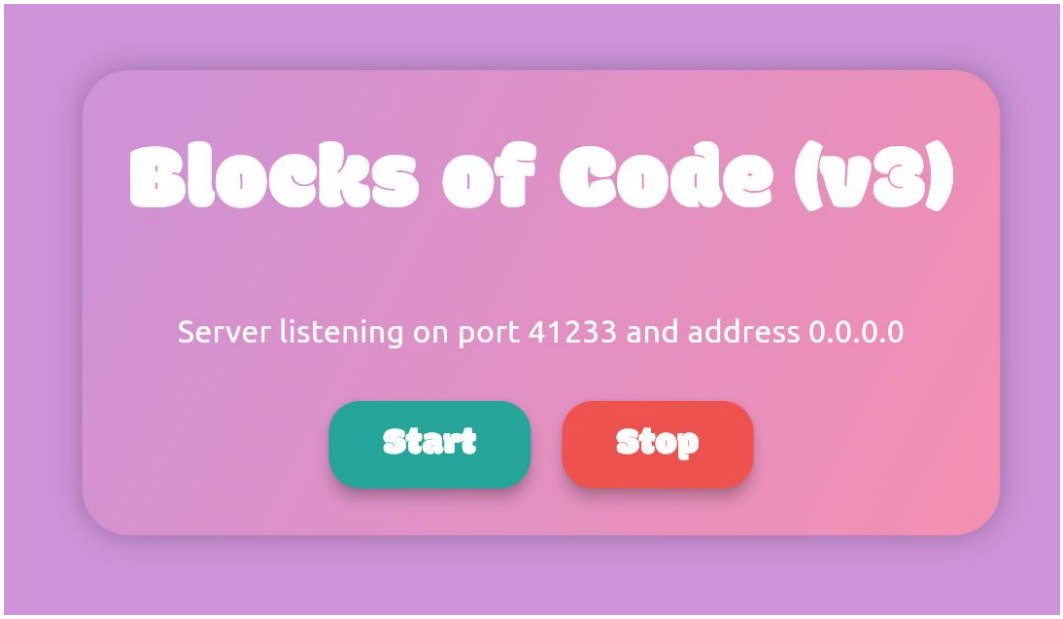
```
I (755) SELFTEST: WiFi started. Connecting…
I (755) TCP: Server listening on port 41233
I (52425) wifi:connected with Jordan, aid = 1, channel 6, BW20, bssid = 96:23:8a:
0b:b9:4b
I (52425) wifi:security: WPA2-PSK, phy: bgn, rssi: -47
I (52445) wifi:pm start, type: 1

I (52445) wifi:dp: 1, bi: 102400, li: 3, scale listen interval from 307200 us to
307200 us
I (52445) SELFTEST: Connected to AP.
I (52525) wifi:AP's beacon interval = 102400 us, DTIM period = 3
I (53445) esp_netif_handlers: sta ip: 172.20.10.9, mask: 255.255.255.240, gw: 172
.20.10.1
I (53445) SELFTEST: Got IP: 172.20.10.9
```

- [170] **Timer Performance Verification:** To verify the software timer implementation, a self-test routine was executed. The system measured the internal clock against a target duration of 1,000,000 µs (1 second).

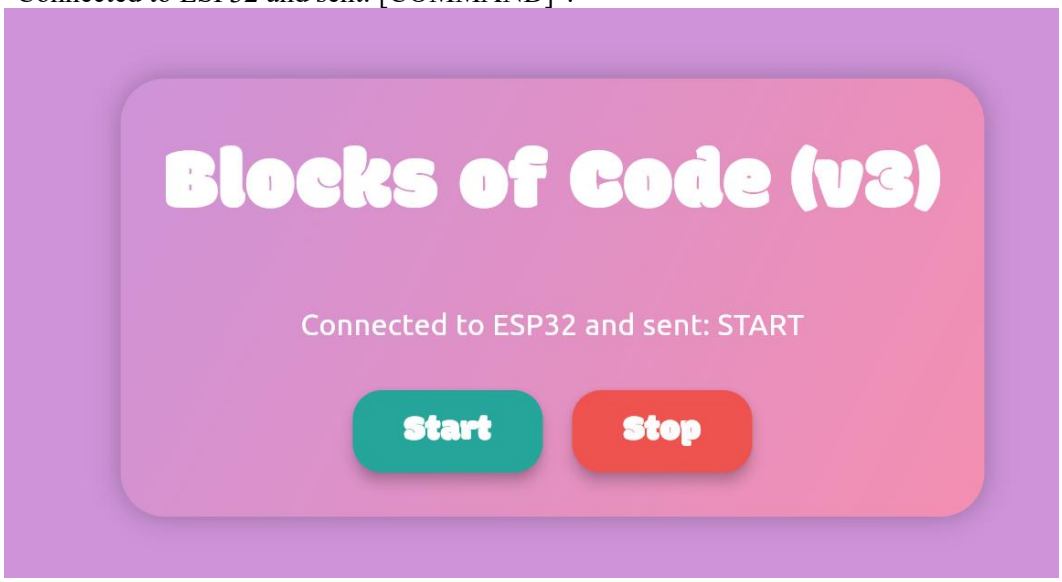
```
I (1977) SELFTEST: Timer sample  1: 999435 us (expected 1000000 us)
I (2977) SELFTEST: Timer sample  2: 999471 us (expected 1000000 us)
I (3977) SELFTEST: Timer sample  3: 999347 us (expected 1000000 us)
I (4977) SELFTEST: Timer sample  4: 999355 us (expected 1000000 us)
I (5977) SELFTEST: Timer sample  5: 999347 us (expected 1000000 us)
I (6977) SELFTEST: Timer sample  6: 999407 us (expected 1000000 us)
I (7977) SELFTEST: Timer sample  7: 999494 us (expected 1000000 us)
I (8977) SELFTEST: Timer sample  8: 999494 us (expected 1000000 us)
I (9977) SELFTEST: Timer sample  9: 999510 us (expected 1000000 us)
I (10977) SELFTEST: Timer sample 10: 999419 us (expected 1000000 us)
I (10977) SELFTEST: Timer summary: mean=999427.9 us, min=999347 us, max=999510 us
```

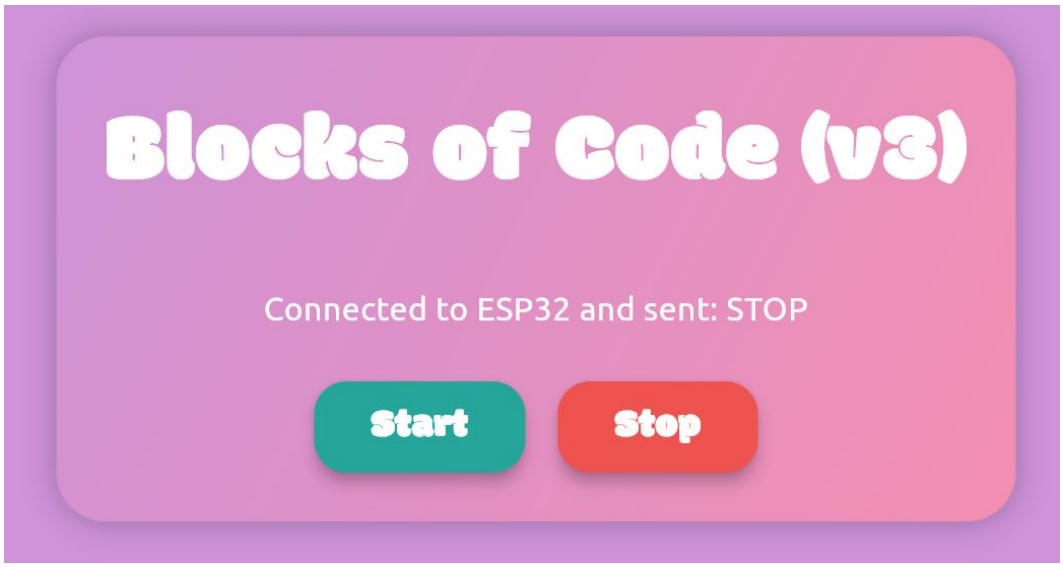
- [171] **Application Interface States (Flutter):** The mobile application ("Blocks of Code v3") provides the control interface for the system. The following states were validated during testing:

1. **Server Listening:** The app displays the local IP and port (41233) while awaiting a connection.

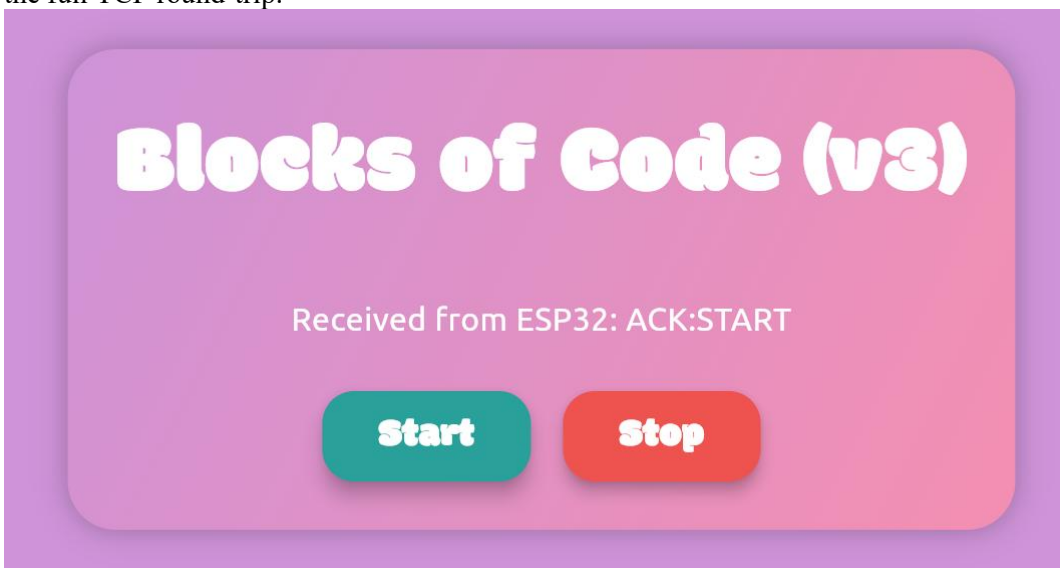


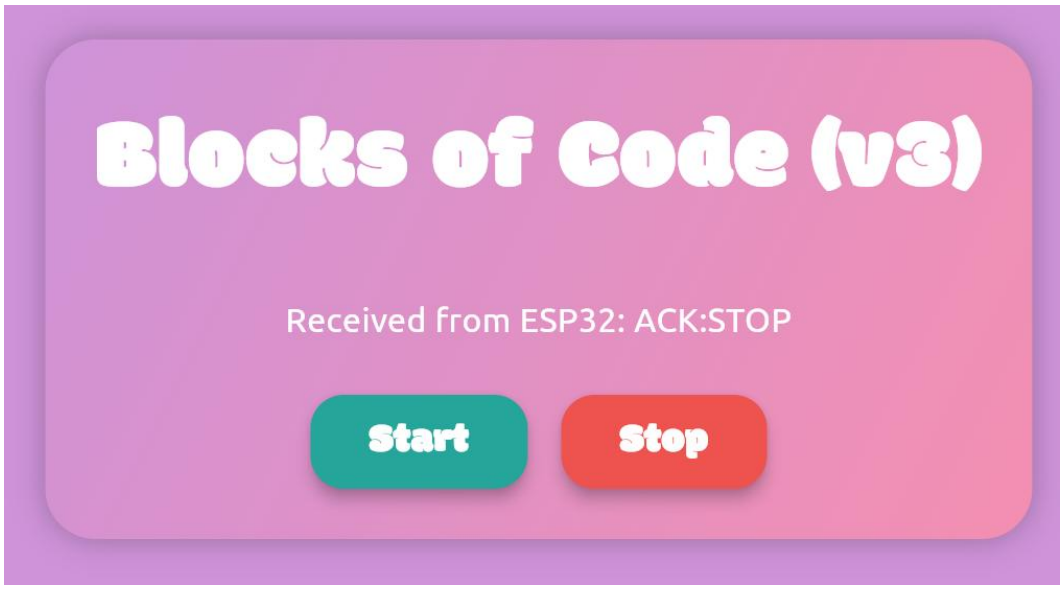
2. **Command Transmission:** Upon pressing "Start" or "Stop," the status updates to "Connected to ESP32 and sent: [COMMAND]".



3. 

4. **Bidirectional Feedback:** The app successfully parses the return payload, displaying "Received from ESP32: ACK:START" or "ACK:STOP," confirming the full TCP round-trip.



5. 

Appendix E - ChatGPT Prompts and Outcomes

- [1] "*Prompt: Explain the benefits and uses of LCDs for embedded systems and write a summary I can put in my report.*" *ChatGPT (GPT-5.1)*, Generated output summarizing LCD advantages, operation principles, and common application areas., Accessed: **Oct. 30, 2025**. [Online]. Available: https://chatgpt.com/share/68eee81f-c490-8000-908e-acebb7216105.
- [2] "*Prompt: Provide a list of amplifiers, speakers, and audio filters suitable for embedded or small-form-factor systems.*" *ChatGPT (GPT-5.1)*, Generated output listing amplifier modules, speaker options, and common audio filter types with brief descriptions., Accessed: **Oct. 30, 2025**. [Online]. Available: https://chatgpt.com/share/68ebfde1-5a90-800d-98d6-9e5269c98d86.
- [3] "*Prompt: power chat (please produce a detailed power systems discussion and list of design considerations)*," *ChatGPT (GPT-5.1)*, response generated with a discussion on power system design, components, and considerations, Accessed: Nov. 22, 2025. [Online]. Available: https://chatgpt.com/share/68d2a636-dc50-8012-b6a2-5a2dbb2ccd28
- [4] "*Prompt: Provide suitable connectors, interface options, and wireless charging methods for an embedded design.*" *ChatGPT (GPT-5.1)*, Generated output listing connector types (e.g., USB-C, pogo pins), interface protocols (e.g., I²C, SPI, UART), and wireless charging standards (e.g., Qi, inductive coupling) with design considerations., Accessed: **Oct. 30, 2025**. [Online]. Available: https://chatgpt.com/share/68dea4d5-2910-8012-911d-3955bc50809b
- [4] "*Prompt: Provide options for lighting and display technologies suitable for embedded systems, including types of LEDs, backlighting, OLEDs, and display interfaces.*" *ChatGPT (GPT-5.1)*, Generated output listing LED types, display backlighting methods, OLED vs LCD comparisons, and interface options (e.g., SPI, RGB parallel), including design considerations., Accessed: **Oct. 30, 2025**. [Online]. Available: https://chatgpt.com/share/68dfe282-b484-8000-af82-7ccacb551926

- [5] "*Prompt: Discuss manufacturing strategies and cost considerations for producing embedded-system hardware at scale.*" *ChatGPT (GPT-5.1)*, Generated output covering manufacturing methods (e.g., PCB assembly, SMT vs through-hole), cost drivers (materials, labor, volume), and tips for reducing cost., Accessed: **Oct. 30, 2025**. [Online]. Available: https://chatgpt.com/share/68dfedab-5e38-8000-8185-e81a8d095082
- [6] "*Prompt: Provide voltage regulator options along with suitable connector and interface choices for an embedded power system.*" *ChatGPT (GPT-5.1)*, Generated output outlining voltage regulator topologies (e.g., linear, switching), recommended connector types (e.g., barrel jack, USB-C, screw terminals), and interface protocols (e.g., PMBus, I²C) for power-management integration., Accessed: **Oct. 30, 2025**. [Online]. Available: https://chatgpt.com/share/68ee7909-8a14-8012-8f8f-f1c66d6a61a8
- [7] "*Prompt: Provide options for lighting and display technologies suitable for embedded systems, including types of LEDs, back-lighting, OLEDs, and display interfaces.*" *ChatGPT (GPT-5.1)*, Generated output listing LED types, display back-lighting methods, OLED vs LCD comparisons, and interface options (e.g., SPI, RGB parallel), including design considerations., Accessed: **Oct. 30, 2025**. [Online]. Available: https://chatgpt.com/share/68dfe282-b484-8000-af82-7ccacb551926
- [8] "*Prompt: Explain the power usage of the ESP32 when driving audio outputs, including DAC output, I²S amplifiers, and expected current draw during playback.*" *ChatGPT (GPT-5.1)*, Generated output describing ESP32 audio-related power consumption, including DAC current levels, I²S peripheral draw, Wi-Fi/BLE impact, amplifier load considerations, and estimated mA ranges during audio playback., Accessed: **Nov. 22, 2025**. [Online]. Available: https://chatgpt.com/share/692259ca-e274-800d-b205-3d13d1ddba39

---

i