University of Central Florida

Department of Electrical and Computer Engineering
Senior Design EEL 4914

Advanced Breadboard
Group 20

Research Paper

Cristian Gutierrez — Computer Engineering

Ammar Mubarez — Electrical Engineering

Sheridan Sloan — Computer Engineering

Committee Members:
Dr. Mike Borowzcack
Dr. Sonali Das
Dr. Suboh Suboh

5 December 2023

Table of Contents

1.0 Executive Summary

The solderless breadboard is a staple tool in electrical and computer engineering. This humble instrument has a simple design but a less than simple use. There is a rise in personal projects that are done outside the lab where there is less access to testing materials. The breadboard can be designed to have more advanced features to make up for this. The Advanced Breadboard will have more capabilities for new users to learn how to correctly build circuits on it.

With components being produced cheaper there is more accessibility to a variety of parts for personal projects. Along with that there is a growing industry in online learning that allows people to try personal projects at home. Due to this, there is a larger audience for those using the basic breadboard. However, these breadboards do little to help identify the mistake a user could have made. Debugging on breadboards is increasingly difficult if the user has little experience with circuit design. Countless hours are spent in labs determining if there is a simple mistake causing incorrect outputs. With a more advanced board, this can be avoided. In the time of automation, this advanced board will become the standard.

The goal of this project is to create an advanced board that will help users avoid common mistakes. This goal is made up of three objectives in order to help the user. The first objective is to take the schematic a user draws and turn it into a layout for how those components should be placed on the breadboard. The second objective is to determine which nodes the components will go into and light them up on the physical breadboard. The third objective is to have an open/short circuit test to alert the user if they made a mistake in placing the components.

A collection of technologies and devices will be used inside a custom-built breadboard. These components include an ESP32 Microcontroller to communicate between the software and the hardware. A Raspberry Pi 4B will house the software with a GUI for the user to interact with. A I2C Digital Wattmeter Sensor will be used to determine open and short circuits. LEDs will be placed inside the breadboard which will be 3D Printed with ABS Filament.

Each component was separately tested to ensure their basic function. Then the components were tested in conjunction with one another and communication between them all were made possible. The breadboard will be developed in layers to separate all of the necessary hardware and create a portable and adaptable product. A PCB will be designed with most of the components including power supply, ESP32, voltage regulator, and other necessary parts. With the completion of the design and testing, the next part of the project will include the production of the part. There will be three main focuses: software algorithms, PCB production and testing, and integration of all components. These sections will need to work to accomplish their individual goal along with their interconnection communication goals. The Advanced Breadboard will change the way circuit design and testing are done in lab settings and home settings.

2. Project Description

2.1 Introduction

The breadboard is an engineering tool that allows the building of a temporary circuit without the use of solder. Without solder, the components on the circuit can be easily assembled and reassembled. The breadboard typically consists of a plastic chassis and lines of metal strips to connect the nodes. Circuits built on breadboards are for demos rather than final product. The simple design of the breadboard makes it so that it serves only one function to its user, but this simplicity allows it to be improved upon.
The Advance Breadboard idea is to redesign the breadboard so that it has advanced features to assist the user in creating a successful circuit.

2.2 Project Motivation and Explanation

The idea for the project came from the team's time in the Linear Circuits Labs. The correct use of the breadboard was integral to success in the laboratory. Understanding how to use the breadboard was straightforward, but applying it in practice proved to be more challenging than expected. Two major challenges of using the breadboard were translating the schematic to a physical board and debugging for components placed incorrectly. The first issue comes from connecting the idea of nodes in a schematic and nodes on the breadboard. Many mistakes come from a lack of understanding of nodes, especially when ground is involved. The second issue consumes a significant amount of lab time because open or short circuits can be hard to spot. Labs are designed to give an application to theory, but unfortunately the majority of the time is spent trying to learn how to use the hardware. The Advanced Breadboard project aims to resolve these issues by adding a graphical user interface (GUI) that will simplify the process of transitioning a schematic to components on a board and identifying open and short circuits.

This new smart-board will include a GUI that a user can use to draw their schematic. An algorithm will then decide how to place that schematic on the breadboard which will also illuminate which nodes to occupy. The breadboard will be made to house LEDs besides the nodes and connect to a device that will house the software for controlling the GUI and LEDs. This breadboard will also have fewer nodes than a typical board to make it easier for the user to learn how to use it. The overall deliverable is a breadboard with LEDs and an open/short circuit test along with a GUI and algorithm for design planning. This project will serve as a stepping-stone for others to develop the breadboard. There is a need for hardware to be more advanced to keep up with the ever-evolving software and advanced algorithms. With the increased accessibility and affordability of microprocessors, at-home projects are on the rise. This project will allow untrained users to become more familiar with circuit design. The conclusion to this project will show how the breadboard can become an advanced feature in teaching of circuit design and integration.

2.3 Goals and Objectives

Objectives:

- Schematic to breadboard wiring algorithm
- LED activation for component placement
- Open/Short circuit detection

Goal

- Redesigned breadboard with advanced features to assist students in learning how to use the breadboard.

This redesigned breadboard aims to reduce the errors that can come from designing and testing on a regular breadboard. This will include five objectives, or deliverables, with three being software focused, and two being hardware related. The first objective is to create a GUI where a user can easily draw a schematic. This GUI will limit what the user can make including how many nodes are available. The focus will be on node configuration rather than what specific components are used as there will not be options for different values. Instead, a list of components will be available including resistors, capacitors, and inductors. The second objective will be to create an algorithm that can transform the schematic into a visual of components on a breadboard. The focus will be on accuracy and speed so that the program does not take time away from the user. The algorithm should ensure that the number of nodes in the schematic are not more than what is available on the breadboard. It will include other checks to make the translation as accurate as possible. The third objective is for the software to send this data to the breadboard to prepare for hardware integration. It will be able to accurately send the virtual breadboard layout to the physical board through a connection between the program and LEDs.

The fourth objective will be for the LEDs in the breadboard to illuminate based on what the algorithm provides it. If nodes A and B are expected to be used then only those nodes should illuminate. This will make it easier for the user to place components on the board without mistake. The LEDs will be bright enough and spaced to not confuse one node for another. The fifth objective is to have a check for open and short circuits. To the untrained-eye that can be difficult to spot, so this check will allow the user to identify this mistake and fix accordingly. This final objective is the program should be able to let the user know that this test failed. Figure 1 shows how the GUI and LEDs will work together to make a guide for the user.

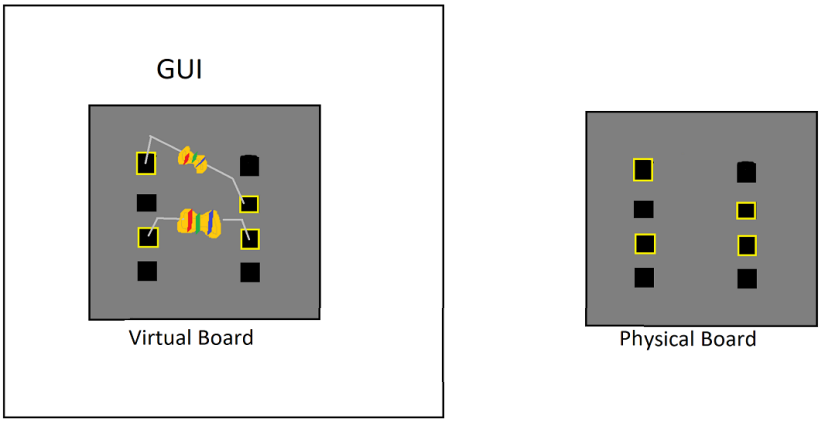

Figure 1*: Virtual and Physical Board Relation.*

To make this project more portable there will be a limited number of nodes as compared to a regular breadboard. With only a handful of nodes there will only be a few holes associated with each. This will create a constraint on the GUI because the schematic will have to consider this limit. The software will have checks to ensure that the user is not building a design that is incapable of being made. These checks will be known to the user if any of them are failed so that a redesign can be made. The breadboard will also need to be designed and built to include the LEDs, PCB, and communication with software. This new board will allow the project to be encompassed with fewer separate parts.

2.4 Prior Related Work

An important part of this project is to create new features to make the breadboard more advanced. Breadboards in most labs are simple solderless boards that allow the user to place components as needed. The goal is to improve on this in a way that has not been done. The four main components of the project are the GUI, schematic to breadboard, illuminated board, and open/short circuit test.

There are many softwares that allow a user to build a schematic and create a simulation. These include, but are not limited to, Multisim, AutoCard, and LTspice. The project will include aspects similar to these examples because of their easy-to-use interface. These applications have a section where components can be dragged and dropped. Keyboard shortcuts are also added to make schematic drawing faster. Lots of circuits can be created and saved to be accessed later. This requires a memory element to the program that will be considered. When those saved circuits are accessed they have the same output as they did originally so the algorithm is reliable and accurate.

There are not many softwares that will convert a schematic to a breadboard design. However, Fritzing is an open source software where the user can create a schematic and convert that to a virtual breadboard and convert that into a PCB design. It allows the user to convert from different views including schematic, breadboard, and PCB layouts. This is an advanced software and provides many capabilities not commonly offered. Fritzing will be useful for guidance through the project on how to visualize the translation process. It gives insight on what the user will prefer to use and what to improve upon. As

breadboards do not have communication ports this information from the software will not translate directly onto a physical board. Rather, the user will have to translate this information. This project will differ in that the software will send this information to the board to illuminate the same nodes.

Most LEDs on a breadboard are tutorials of how to illuminate an LED using the breadboard rather than as a feature of the hardware. Instead there are solderless breadboards that offer different options. There is the Thorlab's Temperature-Controlled Breadboard, and the Powered Breadboard which is more common. Powered breadboards offer a way to keep the power supply with the breadboard in one hardware package rather than separately wiring one in. This will be useful in the consideration of the hardware design because of its compactness and portability. The Temperature-Controlled Breadboard is a special consideration because it is not a typical version of the breadboard readily available. Instead, it is made for specific situations. The Advanced Breadboard will consider the unique design of that board since it will be designed differently from the typical breadboard. The Temperature Controlled board includes components such as electrical tape and mounts as a part of its design [12]. These components will be taken into consideration when designing the hardware because the LEDs will need to be placed close to the nodes without interfering with the circuit. As well as considering how to build the board while keeping components snug in the nodes.

For the open/short circuit test there were not a lot of resources where the breadboard itself was able to detect these. However, there are plenty of resources on the method of how to test for both. The test for an open circuit is when there is an infinite resistance between two points. Continuity tests on a multimeter are another way to test for open circuits. For short circuits, a resistance reading close to 0 will be conclusive. There is another method of using thermal cameras to detect short circuits, but given that we will not be testing with a large enough source this may not prove effective.

2.5 Requirements and Specification

2.5.1 Specifications

The goal for our hardware design would be to let the PCB control the LEDs, check for open and short circuits, and connect to the GUI using a direct connection. This GUI will be able to provide a user interface through a display. From there, we will have LED lights to indicate which nodes should be occupied. Finally, a short/open circuit detection method will be used to determine if the power supply has been shorted or opened when the user places their own circuit on the breadboard. We will be demoing Specification 4, 7, and 8. Specification 4 will be presented by having a prototype GUI that connects to the breadboard without any cabling. Specification 7 will show that the open/short circuit functionality is working and warns the user when an open/short circuit has been detected. Specification 8 will showcase a demo program where if the demo-GUI activated LED1 then that corresponding LED will light up.

Specifications	
Weight	**< 10lbs**
Size	**Fit in two hands (Portable)**
Computer Interface	**ELECROW 5 Inch Touchscreen**
Power Consumption	**5W**
Runtime for Software to Interact with Hardware (LED Activation or Sensor Data Retrieval)	**< 10 Seconds**
Open/Short Circuit Detection	**>95% Accuracy**
LED Activation Placement	**>95% Accuracy**
Battery Powered	**Powers everything except the users circuit**
PCB to GUI communication	**Wired Connection**
Cost	**< $200**

2.5.2 Requirements

This section will focus on the requirements that have been setup for this project. These requirements will be used as objectives that need to be met when designing the device. Furthermore, this will make the designers understand what the users want and what they need.

General Requirements:

- The device shall be used by one person
- The device shall not require wired connection with a computer
- The device shall detect a short/open circuit is less than 5 seconds
- The device shall be small enough to be held with 2 hands
- The GUI shall have a user friendly interface that requires no training to use
- The device shall have red LEDs for light indication
- The LEDs shall be bright enough to be seen up to 3 feet away
- The device shall have a warning for open/short circuit detection
- The device shall be powered by batteries
- The device shall be affordable for an average person

Possible Constraints:

- Power Supply
- Reliability
- Sustainability

- Cost
- Testing Process
- Time

2.6 House of Quality

The House of Quality shown in Figure 2 is a tool designed to reflect the customer's needs and how it relates with the product specification. Using this tool, we can analyze the qualities of our product to match that of the User's Requirements. In Figure 3, we have used the horizontal row to reflect the Engineering Requirements; and, vertical rows to reflect the User Requirements. These two rows cross with each other to show the correlation between the User Requirements and the Engineering Requirements. The roof represents the interaction between the different Engineering Requirements and how these requirements affect one another.

When analyzing our House of Quality, we have concluded that the most affected factor is the cost. The cost directly affects the quality and functionality of our product. For instance, we cannot increase the size and the number of LEDs without directly affecting the cost; and, by changing the cost we are indirectly changing the quality of the other parts in our product.

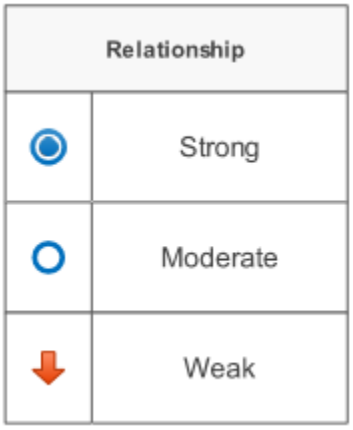

Figure 2*: House of Quality Legend*

Figure 3*: House of Quality*

2.7 Diagrams

2.7.1 Hardware Flow

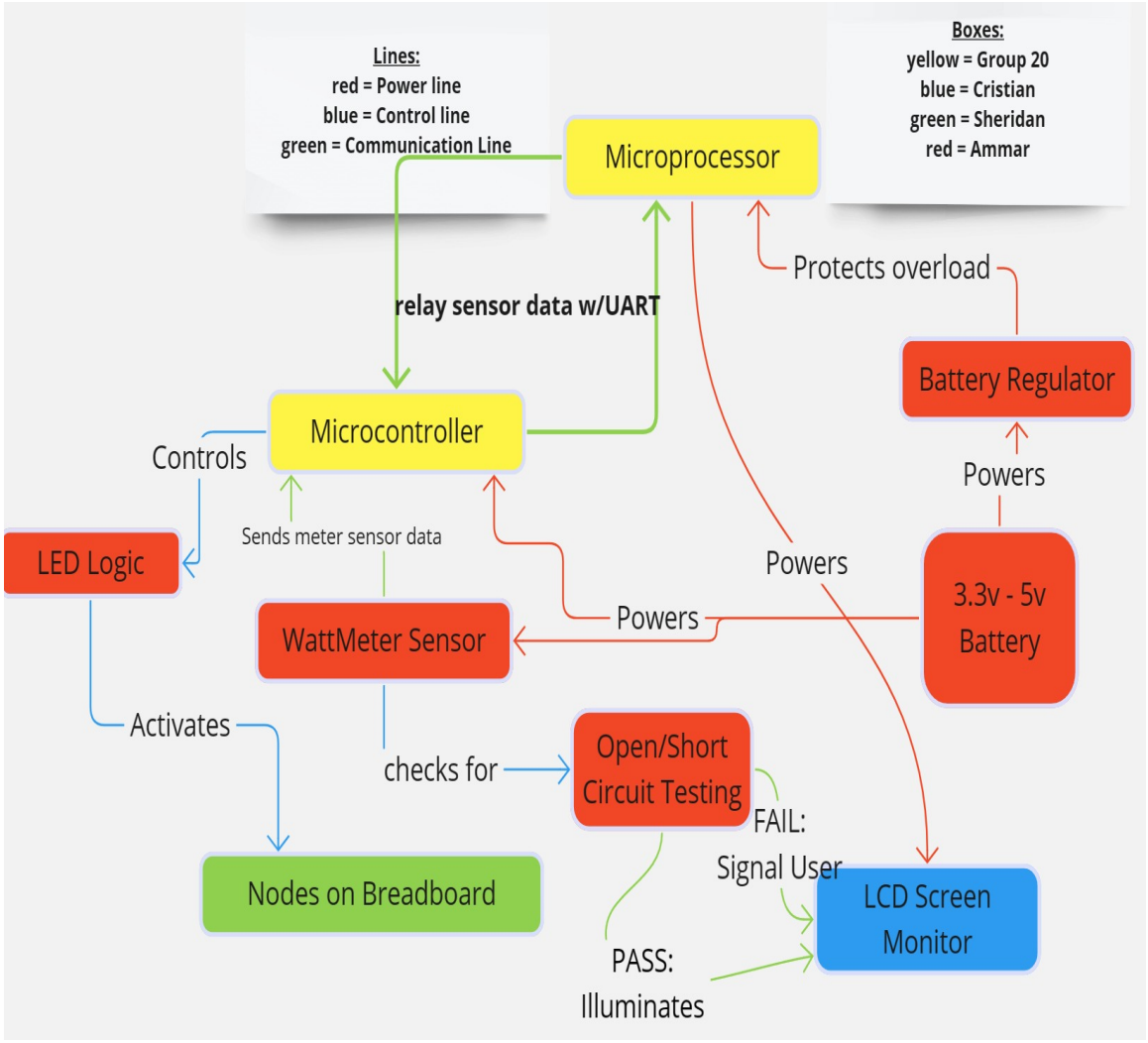

Figure 4*: Hardware Flowchart*

2.7.2 Software Flow

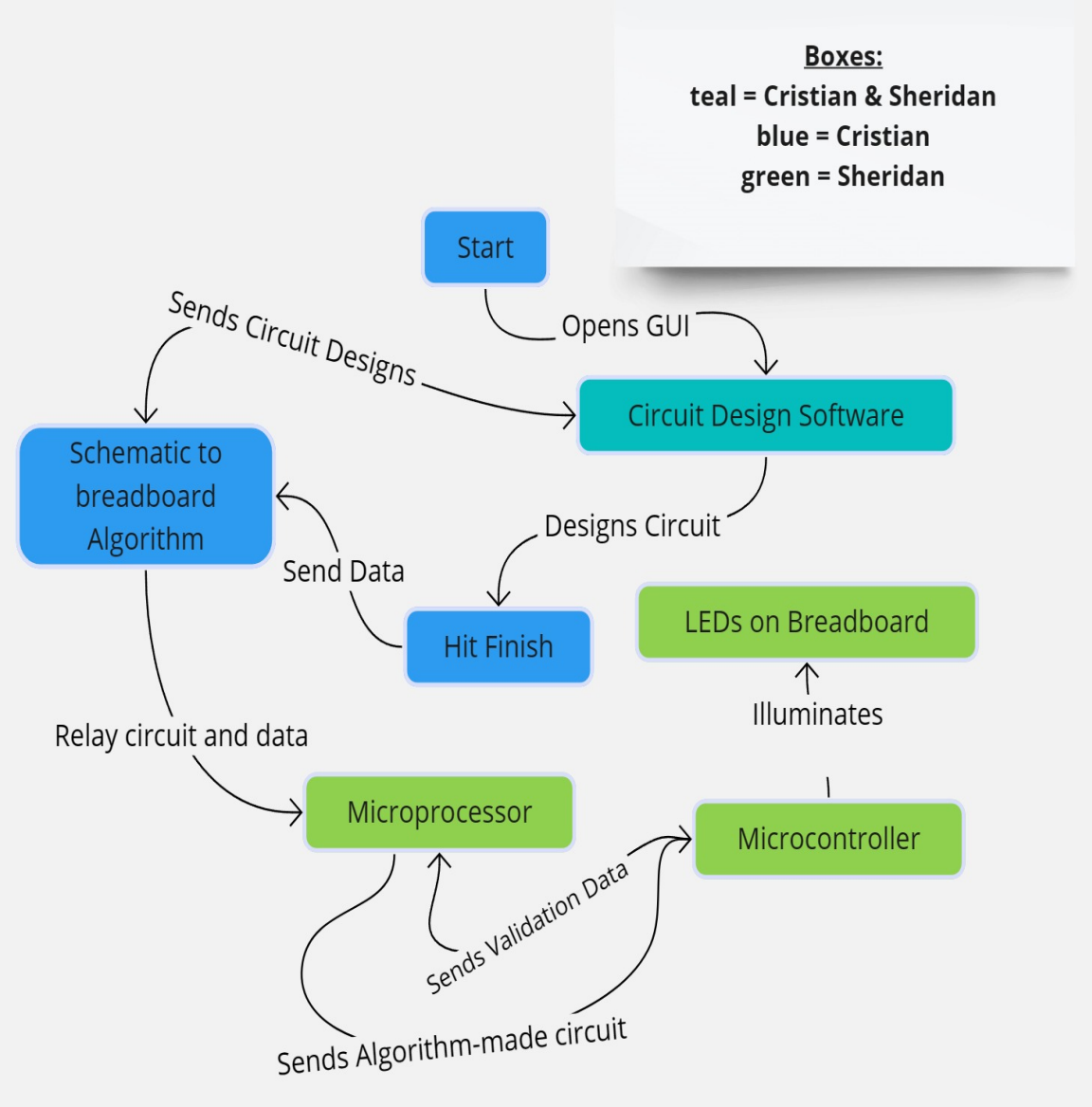

Figure 5*: Software Flowchart*

3. Research and Part Selection

3.1 Breadboard Design and Development

The breadboard will be designed and built to incorporate all of the customizations for the project. This new design will mimic the look of the typical solderless board, but will be able to house the PCB, Microprocessor, Monitor, and LEDs. The material will be chosen with weight, cost, and size in mind. The number of nodes will be limited to help maintain the smaller size. The nodes themselves will be spaced in order to allow components to stretch to them but not be out of reach. It will also be built to hold the components as a typical board will do to ensure the components share the node on the metal strip. The board will also take into consideration the LEDs and will decide their placement to ensure the user can easily see them and decipher which node it corresponds to. The breadboard size, color, design, and shape will depend on a number of factors.

A typical breadboard cannot be used since there are multiple components that must be incorporated into it to make it more portable. The breadboard design will be separated into layers to organize the different capabilities to be added. It will be important to gain access to the PCB since that is where the power supply will be. The board will have to be made out of a material that can handle these constraints. Breadboards are typically made of ABS Plastic [1] so this will be a strong contender. The breadboard must be made of insulator material to ensure the user's circuit is not affected. Some insulators to be considered are wood, plastic, and glass. The weight, electrical resistivity, accessibility, price, and ability to be manufactured will be the factors in determining which material to build the breadboard out of. Wood does have weight and it is difficult to work with thin pieces of wood because it won't be sturdy enough. It is also not easily manufactured for the shapes and design that are required. For these reasons wood will not be considered. Glass will not be considered because of its manufacturability and cost. It would also incur a lot of weight and will be very fragile for the final design. Plastic, however can be manufactured using easily accessible 3D printers, are cheap, and easy to design with on CAD.

There are different types of plastic that can be 3D printed but the three most popular are Acrylonitrile Butadiene Styrene (ABS), Polylactic Acid (PLA), and Polyethylene Terephthalate Glycol (PETG). These are more commercially accessible so they will be considered. The columns in the table below were chosen to be the most important factors. The Shore measurement is important because the breadboard should not be flexible or soft. The Dielectric Constant must be highly considered because the material needs to avoid messing with the user's circuit. The temperatures were taken into consideration because components may heat up and the environment itself may be warm given the current climate. It is also important to have because the printer will need to be able to handle that type of temperature. Based on these considerations in Table 1 it is determined that ABS is the preferred material because it has a high electrical resistivity and higher processing/melting temperatures. It is also the material that is most typically used for commercial solderless breadboards so it can handle the task.

Plastic Type	Hardness (Shore)	Dielectric Constant	Melt Temperature
ABS	76 [5]	2.19-2.9 [6]	437℉-473℉ [5]
PLA [2]	83	1.7-2.8 [6]	293℉-320℉
PETG	70 [3]	2.6 [4]	410℉-455℉ [3]

Table 1: *Breadboard Plastic Material [2], [3], [4], [5], [6].*

With the board being made out of plastic, it can be designed with different layers and different colors. The layers will include the PCB housing on the bottom, a layer for LED placements, then a top layer for the nodes which can be translucent to allow the light to show through. The bottom part of the layers will be white to stay with the standard color of other solderless breadboards. This makes debugging easier for the user because the white background and the colors of the components will contrast well.

The number of nodes will decide what level of circuits can be built. This breadboard will be primarily a teaching tool and therefore will not be equipped for large, advanced circuits. Rather, this board will only use a handful of nodes to ensure accuracy, speed, and simplicity. Having fewer nodes will allow for higher accuracy since there will be less nodes for the short/open circuit test. The speed will be higher for fewer nodes since the circuit planning algorithm will have less possibilities for the diagrams. The simplicity will be better with fewer nodes because the user will not need to worry about trying to build a complex circuit without having a good understanding of the process. One node will not be considered because a circuit requires a minimum of two nodes. To allow for a variety of circuit designs, three nodes will be implemented.

The next consideration for the nodes will be how to connect the nodes with metal contacts. There are two things that will restrict the placing of the components. How long the components themselves can be stretched and the type of metal strips that will be used to connect a node. The connectors are long pieces of metal that contain multiple clips to hold the components in place and connect them to one node [7]. There are a few options for the metal clips which include scrapping them from regular breadboards or designing something that can be made using regular metal clips.

The first method would be more reliable and require less parts for assembly. The metal contacts in commercial solderless breadboards are mass produced and would have the highest quality of the options. Since most commercial metal contacts contain five holes per node, two of those clips will need to be removed. Removing the clips from the board will require more work as they are not built to be disassembled. Using those metal contacts will also limit the spacing of the nodes on the redesigned breadboard. Some extra spacing between the nodes could increase the accuracy for the user because they won’t get confused by the closely placed holes. The second option is the clips can be customized and manufactured. This can be a challenge and will require an advanced knowledge of CAD because of the nature of the clips. The metal clips will need to be designed to hold the component in place and be small. This part would be 3D printed and

a metal strip would need to be attached. All of the designing, testing, and machining may take more time than would be necessary. The third option would be to take the metal contacts from miniature boards because that would avoid the issue of removing the two extra contacts like in the first option. However, the contacts in the smaller boards are different than that of larger boards and may be more difficult to integrate into the new design. These options are seen below in Table 2.

Method	Cost	Time	Customizable
Removing metal contacts from solderless breadboard	<$10	5 minutes	Fits the design. Must remove two nodes
Designing/Manufacturing	<$10	48 hours	Made to specifications
Removing from smaller breadboards	<$10	30 minutes	Correct number of nodes, metal contacts need altering

Table 2: *Breadboard Metal Contact Design Options.*

This project will move forward with sourcing the clips from already manufactured breadboards which are easily accessible. It takes less time than sourcing from the smaller breadboards, and the contacts are made in a way that can be easily integrated into the board. Two of the contacts for a node may need to be removed but that process will not require any advanced skill or tools. A commercial solderless breadboard will have more than enough metal contacts required for the project with enough for testing and prototyping and final submission.

3.2 Microprocessor

In the context of the Advanced Breadboard project, microcontrollers and microprocessors play a pivotal role in enhancing the functionality and user experience of the breadboard. The following devices have been selected as options to be an integral part to the project.

Our objective is to select one from these devices to develop a graphical user interface (GUI) that allows users to translate their circuit schematics into physical components on the breadboard. Additionally, the devices will aid in facilitating the LEDs of specific nodes on the breadboard. However, our consideration extends beyond just the physical size of these devices. We must also consider their operating voltages. Also, the output voltage pins of these devices hold significance.

In summary, the microprocessor selected for the Advanced Breadboard project will serve as the brains behind the system, enabling an intuitive GUI, node illumination, and ensuring the safety and efficiency of the redesigned breadboard for circuit design and integration. Here are the most viable breadboards in question:

3.2.1 Raspberry Pi Zero W

The Raspberry Pi Zero W is a compact yet capable microprocessor board that features a 1 GHz single-core ARM1176JZF-S CPU and 512MB of RAM. It's well-suited for your advanced breadboard project, offering sufficient processing power for tasks like running a graphical user interface (GUI) to assist with circuit design. It includes built-in Wi-Fi and Bluetooth for wireless connectivity, making it convenient for data transfer between the breadboard and a laptop. With 40 GPIO pins, it provides ample options for interfacing with the breadboard's components. However, it's essential to note that it consumes more power compared to the ESP32, which should be considered for portable applications.

3.2.2 Raspberry Pi 4B

The Raspberry Pi 4B comes in multiple RAM configurations (2GB, 4GB, or 8GB) and is equipped with a powerful quad-core ARM Cortex-A72 CPU. This microprocessor offers substantial processing power, making it suitable for complex tasks in your advanced breadboard project, including running a sophisticated GUI. It runs Linux distributions and provides a full desktop environment, making it versatile for software development. Dual-band Wi-Fi and Bluetooth connectivity add flexibility for data transfer and remote control. With 40 GPIO pins, it enables seamless integration with the breadboard. However, the Pi 4 consumes more power compared to the ESP32, which may require a reliable power source.

3.2.3 ESP32

The ESP32 microcontroller is known for its power efficiency and portability. It features a dual-core Tensilica LX6 CPU running at up to 240 MHz and 520KB of SRAM. While not as powerful as the Raspberry Pi models, it's suitable for basic tasks in your advanced breadboard project. It can handle a simplified GUI and offers built-in Wi-Fi and Bluetooth for wireless communication and remote control. With its GPIO pins, it facilitates easy interfacing with the breadboard's components. One of its key advantages is low power consumption, making it an excellent choice for portable and battery-powered applications. However, it may struggle with more complex GUI requirements compared to the Raspberry Pi boards. Table3 and Table 4 below shows the comparison breakdown.

Microcontroller/Microprocessors	**RAM**	**Pros for Project**	**Cons for Project**
Raspberry Pi Zero W	512MB	Offers ample ram for the project.	Maybe overkill for a simple breadboard project.
Raspberry Pi 4B	2GB, 4GB, or 8GB	Provides plenty of RAM for complex tasks.	May be overpowered and more expensive.

ESP32	520KB SRAM	Suitable for basic circuit design tasks.	Limited RAM; may struggle with complex GUI.

Table 3: *Project-Specific Comparison.*

Microcontroller/Microprocessors	**Pros for Project**	**Cons for Project**
Raspberry Pi Zero W	Offers a full Linux environment.	Higher power consumption compared to ESP32
Raspberry Pi 4B	Powerful, suitable for a wide range of tasks.	High power consumption may need active cooling.
ESP32	Extremely power-efficient and portable/	Limited computational capabilities compared to Pis

Table 4: *General Aspect Comparison.*

The Raspberry Pi 4B stands out as the ideal choice among the three devices for the Advanced Breadboard project. Its substantial processing power ensures smooth operation of the graphical user interface and complex algorithms for translating schematics to physical components on the breadboard. With multiple connectivity options, including USB ports, HDMI, Ethernet, and a built-in Wi-Fi (not necessarily guaranteed to be used, but there regardless, displaying further flexibility), it provides the flexibility needed for interfacing with the advanced breadboard and handling software-hardware integration. The Pi 4's capabilities also enhance the GUI's performance to be able to use the monitor to its full potential, offering a better user experience. Additionally, its popularity and reputation is showcased and supported by a large and active community, allowing for customization and access to a wealth of resources. Compatibility with various accessories and sensors, coupled with its longevity and continued support, make the Raspberry Pi 4B the most suitable choice to develop the advanced breadboard project, ensuring efficient development and long-term viability.

3.3 Raspberry Pi 4B

3.3.1 Operating System

The raspberry pi is a very common product that a lot of users have experience with and purchased. As such, it has a large community of people who have developed many software and even OSs that are open source. One of the most popular software is the Raspberry Pi OS because of its many capabilities. It is an optimized system designed for the Raspberry Pi to facilitate project building. Its compatibility spectrum, spanning

various libraries and tools, ensures rapid software development and a frictionless connection between hardware and software. Additionally, it has a lot of technical documentation in case of the need to dive deeper into the understanding of the operating system. This will be an important technology to be considered for the project because the software side will require housing to interact with.

3.3.2 Application Layer

The application layer can house applications tailored for our project, sculpted in Python. This allows the designers and the users to access the program without hassle. With Python's expansive libraries and Raspberry Pi's intrinsic support, we can ensure optimal control over hardware components. This capability can help with the project's objective of maintaining accuracy and efficiency because there is more control of the device with this service.

3.3.3 Connectivity

The project will leverage devices like the ELECROW 5 Inch Touchscreen. This way, direct connectivity will likely negate the need for Wi-Fi because the user can manipulate the program from the device rather than requiring their own separate device. Setup becomes easier for the user because less hardware will be required. Hence, implementing robust security protocols for data sanctity remains paramount, Wi-Fi-based standards like MQTT may be circumvented due to the direct hardware linkage as provided by the ELECROW 5 Inch package. It will also be important to consider the user when interacting with the touchscreen because it should be accessible and should not require strained effort to use or see.

Another option could be a wired connection between a screen monitor and the Raspberry Pi through HDMI. This type of connection would be reliable and would allow the user to change the type of monitor for their needs since HDMI ports are widely used. It is important for the project to be accessible to users so this adaptability would be a benefit. However, having a wired connection will cause the project to be less portable since the wire would be a limiting factor.

3.3.4 External Devices

One of the pivotal external devices enhancing our project's user experience is the ELECROW 5 Inch Touchscreen. This device offers:

· A 5-inch high-resolution display, boasting 800x480 pixels, ensuring clarity and a vivid representation of our software interface.

· Broad compatibility, including Raspberry Pi 4B, 3B+, 3B, and 2B+. This offers flexibility for future adaptations and iterations of our project.

· A resistive touch screen, enhancing the user's interactive experience. The accompanying touch pen ensures precision during operations.

· The provision for manual backlight control, empowering users to manage power consumption based on their requirements.

· While it has been primarily designed for Raspberry Pi integration, its compatibility extends to standard HDMI devices. This positions our project to have a wider application, even beyond the Raspberry Pi ecosystem.

· A comprehensive package that includes not just the screen, but essential accessories like the touch pen, HD adapter, and copper pillars, ensuring hassle-free setup and installation.

3.3.5 GUI (Graphical User Interface)

Envisioning a user interface that encapsulates both simplicity and functionality is one of the most important parts and goals of our software design. Drawing inspiration from the well-established Multisim:

· The GUI will open to an open breadboard layout. This initial setup stage is for the user and it takes them directly into the circuit creation phase.

· Users can drag and drop components onto the breadboard. This interactive method simplifies the learning curve and augments the hands-on feel of circuit design.

· A library of components will be available. Users can easily search, select, and integrate these components into their design.

· To enhance user experience and reduce clutter, unnecessary toolbars and options prevalent in traditional Multisim templates will not be there, offering a clean, concise, and focused design workspace. This is also as a result of the difference in product which naturally requires more simple/less features.

· Incorporating these features ensures our GUI not only resonates with experienced Multisim users but also welcomes newcomers with an intuitive and user-friendly design. This is especially key to users such as students who have had to deal with the original breadboard.

3.3.6 Error Handling

In the realm of software development, especially when interfacing with hardware like the Raspberry Pi 4B, error handling is extremely key. A well-structured error handling system not only identifies and captures errors but also provides meaningful feedback to users, ensuring a good experience with the product. Following the IEEE 1044-2009 standard on classification for software anomalies, our approach is to categorize, prioritize, and manage errors effectively. By implementing this standard, we can swiftly find errors/bugs, classify them based on severity, and address them in a structured and organized manner. The primary objectives are to:

· Inform the user gracefully about any issues, ensuring they are not left in the dark.

· Log errors in a manner that aids in swift diagnosis and resolution.

· Ensure that critical functions have fallbacks, so the software can continue operating even when facing minor glitches.

3.3.7 User Experience (UX) & User Interface (UI) Design

The fusion of an intuitive user interface (UI) with a seamless user experience (UX) is the key of any successful software and especially an intuitive or easy one. It's not just about aesthetics though, it's about crafting a user environment where functionalities are easily accessible, and interactions feel natural. The commitment to the ISO 9241-210 standard, which emphasizes human-centered design for interactive systems, is going to focus on the end-user. By implementing this standard, we ensure:

· A solid understanding of user requirements, tailoring our software to fit their needs.

· Accessibility, making sure that our software can be used effortlessly by everyone.

3.3.8 Testing

Following the guidelines of IEEE 829-2008, our testing won't be haphazard but systematic. This approach allows the Raspberry Pi's software to always deliver reliably when communication to hardware.

3.4 Open/Short Circuit Detection

A crucial component in the Advanced Breadboard project is the ability to detect short circuit and open circuit within the circuits designed by the users. As mentioned in the previous sections of this report, the Advanced Breadboard will be a newly improved version of a regular breadboard that is commonly used in engineering laboratories. A common mistake that occurs when using a breadboard is when components become short circuited or open circuited without the users knowledge. This can be done without intention and is something that could be easily missed by the user; so, the solution we created is the ability to detect when the user’s circuit is short circuited or open circuited; and, warn the user when the condition is true.

The main method we have found that can be utilized in the Advanced Breadboard is the use of Voltage Sensors and Current Sensors. The voltage sensors and current sensors would connect to the main power supply and also connect to an A/D Converter to process the data given by the sensors. If the current is equal to zero then the circuit is open, if the voltage is equal to zero then the circuit is shorted.

3.4.1 Voltage and Current Sensor

The outline of this section is to discuss the Voltage and Current Sensor component. This component has the combination of both a Current Sensor and a Voltage Sensor. The voltage and current sensor is the primary component for the open/short circuit detection.

This component would connect from the Variable Power Supply, which would be the user's power source for their electrical circuit, and connect to the microprocessor. Since most of these sensors already contain a Digital to Analog converter, and the microprocessor we plan to use also contains an A/D Converter then it is not necessary to include an A/D Converter to the PCB design.

3.4.1.1 INA219AID

The Texas Instrument INA219 Zerø-Drift, Bidirectional Current/Power Monitor With I2C Interface is a unique sensor that measures the voltage, current, and power of various electronic equipment. For the purpose of this project, we will only be examining the functionality and accuracy when measuring both the voltage and current of a power supply unit for the open/short circuit detection functionality implemented in this design. This sensor can measure electrical equipment within 0-26V and 8A with a maximum relative error rate of less than ±0.2%.; and, is typically powered by a separate power supply with a voltage between 3V-5.5V. Applications of this sensor include Battery Chargers, Test Equipment, and Power Supplies which make it the ideal sensor that we need since we may need to use all three of these equipment.

The INA219 utilizes an I2C or SMBUS-compatible interface. However, the I2C interface is mostly used throughout the datasheet while the SMBUS Protocol is used when a difference between two systems is addressed. Which is why for this design we may be utilizing the I2C interface for voltage and current sensing applications.

3.4.1.2 INA260AIPW

This sensor is also manufactured by Texas Instrument and offers a ton of digital-output monitoring. For starters, this sensor can measure current, power, and voltage with an I2C and SMBus compatible interface. The sensor enables high accuracy current and power measurements as well as offering an over-current detection at common-mode voltage varying between 0-36V. Just like the other sensors, it offers an Analog to Digital Converter; and its digital interface allows for programmable threshold alerts, ADC conversion times, as well as averaging.

3.4.1.3 INA233AIDGST

This Texas Instrument sensor measures current, voltage, and power with an I2C, SMBus, and PMBus interface. Independent of the supply voltage, the sensor senses current on common-mode bus voltages that vary between 0-36V. The biggest advantage of this sensor is its .1% Maximum Gain Error which results in a highly accurate sensing. This may be a little too much for the purpose of this project, but it is still something to consider. The compatibility between the three interfaces gives it a huge advantage compared to the others. Another feature this sensor has is power detection. Although we may not need this feature since we are more interested in voltage and current, it is still something we may need to consider if we plan to improve the Advanced Breadboard design.Finally. It has one of the fastest lead times coming in at six weeks. Table 5 below shows this comparison.

	INA219AID	INA260AIPW	INA233AIDGST
Cost	$2.70	$6.38	$4.22
Operating Voltage	3-5.5V	2.7-5.5V	2.7-5.5V
Sensing Voltage	0-26V	0-36V	0-36V
Accuracy	.5% Maximum (over)	.15% Maximum System Gain Error	.1% Gain Error (Max)
Interface	I2C, SMBus	I2C, SMBus	I2C, SMBus, PMBus
ADC	Yes	Yes	Yes
Lead Time	6 Week	8 Weeks	6 weeks

Table 5: *Sensor Comparison.*

Based on the table comparison and the short analysis we made on the different components, we have decided to go with the INA219AID sensor. For the purpose of this project we need two conditions to satisfy the short/open circuit detection. The first, when the current is equal to zero. The second, is when the voltage is equal to zero. Based on those two reasons, accuracy of the sensor becomes negligible after a certain point. Although the INA233AIDGST is the most accurate sensor out of all the choices, it is a bit of an overkill for the purpose of the Advanced Breadboard's open/short circuit detection. Furthermore, the maximum output voltage by the user's power supply is 15v. So having a sensor that has a higher sensing voltage does not pose any benefits. Having a six week lead time as well as being cost effective are both benefits that the INA219AID has over the other two sensors. For these reasons, we concluded the part comparison for the Voltage and Current Sensors and have decided that the most optimal choice is the Texas Instrument INA219AID sensor.

3.5 Microcontroller

The Raspberry Pi 4B has the capability to handle the GUI and interact with the sensors and LEDs. However, the Elecrow Display uses Pins 1-26 on the Raspberry Pi 4B, and the Wattmeter Sensor requires I2C communication. The I2C pins are being covered by the

display. A set of options will be considered on how to handle including the sensor. While also considering how this will be incorporated into the PCB.

A Daisy-Chain connection can be done to connect the Wattmeter Sensor to the Raspberry Pi 4B I2C pins. This option will reduce the number of controllers and components that would need to be programmed. However, a more advanced program to handle the I2C will need to be written and tested. Having the sensor data is integral to the project. This would also require that all of the LEDs are wired to the Raspberry Pi 4B. This possibility would make handling the LEDs simpler, but the Raspberry Pi 4B would need to be wired to nine LEDs. This is possible, but can be a hassle handling that many wires on one device.

Another option would be to use a controller that will handle the Wattmeter Sensor and the LEDs. The device would not require processing power like a Raspberry Pi, rather, a microcontroller would be well suited. An earlier section discussed the ESP32 so that will be considered here. The MSP430 will also be considered since that controller is familiar to all members of the team. This method would also need to communicate to the Raspberry Pi 4B since that will be housing the program. Both of these devices have I2C capability and have enough pins for the LEDs. That is why these two will continue to be considered. Table 6 makes this more clear.

Device	**Cost**	**Resources Available**	**Experience**
ESP32	$5-$10	Has libraries for Wattmeter Sensor	Familiar with some features
MSP430Fr6989	$20-$50	Notes from Embedded Systems	Familiar with most features

Table 6. *Comparison of Microcontrollers.*

There is a constraint on the project that the PCB needs some sort of controller so the method of putting everything on the Raspberry Pi 4B will be avoided. Therefore, a microcontroller will need to be chosen. Between the two, the ESP32 will be implemented because it has a lower cost and has more resources available. There are two purposes to ESP32, control the LEDS and receive data from the sensor. SInce this device has specific libraries to handle the sensor then this will make the processes more efficient. The EP32 also has UART capability that will provide an option for communicating with the Raspberry Pi 4B.

3.6 Power Supply

Powering the components of our Advanced Breadboard is an important topic that needs to be discussed. The power supply we choose impacts the rest of the design and the components we choose to use throughout the project. We intend to adhere to the environmental, health, and economic constraints discussed in the following sections of the report. Since the Advanced Breadboard utilizes multiple components to fulfill its

purpose, we need a power supply adequate enough to power the product and keep it portable. To come up with a proper conclusion we need to figure out the amount of operating voltage needed to power all the components in our product.

After analyzing the various components and Power Supplies for our design, we have found that the best option is to separate the Power Supplies for the PCB and the User's Power Source for their circuit. Firstly, most components used in our design need an operating voltage ranging between 3V-5.5V. If we utilize a Variable Power Supply that can output a voltage between 0V-3V then the Advanced Breadboard's components will not function, but it will still power the user's circuit. Additionally, if the Variable Power Supply outputs a voltage higher than 5V then we can use a voltage regulator to reduce the voltage when powering the components to a voltage between 3V-5.5V. Either way, if we combine the Power Supply to power both the user's electrical circuit and the Advanced Breadboard's PCB the design could be expensive, impractical, large, and time consuming. In conclusion, we have found that separating the Power Supplies will pose the best case scenario.

3.6.1 Variable Power Supply

This section will cover a variable power supply that can adjust the voltage output. This is important because we want the user to have the freedom to choose the voltage input for their circuit. Based on further research we needed to decide whether to use that same variable power supply to power the PCB. This includes all of the components connected to the PCB as well.

The disadvantage with this setup is the voltage range that we need to set. Since one of the environments we are analyzing are university engineering laboratories, we have found that the most common maximum voltage input designed with breadboards is 15V. Another reason to limit the maximum input voltage is due to the voltage and current sensors having a limitation. Most voltage sensors can only measure voltage that is around 4-5x the input voltage. Furthermore, we need a power supply that does not pose a fire hazard when shorting the voltage terminal pins on the breadboard. For this reason, we have decided to use a Variable Power Supply to power the user's electrical circuit. The subsections below will discuss the options we have for a variable power supply while the table below will compare and contrast the power supplies to determine which one is the most suitable part for the Advanced Breadboard.

3.6.1.1 Elenco XP-15K Power Supply Variable Kit

The Elenco XP-15K Power Supply is a very common power supply used by many students and lab instructors in university laboratories. Its affordable price and low voltage output makes it a very valuable tool in an engineering laboratory. According to the manual, the power supply contains a transformer that transforms the voltage from 120V to 18V, an AC to DC converter that converts the 18VAC to 20VDC, and a voltage regulator that changes the voltage between 0V to 15V making it one of the top candidates for the ideal power supply that we need. Although we cannot adjust the output current, the manual specifications state that at 12V the output current is .3A and at 15V the output

current is .2A which is something we need to account for. Furthermore, when the power supply is shorted it will turn off and when the short is removed it will recover to full functionality.

3.6.1.2 Korad KD3005D Power Supply

The Korad KD3005D Power Supply is an industrial-grade performance power supply with an affordable price. The power supply offers a system lockout to prevent any unwarranted changes to the voltage and current outputs once they have been set. This power supply offers an output voltage range between 0-30V which is more than double the intended maximum voltage we are searching for. Furthermore, the power supply offers an output current between 0-5A which may be unnecessary considering the scope of the project. The advantage of this power supply is that it offers a full digital control of the voltage and current output and displays using the LED display which allows for easy adjustment to the output voltage and current at the user's discretion.

3.6.1.3 Matrix MPS-3206 Series

The Matrix MPS-3206 Series Power Supply is a high accuracy dc power supply with an output voltage ranging between 0-32V and an output current between 0-6A. Although the price is higher than the Elenco XP-15K, it compensates by having a digital display and adjustable current analog unlike the Elenco XP-15K. The digital display consists of a double four-digit LED display that displays both the voltage and current which can be helpful for the user designing their circuit. Furthermore, the power supply comes equipped with a smart fan to regulate the temperature during long work loads which makes it ideal for laboratory environments. Since we are planning to make the Advanced Breadboard transportable, this power supply is designed to be ultra lightweight weighing in about 1.5kg. Lastly, the power supply offers a short circuit safety measure to prevent any damages to the power supply in the case of a short circuit. Table 7 below provides an organized way of displaying this information.

	Elenco XP-15K Power Supply	**Korad KD3005D Power Supply**	**Matrix MPS-3206 Series**
Cost	$34.99	$89.99	$69.99
Output Voltage	0-15VDC	0-30VDC	0-32VDC
Output Current	.3A @ 12V .2A @ 15V	0-5A	0-6A
Short Circuit Safety	Yes	Yes	Yes

Lead Time	1 Week	8 Weeks	5 days
Digital Display	No	Yes	Yes

Table 7: *Comparing Power Supplies.*

After analyzing the different power supplies we have found that the best and most ideal power supply would be the Elenco XP-15K Power Supply. Although there are some downsides to the Elenco compared to the other two power supplies, its benefits far outweigh its downsides and for those specific reasons is why this came out as the ideal power supply for the Advanced Breadboard.

For starters, this power supply will power the user's electrical circuit and for the most case the highest voltage we would need is under 15V. We came to that conclusion by analyzing the current and voltage sensors, most of them specifically state that an input voltage higher than 15V may alter the accuracy of the sensors. So, choosing a maximum voltage output of 30VDC or 32VDC is a bit of an overkill for this project. Secondly, being able to adjust the output current is a benefit; but in the case of simple breadboard circuit designs by the user, that functionality is not a necessity. Thirdly, the fast lead time is another huge benefit. The earlier we get the power supply, the earlier we can start prototyping and designing the Advanced Breadboard. In conclusion, due to the reasons mentioned in this section, we have decided that the most ideal power supply to use for the User's electrical circuit in the Advanced Breadboard is the Elenco XP-15K Power Supply.

3.6.2 Battery Pack

The battery pack we choose will be used to power the Advanced Breadboard's PCB along with the components connected to the PCB. Since the PCB requires 3V-5.5V to power all the components, we figured that a battery pack will be more than enough to supply the power we need. The subsections below will describe the battery selection we have chosen, as well as the benefits and downsides to the battery pack we have selected. Additionally, There is no specification for rechargeable and non-rechargeable batteries. So, we decided to research which battery pack will provide the most benefit to this project.

We had originally planned on using a rechargeable battery; however, we took into consideration the portability of the device. For instance, if the user would like to transport the device to another facility or to another workstation, we want the battery to last a lot longer so that there are no interruptions with the circuit the user simulated. Unfortunately, rechargeable batteries contain significantly less capacity than most alkaline batteries. Furthermore, we wanted the device to stay within budget while also providing the user with an affordable device. Implementing a rechargeable battery, battery charger mechanism, and the battery charger accessory would have driven the price of the

Advanced Breadboard significantly and proportionally have the user spend more when it comes to replacing the charger, batteries, or repairing the charging mechanism. Secondly, we thought ahead in terms of battery replacement and battery repair. Although, we could've made it simple to replace the rechargeable battery; when we analyzed the advantages and disadvantages of using rechargeable batteries we found that Alkaline batteries posed as a better option for the simplicity of our design.

Since there are many types of batteries, we decided to explore the types and evaluate which battery type will offer the best performance for this project. Alkaline batteries utilize zinc metal and manganese oxide instead of acidic electrolyte to drive the energy. The advantages for this type is a higher shelf life, smaller size, and low leakage. Another common battery type that is mainly used in mobile computing and other applications is the Lithium-Ion. This battery type utilizes Lithium-Ion cells to generate high energy density and low discharge. The last type of battery we analyzed is the Zinc Carbon batteries. This type offers less performance than the alkaline batteries, but can be suitable for devices such as remote controllers, and clocks. Table 8 summarizes all of this below.

	Alkaline	**Lithium-Ion**	**Zinc-Carbon**
Capacity	850-1200 mAh	1150mAh	1100 mAh
Self-Discharge	2-3% per year	1-2% per month	.32% per month
Shelf-life	5-10 years	10-12 years	1-3 years
Temperature Performance	4-129°F	-4-130°F	23-113°F
Internal Resistance	High	Low	High

Table 8: *Battery Type Comparison.*

Based on the research and the table comparison we have found that the most ideal battery for the purpose of this project is the Alkaline batteries. The availability and popularity of this battery type partially influenced our decision since we are trying to comply with the economic constraints that were placed on this design. Furthermore, the battery capacity of a single AAA Alkaline battery is within range of our design compared to the capacity of the other battery types. The major downside to this battery is the safety concern with a short circuit, due to no short-circuit safety measures the battery could potentially cause harm to the user. This can be avoided due to the safety risk assessment we plan to conduct as well as the risk management plan we have in store. Another benefit

that this battery type possesses is the long shelf life and the low self-discharge. This will ensure that the batteries last longer than the other battery types.

3.6.2.1 Duracell AAA Batteries

The Duracell AAA Alkaline battery is a common household item due to its usefulness. Although this battery is non-rechargeable, each battery packs 1.5V and a charge capacity of 1150mAh which is a lot compared to the average rechargeable batteries. The high charge capacity will make it easier to keep the Advanced Breadboard running for a long time, and if the batteries drain, replacing the batteries would not be an issue. Its small size and lightweight is an asset to this project since we intend on making the Advanced Breadboard compact and portable.

3.6.2.2 EN91F3

The EM91F3 battery is a unique Alkaline battery pack manufactured by Energizer. This battery pack utilizes 3 AA battery cells to generate 4.5V which is in the range that we need for the PCB. It weighs at around 64 grams and its volume is around 2 cubic inches which makes it ideal for the Advanced Breadboard size requirements and weight requirement. The only downside is that we need to implement a way to replace the battery pack since it uses a solder tab termination style which can pose a design constraint.

3.6.2.3 Zeus AA

The E92VP is a AA 1.5V Alkaline battery that is manufactured by Zeus Battery Products. This non-chargeable battery offers a capacity of 3000mAh and weighs at approximately 23.8g. A few downsides that this battery has is that it is harder to obtain compared to the other batteries since it is not a very common brand for an alkaline battery. The fact that it is a AA battery means that it can not be used as an alternative battery to the Duracell AAA batteries, for that reason we need to keep in mind the battery we choose since the batteries are non-rechargeable. Since they are not rechargeable we need the user to have an easy and accessible way to replace the battery in case it is depleted. All the devices are described in Table 9 below.

	Duracell AAA Batteries	**EN91F3**	**ZEUS AA**
Cost	$6.195	$3.22	$6.60
Quantity Needed	3	1	3

Voltage	4.5V Total	4.5V	4.5V
Capacity	3450mAh Total	2779mAh	3000mAh
Rechargeable	No	No	No
Lead Time	1 day	5-7 days	12 Weeks

Table 9: *Battery Pack Comparison.*

Based on the table and short descriptions of the batteries we researched. We have found that the best and most ideal battery pack to use is the Duracell AAA Battery Pack. Although these batteries are non-rechargeable, it surely compensates in other ways compared to the rest of the batteries. Due to this battery being inexpensive and a common household item, we have found that the availability for this battery is endless and replacing these batteries are much simpler than the others. Since we are using three of the Duracell AAA batteries, we would need a combination of three batteries to generate 4.5V thus powering the PCB and all its components. Furthermore, the capacity of all three batteries would total to 3450mAh which would be long enough to keep the PCB powered longer than the other batteries we researched. The total cost of these batteries would total to around $6.195 on average which keeps this whole project within the budget and satisfying the financial requirements that were identified in the beginning of this report.

3.7 Voltage Regulator

A Voltage Regulator is a crucial component in a circuit that maintains a constant output voltage regardless of any changes to the input voltage or load conditions. When it comes to voltage regulators they are commonly used for DC/DC conversion while some are used for AC/DC or AC/AC conversion. For the purpose of the Advanced Breadboard, we will be using a DC/DC conversion voltage regulator to take a DC input from a battery source and convert it to a constant voltage output that will be used to power the PCB.

Of this type there are two types of voltage regulators, the first is known as a linear type while the second is a switching type. The Linear Voltage Regulator, are referred to as step-down converters since the output voltage is always lower than the input voltage. Switching Voltage Regulators are referred to as step-down converters, step-up converters, or a combination of both. The input voltage can vary while keeping the output voltage constant. Since we are using Alkaline AAA batteries we will be using a switching voltage regulator since our input voltage may vary between .8-4.5V due to the discharge of the batteries. The following subsections will analyze the different switching voltage regulators and we will choose which regulator will be the most ideal for the purpose of this project. Table 10 below describes all of the comparisons.

3.7.1 TPS82150SILR

The TPS82150SILR is a step-down converter that integrates a step-down converter and an inductor to simplify design and save PCB space. A unique thing about this regulator is that it can automatically enter Power Save Mode to operate at light load currents. Although this regulator may seem like a viable option, the downside to this regulator is that the minimum input voltage is 3V. This may pose an issue considering that we are using batteries that tend to discharge after a period of time. The maximum voltage that will be generated when the batteries are at full capacity is 4.5V and as it discharges the voltage will decrease. Although this regulator has many unique benefits, the minimum input voltage makes it the least ideal of the three choices.

3.7.2 TPS613221ADBVR

The TPS613221ADBVR is a boost converter that provides a power-supply solution for devices powered by alkaline batteries. Thanks to its low input voltage we do not have to worry about the batteries discharging past a certain point. Its small size and high efficiency make it one of the most ideal regulators in the market. Furthermore, its cost allows us to stay within the economic constraint and keeps our Advanced Breadboard within budget. Lastly, the size of the component will help save space on the PCB.

3.7.3 TPS61023DRLTT

The TPS61023DRLT is a boost converter that provides power supply solutions for single cell, two cell, or even three cell alkaline batteries. At low load of current the regulator will enter Power Save Mode to maintain a high efficiency, this feature can be disabled making the converter operate at a fixed switching frequency. At 90% efficiency the regulator is about to output a current of 300mA at 3.3V. Its extremely low minimum input voltage and output short circuit protection grants this regulator a high amount of advantages compared to its disadvantages. Lastly, the small size of this component also makes it ideal for a PCB design since the Advanced Breadboard has a size constraint.

	TPS82150SILR	**TPS613221ADBVR**	**TPS61023DRLT**
Cost	$3.92	$0.58	$1.18
Frequency Switching	2MHz	500kHz-2MHz	1.2MHz
Input Voltage	3V-17V	0.9-5.5V	0.5-5.5V

Output Voltage	.9-6V	1.8-5.5V	2.2V-5.5V
Quiescent Current	20uA	6.5uA	<55uA
Maximum Output Current	1.2A	1.6A	3A
Efficiency	No	>90%	>90%
Size	3.0mm x 2.8mm x 1.5mm	2.90mm x 1.30mm	1.2mm x 1.60mm
Lead Time	6 Weeks	6 Weeks	6 Weeks

Table 10: *Battery Pack Comparison.*

After analyzing the three choices and comparing them together, we have decided that the best choice is the TPS61023DRLT regulator. Since our Advanced Breadboard component requires a minimum of 3.3V to operate, we needed a regulator that can output the same amount of voltage. Furthermore, we plan to use three AAA Alkaline batteries to power the Advanced Breadboard's PCB as well as the connected components which output a maximum of 4.5V. Unfortunately, the batteries discharge after use reducing the amount of output voltage, the TPS61023DRLT requires a low input voltage which makes it a great solution to this issue. Another advantage is the manufacturer, Texas Instrument, designed a public tool that generates an ideal circuit using their parts when given an input voltage and an output voltage. This tool can be used to help design a regulator circuit that meets our design constraints and keeps us within the given project requirements.

3.8 LEDs

For the Advanced Breadboard we plan on using LEDs to reflect which nodes need to be occupied by the user. The plan is to use the GUI to have the users simulate a linear circuit. From there, the GUI will light up the corresponding node on the Advanced Breadboard. We need to search for an LED light that makes it clearly visible which node is activated, but at the same time we do not want it to be too strong that it overlaps to another node unintentionally. In the subsections below we will look at the different LED lights that we researched and choose based on the comparison table and the description of each LED light. Table 11 below has a breakdown of all the necessary information for the comparison to be completed.

3.8.1 LTST-C191KRKT

This LED is manufactured by LITEON OPTOELECTRONICS and offers an ultra bright and super thin AlInGaP Chip LED. This particular LED is 2mm by 1.25mm in size making it a great option to light up a node of a 1mm diameter hole in a regular breadboard. However, due to the design of the Advanced Breadboard, for practical and simplicity reasons we may increase the size of the holes of the Advanced Breadboard. The LED offers a water clear lens with the source color being red, which would make it easier indicators for the user to see. The only downside to this LED is that it may pose an issue when designing the prototype of the Advanced Breadboard on a breadboard due to the soldering terminals on the bottom. This means that this LED does not have pins that can be inserted into a breadboard but rather, we would have to solder on a wire in order to connect it to a breadboard.Although, this would be a great choice for a PCB design.

3.8.2 LTST-C171KRKT

This LED is also manufactured by LITEON OPTOELECTRONICS and offers an ultra bright and super thin AlInGaP Chip LED. The difference with this LED is that it is smaller in size compared to the one mentioned in Section 3.6.1. This may help to keep the light contained to one hole and not have it illuminate any adjacent holes causing the user some confusion. This LED also uses a clear water plastic lens and illuminates a red light. Unlike the other LED lights this one generates the least luminous intensity which is something to consider with how bright we want the LEDs to be. This will all depend on how the 3D printed molding will absorb the light. Just like the previous LED light this one also has a soldering terminal.

3.8.3 WP7113SURDK

This LED is different from the rest of the ones we researched. Manufactured by Kingbright, this LED is a Solid State Lamp LED with a Hyper Red source color made with AlGaInP Light Emitting Diode. The LED has a low power consumption and general purpose leads which makes it easier to hook onto breadboards for the prototype testing of the Advanced Breadboard. Additionally, the LED is bigger in size measuring at 5mm in diameter which would be helpful if we plan on making the Advanced Breadboard holes bigger than the traditional breadboard. Furthermore, the high luminous intensity may pose an issue with accidentally lighting up the adjacent nodes.

	LTST-C191KRKT	**LTST-C171KRKT**	**WP7113SURDK**
Cost of each	$0.25	$0.27	$0.35
Forward Voltage	2V	2V	1.95V

Viewing Angle	130 deg	130 deg	30 deg
Luminous Intensity	18.0-180.0mcd	18.0-54.0mcd	1300-2300mcd
Size	2.00mm x 1.25mm	1.40mm x 1.25mm	5mm
Lead Time	1 Week	1 Week	6 Weeks

Table 11: *LED Comparison.*

After analyzing the different LEDs and the table comparison of the LEDs, we have decided that for now we will use the WP7113SURDK. The high luminous intensity will provide a benefit to the prototype testing to ensure the Advanced Breadboard works properly. Additionally, the general purpose leads on the LED will make it easier to prototype the Advanced Breadboard. If it is successful we may use the LTST-C191KRKT LEDs for the final PCB design. Both of these LEDs are cost effective, provide enough luminous intensity for the purpose of the project design, and can be adjusted to be bright enough for the node indication on the breadboard. There are many things we have to consider with these LEDs and how they affect the 3D printed molding which will be observed during the prototyping phase. The only issue with the WP7113SURDK is that it has a high lead time and costs more than the other LEDs. In conclusion, for the sake of prototyping we will be using the WP7113SURDK and will be observing its effects with the 3D printed chassis and other components; and, we may be using the LTST-C191KRKT for the PCB design and will also be observing its performance.

3.9 Software Communication

The organization of the software and hardware will include many components that will require a reliable way for communication. The software will include the GUI and the algorithm for schematic to breadboard. The hardware communication will involve the LEDs being activated and deactivated and receiving data from the sensor. The communication will need to relay the GUI information to the LEDs and sensor to GUI in a timely and efficient manner. The focus on this section will be to determine possible ways to transport this information. The methods below do not represent all of the ways that communication can be accomplished, but for the scope of this project the following will be examined. These were chosen because they are more readily available in terms of price and experience. The communication systems will need to cover options for controlling the LEDs and sensors, and the microprocessor. Table 12 will list all of the options discussed below and provide a visual for the comparisons.

3.9.1 Wired Communication

High Definition Multimedia Interface, HDMI, is a cable that can transfer data to most applications. This is done with a cable that can connect from the sender to the receiver. HDMI's purpose is to transmit audio and visual data using one cable rather than multiple [8]. This feature is not important to the project because there is no audio being included. It is important that the information be translated with high quality, but the application for HDMI doesn't fit that of the project. It is known for its high resolution ranges between 720p to 4k [8]. The resolution capacity will be important to consider because the GUI will have moving parts and will need to be visually clear for the user. A lack of clarity can cause the user to make mistakes in the circuit design which lowers the accuracy of the project as a whole. HMDIs do not require advanced setup or driver downloads which makes replacing the wire easy for the user incase of damage. There are a wide variety of HDMI cables that can work with many different devices, but it does limit which controllers can be used. Distance is another limitation to this communication style because of the need for wires. Most microprocessors do not use HDMI as they are not equipped for that type of communication but Raspberry Pis are able to handle this. However, there are many options for cables which makes it readily available and affordable.

The Universal Serial Bus, USB, has been a staple to computers communicating with peripherals. The focus will be on USB 3.0 because it is a widely available and newer version. It is capable of two way communication and at many variable cable lengths. USB is widely used and incorporated into many different devices. USBs are convenient because they come in different types that allow for different connections. Acquiring a USB would be very easy since it is very cheap and very accessible.

3.9.2 Wireless Communication

Bluetooth is a type of wireless communication that sends packets of information through radio waves. This two-way connection allows the sender to also be a receiver which in the scope of this project may be necessary during prototyping. The frequency picked up by the antennas is in the range of 2.4 to 2.4835 GHz [9]. It will have to be ensured that the controllers used have a module that can pick up these signals and interact with them. A wireless connection would be ideal because it would make the hardware less complicated and would remove the strict limitations of wires. This would limit our search to microprocessors that contain a bluetooth module or are fit to add a module to it. Many phone apps use Bluetooth to interact with peripheral devices which may be able to be used in this project because it makes it far more portable and shareable. Bluetooth modules can be used to create a bluetooth connection on a microprocessor using some pin designation and programming [10]. This allows more options for wireless connectivity, however, more time will need to be invested to setting up the module and learning about its functionality. Testing will be very important to ensure the data being passed through is accurate.

Wireless Internet would be another wireless connection that is available to most computer devices. Using wireless internet allows for better transferability of data because

results can be sent to nearby devices also on the network. However, this may have a security risk which would require extra care and analysis. There are wifi modules that can be added to microprocessors to achieve this communication [11]. This limits our choices to microprocessors that are capable of adding the module if it is not already included. This increases the difficulty of the setup, but may make the process easier for the user to move the hardware around since it is not wired.

The table below shows a breakdown of what microprocessor options there are for the different communication types. There are other devices like microcomputers and desktop/laptops that can be considered but the majority of those have HDMI, Bluetooth, USB, and Wifi included so they will not be analyzed as there are too many options. The HDMI is not very compatible with microprocessors as none were found to contain modules for that communication. HDMI is for dense data communication which would not be suited for a microprocessor which is set for less complicated actions. Bluetooth can match with a few microprocessors that have the module included while others have the option to have it added. Adding a module can create some difficulty because a setup would be required and would need to be functional. USB is used in many devices including many microprocessors. This makes the process easy because if the microprocessor is required to work with any peripherals it most likely will be able to do so with USB since it is so highly used. The wifi communication will lower the options, but may result in the smoothest process for the user as set up will be very portable. Less options may be better because there will be more resources for that specific device that can handle the wifi communication.

Communication	**microprocessor**
HDMI	Not Compatible
Bluetooth	PIC32CX-BZ2, module to add
USB	Modules to add, many come included
Wifi	“ESP8266”, some options

Table 12: *Communication Options.*

Based on the analysis, a wired connection will be preferred because there will be less effect with noise and reduces the compilation of establishing a wireless connection. With USB there are serial connections that can also be done using UART which is a viable option for most devices including microcontrollers and sensors.

3.10 PCB Design Software

When making the PCB schematic design we had a number of PCB design softwares that we could use. We were able to narrow down our options to three which involved Autodesk EAGLE, KiCad, and EasyEDA. All of which were the three highest recommended softwares among our peers and within the engineering industry. Each software has its own key features that give them an edge compared to the others. The

following subsections will outline each software's features and how they can each provide the best tools to design the best PCB for the Advanced Breadboard.

3.10.1 Fusion 360 Autodesk EAGLE

EAGLE is a software that the group is familiar with due to having a course revolving around this software. Furthermore, its UI and PCB design tools make it easier for beginners such as ourselves to design our PCB. For instance, EAGLE has a well developed routing engine that offers route optimization and PCB navigation. Another thing that may be useful is that EAGLE is a very useful tool for PCB design beginners. The support system as well as the tutorial videos make it very useful to always reference techniques and designs to your own PCB design. Since we are testing designs using simulations, the SPICE simulation integration with EAGLE makes it so we have access to precise analysis and simulation for our design. A major disadvantage to the software is the access of the full capability of the software requires a purchase payment. The payment of the software would make it harder to stay within the budget of the project. However, we were able to combat this by accessing the student license of the software which provides us with all the features we need in order to create the best PCB possible for our design. In short, EAGLE is a good software for beginners who have a complex PCB design, its massive libraries offer a faster design process, and it's a very effective tool when it comes to improving the design quality of our PCB. Table 13 summarizes the comparison below.

3.10.2 KiCAD

KiCAD is a free EDA tool that creates ease and establishes a smooth PCB designing tool. The open-source software makes it easy to receive help when using the software and allows for design debugging and PCB design optimization. This software utilizes a tool that creates layouts and routings of PCBs at an incredible speed to save time and avoids obstacles. Furthermore, it includes a feature called Eeschma to offer efficient schematic capturing. A key feature to perfect a PCB design is the rule checking of PCB designs, which you can't approve a design without first running a check. This feature uses a DRC tool to rule out any chances of electrical failures, short circuits, current leaks, and faults. Lastly, the software is compatible with all operating systems which makes it easier to freely cross platform between all the operating systems when designing our PCB. A few downsides to this software is the difficulty in developing a BOM from the schematics as well as making footprint and library management difficult. Although there are a few downsides to this software, the overall use of the tool brings a different aspect to PCB designing in which this tool has many solutions to creating and improving PCB designs.

3.10.3 EasyEDA

The last PCB design software on our list is EasyEDA which is an online program used to design PCBs and schematics. Since this is an online program it can be accessed using any browser from any device and stores all its files on the cloud. This makes it the most portable design software on our list making our design easier to access in case of needing to change workstations or devices. Because EasyEDA is an online program, multiple

users can collaborate on the same project and the edits made by any of the designers will be reflected to the other users in real-time which makes it a unique feature compared to any other PCB software program. Although it is better to manually route the design, EasyEDA has the option to use the online platform for routing or the option to download a package that allows for manual and autorouting on our desktop.

	Fusion 360 EAGLE	**KiCAD**	**EasyEDA**
Cost	$545/year Or Free Student License	Free	Free or Premium for $19.90/month
Beginner Friendly	Yes	Yes	Yes
Operating System	Windows, MacOS, Linux	Windows, MacOS, Linux/Other, Docker	Browser (Any)
Autorouter	Yes	Yes	Yes
Group Experience	90%	0%	0%

Table 13: *PCB Design Program Comparison*

Due to the comparison table and the description of all the PCB Design Software that we have in mind, we decided that Fusion 360 EAGLE is the most optimal software for our PCB design. Thanks to our familiarity with EAGLE and the advanced features that come with the program, we have found that it is the best beginner-friendly PCB design tool out of the other softwares. The tutorials and beginner's guide will skyrocket the amount of time and troubleshooting needed when working on our design. Furthermore, since we have access to the student license, we will have access to all of the software's premium features without the need of increasing our budget. Although the other softwares has a free version, we have found that EAGLE possesses more features and opportunities to create the best PCB design for the Advanced Breadboard. EAGLE's feature of being able to generate a BOM will be useful in keeping track of the PCB design and help maintain the budget that was set for the Advanced Breadboard. In conclusion, we have found that EAGLE possesses many features and guidance that will help our group design the perfect PCB for the project.

3.11 Computer Aided Design Software

To print the breadboard on a 3D printer there has to be a design file created. This file will include a 3D model with specifications and measurements of how exactly the parts of the breadboard will be. Therefore, Computer Aided Design Software, or CAD, will need to be utilized. There are plenty of options, but due to limited resources and experience only the most common tools will be considered. SolidWorks, AutoDesk, and Creo. The team has limited experience doing this type of design so the tool that is more user-friendly will be considered.

SolidWorks is a very common software for beginners. It has plenty of resources and tutorials online that can be used as guides for building the designs. This software is also available for free through the University which will be considered highly. A member of the team has used this before so the experience is highly regarded. AutoDesk has the application AutoCad that can perform the necessary functions. This option is also provided by the University which can be very helpful given the budget constraint. The final consideration is Creo which is not as common as the other two. Creo provides university students the opportunity to use their software. This software is more advanced than the other two and works in a very different way than SolidWorks in the way it manipulates its parts.

Based on the options and the analysis, SolidWorks will be the choice for CAD software. It is free to use, available on campus computers, and a team member has experience with it. The designs of the breadboard will not be too complicated since the shape itself is box-like. Therefore, a sophisticated program that needs to be learned will not be necessary.

4. Standards and Design Constraints

The following will describe the standards and constraints that are expected to be placed on the specifications of the project. The standards included were determined to be the most important for the specific goal of developing a smart-breadboard. Details will be given on how they are expected to affect the project and how this will be considered in the design.

4.1 Standards

The following will describe the standards on the specification of the project. The standards included were determined to be the most important for the specific goal of developing a smart-breadboard. Details will be given on how they are expected to affect the project and how this will be considered in the design. As we are integrating software when using the GUI interface we need to cooperate with the standards related to the software. As well as standards related to the electrical components used. Utilizing these standards will ensure that our product works efficiently, effectively, and within the overall guidelines that are set by these standards. At the heart of our project in the software compartment is the Raspberry Pi 4B, an innovative microprocessor that perfectly uses traditional computing with embedded systems. This is especially perfect for a project

such as this that relies on a great connection between hardware and software. And while upholding software reliability, efficiency, and adaptability isn't a luxury it is a necessity for the advanced breadboard. Implementing these standards ensures the seamless functioning of our project.

4.1.1 EEE 730-1984 (Quality Assurance)

The IEEE 730-1984 is about the commitment to a high quality project. Set within the realm of Software Quality Assurance (SQA), this standard isn't just about evaluating the end software product. It also offers a more visual approach that highlights the underlying processes, methodologies, and protocols that lead to said final product. The framework provided is complete as it has, encompassing guidelines, requirements, and best practices, all unified towards ensuring that the SQA processes begin, then are planned, controlled, and executed with a degree of professionalism.

IEEE 730 will serve as the quality backbone of our Advanced Breadboard project. We can use the Software Quality Assurance Plan to help us focus on areas such as version control and issue tracking. We'll use Git for code version control, ensuring that any changes made to the GUI or algorithm are logged and reversible (if needed). A Jira board will also be set up for issue tracking and reporting to make sure all software glitches, big or small, are dealt with promptly. Otherwise, this communication can also be tracked via Discord.

A pivotal component within this standard is the Software Quality Assurance Plan (SQAP). The SQAP is a structured document, laying out in definitive terms the methodologies, tools, processes, resources, and responsibilities that will be employed to guarantee software quality. It is typically used as the anchor, ensuring that the entire team has a clear vision of good objectives and the means to achieve them. But on top of having a plan, there is a lot more that this process contains. Periodic quality audits/checks, make sure that the processes and the resulting software remain in alignment with the SQAP. These frequent checks and audits are the checkpoints that make sure that the team remains on the right path towards a finished software product.

Additionally, with the incorporation of quality metrics, the standard pushes towards an approach to quality and also quantity. These metrics that are being measured are things such as code churn rate, or response time, are used as indicators of good or bad software quality. They facilitate the ongoing assessment, pointing out areas of excellence and those needing improvement. Basically, IEEE 730-1984 creates an almost sort of ecosystem of continuous improvement. With structured problem reporting and a way to correct actions in certain situations, issues are not just identified but are resolved, making the software more robust with each iteration.

For a project like ours, where the Raspberry Pi is intertwined with hardware components, this standard's adherence takes on even more significance. Each software module and each functionality, especially those interfacing with hardware, demand rigorous validation and testing. This standard certifies that such consistent checking is not more than it needs to be but rather systematic in nature. Furthermore, it promotes a culture of

accountability and continuous enhancement within the group working on the advanced breadboard, ensuring that our Raspberry Pi's software remains agile, robust, and in alignment with our quality aspirations and requirements.

This pivotal standard emphasizes the essence of software quality. Applying quality to our project translates to a lot of software testing and ensuring that our Raspberry Pi's software consistently meets expectations. Every single functionality, especially those that are directly connected or related to hardware.

4.1.2 IEEE 829-2008 (Test Documentation)

When it comes to software, functionality is very important. But, equally important is software with reliability and stability. The IEEE 829-2008 is a standard that speaks to this importance of testing and providing a structured approach to test documentation. This standard doesn't just acknowledge the act of testing but emphasizes the imperative of documenting every single little detail throughout the testing process. At its core, the IEEE 829-2008 presents a structured framework for eight distinct types of test documents. Each document, from the Test Plan to the granular Test Incident Report, serves a unique purpose which makes sure that every aspect of testing is captured, analyzed, and archived. For example, the Test Plan is not just a strategy document. It is the compass, detailing out the scope, resources, schedules, and methodologies that will be used during the testing phase.

Equally as important is the Test Design Specification. This stage describes the reasoning behind specific test conditions, the expected system states, and the criteria that determine the success or failure of a test case. This is further explored and explained in detail in the Test Case Specification, which dives into the specific inputs, the expected outcomes, and the conditions under which the test would be executed. This level and amount of detailing makes sure that the testing process remains clear, replicable, and consistent.

In compliance with IEEE 829-2008, we'll develop a detailed Test Plan with multiple phases. Firstly, in the "Initial Algorithm Analysis" phase we have specific scenarios through the circuit translation to ensure it accurately translates electronic schematics to physical breadboard layouts. Then in the "GUI Operation Verification" phase there will be user acceptance testing, where we verify that the GUI is intuitive and responsive. In this phase, it can be tested amongst the group as well with random people. Finally, "Breadboard Limitation Testing" will check the compatibility limits of our algorithm with different circuit models and sizes.

For a complex project such as ours, based on the Raspberry Pi 4B for the software , such structured test documentation is of utmost importance. Each software module and each line of code, undergoes this regime/process. Detailed test logs, error reports/logs, and summary reports not only ensure that bugs/issues are identified and rectified but also offers users a transparent view of the software's quality and readiness. In essence, the IEEE 829-2008 serves as the guide in the testing process, ensuring that it remains comprehensive, clear, and most importantly effective in validating the software's reliability.

Structured and organized test documentation is crucial to the research and development of this breadboard. By adhering to this standard, we can assertively claim that each aspect of the Raspberry Pi 4B's software has been checked for any bugs or errors. In a project that demands precision in operations, meticulous documentation plays a pivotal role in diagnosis and version control.

4.1.3 IEEE 830-1998: Software Requirements Specifications

For our Advanced Breadboard, the IEEE 830-1998 standard will serve to develop a typical Software Requirements Specification (SRS) document. Such as the state of the GUI's functional requirements, like a drag-and-drop feature for electronic components, and the limitations on the number of nodes and connections. On the algorithmic side, performance metrics like speed of schematic translation can be tracked and clearly outlined.

Clear and unambiguous software requirements are a project's bedrock. By following this standard, we will be made fully aware of the precise capabilities and expectations of the Raspberry Pi 4B, eliminating any ambiguity. Additionally, this will almost eliminate any need to further buy new parts or technologies.

4.1.4 IEEE 1016-2009: Software Design Descriptions

In the context of the Advanced Breadboard project, we have to follow the IEEE 1016-2009 standard documenting the software design. This standard will outline each GUI component and their functionality, from drag-and-drop functionalities to the typing (in general). It will specify how the GUI and breadboard interact. This standard also helps us detail how the software triggers or activates, like when adding a resistor in the GUI and how it corresponds to physical changes on the breadboard, such as lighting up a specific LED. Additionally, IEEE 1016-2009 informs how we document the system's (possible) scalability and error handling mechanisms, preparing the project for future enhancements and troubleshooting. By adhering to this standard we'll have a well documented project.

A well-defined software design description is the roadmap that guarantees the software's modularity, scalability, and maintainability. As our project takes shape and scales, this blueprint becomes the guiding factor for not just the software but the entirety of the project. A well practiced use of software design descriptions increases its helpfulness down the line of developing the advanced breadboard.

4.1.5 Impact of IEEE Standards on Software Lifecycle

The IEEE 730-1984 standard helps refine how the quality of the code will be. Under its umbrella, our software will not just *do* what it has to but also achieve reliability and efficiency, ensuring the Raspberry Pi 4B will complete its important tasks.

The standards don't just emphasize but mandate software safety. This focus means that the software running on the Raspberry Pi remains secure to its hardware counterparts/components.

Robust documentation, anchored in these standards will act as the project's memory. This detailed record-keeping becomes the touchstone for problem-solving, enhancements, and maybe even future expansions.

The rigorous software implementation standards for the Raspberry Pi 4B, steered by the revered IEEE standards, isn't about following a rulebook. It's about a reliable, efficient, and adaptable product.

4.2 Constraints

The following subsections will discuss the constraints that are being encountered with the design of this project. These constraints will surely affect the specifications of this project, but we need to be able to limit these constraints so that ultimately we can design an effective, accurate, efficient, and functioning Advanced Breadboard.

4.2.1 Software Constraints

One big focus for the software of the advanced breadboard is on real-time performance. This is essential to keep the GUI and LED indicators on the breadboard in sync. This is especially important given the limited number of nodes available on the specialized breadboard which sets constraints on the software's efficiency and the types of circuits that can be built on the GUI. Data integrity and memory also seem to be key factors, requiring the software to include error checking and possibly error recovery mechanisms to ensure a seamless and accurate transfer of data between the GUI and the advanced breadboard.

Resource utilization and scalability are other key considerations. The software should be portable enough to run on a variety of hardware specifications without consuming too many resources. Also, one of the goals is that its designed with modularity in mind, allowing for future scalability in terms of additional nodes, features, or components. This would pave the way for easier upgrades, both in terms of hardware and software features in the case of a different/bigger breadboard.

User experience should be front and center, which involves creating an intuitive, user-friendly GUI that minimizes a learning curve as much as possible. The GUI should also have validation checks to guide users in creating feasible projects given the hardware constraints. However, the breadboard is custom and as a result, the validation checks might not be necessary since the breadboard and software are supposed to work in tandem and as such are unique. In a multi-user setting, account management and individual project saves could become necessary, adding another layer of complexity to the software design, although not currently under development.

File format compatibility, especially if schematic importing is to be supported or added as a future feature, this will also be important. This requires the use of parsers for circuit design file formats, adding another layer of complexity. While exporting the file itself may not be a feature, the information from the GUI/schematic will have to be transferred over to the breadboard in some way, so a file transport protocol should be considered. Security is another crucial aspect, with secure protocols being potentially necessary to make sure of the data integrity during the communication between the software and hardware components.

Documentation is essential for a tool designed to be educational and user-friendly. While, the software is intended to be simple to use anyways, user manuals, help features, and possibly even built-in tutorials should be developed to assist those using our product. If the software is to be deployed in educational settings, compliance with relevant educational standards or accessibility laws might also be necessary, which would vary from region to region.

By addressing these software constraints early in the development process, the Advanced Bread project can create a more robust, efficient, and user-friendly product. These constraints, when carefully managed, will contribute to the project's goal of making circuit design more accessible, educational, and simple.

4.2.2 Electrical Constraints

In terms of this project the Electrical Constraints can come in many forms, and our job as engineers is to identify these constraints and minimize the overall effect they have in the project we are working on. The Advanced Breadboard is a unique project that may require a sophisticated PCB design. With all the different components we require and the limited space we have, it may leave room for a lot of mistakes and errors. As you may know, when dealing with a lot of components there will be some form of signal delay. Since we are expecting to use a microprocessor with Wifi communication we want to be able to limit the signal delay between the microprocessor and the LEDs. Furthermore, the voltage and current sensors use an I2C interface and we need to be able to minimize the delay between the sensor and the microprocessor it is connected to.

Another Electrical Constraint we need to focus on is the PCB footprint. Because the Advanced Breadboard has a size requirement we need to be able to save as much space as possible. Good engineering work requires being able to judge the situation and figure out a solution to the problem. This includes properly designing a PCB that does not leave any unnecessary space. On the other hand, our design utilizes many components such as a battery pack and ten LEDs. These alone take up a lot of space on a PCB which leads us to an electrical constraint. Our choice in these components, how we place these components on the PCB, and how we wire them will all be a challenge we need to take into account in order to minimize the electrical constraints.

Furthermore, another Electrical Constraint we need to keep in mind is heat transfer and manufacturing issues. When designing the PCB we have to factor any parts or PCB manufacturing issues as well as how the heat is being transferred throughout the project.

Since we are using small components we may not need to worry too much about overheating. This requires more research. As for manufacturing issues, we would need to minimize this issue by ordering a few components of each part that we need as well as ordering a few PCBs to minimize this constraint.

4.2.3 Economic Constraint

The funding of the project is an economic constraint that will affect major portions of the project. There are no outside funders so all expenses will be taken care of by members of the team. Most products offer more benefits, or features, with higher prices. The breadboard could have the option of being sent to a manufacturing company to get machined to perfect specs but the limit of the budget encourages the team to find a more creative alternative. The team will need to determine what parts of the project do not need high accuracy to still perform as intended. The breadboard will be 3D printed which comes with a cheaper cost and will not require shipping. However, the board may not be printed with the most accurate measurements due to cheaper 3D printers not having the capability to print to certain decimal points.

Along with this, microprocessors will have to be chosen based on information gathered in research and personal experience by the team rather than testing all of our options. The team will not be able to purchase multiple microprocessors to test because that can become costly very quickly along with rising shipping costs. A lot of the resources will be sourced on campus as there are many labs and researchers that have access to tools the team may attempt to use during prototyping. This includes resistor components and power supplies.

4.2.4 Environmental Constraint

The environmental constraint on the project comes in two forms: environmental impact and the environment of the user. The environmental impact of this project will be considered during every part of its production. ABS was found to be the preferred plastic for the makeup of the breadboard but a professor of mechanics in malaysia found that “ABS is recyclable”. This finding will encourage the rest of the project to find ways to make it recyclable or reusable. The electronics in the project will be designed to produce as little heat as possible to reduce the waste of energy. This will be done with calculations and simulations to ensure that the design is efficient.

The next consideration will be the constraint imposed on the project based on the environment the user will handle the product. This may include lab rooms which may contain many instruments and tools surrounding the work bench. Because of this, the breadboard will need to be designed to work in smaller environments. Since solderless breadboards are often portable it will be important to maintain that ability so the majority of the hardware should be within the breadboard set up in layers. Choosing the breadboard material out of plastic comes from this constraint because wood or glass would be too heavy. This constraint also influences how a connection from the GUI to the breadboard will look. Using the least amount of hardware and wired connections will help with the adaptability to the environment. Considerations will be made to make

wireless connections. The overall design of the breadboard will take into account the ability for recycling materials and for keeping a tight and organized design. These constraints will be made by using a recyclable material for the breadboard and by making it small and portable.

4.2.5 Time Constraint

The time constraint was imposed within the guidelines of Senior Design I. There are hard deadlines for specific aspects of the project. These deadlines create a constraint on the design of the pcb and part selection because of shipping time. This deadline also will affect the use of resources on campus which may be used by others. There are deadlines for the research of the project, the prototyping for proof-of-concept, the final report, and the presentation of the final project result. These deadlines will drive every aspect of the project. The research will need to limit which parts can be analyzed due to lead times and shipping costs. The PCB will also need to be designed with lead time in mind because shipping time has been extremely long as of recent. There are materials and instruments available on campus, but their use will need to be strategic because they may be occupied or inactive which can affect the manufacturing and testing of the breadboard.

The PCB is one of the components of the project that must be sent to be manufactured and may have the longest lead and shipping time. Priority will be on the PCB design and ensuring it is made without mistake. Yedda Yu recommends “some tips to help with delay which includes not having an error in the design including drill hole placement”. It will be important to design the PCB with specific goals in mind. These goals will be described further into the report once all parts are chosen and constraints are considered. It is important that the PCB is done correctly the first time because redesigning and manufacturing another will not allow for enough testing time for the final products. The PCB will need to successfully alter the state of the LEDs and assist with the open/short circuit testing. This is a vital part of the project and will need optimal testing to ensure a finished product is presented by the deadline.

The economic constraint on the project led to the decision to resource parts from campus which are available. This will prevent shipping and lead time for some of the parts. However, since campus resources are often available to a large number of students it will be important to plan out their use. There are 3D Printers available that can handle printing the breadboard to its specifications. Printing can often take hours and may need reprinting if not done properly. There is also a process to refill the filament and to ensure the correct type is ordered which can take a few days. Most of the electronics and the hardware will be placed in the breadboard chassis so that will need to be designed once all of the parts are selected and measured. The print should not take longer than 24 hours to print, but may need to be reprinted for accuracy or if changes need to be made. This will be a high priority item to have completed to finish the rest of the project.

The time constraint will drive the design, production, and testing of the project. The research will only consider parts that can be easily acquired in time for the deadlines. Local resources will have priority consideration since they will require little to no shipping time. The PCB design will take special care to ensure that it comes in time. The

delay of that part can cause major consequences as time will need to be allotted for testing which leaves little time for shipping since it may be delayed as has been the recent trends. The time constraint has one of the biggest impacts on the scope and part picking since it creates limits, but these limits are useful to help narrow down the vast amount of options available in the market. Creating a project with affordable, and easily accessible parts will allow for it to be easily replicated which is ideal for product advancement and availability.

4.2.6 Health Constraint

The health constraint will be considered one of the more strict constraints because of its importance. This project will require human interaction and therefore will require safety for its users. The focus on safety will involve the materials being used to create the project, and the electrical hazards that come with it. The rest of the design will take into consideration the health and safety of those building the project and those using it. This will include the proper personal protection equipment, PPE, is acquired and that protections are taken during testing. There are a lot of electrical components that will be tested and will be done safely in the lab.

For the safety of the user, all wired components will need to be insulated to avoid electrical shock. They will also be placed in a way that will avoid getting caught or ripped during transportation or use. The breadboard itself will be designed to avoid injury by ensuring corners are rounded and other aspects will consider safety into its design. The GUI will have the most interaction with the user so its display will have to take health and safety into consideration. It will be important to make sure there is nothing flashing on the screen that can induce a negative response to the user. The font and coloring on the screen will be considered as well to ensure the user can properly see all of the information without any strain.

Printing ABS material has a level of toxicity to it so it will be important to take proper precautions. The printer will need to be encased to direct the fumes away from the public. Other precautions will be taken when 3D printing including checking the temperatures of the bed and nozzle to ensure no skin contact is made to a heated part. This will be considered when removing the part from the bed and replacing the filament in the nozzle.

4.2.7 Social Constraint

A social constraint will be considered because it is important that this project is made to be accessible to everyone. The project will be made accessible by making it portable and adaptable. The users of the project are diverse in ability and skill and therefore will require different specifications. The best way to cater to all will be to ensure effort is placed into the design with this in mind so that equal opportunity is given to all. The LEDs on the breadboard will need to be visible enough so that extra effort is not required to identify them. Multiple colors will not be used to avoid confusion.

Portability will provide a large amount of accessibility because it can adapt to the environment. The breadboard is being designed to be lightweight by decreasing the infill

percentage of the print. The electrical design is taking into consideration portability so the wires must be placed in a way that allows them to be easily moved or reconfigured as needed. The majority of the hardware is going to be placed inside the breadboard to facilitate its portability so that less items need to be handled.

5.0 Comparison of Chat GPT

AI tools, like ChatGPT, are making their way into every industry. For the scope of this project, ChatGPT will serve as a tool that can take an input like a question or idea and expand on it based on what is available on the internet. There are many AI software tools available, but ChatGPT will be the primary tool because it is free, easily accessible, and easy to use. ChatGPT will be used as a starting point rather than a final product. The following sections will describe how ChatGPT will benefit the project and its constraints, followed by examples on its integration.

5.1 Pros

Integration of AI is inevitable, so it is important to take advantage of its use in this project. ChatGPT has the ability to receive an input and provide an output based on data that it accesses on the internet. This is considered a pro because it can help grow an idea. If there is a subject in the project that the team doesn't have a starting point for, ChatGPT can reveal more information about that subject to provide the team with items to search. AI software will help to save time which is one of the largest constraints on the project. It can help with quick information and facilitate research. Sometimes the internet can be difficult to navigate since there is so much information out there that ChatGPT can spend the time sorting through rather than the team spending far more time doing. ChatGPT is also able to take in information and summarize or simplify. This will be an important tool because research includes reading scholarly articles that may be too advanced. Inputting that to ChatGpt will provide a summarized excerpt that will help with understanding and explain how it can be relevant in this project.

ChatGPT offers a tremendous amount of benefits in terms of researching and developing the advanced breadboard. Due to the large database that ChatGPT has from the internet (up until 2019), practically any topic can be verbalized and understood in a matter of seconds. Difficult concepts and ideas could be grasped much quicker as opposed to watching a whole video or reading a textbook/paper on a certain topic. For this project, a lot of electrical engineering concepts and software concepts will be used to develop the breadboard, combining programming with hardware and circuitry. ChatGPT can reduce research time in these topics and help in making templates or skeletons of the project to have somewhere to start in some cases.

Unlike people, ChatGPT is available 24/7, which can expedite research and problem-solving, especially in situations where you need assistance outside of the typical availability hours of group mates or professors. ChatGPT can assist in brainstorming and generating innovative ideas for the advanced breadboard project. It can provide inspiration and suggest alternative approaches to solving complex design problems that otherwise could have unlikely been thought of.

The benefits of using ChatGPT are endless, along with its vast improvements and quick uprising in the world. Rather than viewing ChatGPT as a program that inputs a web of information and outputs answers; instead, it should be seen as a new tool that can be used in many fields including the engineering field. A benefit of using ChatGPT is that it can create circuits to use when being asked. Although the accuracy and function of the circuit can vary, it still has potential to become a very powerful tool in engineers' everyday work.

5.2 Cons

AI tools have their use in research, but they also have constraints as the technology is still improving. ChatGPT is not always accurate and it is not always obvious if it is or not. Anything that ChatGPT produces must be checked to ensure it is correct before using that information for the project. An error in any basic calculation from ChatGPT can disrupt the flow of the project and can cause issues to the software and hardware. Because of this risk, ChatGPT's use will be limited to ensure that the bulk of the research comes from reputable sources rather than quick responses. ChatGPT may provide a correct response but for a different input than the one intended which can cause confusion and errors. It will be important to work all phrases properly so that the tool does not mistake what its output should provide. The safeguard for these is ensuring all outputs are verified through other sources.

Another issue may be that ChatGPT does not cite the source they use in their response which may have been taken from another author. This can create an issue when publishing information about this project that may have come from an unknown source. According to information provided directly from ChatGPT, its knowledge is dependent on when its update was last done so there will be a lack of up to date knowledge (ChatGPT, 2023). This must be considered a con because research should always consider recent discoveries because those may make large impacts on the project.

Developing some aspects of the advanced breadboard can be complex and time-consuming, requiring specialized knowledge and resources. As a result, ChatGPT is unlikely to be effective in EVERY aspect that it can be called during development. The questions we may prompt to ChatGPT can be difficult to put into words and as such, difficult to respond to. For efficient responses from ChatGPT, a relatively new job has recently become popular, prompt engineering. The core concept of using artificial intelligence in general is that what you get from it, is what you put in. Therefore, whenever we might have questions about large scale portions of the project, or maybe a very specific circuit, a large amount of detail will be required to include in the prompt. At some point, the amount of energy/time used towards making a *perfect* prompt is not worth the time it takes to do so. Because we *could* benefit higher from using non artificial intelligence resources in some cases

Relying heavily on AI may reduce the understanding of the underlying principles of the advanced breadboard's design, which can be a drawback in future settings. Whenever ChatGPT is not available, fixing and/or designing the breadboard could be much more

difficult. A reliance on ChatGPT deteriorates the learning curve, disallowing a deeper understanding of the product.

Developing electrical equipment is not an easy feat. It requires advanced research, trial and error, finances, and a ton of intuition and creativity. Sadly, some of these things cannot be attained through the use of ChatGPT. For starters, as engineers grow experience in their field they tend to grow an intuition. This intuition affects our decision-making by providing context to the given situation and using our own experience to derive a solution. ChatGPT is capable of providing an answer to a question one asks, but will it consider the context of that situation and make decisions based on that intuition? As engineers, our job is to innovate and create something new or to improve on something old. With ChatGPT the creativity of the human mind does not exist; thus, the lack of new ideas will drive a depression on innovation. That is why ChatGPT can never replace engineers in their craft; rather, it can be used as a tool by engineers to drive innovation by accessing and utilizing the information across the internet.

5.3 Examples

The first example of using ChatGPT as a tool is asking what material would be best to print the breadboard with. The exact prompt given to the tool was “I am working on a senior design project where I am creating a breadboard with advanced features like an open/short circuit test. There will also be LED lights that light up the difference in the circuit. I want to 3D print the breadboard, what material should I use to 3D print it”. This prompt provided all the information that the team had on-hand so that ChatGPT was working with the same amount of detail. It was important to give context to the question such as it being a Senior Design Project. This detail may inform someone that this project is being made with limited funds, resources, or time. However, when repeating the processes without this detail ChatGPT’s response did give any change to reflect the altered input.

The AI tool’s response to the prompt included three different sections even though the prompt asked for one answer. The first section includes a list of considerations to take when choosing a material. These considerations were exactly what was considered in the preliminary research: Electrical insulation, durability, thermal resistance, and printability (ChatGpt, 2023). Earlier in the report these constraints were broken down for a set of possible materials. The second section provides a list of four optional materials: PLA, ABS, PETG, and Nylon (ChatGPT, 2023). These materials were also previously considered and narrowed down to PLA, ABS, and PETG as they are more standard. The third section includes more information to consider when building a breadboard. This information does not involve the type of material, but goes into details in other parts of the project. This may be due to the extra information provided in the prompt.

ChatGPT provided more information that prompted it, but this information was found to be useful. Before the question was given to the AI, research was done to answer the question so that a comparison could be made. The response was given in less than 30 seconds, while the research took around 12 hours. The quality of the response was proven

when compared to the preliminary research because it matched it very well. However, the response provided less details and more of a simplified explanation. Another prompt was given to test the limits. The AI was asked to choose from the given material list to see if it could make that decision rather than providing another list. ChatGPT then responded with PETG as its choice of material for the given situation (ChatGPT, 2023). The reasoning for this choice was not very detailed, rather, just a sentence with a basic overview as to why it is preferred. The response did not include any data or sources.

For this example, ChatGPT would have provided a strong base for continued research. It provided quality options for the type of materials to consider. Rather than be a source for a final decision, it opened an opportunity for research on more specific ideas. The reasonings and details were lacking because data was not provided to back up its claims, but that is why this tool cannot be used as a source of information. ChatGPT allows a follow up question to reference its previous response which is very convenient because follow-up details can be provided within the same reference frame. This example shows that ChatGPT is quick, can provide a quick overview of the subject, and cannot be used as a direct source because of the lack of details and source citing.

Another example is that, in the early stages of designing the graphical user interface (GUI) for the advanced breadboard project, we would want to ensure that the interface is user-friendly and intuitive. Here, ChatGPT can be used to brainstorm ideas for the GUI layout and user interactions. It can also offer suggestions on how to simplify the user experience, improve accessibility, and make the GUI aesthetically pleasing. All of these are essential aspects to the software portion of the project.

Now consider a scenario during the design of the printed circuit board (PCB) layout that requires specific knowledge and expertise. In this situation, we decided to rely heavily on ChatGPT for guidance. However, despite its vast knowledge, ChatGPT struggles to provide the detailed, nuanced information required for our specific PCB design challenges. Because of that, we spend a lot of time creating the supposed *perfect* prompt to give to ChatGPT about the PCB layout which will become increasingly complex, and ChatGPT's responses, while quick, lack the depth and accuracy needed for making a PCB. This over-reliance on ChatGPT could hinder the team’s ability to design the PCB effectively since we wouldn’t know any details, we would just follow instructions on how to make/fix the PCB. Here we would realize that in such specialized and complex areas, using non-artificial intelligence resources, such as consulting with a professor, watching YouTube video, or conducting in-depth research, would have been more productive and yielded better results. In this case, ChatGPT's limitations in handling highly specialized topics and complex tasks prove detrimental to the research and development of the advanced breadboard.

6. Hardware Design

The following section will outline a more in depth explanation of the Advanced Breadboard’s hardware design and hardware specifications. The hardware design will cover the PCB, the 3D printed housing, and all of the hardware components as well as

their respective schematics. All of these hardware components will interact with each other to fully operate the Advanced Breadboard.

6.1 Subsystem Block Diagram

The Advanced Breadboard is made up of many subsystems that collectively operate together in order for the whole thing to function properly. In Figure 6, we decided to label all of the parts of the Advanced Breadboard and how they would operate with one another. The things we had to consider are the components that are operating outside of our product, components that are operating within our product, and the components operating within our PCB. Since we are planning to use our product to operate a user's circuit, every component outside and within our Advanced Breadboard needs to be considered as well as needing to be operating with one another. Within our block diagram we included the relationships between the components to assess how the system works as a whole. For simplicity reasons, we decided to include the amount of power being supplied to each of these blocks. Since the values can change as we progress through the prototyping phase and enter the final product phase, we decided to leave some of the values out until we have a better understanding of what is required. The main issue that we need to resolve during prototyping is the amount of power we can save. Since the Advanced Breadboard is using Alkaline batteries as its main source of power, the amount of power we have is constrained by the batteries. The versatility of the batteries comes at a tradeoff to how long we can power the device.

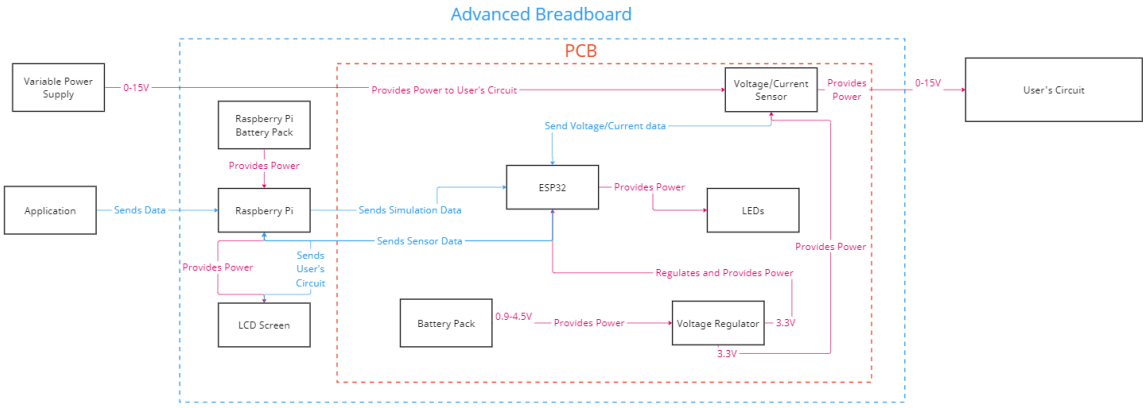

Figure 6*: Subsystem Block Diagram.*

6.2 Schematic Diagram

The following subsections will outline the schematics for each portion of the PCB and then the entire schematic for the PCB. Since we are using many components for our design, there will be many connections made to each of the parts. A major element that we had to consider when making the schematic is making sure the PCB board can fit into the Advanced Breadboard. As mentioned in a previous section, we will be using Fusion 360 EAGLE software for the PCB design, and Texas Instruments' Webench power designer to assist in the DC/DC conversion voltage regulator.

6.2.1 Power Supply/Voltage Regulator Schematic

When designing the voltage regulator we decided on the main component from Texas Instrument. Luckily, Texas Instrument provides a free public tool known as Webench to assist their customer in designing AC/DC conversions or DC/DC conversion. For our case, we needed to design a DC/DC conversion system that converts the voltage of the batteries into a constant DC output voltage into our system. Since our system is being powered by Alkaline batteries, we may have an input voltage between 0.9V-4.5V which is why we needed a boost-passthrough voltage regulator. When using Alkaline batteries at full capacity they generate 1.5V each cell, and we are using three cells for a combined total of 4.5V at maximum capacity. Unfortunately, the batteries will not always be in full capacity due to the shelf life of the batteries as well as safety risk. Furthermore, the batteries have a self-discharge rate which decreases the battery capacity when the batteries are idle.

Eventually, as the batteries are being used the capacity gets lower thus the voltage decreases as well. The minimum voltage output of the alkaline batteries average around 0.9V while the maximum could be 4.5V. Overall, this means that the batteries will never output a constant voltage which may alter the overall functionality of the PCB. That is why a voltage regulator is necessary and a crucial part for the function of the PCB, and the boost-passthrough voltage regulator is the most ideal regulator for our specific design. This is so if the input voltage is above the output voltage, the regulator will operate in Pass-Through mode. However, when the input voltage is lower than the output voltage, the regulator will operate as a boost converter to increase the voltage to the desired output voltage.

Since we chose to power our PCB using Alkaline batteries, as seen in Figure 7, we needed a more strategic approach to utilize their full potential. For starters, we need to be more strategic with the capacity of the alkaline batteries. Although they carry a high capacity compared to rechargeable batteries, we wouldn’t want the user to keep replacing the batteries every so often. The solution we came up with is integrating a power switch on the Advanced Breadboard to save energy. By having the user turn off the Advanced Breadboard when it is not in use, the batteries will last significantly longer thus replacing the batteries will become more infrequent, which will overall save the user a lot more money.

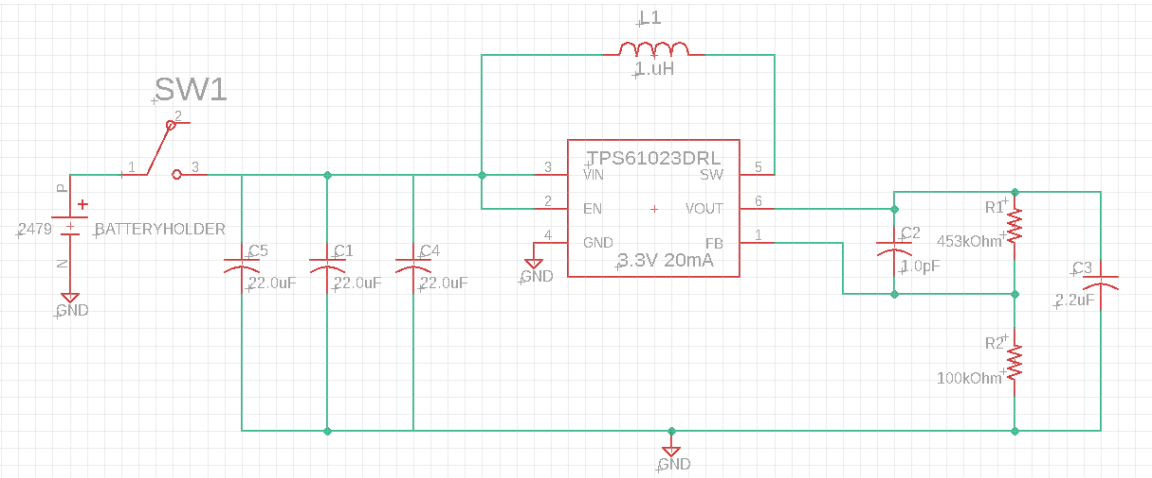

Figure 7: *Power Supply Schematic.*

6.2.2 Voltage/Current Sensor

For this detection method we had to choose between a limited number of methods. As mentioned in a previous section, the test for an open circuit can be measured when the resistance between two points becomes infinite. Another way to detect an open circuit is measuring the difference voltage between two points. When the difference voltage between two points is equal, that is an indication of an open circuit. Finally, the last representation of an open circuit is when the current is equal to zero. All of this relates back to Ohm's Law ($V = IR$).

Similar to the test methods for an open circuit, the short circuit detection works similarly in relation to Ohm's Law. For short circuit detection the resistance measured between two points becomes zero. Another way to detect a short circuit is when the difference voltage between two points is equal to zero. By this reasoning we decided that when using the voltage/current sensor that we must come up with a logic that is both simple and effective. Relating back to Ohm's Law we found that the best representation of a short circuit is when the difference in voltage is equal to zero. We have also found that the best representation for an open circuit is when the current is also equal to zero. By keeping the logic of the entire system to these two conditions we can utilize the voltage/current sensor at its highest potential in order to ensure the highest accuracy for the entire system. In order to analyze and process the data we will be utilizing the ESP32.

Our objective is to save as much space as we can, while also making the cost of the product practical. Furthermore, we are also aiming to achieve efficiency while also keeping our product effective and accurate. For these reasons, we need to observe the implementation of the short circuit detection and how we can design it in a way that meets all these expectations. After observing the use of breadboards in engineering laboratories we realized that the most common form of short/open circuits originate from the positive and negative terminal strips on the breadboard that connect to the power supply or voltage source. By minimizing the short/open circuit detection to these two

terminal strips, we can greatly improve the efficiency without neglecting the accuracy of the detection. This idea comes from the phrase “more does not mean better”.

If each node of the breadboard had some type of short/open circuit detection then the cost of the parts to manufacture the product will increase significantly. Our goal is not to make it more expensive than the original breadboard, but to be a convenient upgrade that is affordable and can be used in all engineering laboratories for learning and training purposes. Furthermore, the Advanced Breadboard has a design requirement that shall not exceed a certain size. In order to keep it compact, we need to evaluate and strategize the implementation of the short/open circuit detection. That is how we reached our conclusion, that implementing a short/open circuit detection in each and every node is impractical from a design perspective.

For this schematic, as seen in Figure 8 below, we first started with the connection required to power the sensor, which is also known as VCC. This connection would come from the voltage regulator and be connected to the VCC pin along with a bypass capacitor, and the pull-up resistors for the SDA and SCL connections. The bypass capacitor needed to be placed as closely as possible to the power supply and the ground pins as mentioned in the datasheet. While the pull-up resistors may not have been necessary if there are already pull-up resistors on the same lines somewhere along the system. In our case, there were no pull-up resistors to the ESP32 connection which is why they were placed closer to the sensor. After we made the VCC, SDA, and SCL connections we progressed to the next set of connections which are the A0 and the A1 connections. These connections are the Serial Bus Addresses for the INA219. In the datasheet, there is a table that indicates 16 possible addresses for which we can choose from. Ideally, we could have implemented a type of switch that allows us to change between the addresses; however, for our specific design we felt it was best to keep it simple and to stick to one address.

As shown in the schematic above, we decided to connect A1 and A0 to GND for the slave address 1000000. This connection was done for simplicity as well as efficiency. Finally, the last connection that was made to the sensor was the VIN+ and VIN-. In the datasheet, there was a feature implementation for an input filter circuit consisting of two resistors and one capacitor. It was indicated that for most applications it was unnecessary. This filtering feature was designed for measuring current when such noise can be difficult to determine. Furthermore, the datasheet states that filtering the input is only necessary when there are transients at exact harmonics of the 500-KHz sampling rate. So for our design the filtering circuit is not required; however, in the case that we may need the filtering feature it is recommended that we reserve the board space for the components by installing 0-Ω resistors for the resistors and leaving the capacitor unpopulated. This is in the event that if we may need to add a filter we can easily solder in the resistors and the capacitors in such a scenario. The final connection is the Rshunt in between the power bus and the load.

In our particular case, the power bus is an Elenco XP-15K power supply that generates a voltage between 0-15 DC voltage which is well within the 0-26V range suggested in the

datasheet. The Rshunt resistor is a current-sensing resistor that when current passes through it measures the voltage across it. The INA219 will then sense the drop across the shunt resistor for the shunt voltage, and then it will sense the voltage with respect to the ground from VIN- for the bus voltage. Typically the shunt voltage would be the VIN+ - VIN- and would measure at less than 50mV. After the connection to the shunt resistor, the line would then go into the load. The load in this case would be the input voltage of the user's circuit.

After the sensor captures the voltage and current data from the sensor the data will be sent to the ESP32 so that it can process the data. The next sections will elaborate further as to how the data is processed and what happens to the data once it is received by the ESP32.

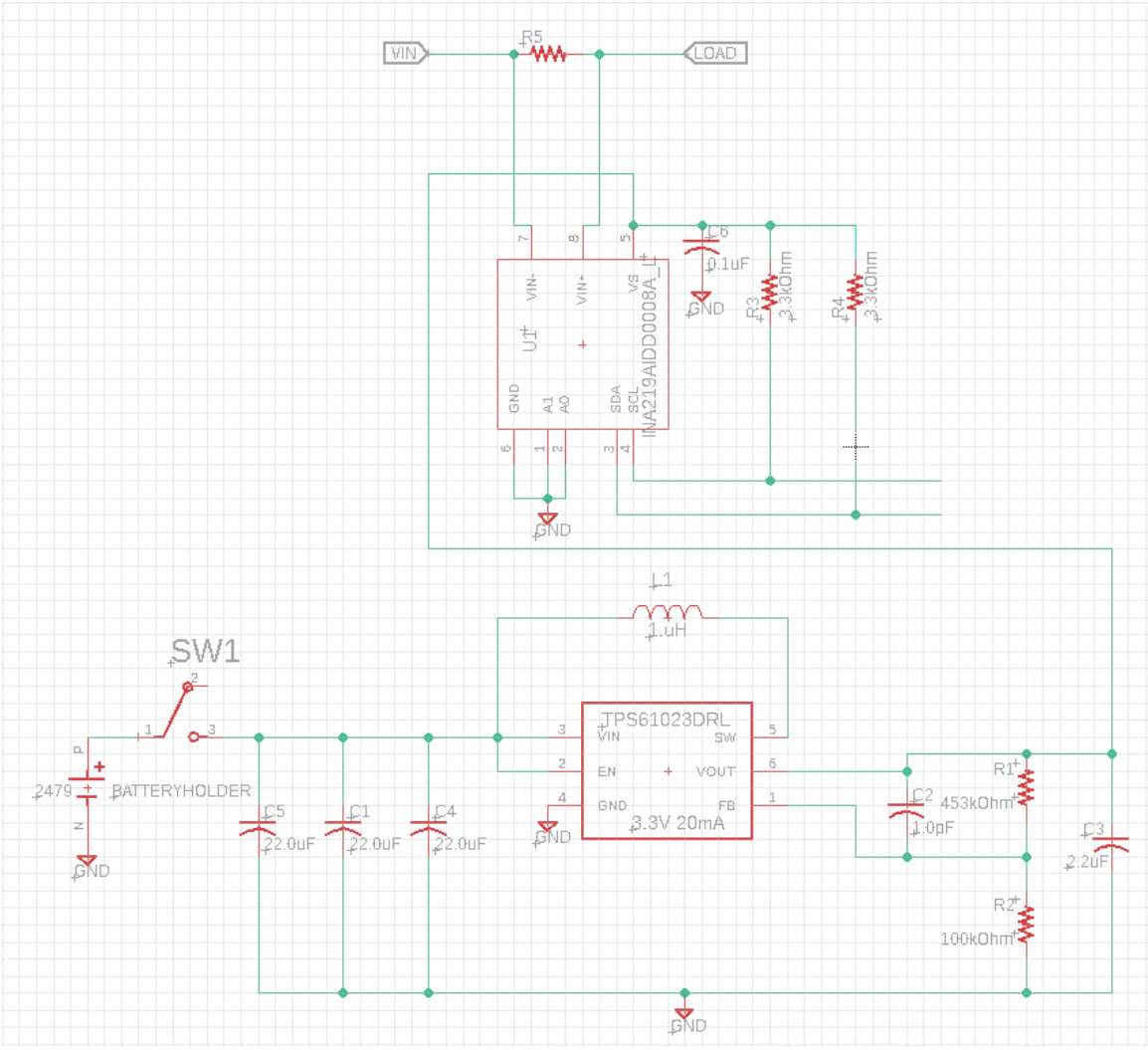

Figure 8: *Voltage/Current Sensor Schematic.*

6.2.3 ESP32 and LEDs Schematic

For this portion of the schematic we added in the ESP32 Microcontroller along with the nine LEDs we plan to use in our design. The ESP32 is connected to the voltage regulator

that is currently outputting a voltage of 3.3V. According to the datasheet a power supply voltage of 3.3V is necessary to power the ESP32. Connected to the ESP32 are the nine LEDs that we plan to use to indicate to the user when a node should be occupied. In an ideal circumstance, the LEDs require 20mA to properly function at their peak performance which is why it is necessary to know how much current is being outputted by the GPIO pins on the ESP32. According to the datasheet, the amount of current that is drawn from the GPIO pins is typically around 20mA. This satisfies the amount of current needed for the LEDs, so a resistor is not required to keep the LEDs from taking in too much current. However, for the purpose of this design we do not want to take the word of just the datasheet since there could be some technical issues with the ESP32.

In the System Testing section, we detailed a testing process for the ESP32 to ensure the functionality of the MCU. One of these tests will include a method to properly measure the output current of the GPIO to mitigate any potential issues that may arise in the future if these pins are not outputting the correct amount of current to the LEDs. If the current measures at above 20mA, we will edit to the schematic to add resistors to lower the current. If the current measures at below 20mA, we will leave it as it is since the LEDs can function below 20mA. Although the LEDs can function at below 20mA of current, the tradeoff is that the luminous intensity of the LEDs will fluctuate judging by the amount of current entering the LEDs. Regardless, a test on the LEDs will be performed to ensure that if the GPIO pins are outputting current below 20mA, we need the luminous intensity to correlate with the LED requirements stated in a previous section.

This entire system will work in a way that allows the components to work cooperatively to achieve intended function. The battery pack will power the voltage regulator and output 3.3V. This output line will connect to both the Voltage/Current Sensor and the ESP32 and provide power to both of these components. The sensor will then transmit the data to the ESP32 for processing. Once this data is processed, the ESP32 will power the LEDs in a specific order to indicate that a node needs to be occupied. While that is happening, the Power Supply connected to the sensor will power the Load. This is how the power is being distributed in the Advanced Breadboard. Which can all be visualized in Figure 9 below.

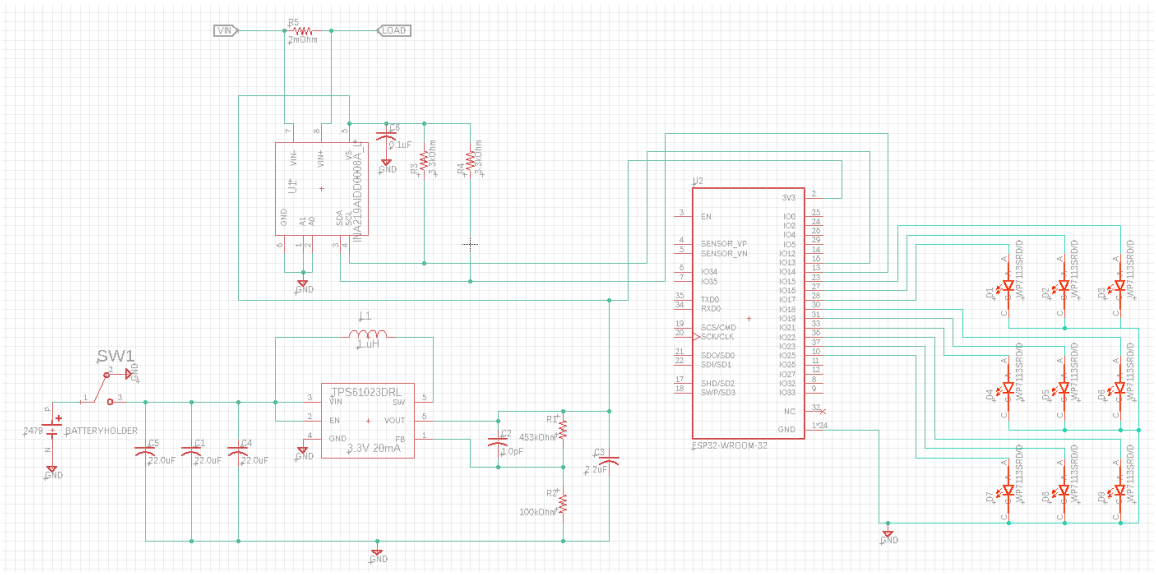

Figure 9: *ESP32 and LEDs Schematic.*

6.2.4 Final Overall Schematic

After all the necessary hardware testing has been completed, we designed the final overall schematic for the Advanced Breadboard as seen in Figure 10 below. As indicated in the datasheet for the ESP32, pins 35 and 34 are one of the UART pin connections on the ESP32; so, compared to the previous subsection, in the final schematic design we outlined the UART connections that are needed for the communication to the Raspberry Pi 4 Model B as discussed in previous sections. At first, we had planned to use a USB connection to assist our PCB in communicating with the Raspberry Pi. However, the Advanced Breadboard has a size constraint, so we needed to save as much space on the PCB as possible. So instead, we decided that it would be best to solder the UART connection pins to the Raspberry Pi for a more direct connection while also saving footprint on the PCB.

UART is a necessary connection for the ESP32 to communicate with the GUI simulation designed using the Raspberry Pi. From there, the ESP32 will control the LED logic to turn on the sequence of LEDs. A more detailed explanation as to how the LED logic will function will be explained in the Software Design Section of this report. Furthermore, a more detailed explanation for the software connection between the Raspberry Pi and the PCB will also be outlined in the Software Design section of this report.

A great feature within EAGLE allows us to check the compatibility of the schematic to ensure that theoretically it can function properly. As we run the ERC check tool, we can see that there are no major errors within our designed schematic. From there we are ready to create the PCB Layout Footprint for the Advanced Breadboard.

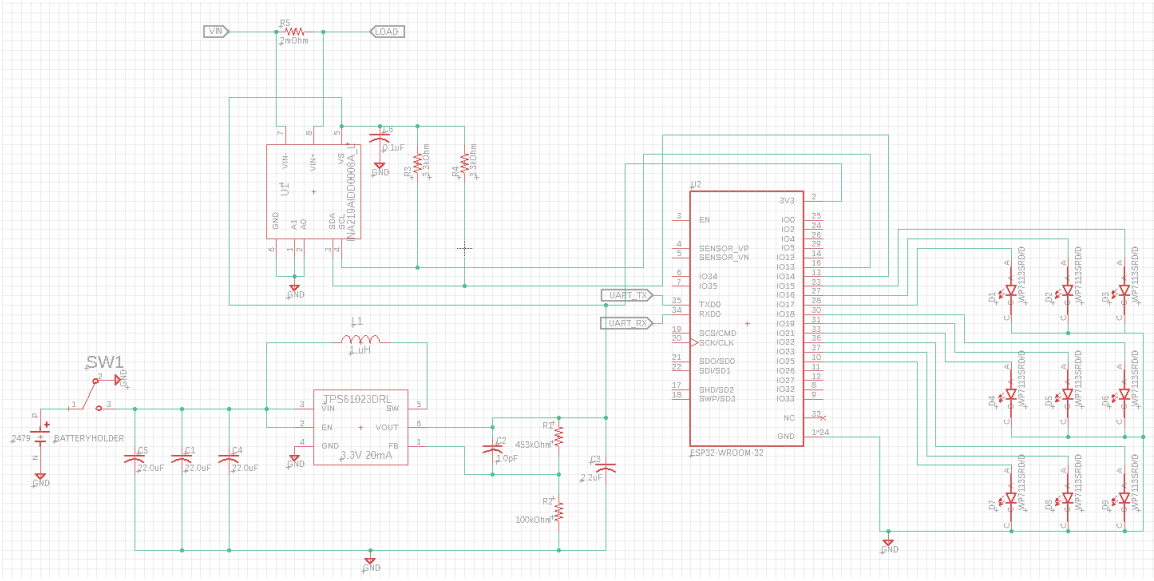

Figure 10: *Final Overall Schematic.*

6.4 Breadboard

The design of the breadboard will be very adaptable to the needs of the project because it is being 3D printed. Solidworks will be used to design the layers because it will be capable of building the layers as separate parts and then combining them to make a final assembly as seen in Figure 11. There are a few separate pieces of hardware that can be housed by the breadboard to make it more portable for the user. To fit all the pieces efficiently, the breadboard will be made with stacked layers for the different components. There will be a bolt in each corner to connect the layers and keep them together. This will allow for easier access to the components because the layers can be taken apart, but also provides a sturdy product. The bottom most layer will house the PCB and the layer above that will contain the metal contacts and the LEDs. The top layer will cover all the previous layers and will hold the raspberry pi. As prototyping and testing continues this design will be added to. However, this base will work for different configurations since every layer is separated and can be reconfigured.

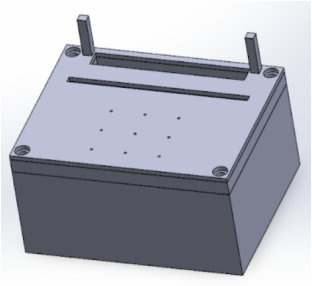

Figure 11: *Complete Assembly.*

6.4.1 PCB Layer

The PCB is the bottom-most layer of the breadboard. The underside of the layer will have rubber stubs on the corners of the board. The rubber will keep the board from easily

falling off the table which can bring harm to the circuit, board, and user. These will be super glued to the bottom of the board. Small stops can be designed for the bottom of the board but to get more accurate printing this will be avoided. 3D Printers print from the bottom layer up, so having the entire bottom layer printed above the stops could cause some issues. Therefore, this will be avoided by adding rubber stops after the printing process. Since the layers of the board are being held together by a bolt, there will be a nut on the underside of the board to secure the bolt. There will be a cut-out on the underside of the board for the nut to be super glued into. The underside of the layer will also contain ventilation vents for the PCB incase of overheating. These thin-slits will be made to avoid any large parts falling out or falling in to protect the board.

Within the breadboard there will be space for the entire PCB including battery packs, LEDs, and sensors. As seen in Figure 12, there are two holes that will be used as cable management that lead to a rectangle opening. All of the layers will have this same rectangle opening to allow for the cables to get to their respective components. The cables will connect to the open/short circuit sensor, nine LEDs, and the Raspberry Pi 4B.

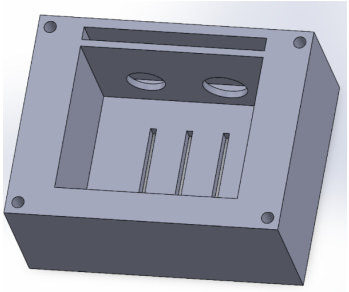

Figure 12: *PCB Layer.*

6.4.2 Contact/LED Layer

The next layer will house the metal contacts and LEDs. The metal contacts are sourced from regular solderless breadboards. Breadboards have metal contacts that connect five holes to make one node. For the purpose of this project, only three holes will be used per node with a total of three nodes. Therefore, three contacts will be required with spots for at least three holes as seen in Figure 13. The nodes and corresponding holes will be spread out further than regular breadboards to make it easier for the user to differentiate the nodes so two sets of contacts may be connected together. Between the different nodes there will be 0.7 inches of space as this was found to be a length that was far enough to make it easy to determine which node is which and not overstretching the component. Since the metal contacts are being sourced from pre-made breadboards, the contacts that line the source and ground will be collected. The reason that those will be used is that they have 10 different contacts, 5 holes each, all connected as one node. SInce this board aims to make circuit building easier for the designer, having the holes of each node spread out will help. The most efficient way to do this is by taking the node with 10 contacts connected and using only three of them in connection. This breakdown is discussed in FIgure 14. The metal contacts will be placed inside the layer so that the component does not need to be pushed down too far to reach the metal clip. To avoid having the clip go too high there will be a groove inside the layer to drop the height and give space to super glue the metal contacts to the layer.

Like the PCB layer, this layer will have holes in the corner to allow a bolt to go through. This will then fasten this layer to the bottom layer where the nut will hold the screw into place. The total height of this layer will be 0.3 inches because the metal clip heights are 0.3 inches so that the top of the layer is flush with the top of the metal contacts. There are nine holes that the LEDs will sit on top of and their wires will go through the holes to the PCB layer to have an easy connection to the PCB itself. These LEDs will not need to be fastened down because the diameter of the hole is smaller than that of the LED. The placement of the lights will allow the user to differentiate which hole or node should be occupied. Further testing will be required to ensure that the illumination of an LED does not interfere with the nodes. As seen in the PCB layer, this layer has the rectangular opening to allow for cables to go between each layer. This will be important for allowing wired connections to the Raspberry Pi 4B. This layer is far thinner than the PCB layer, and the further up the layer stack they will get increasingly smaller to ensure the components do not have to be pushed down too much to get to the metal contacts. Otherwise there can be issues with the circuit which is what this project aims to reduce.

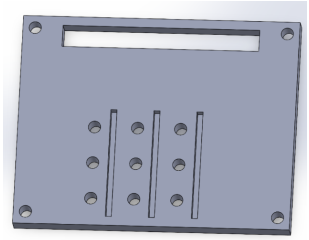

Figure 13. *Metal Contact and LED Layer.*

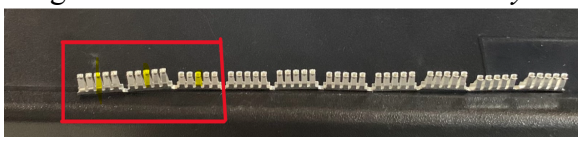

Figure 14. *Metal Contacts.*

6.4.3 Display Layer

The display layer covers the rest of the hardware, and will house the Raspberry Pi. This layer is very important safety-wise because it will separate the user from the electrical hardware. This layer will also be clear so that the LEDs can be seen from a top-view. The layer will need to be thin enough to see the LEDs but thick enough to withstand constant use from the users. A section of this layer will be dedicated to holding the Raspberry Pi and display. The display will be angled upward so that the screen is easier to read from many angles. Figure 15 shows two pillars and a thin rectangular opening, separate from the rectangle opening for the wires. Since the display is directly on the Raspberry Pi 4B, both hardware components will need to be on this layer so the user can interact with the screen. The Pi will be angled rather than sitting vertically otherwise the screen will be too difficult to see. The pillars are where the Raspberry Pi 4B will be rested against, while the slit will be where the bottom of the Pi will rest. This design will be mainly used for testing which will allow easy access to remove the Pi as needed. The final design will cover the Raspberry PI 4B to ensure no hardware is damaged during use. This will help to protect the user and the program.

This layer also has the holes for the components to be placed through. These holes will be the same diameter of a typical breadboard since those work well with all components. This layer will be printed with a thickness of 0.3 inches. This will be altered after some testing to ensure the LEDs can be seen through the clear filament. If the LEDs are clear, but the light doesn't make it easy to decipher between the nodes then the holes will need to be placed further apart from each other which will need to be done on this layer and the contacts/LED layer. The final accessory of this layer is the larger hole for the bolt to go through so that the top of it is flush to the layer so that nothing is sticking out that can harm the user or affect the circuit in any way.

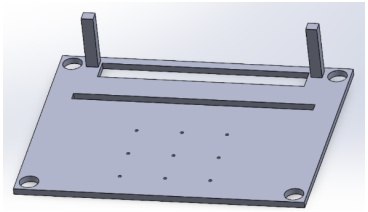

Figure 15. *Top Layer.*

6.4.4 3D Printing Preparation

With the design of the breadboard completely it must be prepared for printing. The design uses SolidWorks, but the file must be transferred to a software slicer to create the gcode for the printer. This can be done using Ultimaker Cura which can take the SolidWorks file and turn it into another file that most printers can read. Once uploading the file into the slicer software, a couple settings need to be adjusted for printing these layers. First is the infill density which will be set to 15% for testing, but will most likely be set to 20% for final printing. This is not a low percentage, rather, this is average for most prints with plastic as it maintains durability and doesn't require more printing time. Another setting are supports which will be added during test printing and final printing. 3D printers print from the bed of the printer up to the top by layers. Any part of the print that has a gap in the layers will require supports to ensure it prints properly. This includes the holes in the PCB layer for the wiring. Another setting is the orientation that the part will be placed on the bed. A more precise print is done when the part is orientated in a way that requires less supports which can be done by ensuring the flattest part of the design sits on the bed. Since all of the layers are flat on the bottom they will keep their orientation. With these settings the file can be saved and uploaded to the printer.

7. Software Design

7.1 Software Subsystem Block Diagram

The Advanced Breadboard's software subsystem, as shown in the Figure 16 block diagram below, consists of an intricate network that works togethers between software programming and tangible hardware interactions.In the literal center of the block diagram

what you will find is the schematic-to-breadboard translation algorithm, a pivotal script that transforms user-generated circuit schematics into practical, physical configurations. The algorithm is the main component of the system since it interprets electronic diagrams made by the users and effectively dictates the subsequent activation of LED indicators on the breadboard, which guides users in placing components accurately.

Inside of the Pi we have the entire program and aside from the algorithm which is not available for the users, we have the GUI. TheGUI emerges as the user's main interface, a digital canvas where electronic components are not merely represented, but where circuits can be connected, redone without consequence, stylized, and labeled. The GUI's design space is where users, through intuitive controls and a comprehensive toolbox, can drag and drop components to assemble their schematics. This interactive area allows for the design of circuits that range from the fundamental to the complex. However it is important to note that mostly simple circuits work well with the algorithm. The GUI's role extends beyond that of a simple design tool since it is an educational platform that provides immediate visual feedback, error detection, and the verification of circuits, thereby reinforcing the principles of electronic design through a hands on approach.

Beyond the immediate user interface, the system's architecture includes a simple communication framework that ensures a smooth connection between the software and the hardware components. This dynamic connection is facilitated by the Raspberry Pi and the ESP32 microcontroller, which work together to relay/send sensor data, including wattage readings and circuit integrity checks. The flow of information is not just one way either because the GUI is also designed to receive inputs from the hardware (pi), including the crucial open/short circuit test results. This back and forth flow of data is crucial for maintaining a real-time update of the system's status and ensuring that the user is continually told about the state of their circuit.

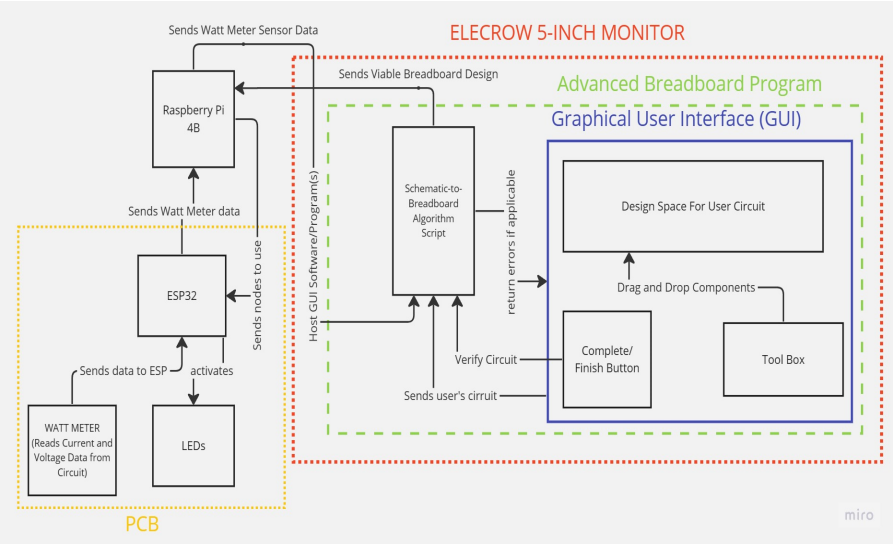

Figure 16: *Software Subsystem Block Diagram.*

7.2 Software Integration

The software will be housed on the Raspberry Pi 4B and will contain an algorithm for converting a schematic to breadboard layout and a graphical user-interface. The Raspberry Pi 4 will communicate with an ESP32 on the PCB to get sensor information including the value of the open/short circuit test and command the LEDs. The GUI will be displayed on an Elecrow Display that is attached to the Raspberry Pi 4B. This makes the breadboard portable and easy to use. The PCB with the ESP32 will be placed inside the layers of the breadboard to avoid user interaction.

7.2.1 Graphical User-Interface

The script with the GUI will contain the schematic to breadboard algorithm in it so that the user doesn't have to run different programs. The Raspberry Pi 4B is a strong enough microprocessor to handle the calculations given the limited functions required. When the user is ready to run the program they can turn on the Pi using the power button. They can then navigate to the desktop icon for the GUI. Once that window is opened the user will be able to begin working on their schematic. The algorithm will then decide what node's LEDs should be activated. The GUI will eventually receive information from the open/short circuit test and display a message to the user. This information will be sent out or received through UART to the ESP32 on the PCB.

The program will be placed on the Raspberry Pi because of its configurability for a display and its strong processor. The display will occupy pins 1-26 on the Raspberry Pi 4B which limits the amount of pins to be used for the sensors. Specifically the I2C communication pins and some of the UART pins are used by the display. It will have 8Gbs which is more than enough for the program. The Raspberry Pi 4B has USB ports for a mouse and keyboard. The user will only require a mouse for the operation of the GUI. Wired or wireless mice will be compatible. The Raspberry Pi 4B is compatible with the Raspian Operating System which makes development easier because it is common and free. Kivy will be used to create the GUI because it is open source and can be programmed in Python which is a more accessible language.

A Python program can be made an executable on the desktop with a customizable icon. This makes navigating the Pi very simple for the user because they will have experience with locating icons on a desktop. The user can then navigate the GUI with a mouse connected to the Raspberry Pi. Here, the user will follow on-screen instructions for building a schematic of their circuit. After some calculation, the display screen will show a picture of the breadboard with components that the user should copy onto the real board. The software will then connect to the board to begin the next part of the process. The Raspberry Pi will need to send data to the LEDs to activate or deactivate them based on the schematic. Then the Pi will need to receive data from the sensor to determine open/short circuits. The Raspberry Pi does not have GPIOs available to communicate with the open/short circuit sensor and the LEDs because the display screen is covering the required pins. There will be a microcontroller that will facilitate this communication.

7.2.2 Peripherals

The open/short circuit sensor and the LEDs are two peripherals that must be integrated with the software. These devices cannot be directly connected to the Raspberry Pi because of the lack of available GPIO Pins that they require for communication. The open/short circuit sensor requires I2C communication which is being used by the display on the Raspberry Pi. Instead, the sensor will be connected to the ESP32 because of its ability to handle I2C. There will also be nine LEDs that must be connected to GPIO Pins and they must be programmed to be output pins. The Raspberry Pi will not have enough pins for this and the ESP32 can handle this configuration. In order to retrieve the information from the ESP32 about the sensors and LEDs the ESP32 will be connected using UART on the Raspberry Pi.

The ESP32 has a USB connector that typically connects to a PC(host) to receive and send information. This host will change to the Raspberry Pi 4 where there will be serial communication with UART. The Raspberry Pi supports the serial library which allows it to open a port for communication with the ESP32. UART functions will be created to send and receive data. On the Raspberry Pi 4B, pins 12 and 13 are used for receiving and sending UART communication [Raspberry Pi 4B]. On the ESP 32 side, GPIO 16 and 17 are UART capable[ESP32]. This will allow for LED data to be transmitted from Raspberry Pi to MSP, and open/short circuit sensor data to go the opposite direction. It will be important to ensure both devices maintain compatibility by ensuring their baud rate is the same. The program running on the ESP32 will use interrupts to determine when data should be sent or received. This will allow data to be transferred when the GUI program requests communication. This is more convenient to the user who only needs to interact with one device rather than both.

7.2.3 Open/Short Sensor

The open/short circuit sensor requires I2C communication which can be configured on the ESP32. GPIO 22 and 21 will be used for SCL and SDA since those are designated on the pin layout [esp]. Device configuration will first require the internal register address for the sensor which is 0x40, 0x41, 0x44, and 0x45 where the last address is default so that will be programmed. There are four different outputs that can be given to the program: Shunt Voltage, Bus Voltage, Current, and Power [Wattmeter Sensor]. This internal register should have data on what the current or voltage is. The microcontroller will ask for the data in that address and compute whether an open or short circuit is detected. This will occur when the GUI requests this information from the microcontroller. This request will be when the user is done building their circuit and presses a button asking for the test to be completed.

7.2.4 LEDs

Since there is an LED for every hole in a node, there will be nine LEDs. The two LEDs for the voltage source and ground can be on the same pin since those will always be activated together. Their pin will also always be activated for every circuit. The remaining pins will require separate configurations. This will include assigning a pin to

each LED then setting those as outputs. GPIO Pins 14, 12, 13, 32, 33, 25, and 26 will be used since it has seven available ports which will be all the ones we need without using any other important pins like the ones with I2C or UART. The LEDs will be activated by the ESP32, but the program on the Raspberry Pi 4 will decide which LEDs based on the created circuit.

Since the pins were decided, they need to be directed to output using the command PinMode(PinX, OUTPUT). Here, X represents what number LED we are using starting from 0-7. At the start of the program all LEDs should be off so writing LOW to all of the pins will do that. Then there will be an interrupt set up so that when the program asks for the LEDs to be activated it can perform the operation of setting the GPIO to HIGH.

7.3 Schematic-to-Breadboard Algorithm

The design of the algorithm to translate user created/made circuits into real-life breadboard designs is the most complex and intricate task that requires a deep understanding of electronics, circuit theory, and the constraints imposed by the physical breadboard. To begin making the algorithm we had to first consider the limitations and specifications of the breadboard in question.

With a breadboard that offers three nodes, each with three holes (total of nine connection points). The constraints are further complicated by the fact that certain basic electronic components (resistors, inductors, and capacitors, collectively known as RLC components) will only be used, and each of these components will occupy at least two holes. This could be further complicated if other devices were available to put in the circuit design (but this breadboard only deals with RLC components). Moreover, in any given circuit, one node will be dedicated to the power line, and another for the ground line. The absence of a ravine (the gap that usually runs down the middle of standard breadboards) makes it so that the use of Dual In-line Package (DIP) components cannot be included in the circuits. Given these constraints, the algorithm must execute a series of steps, each crucial to the successful translation of a virtual circuit schematic into a physical breadboard layout.

The algorithm will be running in the background so that the user does not interfere with the process. It is integral that this part can be completed in a timely manner or it will defeat the purpose of the project. The process begins with the algorithm analyzing the circuit components inputted/placed by the user. It identifies each resistor, inductor, and capacitor (RLC), tagging them for the specific roles they play within the circuit. This classification is crucial as each type of component not only behaves differently but also has different physical dimensions that affect how they fit on the breadboard. Once identified, the algorithm calculates the number of holes each component will occupy. Given that the breadboard has nine holes across three nodes, this step is essential to prevent overcrowding since only one component can take up one tiny socket at a time.

Now that the algorithm has a list of the components to be used, they now need to be assigned two nodes for every component. With the components analyzed, the algorithm allocates nodes for the power and ground lines. The allocation is not random, instead, the

algorithm assesses the circuit's needs. With this in mind, it can place the power source line and ground to maximize the efficient use of the empty node(s) for the other components like resistors or capacitors.

Each component is analyzed and provided nodes for placement. This is half of the algorithm since a representative circuit will need to be shown on a breadboard. This is the final result of the algorithm process. Here, the algorithm employs a virtual representation of the breadboard, upon which it begins the complex task of placing the components. This process is iterative since it experiments with various configurations, but gives the user the “best” one. The mapping takes into account the electronic nuances of each component like how resistors might be placed horizontally or vertically. The main goal is to create a map that is not only compact but also logically organized, facilitating easier understanding and physical assembly. This circuit made by the algorithm is then sent over to the breadboard and also displayed to the user on the GUI.

With the design for the breadboard complete, information will need to be sent to the physical breadboard. As an integral part of the circuit mapping process, the algorithm also undertakes the task of LED mapping. It determines which LEDs beneath the breadboard should be illuminated to reflect the current layout, providing an intuitive, visual guide for the user during the assembly process. It is important to note that this does not mean it turns on the LEDs, it simply decides which ones would be used. This step is synchronized with the component placement, ensuring that each LED corresponds accurately to the designated node for a component. The LED mapping is a direct translation of the virtual design into a physical guide which is then sent to the ESP32 so that it can run its code for turning on the specified LEDs.

7.4 Graphical User Interface Program

The algorithm has done its part in showing the user what their design would look like. Following the placement of components on the virtual breadboard, the algorithm shifts into a validation mode. With help of the information returned from the PCB, it checks for potential electrical errors such as short circuits or open circuits that might arise as a result from the user making the physical breadboard circuit. If an error is found, the algorithm flags it and returns this information back to the user via the GUI, demonstrating an adaptive approach to this design process.

The GUI will have a popup for when an error is encountered because the user must be made aware. When the algorithm encounters a layout that is unfeasible or detects an error in the circuit, the algorithm provides the user with constructive feedback. It might suggest alternative components that occupy fewer holes or recommend a reconfiguration of the circuit to fit the limited breadboard space. If the error is in regards to an open or short circuit, it will let the user know as well. This phase is crucial for educational purposes, as it helps users learn from their design choices and understand the practical limitations of physical circuit design.

There is a lot of programming happening in the background, but this must be brought to the user. The algorithm's interaction with the GUI is what brings the virtual design to life.

Once an optimized layout is achieved, the algorithm sends updates to the GUI, which in turn reflects these changes in the virtual breadboard display. It's a dynamic conversation between the algorithm and the GUI, with each component placement or wire route immediately visible to the user. This interaction is not static but continues throughout the design process, ensuring that the virtual breadboard is always an accurate representation of the user's intentions.

One of the objectives of the project is to reduce the time it takes to design and test a circuit on the breadboard. The integration of the algorithm with the GUI is seamless and real-time. As the user places components and wires, the algorithm works in the background, continually updating the layout and ensuring that each action is valid within the constraints of the physical breadboard. On selecting the button to make the circuit on the GUI, the full information from the circuit is sent over to the algorithm. Note that the algorithm is part of the full program but not the GUI itself, it works in the background and is instead called from the GUI.

The GUI, crafted with Kivy, not only allows users to create and visualize their schematics but also the overall circuit within the virtual breadboard space. The designed interface will mirror the physical limitations and configurations of the Advanced Breadboard, so that users gain practical knowledge applicable to real-world circuit construction.

Inside of the GUI, the user's interactions with the schematic editor are more than just drag and drop motions, they're a series of inputs that define how the underlying algorithm would run in the background when making the real life schematic. As the user places each component and information and draws every connection, the software analyzes the schematic for electrical logic, adherence to breadboard constraints, and optimal LED node activation.

7.5 I2C

I2C communication, also known as Inter-Integrated Circuit, is a communication protocol necessary to this project. This is a type of serial connection that uses two lines, SCL and SDA. [46]. I2C differs from other protocols because multiple devices can be connected to it at the same time. This can be helpful in some applications, but for the purpose of this project it will not be necessary. The rest of this section will discuss how I2C is configured and how this will be done for the purpose of this project. While I2C is not the only protocol being used it is an integral one that requires precise programming.

I2C can handle multiple devices at once, but can only have one Master. The Master sets the clock signals on the SCL line, while data is transferred both ways between Slave and Master. A pull-up resistor is used to control which device can communicate so that only one device can do so at a time. The Slaves are active low so the bus is always set to high and eight bits of data can be sent at a time [46]. Since the Master, ESP32, will not be sending data to the device there will not be a need to take care of sending data in eight bits. Rather, there will need to be a method to handle receiving eight bit data. To begin

and end communication the SCL line should remain high while the SDA gets moved from high to low, then low to high [46].

Both the ESP32 and the Wattmeter Sensor will be prepared for I2C communication. This is because the Wattmeter Sensor is designed for this protocol specifically. It has four specific registers that contain output data, but only the values for the voltage and current will be used. The pull up resistor will be set to 2.4KOhms which is a nominal value for 3.3v devices which the ESP32 is [48]. The ESP32 will be set at the Master since that will be requesting data from the sensor. The first configuration to set is setting the Master and the Slave to correspond to the correct device. Then the selected pins will be set to I2C to prepare for this specific communication protocol. For Arduino, an I2C bus instance must be created and then the frequency can be set for the clock [48]. According to the device's data sheet the default address is 0x45, but there are four different options for addresses in case other devices connect with the same address.

For the ESP32 a read function does not need to be created because there is a built-in one in the libraries. This is one of the pros of using the ESP32 over another microcontroller. The process of the Master reading from a Slave device includes first sending a read request to the device address by also setting the SDA line to low. The Slave then acknowledges that a read request has been made. The data is then sent from the Slave through the SDA line to the Master. Upon receiving the data the master then acknowledges back to the Slave that the data was delivered. The pull-up resistor is then used to bring the SDA line back up to high to end the communication. The interactions would look different if more devices were connected but with the current design this will not be necessary to consider.

The only other device using I2C is the Elecrow Display but this is with the Raspberry Pi 4 and will not be involved with the I2C of the ESP32. A Daisy-Chain could have been created to get both devices using the I2C directly with the Raspberry Pi 4, but there were other peripherals to handle that required more pins so the ESP32 addition can handle all of these controls.

7.6 UART

UART is the Universal Asynchronous Receiver Transmitting communication protocol. This is another type of serial communication that is widely used.UART communication requires more configuration than the I2C protocol, but the ESP32 has many libraries to assist with this. The following section will describe what UART communication looks like and how it will relate to the working of the project. UART will be used to communicate between the ESP32 and the Raspberry Pi 4.

To configure the proper communication both devices must agree on a set of communication rules. These include the baud rate, number of data bits, presence of a parity bit, and stop bit [47]. The ESP32 provides a configuration function so all that needs to be done is to pass through each value. The baud rate is how fast data flows through from one device to another. The number of data bits is how many bits the receiver should expect from the sender for the specific data. The presence of a parity

helps the receiver to check if the data received is authentic. The stop bit tells the receiver where the message ends. For the Raspberry Pi, a UART Port can be activated using the terminal and the device, ESP32, can be given a name when connected.

The UART pins on each device will be connected using wires. Drivers will need to be installed on the ESP32 but will require some specific information on what the data transfer will look like [47]. Since buffers are used for sending and receiving the size of them will be set when installing the drivers. Both the ESP32 and the Raspberry Pi 4 will send and receive data so both functions will be considered here unlike the I2C. There is a finite state machine that handles the majority of the communication so the program will only need to call the prebuilt read and write functions [47]. These pre-built functions reflect the ESP32 side, for the Raspberry Pi 4 there are also read and write functions that can be used in Python once the port is opened. For the ESP32, reading and writing are similar but opposite operations which are not handled by the programmer.

There will be a lot of communication between these two devices. The Raspberry Pi 4 will write to the ESP32 which LEDs should be activated or deactivated. This can be done by assigning a number to all of the LEDs 0-8 where the number 9 will be sent to deactivate all of the LEDs. The ESP32 will receive that number and set the corresponding LEDs to high, unless 9 is received and then they will be set to low. The Raspberry Pi 4 will request Wattmeter Sensor data from the ESP32. This can be done by sending the number 11 to the ESP32 when this information should be fetched. The ESP32 will use I2C communication with the sensor and get the data it needs. Then this data can be formatted in a way that the Raspberry Pi 4 will accept. UART communication is the preferred method for these two devices since it is a wired connection and has more stability. There is a possible wifi communication option that will be considered if UART doesn't work as intended, but this is not preferred since wifi connections are not alway the most reliable or safe in high traffic areas. Wifi can be considered for future advancement of the board.

8. System Fabrication

8.1 Hardware Prototyping

With the components acquired and the design completed, fabrication and prototyping are necessary to allow for testing. Prototyping for the hardware is important because it is the basis for the software. The objectives for the hardware prototyping are to print one layer of the breadboard, set up some breadboards for sensors and LEDs, and getting metal contacts prepped for integration. Setting up breadboards for some of the devices is important because it allows for the components of the project to be tested separately which makes debugging and experimenting easier. The following sections will cover how the hardware will be prepared to allow for testing.

8.1.1 Materials and Components

The components included in the hardware prototype are the ESP32, Open/Short Circuit Sensor, LEDs, metal contacts, and ABS Filament. To complete the prototypes, secondary items will be needed like breadboards, jumper cables, LED, resistors, and multimeters. For the ESP32, out of the box the microcontroller is ready to use and does not require any physical setup. It does require a USB to microUSB connection to communicate with a PCB. For testing it required a breadboard to safely interact with other devices. Here, jumper cables were needed to avoid soldering anything to the pins this early on. For the open/short circuit sensor a breadboard was not required. Instead, the sensor had to be prepped using a flat head screwdriver so that its ports could be connected. The sensor and the ESP32 would need to interact so with this setup they can communicate using the breadboard and jumper wires. Multimeters will be sourced to test the output voltages and currents of these two devices to determine whether any resistors need to be added. Using the breadboard with the multimeters for this testing will help to avoid issues like blowing out an LED. The purpose of this prototype is to safely handle the devices and avoid shocks to handlers or ruining a device.

The prototype for the actual breadboard will be a printed Metal Contact Layer. This will include a layer printed using the ABS Filament and will have metal contacts secured. The layer was designed using Solidworks since it has a basic design and currently is not including an entire assembly. All of the layers are important and differ in their design and function, but the actual process of printing will be similar. It would also be a waste of filament to print all of the layers before testing. The prototype will include the metal contacts which are being sourced from a breadboard. Since the node will be spread out it will be more efficient to use the metal contacts that are under the positive or negative ends of the breadboard because those nodes are typically longer. The metal contacts will be removed and fastened into the printed layer. One of the breadboard's main functions is to hold components and connect them using metal strips. Therefore, this layer will be important to prove the functionality of the built breadboard. Without the success of this layer the rest of the project will not serve its main goal of being a functional breadboard that teaches the user how to avoid mistakes. The metal contacts will need to be secured in a way that does not allow them to be moved or broken when placing a component.

8.2 Software Prototyping

The hardware setup drives the design of the software prototyping. Software prototyping provides a way for the team to determine if the hardware they use are capable of handling the design of the project, and tests the software to see if it should be swapped for another. Software prototyping leads to testing of the software which will be important to accomplish before the next phase of the project because it will provide insight on what needs the most attention and what resources will be needed. Software prototyping in the context of the Advanced Breadboard project is an iterative process of creating a functional model of the proposed GUI and algorithm, made to assist users in transitioning from circuit schematics to physical breadboard layouts. This process is integral to the development of the project, ensuring that the software components, such as the GUI,

schematic-to-breadboard wiring algorithm, and the open/short circuit detection mechanism are made in alignment with the project's goals.

In order to prototype all of the software used in the advanced breadboard, we first have to analyze all the aspects that are involved. Below are the main components that are pivotal towards a successful completion of this project:

1. ESP32: This is the microcontroller in charge of connecting to the wattmeter sensor. Information is collected via i2c and wireless transmitted to the raspberry pi. The ESP is used instead of the raspberry pi due to the use of the Elecrow 5-inch monitor which uses the i2c pins on the pi. The ESP is an easier and viable option as opposed to daisy-chaining the two devices (sensor and 5-inch screen).
2. Open/Short Circuit Sensor and Detection: A vital feature for troubleshooting, the software will incorporate mechanisms to detect current responses and alert users of any open or short circuits in their design, which is crucial for learning and error correction.: This sensor proves integral to the project in user
3. Graphical User Interface (GUI): The GUI is a critical component that enables users to visually draw and plan their circuit schematics. The design and usability of the GUI are pivotal in simplifying the process of circuit design for users, especially those new to breadboarding.
4. ELECROW 5-Inch Touchscreen: This screen is one of the most important components for the entirety of the project as it adds onto the easy-of-access for the user. This screen will be used to digitally design the circuit that the user desires to make.
5. Schematic-to-Breadboard Algorithm: This algorithm is the backbone of the software, transforming user-input schematics into a practical layout on the breadboard. It ensures accuracy in translating the design while considering the limitations of the physical breadboard, such as the number of available nodes.
6. LED Activation Logic: Part of the software's functionality is to control the LED indicators on the breadboard, guiding the user in component placement. The software needs to accurately illuminate the relevant nodes based on the algorithm's output, enhancing the ease of circuit assembly.

8.2.1 ESP32

The ESP32 requires a way to receive programs to execute on its hardware. This is done by connecting a micoUSB to the USB on a PC. Since ESP32 is Arduino based it can be interfaced with the Arduino interface. First, the IDE can be downloaded freely since it is open source which makes it convenient to use in any environment. The ESP32 is not a board that immediately appears in the IDE so it must be downloaded [45]. Once the ESP32 Dev Board is on the IDE it can now be used to program the ESP32. The IDE must connect to the ESP32 by ensuring the type of board it communicates with and which COM port is being used. After some testing, it is determined that it is the DFRObot Beetle ESP32-C3 using Port 7. This is important information for testing because it will ensure that the bug is not coming from the communication between the IDE and the board.

The Arduino IDE has an open-source example program that can be run to see if the prototype set up is correct. After running a few of these it was determined that the prototype connection was active. The hardware prototype has the ESP32 on a breadboard with jumper cables connected to prepare for testing. Now that the software is prepped, testing can commence because there is now a way to transfer a program to the ESP32 to carry out. This wired connection also serves as power to the board which will power the prototype circuit.

8.2.2 Open/Short Circuit Sensor

The sensor is more involved with the hardware prototyping, but it requires calibration before testing can begin otherwise the output values may be off. This sensor also has a library for Arduino so this will require setup [Datasheet]. Downloading the library can all be done through the IDE with the provided link on the sensor's datasheet. The calibration code is also provided in the data sheet so its integration will be simple. The calibration does not involve the voltage responses, but instead is for the current response which will be integral to have correct since that value can help determine open or short circuits. Since calibration can be affected, and the nature of the project making it portable and adaptable, the calibration code will be included in the final product to ensure every measurement is accurate. This will be important to test during initial testing to ensure the calibration is done correctly. The sensor will be connected to the ESP32 on the breadboard and will be powered by a power supply for testing to ensure it receives the voltage and current it needs. The ESP32 will have the libraries installed and the calibration code ready for testing to be completed properly. It is important that the prototype software is accurate because it is the base for the rest of the project.

8.2.3 Graphical User Interface

The Advanced Bread project's success hinges significantly on the development of a robust and user-friendly Graphical User Interface (GUI). This GUI plays a critical role in enabling users to visualize, draw, and plan their circuit schematics efficiently. The development of this GUI was undertaken using Python, a default language in the Raspbian operating system, and was developed using Geany Programmer's Editor version 1.38. This report outlines the software prototyping process for the GUI component of the project. Below is an overview of all main components used in the development and design of the GUI, including the tools and the process of making a prototype.

The first step towards making the GUI for this project is to evaluate what tool we can use for development. This is in reference to a programming language, the operating system, and in what environment the GUI could optimally be created. Firstly, for the programming language, Python was chosen for its simplicity and the fact that it is pre-installed in the Raspbian OS, eliminating the need for additional installations.
The Integrated Development Environment (IDE) that we picked to develop the code for the GUI was Geany Programmer's Editor version 1.38 was selected for its lightweight and functional nature. Geany supports a variety of languages and offers features like syntax highlighting, code folding, autocompletion, and auto-closing of tags, making it

ideal for GUI development. It is also important to note that Geany is an IDE pre-installed when configuring the Raspberry Pi OS onto the board.

Within the code of the GUI we found that python has a large amount of development tools, libraries, and frameworks. Of the popular choices, we had Tkinter, Kivy, and PyQT. All were valuable options but PyQT stood out as it has a short learning curve and works well for desktop devices. While Tkinter was too barebones and Kivy is great but works better with mobile applications, which we were not using. And so the PyQT library was utilized for GUI development, owing to its comprehensive set of tools and widgets that simplify the creation of intuitive user interfaces. This library had to be installed via the terminal, as it was not a default component of the Raspbian OS. PyQT is renowned for its versatility in designing custom GUI elements, which was essential for this project.

Now that we had the main frameworks and sandbox for development ready, we had to begin the process of prototyping the actual GUI.

In order to have a successful GUI we had to create mockups and designs that were not only aesthetically appealing, but easy to understand, and still useful in their utility. The first step in the prototyping process involved creating initial designs and mock-ups of the GUI. This phase focused on the layout, ensuring that it was intuitive and user-friendly, particularly for beginners in breadboarding. This took place mostly on paper and/or Figma, an online graphic vector editor. On Figma, the user interface was made and then firstly evaluated amongst group mates, then other peers and users of breadboards. This step is key considering that an important aspect of making a “successful” board was to make it easier for other people to use.

After the multiple mockups were made and evaluated, we continued onto the actual development of the GUI. Following the initial design, the development entered an iterative phase, where features were incrementally added and tested. This approach allowed for continuous feedback and improvements, ensuring that the GUI met the specific needs of the project. Here we focused heavily on the ease of drag and drop features with the “open canvas” available to the users. Afterwhich, the users have features like a wire select to be able to connect their components together.

While the development phase is crucial it is also important to us to always take a step back and see how other people would evaluate our work. This is a key step because as an education tool, if the design was not intuitive to others, then users would be demotivated to use the advanced breadboard for learning purposes. Overall,user feedback was a critical component of the prototyping process. Trial runs with potential users helped in identifying usability issues and areas for improvement, allowing for adjustments in the GUI design and functionality. Another thing that was of importance to the GUI was its speed. Although the refresh rate for the 5-Inch monitor is 60hz, it would lag in a couple of moments, which led us to find a lack of efficiency within the python code.

While the GUI itself is a key part of the project, it was also important to work on how it integrates with the rest of the advanced breadboard. In that regard, the GUI was designed

to seamlessly integrate with the advanced features of the breadboard, such as LED activation and open/short circuit detection. This required careful planning and testing to ensure that the GUI accurately reflected the breadboard's state and user interactions. However, these features are not necessarily related directly to the GUI, instead they are actions that are caused as a result of the users actions within the GUI. For example, once the user is done making a schematic, they can finalize and “send” it to the breadboard. Depending on the status of their circuit, something will happen, which will be touched upon later in the software prototyping sections. For the relevance of the GUI, this is led by incorporating buttons with detailed text to instruct the user what to do. The final stage of prototyping involved refining the GUI based on accumulated feedback and tests. This phase ensured that the GUI was not only functional but also aesthetically pleasing and easy to navigate.

8.2.4 ELECROW 5 Inch Touchscreen

The ELECROW 5 Inch Touchscreen offers a high-resolution display and is noted for its user-friendly features, including a resistive touchscreen with a touch pen. Below will be the specifications of the touchscreen, its detailed setup, including driver installation and configuration for optimal performance.

Delving into the features and specifications, this device boasts a range of impressive elements.The key features and specifications of this device include a 5-inch TFT resistive touchscreen with an 800x480 resolution, compatibility with Raspberry Pi 4, 3B+, Banana Pi, Jetson Nano, and Windows 11/10/8/7, touchscreen functionality, a 60 Hz refresh rate, and the package contains the touchscreen, a CD, touch pen, HD adapter, and copper pillars, while setup requires an additional SD card to USB adapter and an SD card.

In order to use the ELECROW screen we must first install it and fill out all the prerequisites necessary. Firstly, we have to do the pre-setup to make sure we are set with the correct tools prior to using the 5-inch monitor. The pre-setup consists mostly of readying the SD card. The SD card used was not brand new, so it had to be formatted in order to install the proper operating system onto it. Additionally, the operating system installed onto the SD was Raspberry Pi OS, previously known as Raspian. In order to do this, the official raspberry pi website provided a download link with instructions. After installing the imager, it can be transferred onto the SD card by plugging in the USB to SD card adapter (with the SD card inside) into a computer or laptop and then selecting the proper operating system, drive, and raspberry pi model (4B in this case). Finally, after some time, all the information required to run the operating system on the 5-inch touchscreen is uploaded onto the SD card.

For the physical assembly of the monitor we had to carefully pay attention to the details of the pin layout both on the screen and on the raspberry pi. The ELECROW screen uses SPI and i2c to communicate with the pi, as a result, it has to use pins 0-26. And so, the physical setup involved attaching the 5-inch LCD to a compatible Raspberry Pi 4b board. The package included copper pillars for secure mounting, ensuring a stable connection for both display and touch functionality. Additionally, a connection to the pi and the screen is ensured with the HD adapter provided by the touchscreen.

As noted in the pre-setup, we configured an SD card for use. Its use here focuses on adjusting the contents within the actual SD itself. After following the pre-setup, there should be multiple files in the SD card. One of the files that we have to change is the config.txt file, which remains a crucial step in the setup. The additional information emphasizes the need for specific adjustments, especially for different versions of the Raspberry Pi OS. For instance, systems running on Bullseye required a modification from "dtoverlay = vc4-kms-v3d" to "dtoverlay = vc4-fkms-v3d" in the config file, a critical step for the display to function correctly. However, that adjustment in specific, was not needed. Instead, a couple extra lines detailing the HDMI mode and overlay were added to the end of the config file and saved on to the SD card. After all the proper changes are made, it is saved onto the SD card and it is removed from the USB adapter and then placed into the Raspberry Pi 4B's SD card slot.

One of the final steps of setting up the screen is to give it power and as such, test its general capabilities of turning on and displaying the correct things on the screen. Upon powering the Raspberry Pi, the initial display check was to ensure that the hardware setup was correctly done. This verification step was vital before proceeding to driver installation. After this, steps on the screen were followed for initial touchscreen preferences and setup. Afterwhich, the device loads up the main desktop screen.

When we were finally inside of the actual screen and could interact with it as a result of a mouse, keyboard, and a functioning operating system. We had to install multiple drivers, of which we tried both options (an online installation and an offline installation), of which they both worked. For systems with internet connectivity, driver installation involved downloading the driver from GitHub and executing a series of commands. This process was straightforward but required a stable internet connection.

When we restarted the entire process and tested operating the screen without internet access, we used the offline installation. In scenarios without internet access, the driver had to be manually downloaded and transferred to the Raspberry Pi. The included CD in the package could be used for this purpose, although it was recommended to download the latest drivers from the official website for the most updated support.

8.2.5 Schematic-to-Breadboard Algorithm

The development of the Schematic-to-Breadboard Algorithm within the Advanced Bread project is arguably the most important task aimed at bridging the gap between conceptual circuit designs and their physical realization on a breadboard. The algorithm serves as the core of the Advanced Bread project as it is the main reason people might be enticed to use this breadboard as opposed to regular ones. Its main role is to translate user-input schematics into a practical and accurate layout on the breadboard. This involves intricate processing to understand the circuit components, their connections, and how these can be effectively mapped onto the physical constraints of the breadboard, like the limited number of nodes and specific node configurations.

The actual prototyping of the algorithm was a long and complicated process, the initial development phase of development focuses on establishing the foundational elements of the algorithm. Here, the main objective is to create a basic version capable of handling simple circuit schematics. This stage involves coding the core logic that interprets these schematics and defines how the nodes on the breadboard will be utilized. A significant part of this phase is devoted to setting up the fundamental rules for schematic interpretation and node allocation, ensuring that the algorithm can accurately translate the schematics within the limitations of the breadboard's layout. Early testing in this phase involves inputting simple circuit designs to verify the algorithm’s capability in correctly mapping these onto the breadboard.

Following the first phase of the prototyping process we looked at the Feature Enhancement and Refinement step. As the algorithm progresses into the second phase, the focus shifts to enhancing its capabilities to handle more complex circuit designs. This involves introducing sophisticated logic capable of understanding intricate connections and multiple components within a circuit. An essential aspect of this phase is the development of robust error handling and user feedback mechanisms. The algorithm is refined to detect potential errors in schematic designs, such as impractical connections or layouts that exceed the breadboard’s capacity. This phase's testing becomes more rigorous, with complex circuit schematics being used to assess the accuracy of the algorithm's translation capabilities and its efficiency in detecting and communicating errors.

Now that we have considered the algorithms capabilities and use, we have to then integrate it with what we already have, which begins our integration testing. Integration testing is a critical phase where the algorithm is tested in conjunction with the other components of the system, particularly the graphical user interface (GUI) and the breadboard hardware. The objective here is to ensure that the algorithm seamlessly integrates with the GUI for schematic input and effectively communicates with the breadboard hardware, especially regarding the activation of LED indicators for node allocation. Tests conducted during this phase are comprehensive, encompassing the complete system's functionality from the point of schematic input through to the visualization of the layout on the breadboard.

Now that we have integrated the algorithm and completed the previous processes, we have to see how this algorithm fairs with other people aside from the primary group working on this project. The final phase involves putting the algorithm into the hands of actual users, providing invaluable insights into its real-world application and usability. This phase is centered around conducting user testing sessions, where the participants are asked to input their circuit designs and use the algorithm to map these onto a breadboard. The feedback gathered from these sessions is crucial in identifying any usability issues, practical challenges, and areas for further improvement. The algorithm's performance is evaluated not only in terms of its technical accuracy but also in how effectively it aids users in the process of circuit design and assembly.

8.2.6 LED Activation Logic

Building upon the foundational Schematic-to-Breadboard Algorithm, an essential part of the software prototype for the Advanced Bread project is the LED Activation Logic that is associated with it. This logic is linked with the algorithm, serving as a visual guide for users in placing components accurately on the breadboard. The primary aim is to illuminate specific nodes on the breadboard in correspondence with the schematic design processed by the algorithm.

The development of the LED Activation Logic began with integrating it into the existing framework of the Schematic-to-Breadboard Algorithm. This integration makes sure that once the algorithm maps a schematic onto the breadboard layout, it simultaneously identifies and signals which nodes should be lit up. The LED logic is programmed to control the brightness and on-off states of the LEDs under each node, providing a clear and intuitive indication for component placement.

Testing for the LED Activation Logic was done at the same time with the Schematic-to-Breadboard Algorithm, focusing on its accuracy in highlighting the correct nodes as per the circuit design. The prototype underwent several iterations to correctly get the synchronization between the schematic translation and LED illumination. In the final stages of prototyping, user testing played a critical role. Feedback was gathered to assess the effectiveness of the LED indicators, mostly seeing if it was easy to tell which nodes were lit up. This feedback informed minor adjustments to the logic, optimizing its functionality and user experience.

The LED Activation Logic stands as a vital component of the software, enhancing the usability of the Advanced Bread project. Its successful integration with the Schematic-to-Breadboard Algorithm not only makes a better flow with the circuit assembly process but also significantly aids in the educational aspect of the project, making it a better, more effective tool for learning and experimentation in electronics.

8.3 PCB Layout

For the PCB layout, we were able to use EAGLE to create the footprint as seen below in Figure 17. Since we have a size constraint for the Advanced Breadboard, we decided to first design the outer shell of the Advanced Breadboard and allow that size to determine the size of our PCB. For that purpose we decided that we needed to limit the size of the PCB to 3 inches by 3.5 inches. This limit was necessary to fulfill the size requirement and constraint mentioned in previous sections. To conserve the space on this PCB we decided to use a double-sided PCB to place our components. The two components that we placed on the opposite side are the battery holder and the switch. This is so we can design a compartment on the bottom of the Advanced Breadboard to turn off the batteries and to allow easy access to the batteries for replacement in the case that they are drained. Another thing we added to the PCB layout are the drill holes around the corners to screw in the PCB for stability into the 3D printed housing. This will ensure that the PCB does not suffer any damage due to movement, as well as keep the components and hardware of the PCB in a stable position.

When designing the PCB layout, I needed to measure the distance between each LED to ensure there is enough space to place the metal clip for the Advanced Breadboard. The space between each LED horizontally is 1 inch, while the distance vertically is 0.9inches. This may be subject to change as we start fabricating and ordering the PCB and physically testing the parameters of the entire design. To prevent too many changes between the initial design and the final design, we need to make physical measurements for the most ideal spacing between these LEDs. Another thing we needed to account for is the Shunt resistor, resistor R5, connecting the Power Supply and the Load of the user. That is why it is placed at the top left of the PCB layout, to allow easy access to the resistor pins. Finally, we also needed to keep the UART pins of the ESP32 easily accessible to allow the Raspberry Pi 4 to connect to the ESP32 for the GUI simulation. When looking at the PCB layout, you can see that we left the pins pointed outwards for easy accessibility and enough space to solder the UART connections to the Raspberry Pi 4B.

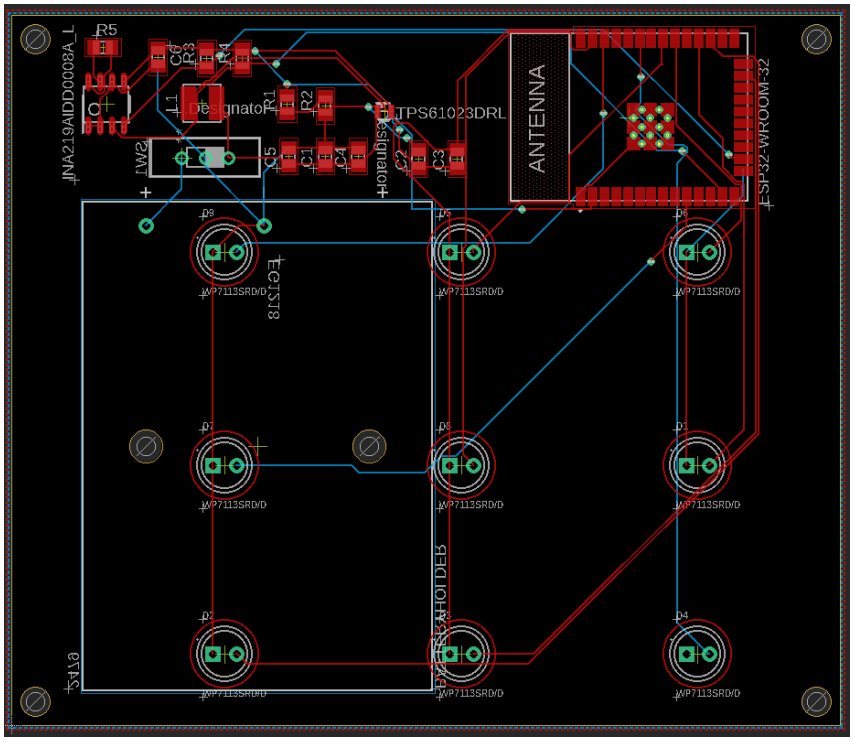

Figure 17: *PCB Layout.*

9. System Testing

In this section we will be discussing the testing procedures that we used during our prototype testing as well as system integration. The details of our testing procedure will be extensive to ensure that our prototype as well as the final design will work as intended while meeting all of the hardware and software requirements. Since our design requires many components, there are many ways to test these components to ensure functionality.

The overall testing goal is to check if the individual components work as intended and if they can interact with each other according to the design. The software housed on the Raspberry Pi will need to communicate with the ESP32 because that will have sensor information and will be connected to the LEDS. The ESP32 is the go-between for the software and the peripheral hardware. This device being able to receive a command from the Raspberry Pi4 and reacting to the hardware accordingly will be the most important part of the integration process. This process will be tested by first separating each component and testing it. Then these components will be tested in groups. This includes separate hardware and software testing that will be discussed below. Once everything is tested separately, the components will be paired up to test such as the ESP32 connecting to the LEDs.

The LEDs will be tested to see what resistance level will be required to maintain the nominal voltage and current it expects. The ESP32 will undergo software testing to ensure that it is working properly. At the success of those tests, the ESP32 will be given a program to activate and deactivate the LED on a breadboard. The conclusion of this test will prove the concept of integrating some components together. Another integration test will include connecting the LED display to the Raspberry Pi 4b and using a mouse and keyboard to operate the Pi using the screen. This integration is integral to the users ability to interact with the GUI. The open/short circuit sensor will need to communicate with the ESP32 as another integration test, specifically with I2C communication.

The overall integration between the hardware and the software will be focused on the ESP32. This microcontroller will have its own program for giving commands to the LEDs and receiving data from the open/short circuit sensor. The Raspberry Pi 4b will handle requesting data from the ESP32 through UART communication. This will be a software based exchange besides connecting pins through wired connections. There will be very little testing in regards to the plastic part of the breadboard because that will be designed around the other components. However, one layer will be printed to test the lengths of the board in comparison with all of the components. Testing all of these separately and together will create a stronger foundation in the experimentation of the project because the results will provide the team with direction. Depending on the difficulty of the integration, more effort and resources can be focused on those parts. Understanding the overall integration also makes PCB design more efficient because it will be clear which pins are required and what components are going to be needed. These components may also have operating voltage and currents that will be observed and noted. The following section will cover the different specifications for testing the hardware and software separately. This will provide more details and give way to more specific conclusions.

9.1 Hardware Testing Procedure

Our design is mainly broken up into two sections, the first being hardware components while the second is the software application. The hardware components mainly compose of electrical components that need to be tested for voltage, current, and the amount of power. While the software application needs to be tested to ensure there are no bugs or unwanted implications. Overall, these components will eventually end up needing to

cooperate with one another, so it is crucial that all the components and applications are tested to ensure the entire design is fully operational and accurate.

For the hardware components, since they are mainly all electrical components we need to test for proper power output and input. Out of the entire testing phase, this is the most crucial. The team needs to ensure that the hardware components work to the best of their ability and that they are fully operational. Especially when it comes to releasing a product into the market. The reason why it is more crucial for the hardware components to be thoroughly tested and ensure that they are operational compared to software applications, is because once it is released it will be a lot harder to fix than a software application. That is why we need to ensure that our hardware components are functioning properly now to ensure that the PCB is designed with correct hardware functionality and tested components. In the end, this will help our team avoid any future issues with the PCB and overall any issues with the Advanced Breadboard. To test the hardware components we will be using tools and equipment from the University Labs as well as any equipment we have in our personal workstations.

9.2 Hardware Testing

The following subsections will outline the hardware testing the electrical components used for the Advanced Breadboard. We will outline the theoretical outcome of using these components, testing of these components, and the actual outcome of the components. We will also outline how testing these parts are relevant to the overall design of our final product. Furthermore, we will outline the strategy we developed as a team to test and create the final design of the Advanced Breadboard.

9.2.1 Battery Testing

The main crucial part of our design are batteries that are being used to power our PCB along with any components connected to the PCB. We need to ensure that we test the output voltage of the batteries so that they align with the voltage that is needed to power the PCB. According to the datasheet given by the battery manufacturer, we should expect to see a voltage between 0.9V-4.5V. Since these are non-rechargeable alkaline batteries, we can expect that the battery capacity will not be at 100% capacity. We can expect the voltage to be less than 4.5V as seen in Figure 18. To test the output of the batteries, we can place the three AAA alkaline batteries in a battery pack. Once placed in the battery pack, we can connect a digital multimeter to the positive and negative output pins from the battery pack holder with the batteries inserted inside. If the multimeter reads a voltage between 0.9V-4.5V then we know that batteries are good enough to power the PCB. However, since the capacity of the batteries are reflected by the voltage they output, we want to ensure that the batteries can last as long as possible. For this reason, we believe it is best to make sure the batteries can output at least 3.85V to ensure that they are at least 85% capacity. Once we are sure that the battery holder along with the batteries are working properly, we can confidently say that the battery testing is a success. We will then be able to solder the pins of the battery holder to the PCB knowing that this hardware component is working as intended.

Figure 18: *Battery Pack Measurement.*

9.2.2 Voltage Regulator Testing

Testing the Voltage Regulator posed an unexpected issue. Unfortunately, the TPS61023DRLT was a lot smaller than we were expecting, thus making it impossible to test using a traditional breadboard. Instead, we came up with a solution that can output the same values as the voltage regulator, and help us determine the least amount of current needed to power our PCB. Since we are using batteries, we want to be able to conserve as much power as possible. For this solution, we used a power supply to output the same values as we were intending for our voltage regulator. These values were 3.3V at 20mA. Although we could not test the actual voltage regulator, we were still able to simulate it in person to measure the actual output of what it would output. To test this, we used a digital multimeter to make sure we were outputting the exact values as we entered on the power supply. Once we ensured it was correct we decided to connect the output of the power supply acting as the voltage regulator to the VCC of the Voltage/Current sensor as well as the ESP32. Once the components powered on, we tested their voltage and current to ensure they were receiving a proper supply to properly function without any future issues.

9.2.3 Voltage/Current Sensor Testing

Testing of the Voltage/Current sensor is split into two parts. The first part is the physical testing of the component and ensuring the purpose of the sensor is functioning as intended. While the second part requires software testing to ensure the sensor transmits the data using I2C. For the purpose of this section we will only focus on the hardware testing of this component; and, on the software testing we will discuss the details of how we tested the sensor from a software perspective.

In regards to the hardware testing we needed to test a variety of features that are offered by this sensor. For the sake of our design, the input voltage of the sensor is different from the “user’s” power supply and the load. That means there are two power sources going into the sensor that need to be differentiated from the other. The first power supply source will be the power supply used by the user, this is the power supply that is being tested for current and voltage. While the other power supply is coming from the voltage regulator and hooked up to the battery pack. The voltage regulator will output 3.3V and that will

connect to the VCC of the sensor. To ensure that the correct amount of current and voltage is being supplied, we will be using a digital multimeter to measure the VCC of the sensor as seen below in Figure 19.

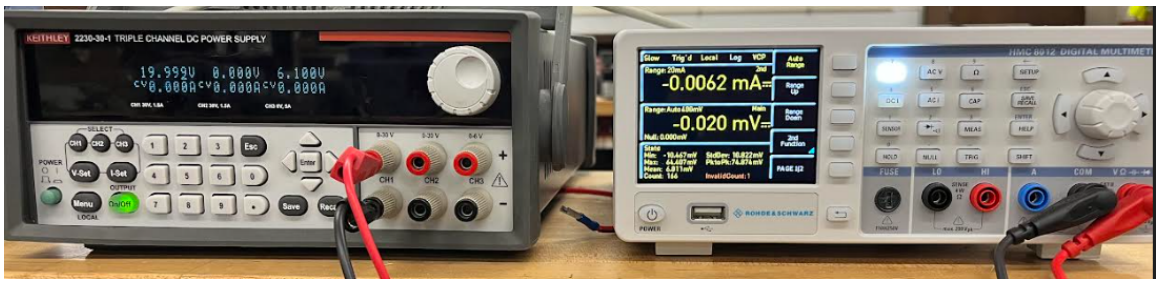

Figure 19: *Bus Voltage Measurement.*

Next, we need to test the VIN+ which is also referred to as the user's power supply. This can be done by testing the voltage and current using a digital multimeter, if the voltage and current is equal to the voltage and current displayed on the power supply, then we can confidently say that the VIN+ of the current sensor is functioning as intended. Next, we can test the VIN- which is indicated as the load in the sensor's datasheets. In our case, this can be referred to as the user's circuit. A visual of this can be seen below in Figure 20. We can measure the voltage and current using a digital multimeter and ensure that the voltage and current displayed is similar to the voltage and current displayed in our VIN+ testing. The reason why we state that the measurements will be similar is for a particular reason. In the Electrical Characteristics section of the datasheet for the INA219 it states that the measurement between VIN+ and VIN- is referred to as the differential voltage across the shunt resistor, otherwise known as Vshunt or Bus Voltage. Because the shunt resistor has a very small resistance, we can expect that the voltage across the resistor may be around 32mV. Although this is a very small voltage, relative to the voltage of the power supply, we have to still expect that the voltage input of VIN+ will not be exactly the same as the voltage output of VIN-.

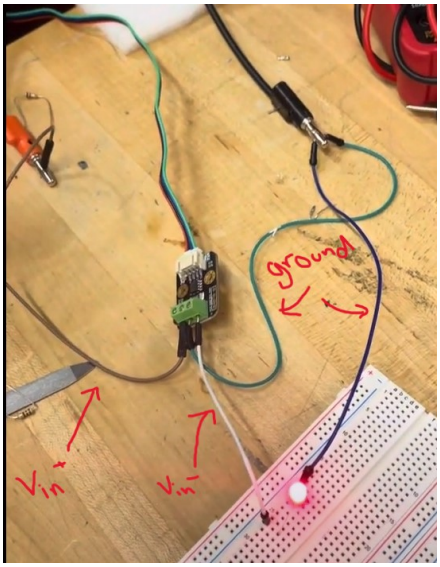

Figure 20: *Power Distribution Through Sensor.*

For the hardware testing of the sensor, we connected the VIN+, the variable power supply, and the VIN-, the user's circuit. For this demonstration, the user's circuit was an LED to give a visual representation of the power being transferred from the power supply to the load through the sensor. This will ensure that the sensor is able to perfectly allocate the power from the power supply to the load without any issues. If the LED lit up, then we have a visual demonstration that one of the key features of the sensor is properly functioning. If the LED did not light up, then we also have a visual demonstration that there must be an interrupted connection in the sensor that is preventing the power from being delivered to the load from the power supply. This testing step is very crucial and if the sensor was not functioning as intended; then, it would have posed a major issue within one of the major components of our design.

9.2.4 LED Testing

Testing the LEDs from a hardware perspective is a very small, but important test for the case of our design. From an outside perspective, it seems to be designed in a very basic way for the Advanced Breadboard, but essentially it plays a large role with the interaction between the user, the software, and the other components. In this subsection, we will cover the purpose as well as the testing procedure we used to test LEDs. In a previous section we outlined that the ESP32 GPIO pins will connect to the LEDs to control the sequence logic as to when the LEDs will turn on. We outlined that the issue arises with the amount of current being outputted by the GPIO pins. For this test we will use a digital multimeter to measure the current entering the LEDs and come up with a viable solution to ensure the proper functionality of the LEDs. According to the datasheet for the LEDs, the typical amount of current for the highest luminous intensity is 20mA. The use of the digital multimeter is to ensure the LEDs are drawing a maximum current of 20mA. The result of this test may change our design schematic as well as the entire functionality of the LED system which is why this is a very crucial test. This test will have one of two results. Either the current absorbed by the LEDs is above 20mA or below 20mA. If it is above 20mA then we can resolve this by incorporating resistors to limit the current flow. If it is below 20mA, we can keep the current schematic design since the only thing that will change is the luminous intensity of the LEDs. However, we must ensure that the current is not too low to the point where we do not satisfy the LED requirements covered in Section 2.4. Since the datasheet for the ESP32 states that the output current for the GPIO pins are around 20mA, then we can safely infer that the LEDs will function in their most ideal performance for the highest luminous intensity as seen happening in Figure 21.

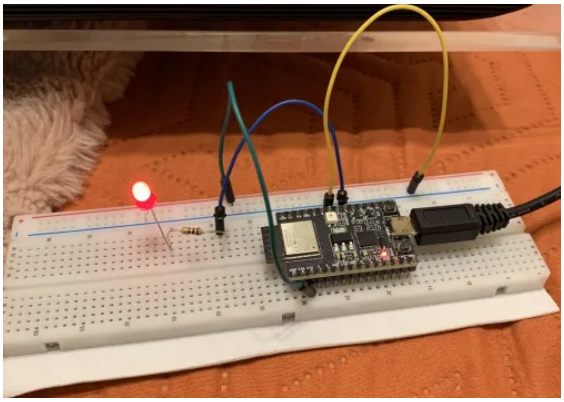

Figure 21: *LED Test w/ ESP32.*

9.3 Software Testing Procedure

For the software application, we plan to test each part of the application to ensure the application works as it is intended. By first ensuring the application works, we can then move on to testing the software with each of the hardware components to ensure that they are compatible and are able to work together in the final design. For the software testing, it is crucial that the software is able to function without any bugs or unforeseen issues. However, compared to the hardware components these issues can easily be debugged and resolved in order for the Advanced Breadboard to function as a whole with all of its components. That is why it is best if we focus the majority of our testing to the hardware components, and focus a minority of the testing to the software. This will ensure that the final PCB design can properly work from a hardware perspective, while also ensuring that the software is able to function as intended. If not, any changes to the final design can be easily made without any implications if it is a software issue.

9.3.1 GUI Testing Detail

Before the actual development of the GUI we had several wireframes developed in Figma to determine the UI/UX design and its viability to ourselves and potential future users. After testing multiple designs with other people not involved with the project, we found that most people liked the minimalist design we made. With a somewhat lack of intricacy, it was easier to focus on the circuit design a lot more as opposed to trying to understand what is going on, or get distracted by a lot of color.

After we replicated our wireframe into an actual GUI, like in Figure 22, the testing phase ensured that the user's interaction with the system was smooth and intuitive. This phase begins with usability testing, where the layout, design, and flow of the GUI are evaluated. We observed ourselves as we navigated through the interface, with particular attention paid to how easily they can access various features and whether the interactive components are behaving as intended. Common tasks like selecting and placing components from the toolbox (drag and drop), routing wires, and saving designs were checked for ease of execution and any potential confusion or frustration. We took feedback with a strong scrutiny as it is one of the most important parts of the project. If the GUI was too complicated to use or had a steep learning curve, then the necessity of

our advanced breadboard would diminish. Some common things we found to be of “hindrance” to a lot of people was the lack of an undo option. While it seems like common sense to add such a feature, it was not initially thought of, instead a drag and drop towards a trashcan was initially encouraged. But after user testing, the trashcan was removed and a redo function was added. Additionally, we found that the addition of a short guide upon opening the GUI, significantly increased user understanding.

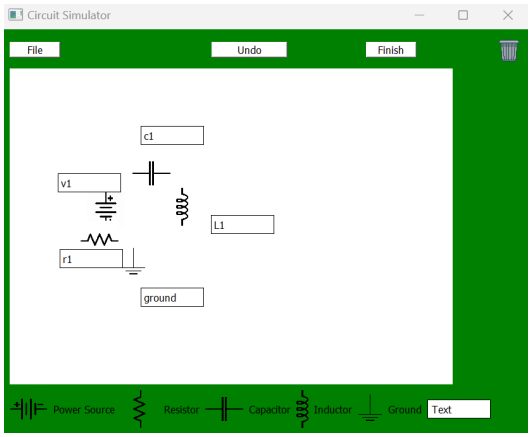

Figure 22: *Graphical User Interface (GUI) for user circuit design*.

Next, the GUI's functional testing involves rigorous checks of all interactive elements within the software. Each button, slider, and input field is tested for the correct action and response time. The drag-and-drop functionality was tested a lot of time so that the components snap to the grid accurately and that wires bend and route with precision but also with respect to the components. We found that keeping track of the components coordinates and then sending them over to the algorithm along with a label, was the best option to make the breadboard schematic. The feedback system within the GUI is also key considering the fact that when users make errors, the system should provide clear and helpful messages to guide them towards the resolution. For example, the algorithm checks for loose components, which is when a component is simply not connected to anything on the circuit.

For component placement validation, a series of predetermined circuit designs were made and re-designed into the GUI to test the virtual breadboard's accuracy. This involves a comparison between the user's schematic and the breadboard's response in terms of component placement and wire connections. Additionally, we tested both circuits and their currents/voltages at each node. The user made circuit was replicated on Multisim, and tested. If those values mentioned the physical data collected from the breadboard schematic made from the algorithm, then the circuit is correct. The virtual breadboard must reflect the user's input precisely, with each component and connection accurately depicted. Errors were also purposely and deliberately introduced to assess the GUI's ability to detect and communicate these issues effectively to the user.

9.3.2 Algorithm Testing Detail

The algorithm's testing is centered on its ability to translate the user's schematic into a viable breadboard layout. The logic verification involves feeding the algorithm a set of predefined schematics that represent various levels of complexity and design variations. The output is then examined to ensure that the algorithm not only places components correctly but also adheres to the physical constraints of the breadboard, such as the limited number of nodes and the specific requirements for power and ground lines. Error handling is a significant aspect of the algorithm's functionality especially if we want it to run smoothly. The software is presented with problematic schematics that are either incorrect or impossible to implement given the breadboard's limitations.

The algorithm must not only find and see, then reject these schematics but also provide feedback that helps the user understand why the design was unfeasible and what steps they might take to correct it. This feedback should be constructive and instructional, aiding the user in learning from their mistakes. We ran multiple tests with obvious issues. For example, if we added too many components, then the algorithm would return a log saying that there are too many components, which would be physically impossible to map to our custom breadboard. The errors and successful attempts were logged via the terminal and text file. It would show in real time when a user placed down a component, the label that component has, and the connection it has to other components. This information was then processed by the algorithm and as such used to make the real life schematic.

An example of an Unsuccess Translation can be found below in Figure 23. Here we have all the details placed by the user during the circuit design process within the GUI. Here we keep track of a multitude of important details. These details include the type of component, the label it is given, and its connections. Originally we kept track of its location on the GUI working space and returned that into the log. However, it is not actually relevant because what is more important is not where the components are, but what they are connected to. This is shown through the following section of the logs, unless there is an error. Errors could include a lack of ground, too many components, and open/short circuit, and a missing connection. The attempt made below shows that not only does the circuit use too many components at a time, but those respective components do not have any actual connections to each other, therefore making the circuit invalid. When an error is detected, the log ends.

```
(myenv) PS C:\Users\skari\OneDrive\Desktop\SeniorDesign GUI\myenv> python .\GUI.py
Circuit Components and Connections Log

Components:
- Power Source placed at 2023-02-10 06:15:23
- Resistor R1 placed at 2023-02-10 06:16:05
- Resistor R2 placed at 2023-02-10 06:16:40
- Capacitor C1 placed at 2023-02-10 06:17:12
- Inductor L1 placed at 2023-02-10 06:18:00

Error: Circuit has too many components, translation to a graph is impossible.

-- End of Log --

(myenv) PS C:\Users\skari\OneDrive\Desktop\SeniorDesign GUI\myenv>
```

Figure 23: *Unsuccessful User-Circuit Translation Log.*

Below, in Figure 24, we can find a successful translation for a user made circuit via the GUI. Here the user fits their circuit to be within all the guidelines of a safe circuit and within the restrictions imposed by the physical space of our custom breadboard. Additionally, all the connections are made and logged onto the terminal. Since the connections are denoted, the algorithm will know which respective components have to be connected to each other. As a result, it then outputs a graph representation. Showing that the resistor connects to the power source and the capacitor connects to the power source. This is representative of a closed loop circuit.

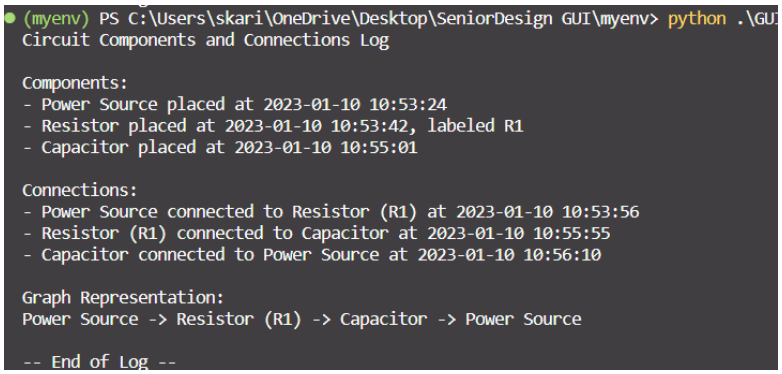

Figure 24: *Successful User-Circuit Translation Log.*

LED mapping accuracy is another critical component of the algorithm's responsibilities. Tests are conducted to ensure that the correct LEDs are activated for each given component placement within the schematic. This feature is vital for guiding users during the physical assembly of the circuit on the breadboard. For each test case, the expected LED behavior is documented, and the actual behavior was observed and recorded. However, the software portion of controlling this was not difficult. The algorithm first determines which of the LEDs would be most appropriate to turn on and sends that information over to the ESP32 in the form of a matrix, where the matrix depicts the breadboard nodes and holes in the breadboard, where the initial values are populated with the number 0, meaning that every LED is turned off. Where there is a 1 in the matrix, an LED is turned on in that position within the breadboard.

An important factor to consider for the algorithm is how it is integrated with the GUI. We have to look at the inputs and outputs between the two systems and scrutinize the results and the performance. Integration testing ensures that the GUI and the algorithm communicate effectively with each other and the hardware. As users interact with the GUI, any changes they make to their schematic design must be reflected in the virtual breadboard representation and the corresponding LED guide. Performance testing focuses on the software's efficiency, especially when handling complex circuits that may demand more resources. However, the focus on this aspect was not too heavy because the amount of nodes was limited. So while users “could” design very large and complex breadboards, they would not be translatable. Within the realm of possibility for the largest circuit a user can make, the software ran perfect and smoothly, but it would occasionally

lag with large circuits (once again not focused on too much because large circuits were not translatable anyways). The software's processing time and resource utilization are recorded for a variety of circuit complexities. The aim was to ensure that the software operates within acceptable parameters, avoiding excessive lag or strain on the system that could detract from the user experience. In our tests, they passed almost every time on runtime and lag/frame rate.

9.4 Plan for Senior Design 2

Throughout this report the reasoning, research, design, and testing of prototypes were discussed. Prototyping and testing gave way to answers and questions. Some of these answers include whether the design is achievable and provide specifics on how it can be achieved. However, it left questions that must be answered in the next coming weeks before a final project is delivered. The focus of Senior Design 2 will be to get the PCB manufactured and start testing, gather all components and complete hardware design, get the ESP32 to grab data from the Wattmeter Sensor and send this to the Raspberry Pi 4, and get the GUI to convert schematic to breadboard circuit.

One of the most important goals of SD2 is to ensure the PCB is well designed and have it manufactured. This component is integral to the project and is more complicated because it has to be sent out to be manufactured. This can be an issue in regards to shipping time. The completion of the PCB relies on the completion and solidification of the project design otherwise a new PCB will be required. Over the break more testing will be completed to ensure the design is well thought through.

The design of the breadboard used estimations on how large the components are, but once the PCB is completed the parts and breadboard can be assembled. There will require some tweaking which can take some time to edit the design and reprint. Printing the layers can take at least a few hours so this will need to be done efficiently. Luckily each layer is printed separately to ensure no layer needs to be reprinted if found it works properly.

The ESP32 during testing focused on communication with the LEDs. During the break and getting into SD2 the ESP32 will need to grab four pieces of data from the Wattmeter Sensor using I2C communication. The sensor itself comes with a comprehensive document that guides the user on how to calibrate the device and interact with its specific registers. The ESP32 will also be worked on to communicate with the Raspberry Pi 4 using the UART communication that both are able to do. The ESP32 will be placed on the PCB, but until then a development board can be used in place.

The GUI is the final part that will require attention during SD2. Specifically the algorithm that converts schematics to a circuit on a breadboard will need to be created and tested. During the break a preliminary list of rules will be created for the program to consider when creating the design. The GUI will be designed to reference this program when dictated by the user. This part will also be worked on during SD2 since the user will require an interface with buttons and other features. These will be created on the Raspberry Pi 4 using Python which has plenty of libraries for GUI implementation.

Each member of the group has specific objectives to ensure the overall goal of the project can be met. With the current design, there are no other components besides the PCB that need to be ordered. All components have been tested to create a baseline of what their nominal operations are. The components chosen also have plenty of resources to reference so that this project can be successful. The overall plan for Senior Design 2 is to implement the design based on the conclusion made during prototyping and testing.

10. Administrative Content

10.1 Work Distribution

The work for the provided research will be broken down into five different sections: software algorithm, graphical user-interface, microprocessor, breadboard design, power supply, circuit testing, and PCB design. The software algorithm will include the algorithm that converts a circuit drawn as a schematic to its placement on a breadboard. The design and implementation will be completed by the two Computer Engineers on the team with one taking a lead on it. The GUI will require communication to the LEDs on the breadboard and that will have the focus of one of the Computer Engineers. The use of the microprocessor and how it will interact with the GUI will be the responsibility of one of the Computer Engineers. The breadboard design will similarly go to one of the Computer Engineers with input from the Electrical Engineer to ensure all parts are considered for the design. The power supply and circuit testing will be considered by the Electrical Engineer. Finally, the PCB Design will be primarily the work of the Electrical Engineer but in coordination with both Computer Engineers.

All parts will contain a primary and secondary that are overseeing each section. This is to ensure that each section is considered by another engineer that can provide another perspective and can ensure coordination with their respective parts to ensure a cohesive design. Secondaries will be in charge of assisting the primary in research and testing, but will also be involved in checking for mistakes or edge cases. All teammates will be involved in decision making for each section regardless of role but it will be integral that the primary provides the details necessary to making the correct decisions on part selection, methodology, and testing.

Work	Primary	Secondary
Schematic to Breadboard algorithm	Cristian Gutierrez	Sheridan Sloan
GUI to Breadboard Communication	Sheridan Sloan	Cristian Gutierrez
microprocessor	Cristian Gutierrez	Sheridan Sloan
Breadboard Design/Build	Sheridan Sloan	Ammar Mubarez

Power Supply	Ammar Mubarez	Sheridan Sloan
Open/Short Circuit Detection Implementation	Ammar Mubarez	Sheridan Sloan
PCB Design	Ammar Mubarez	Sheridan Sloan & Cristian Gutierrez

10.2 Budget Estimates

The project was designed in a way that allowed the team to financially support it since no sponsors are involved. It is expected to find the majority of the parts fairly cheap as long as lead times remain reasonable. These estimations do not include campus resources and personal resources because inventory has not been taken by the team as of yet. These resources include the Senior Design Lab and the Robotics Club which may have some parts available at cheaper rates. Additionally the team has personal access to 3D printing that can reduce this cost as many parts can be designed and printed. Electronic components like LEDs will depend on the decision of whether or not a microprocessor will be necessary or if it will be incorporated into the pcb. All three members will equally commit to purchasing the following items as needed:

Part	**# Units**	**Estimated Price Per Unit**	**Total**	**Estimated Place of Order**
PLA Plastic	2 rolls	$12	$25	Amazon
Raspberry Pi	1	$70	$70	Amazon
LEDs	10	$.30	$3	Amazon
Custom PCB	1	$80	$80	Texas Instruments
Variable Power Supply	1	$34.99	$34.99	Digikey
Duracell AAA Batteries	3	$2.065	$6.195	Walmart

Voltage and Current Sensor	1	$7	$7	Texas Instruments
ESP32	1	$8	$8	DigiKey

10.3 Milestones

This section will outline the milestones we plan to achieve as we are working through this Senior Design project. The Senior Design I milestones represent the report and prototype. These milestones are assigned to the corresponding group member and have an end date to signify the completion of these milestones. On the other hand, the Senior Design II milestone represents the finishing product and our final milestone to completing this project.

10.3.1 Senior Design I Milestones

Number	Milestone Task	Assigned To	Start Date	End Date	Status
1	Initial Team Meeting	Group 20	8/21/2023	8/23/2023	Completed
2	Project Clarification	Group 20	8/24/2023	8/28/2023	Completed
3	10 Page DC Initial	Group 20	9/1/2023	9/15/2023	Completed

4	Short Circuit Detection Research	Ammar	9/11/2023	10/31/2023	Completed
5	LED Integration and Wiring	Ammar & Sheridan	10/1/2023	10/31/2023	Completed
6	Software Design	Cristian	9/11/2023	10/31/2023	Completed
7	60 Page Draft	Group 20	10/20/2023	11/3/2023	Completed
8	Update Website	Group 20	9/15/2023	1/1/2023	Completed
9	Prototype Testing	Group 20	11/21/2023	11/30/2023	Completed
10	90 Page Report	Group 20	11/25/2023	12/5/2023	Completed

10.3.2 Senior Design 2 Milestones

Number	**Milestone Task**	**Assigned To**	**Start Date**	**End Date**	**Status**

1	Review Senior Design 1 Progress	Group 20	1/8/2024	1/12/2024	Pending
2	Complete Project Design/Testing	Group 20	1/15/2024	3/20/2024	Pending
3	Prepare Presentation	Group 20	3/21/2024	4/1/2024	Pending
5	Final Presentation	Group 20	4/1/2024	TBD	Pending

11. Conclusion

The Advanced Breadboard is a custom breadboard designed and built to provide more features for its user. The goal of the project is to create an environment where a new user can learn how to use a breadboard and avoid common mistakes. These mistakes include creating an open or short circuit without knowing, and not transferring schematic to breadboard properly. This new board will use different components together to create a system that helps users to avoid these large mistakes. The work of the project will be split into three distinct groups with heavy overlap to ensure compatibility. One of the Computer Engineers will focus on the algorithm design and GUI. The Electrical Engineer will handle the power supply to all of the components, the sensors, LEDs, and PCB design. The other Computer Engineer will focus on integration of hardware and software, including the program communication with the sensors, LEDs, and PCB.

The components for the program will be a Raspberry Pi 4B which will be used to house the program and the Elecrow Monitor Display. The display will connect to the pins directly on the Raspberry Pi. The GUI will be displayed to the user using these components. Since the Raspberry Pi has the Raspian OS, the users can use a mouse and keyboard to interact with it. The display on the Raspberry Pi uses the majority of the available GPIO pins, so a microcontroller will be used as a controller for the peripherals. This microcontroller is the ESP32 because it has the required capabilities to handle the sensors with an easy IDE.

The ESP32 will communicate with the I2C Digital Wattmeter Sensor to determine if there is an open or short circuit created by the user. This sensor uses I2C communication to give four different useful pieces of data including the voltage and current. The ESP32 also has to communicate with the LEDs to activate and deactivate them as required by the main program. This can be done by connecting the LEDs to the GPIOs and setting them as output and setting them to HIGH. However, all of these parts need to be powered so

there will be a separation between the power supply for the user's circuit and the components for the breadboard. The components will be powered by three AAA batteries placed in a battery pack. Since the output can vary a voltage regulator will be used to ensure the components are receiving a steady input.

A regular breadboard cannot incorporate all of these components so a custom one will be designed. It will be created on Solidworks and 3D printed using ABS Plastic. The breadboard will be separated into three different layers which can be reorganized or separated since a bolt and nut will hold them in place. The bottom layer will house the PCB to keep away from the user or dangerous from the environment. It will have slits to reduce overheating issues. All of the layers will contain a rectangular cutout to allow for wires to go to each layer. The next layer will have the metal contacts and the LEDs. The metal contacts are being sourced from pre-built breadboards because they are operational. The top layer will cover the breadboard and will have tiny holes for the components to be pushed through like any other breadboard. There will be pillars that will hold the Raspberry Pi 4 at an angle so it is easier for the user to see the screen. The Pi will be encompassed to ensure that any part not needed for the user is covered and protected.
The prototype of the project will be separated into different parts to simulate the project as a whole. For hardware prototyping, the following components will be prepared for testing: ESP32, Wattmeter Sensor, and LEDs. The ESP32 does not require much hardware setup except for the microUSB to USB cabling. Since this device has pins it can be directly placed on a breadboard. The Wattmeter Sensor needs a flathead screwdriver to open the port for wired connections on one end. The other end can be connected to the breadboard using jumper cables. The LEDs have two pins that can be placed directly onto the breadboard. For the software prototyping, the ESP32 requires the Arduino IDE be downloaded. Since the ESP32 board is not in the basic download, it will need to be added to get access to all of its libraries. The Wattmeter Sensor is compatible with the ESP32 and has its own library that can be downloaded and used. For the GUI, Geany Programmer's Editor was set up in order to create and edit the program using Python. The Elecrow Display required specific drivers to be installed along with configurations for the purpose of this project. For the algorithm of the program, a set of rules will be created so that the program can turn a schematic into a diagram of a circuit on a breadboard. These rules will be integral to the rest of the project working.

After prototyping was completed, testing could be done to see if the products work as intended separately and together. From a hardware standpoint, the components are tested to see what their input and output voltages are to ensure each component is receiving their nominal values. For the batteries, they had an expected output voltage of around 4.5 V which corresponds to their data sheet. For the Wattmeter Sensor, it was connected to a circuit to light up an LED which it was successful in doing. This shows that the hardware of the device is working properly while also showing the LEDs are working. The nominal current for the LEDs is 20mA which the ESP32 outputs from its GPIOs according to the tests. The software testing focused on the communication ability of each device including pin locating and planning. The Elecrow Display was able to connect to the Raspberry Pi 4 Pins successfully while using a wired mouse and keyboard. The ESP32 was able to activate the LEDs using a program and a breadboard set up. The pins for the UART and

I2C were identified and set aside to ensure no other device needs those pins. Further testing on the communication protocols will be required. Each component was tested separately and in conjunction with another component to ensure integration is possible for this project.

Based on the tests completed it is concluded that this project will continue as designed. The three different parts of this section were tested to see their compatibility with each other. The battery pack with voltage regulator can supply the needed voltage to the different parts. The ESP32 can work to communicate with the peripherals for the Raspberry Pi 4. The Elecrow Display can connect to the Raspberry Pi 4 and with a mouse and keyboard it can be interfaced with. These tests help to validate the design and create a focus on what needs to be accomplished to finish the project successfully. There will be a great focus on communication between the Wattmeter Sensor and the ESP32, and the ESP32 with the Raspberry Pi 4. This process has been done before and with further time will be developed into a working system during Senior Design 2. This project has great potential to change the way students and professionals perform circuit designing and testing. Automation is a key part in technological advancement, and the simple breadboard is no exception.

Appendix

A1 References

[1] “Guide to solderless breadboards,” ProtoSupplies, https://protosupplies.com/guide-to-solderless-breadboards/#:~:text=The%20breadboard%20consists%20of%20a,connect%20the%205%20holes%20electrically. (accessed Oct. 1, 2023).

[2] “Technical data sheet PLA.” Farnell An Avnet Company

[3] “PETG Technical Data Sheet (TDS).” IEMAI 3D printing solutions for high performance materials

[4] “Polyethylene Terephthalate Glycol Modified (PETG).” IAPD

[5] “Technical data sheet ABS.” Farnell An Avnet Company

[6] I. Kuzmanic, I. Vujovic, M. Petkovic, and J. Soda, “Influence of 3D printing properties on relative dielectric constant in PLA and ABS materials - progress in additive manufacturing,” SpringerLink, https://link.springer.com/article/10.1007/s40964-023-00411-0#:~:text=Results%20show%20relative%20dielectric%20constant,density%20for%20various%20infill%20shapes. (accessed Oct. 1, 2023).

[7] “How to use a Breadboard,” Science Buddies, https://www.sciencebuddies.org/science-fair-projects/references/how-to-use-a-breadboard?gclid%5Cu003dCLzkrr-viLACFcyb7QodFVPMNA (accessed Oct. 1, 2023).

[8] C. Chaitanya, “What is HDMI? - all you need to know,” ElectronicsHub, https://www.electronicshub.org/what-is-hdmi/ (accessed Oct. 30, 2023).

[9] S. Thill, “How do bluetooth and wifi work?,” Jolt Magazine, https://jolt.sites.haverford.edu/columns/quirky-queries/quirky-query-how-do-bluetooth-and-wifi-work/ (accessed Oct. 1, 2023).

[10] M. Li, “Complete Guide on Bluetooth Module,” MOKO Blue, https://www.mokoblue.com/complete-guide-on-bluetooth-module/ (accessed Oct. 1, 2023).

[11] “Wi-Fi Module,” ERC Handbook, https://erc-bpgc.github.io/handbook/electronics/Modules/wifi_module/#:~:text=Internet%20of%20Thnigs.-,ESP8266,make%20simple%20TCP%2FIP%20connections. (accessed Oct. 1, 2023).

[12] “Temperature-controlled mini-series breadboard,” Thorlabs, Inc. - Your Source for Fiber Optics, Laser Diodes, Optical Instrumentation and Polarization Measurement & Control, https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=12149 (accessed Oct. 31, 2023).

[13] Raspberry Pi, “Rasperberry Pi Zero W,” Raspberry Pi, https://www.raspberrypi.com/products/raspberry-pi-zero-w/ (accessed Oct. 15, 2023).

[14] Raspberry Pi, “Raspberry Pi 4B3 Tech Specs,” Raspberry Pi, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (accessed Oct. 20, 2023).

[15] ESPRESSIF, “ESP32-C3-DevKitM-1,” ESPRESSIF, https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/hw-reference/esp32c3/user-guide-devkitm-1.html (accessed Oct. 20, 2023).

[16] [1] S. Cope, “Beginners Guide to the MQTT protocol,” Steve’s Internet Guide, http://www.steves-internet-guide.com/mqtt/ (accessed Oct. 21, 2023).

[17] Texas Instruments, “INA233AIDGST,” Texas Instruments, https://www.ti.com/product/INA233/part-details/INA233AIDGST (accessed Oct. 27, 2023).

[18] Texas Instruments, “INA260AIPW,” Texas Instruments, https://www.ti.com/product/INA260/part-details/INA260AIPW (accessed Nov. 3, 2023).

[19] Texas Instrument, “INA219AID,” Texas Instruments, https://www.ti.com/product/INA219/part-details/INA219AID (accessed Oct. 27, 2023).

[20] Kingbright, “WP7113SURDK T-1 3/4 (5mm) solid state lamp - kingbright USA,” Kingbright, https://www.kingbrightusa.com/images/catalog/SPEC/WP7113SURDK.pdf (accessed Oct. 29, 2023).

[21] LITE-ON Technology Corp., “LTST-C171KRKT product data sheet SMD LED - lite-on,” Optoelectronics, https://optoelectronics.liteon.com/upload/download/DS22-2000-109/LTST-C171KRKT.pdf (accessed Oct. 22, 2023).

[22] LITE-ON Technology Corp., “LTST-C191KRKT product data sheet SMD LED - lite-on,” optoelectronics, https://optoelectronics.liteon.com/upload/download/DS22-2000-223/LTST-C191KRKT.PDF (accessed Oct. 26, 2023).

[23] Globtek, Inc., “IEC62133 Certified Batteries,” GlobTek, Inc., https://ru.globtek.com/battery-packs/rechargeable-and-primary-battery-pack-assemblies (accessed Oct. 27, 2023).

[24] Dantona, “PART#: CUSTOM-145-18,” Dantona, https://dantona.com/custom-145-18 (accessed Oct. 27, 2023).

[25] Duracell, “AAA - duracell batteries,” Duracell, https://www.duracell.com/en-us/products/aaa/ (accessed Oct. 27, 2023).

[26] Matrix, “DC power supply MPs-3206 Series,” MATRIX, https://www.szmatrix.com/products/dc-power-supply-mps-3206-series (accessed Oct. 25, 2023).

[27] Korad , “KD series,” Korad, https://www.koradtechnology.com/product/84.html (accessed Oct. 27, 2023).

[28] Elenco, “Variable Voltage Power Supplies Kit,” Elenco, https://shop.elenco.com/consumers/variable-voltage-power-supplies-kit.html (accessed Oct. 26, 2023).

[29]"IEEE Standard Software Quality Assurance Plans," in ANSI/ IEEE Std 730-1984 (Revision of ANSI/ IEEE Std 730-1981) , vol., no., pp.1-12, 9 Aug. 1984, doi: 10.1109/IEEESTD.1984.7435198.

[30]"IEEE Standard for Software and System Test Documentation - Redline," in IEEE Std 829-2008 (Revision of IEEE Std 829-1998) - Redline , vol., no., pp.1-161, 18 July 2008.

[31]"IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

[32]"IEEE Standard for Information Technology--Systems Design--Software Design Descriptions," in IEEE STD 1016-2009 , vol., no., pp.1-35, 20 July 2009, doi: 10.1109/IEEESTD.2009.5167255

[33] “Duracell® Alkaline Batteries vs. Zinc Carbon Batteries,” Duracell vs Carbon Zinc, https://www.rjschinner.com/blog/literature/ProcellDuracell/2015_Duracell/Duracell%20vs%20Carbon%20Zinc.pdf (accessed Nov. 1, 2023).

[34] “Alkaline batteries vs. carbon batteries: What sets them apart?,” LinkedIn, https://www.linkedin.com/pulse/alkaline-batteries-vs-carbon-what-sets-them-apart-zhongyin/ (accessed Nov. 1, 2023).

[35] A. K. Sinha, “Different types of batteries,” Electronics For You, https://www.electronicsforu.com/technology-trends/learn-electronics/different-types-of-batteries (accessed Nov. 1, 2023).

[36] TPS6120x Low Input Voltage Synchronous Boost Converter With 1.3-A Switches, https://www.ti.com/lit/ds/symlink/tl431.pdf (accessed Nov. 1, 2023).

[37] TPS82150 17-V Input 1-A Step-Down Converter MicroSiPTM Module with Integrated Inductor, https://www.ti.com/lit/ds/symlink/tps82150.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1667817710335&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftps82150 (accessed Nov. 1, 2023).

[38] TPS6120x Low Input Voltage Synchronous Boost Converter With 1.3-A Switches, https://www.ti.com/lit/ds/symlink/tl431.pdf (accessed Nov. 1, 2023).

[39] TPS61322 6.5-µA Quiescent current, 1.8-A switch current boost converter, https://www.ti.com/lit/ds/symlink/tps61322.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1695101333035 (accessed Nov. 1, 2023).

[40] “Voltage Regulator Types and Working Principles,” MPS, https://www.monolithicpower.com/en/voltage-regulator-types (accessed Nov. 1, 2023).

[41] “PRODUCT DATASHEET,” 1215, https://data.energizer.com/pdfs/1215.pdf (accessed Nov. 1, 2023).

[42] “How long do carbon zinc batteries last?,” Dry Battery | Car Battery｜Solar Battery ｜Motorcycle Battery｜Tiger Head Battery, https://www.tigerheadbattery.com/article/how-long-do-carbon-zinc-batteries-last.html (accessed Nov. 12, 2023).

[43] Battery University, “BU-802B: What does elevated self-discharge do?,” Battery University, https://batteryuniversity.com/article/bu-802b-what-does-elevated-self-discharge-do (accessed Nov. 1, 2023).

[44] R. Senior, “Ideal operating temperatures for lithium batteries,” Fortress Power, https://www.fortresspower.com/ideal-operating-temperatures-for-lithium-batteries/ (accessed Nov. 1, 2023).

[45] “Installing ESP32 board in the Arduino IDE: Step-by-step guide,” Last Minute Engineers, https://lastminuteengineers.com/esp32-arduino-ide-tutorial/ (accessed Nov. 15, 2023).

[46] “I2C Communication Protocol,” GeeksforGeeks, https://www.geeksforgeeks.org/i2c-communication-protocol/ (accessed Dec. 1, 2023).

[47] “Universal asynchronous receiver/transmitter (UART),” ESP-IDF Programming Guide, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/uart.html#:~:text=The%20ESP32%20chip%20has%203,bits%2C%20parity%20bit%2C%20etc (accessed Dec. 1, 2023).

[48] “ESP32 I2C communication: Set pins, multiple bus interfaces and peripherals,” Random Nerd Tutorials, https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/ (accessed Dec. 1, 2023).

[49] R. Ming, “What is the difference between Eagle vs. Kicad? complete guide,” RAYPCB, https://www.raypcb.com/eagle-vs-kicad/#:~:text=You%20need%20to%20select%20the,getting%20started%20with%20PCB%20designing. (accessed Dec. 1, 2023).

[50] “Easyeda vs KiCad Eda - which one is best? in [2023],” Software Radius, https://www.softwareradius.com/easyeda-vs-kicad/ (accessed Dec. 1, 2023).

A2 Datasheets

[Raspberry Pi 4B] https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

[ESP32]
https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[Elecrow Display]
https://www.elecrow.com/wiki/index.php?title=RC050_5_inch_HDMI_800_x_480_Capacitive_Touch_LCD_Display_for_Raspberry_Pi/_PC/_SONY_PS4

[LED] https://www.kingbrightusa.com/images/catalog/SPEC/WP7113SURDK.pdf

[Voltage Regulator]
https://www.ti.com/lit/ds/symlink/tps61023.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1701737621185&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftps61023

[Wattmeter Sensor]
https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/313/SEN0291_Web.pdf

A3 Copyright

[DFRobot] Re: UCF Senior Design

Oct 27, 2023, 18:23 GMT+8

Dear Ammar,

Thank you for reaching out to us regarding your senior design project at the University of Central Florida. We're glad to hear that our Gravity: I2C Digital Wattmeter has caught your interest and could be of use in your project.

We are pleased to grant you permission to use and cite the information from our website and datasheet, as well as the images posted therein. However, we kindly request that you ensure that the source is clearly stated as DFRobot.

Please do not hesitate to reach out if you have any further questions or need additional assistance.

Best Regards,
DFRobot Team

P: +86-21-61620183

W: www.DFRobot.com

A: Room 603, 2 Boyun Road, Pudong, Shanghai | 201203 P.R.China

Ammar Mubarez
Oct 27, 2023, 13:22 GMT+8

To whom this may concern,

My name is Ammar Mubarez and I am a student at University of Central Florida. My team and I are working on a Senior Desing Project that involves using a particular part. I came across this part during my research and I wish to use and cite the information on your website & datasheet as well as the images posted on the website & datasheet. We are currently writing a Senior Design report and I am asking for your permission to use the content as a reference for our report.

The particular part I am referring to is: Gravity: I2C Digital Wattmeter

Found on the website:https://www.dfrobot.com/product-1827.html?search=SEN0291

I hope to hear from you soon.