

C.A.P.E.R.

Compact Animated Parrot with Enhanced Responsiveness

Group 12

Authors:

Kellen Danielsen	Paco-Jaleel Balthazar	Sarah Tse	William Genevrino
<i>Electrical Engineering</i>	<i>Computer Engineering</i>	<i>Electrical Engineering</i>	<i>Electrical Engineering</i>

Reviewer Committee:

Dr. Piotr Kulik	<i>Assistant Professor, Ph.D., Electrical Engineering</i>
Dr. Vikram J. Kapoor	<i>Courtesy Professor, Ph.D., Microelectronics/Nanoelectronics</i>
Dr. Matthew Gerber	<i>Associate Professor, Ph.D., Computer Science</i>

Mentor:

Dr. Chung Yong Chan



Table of Contents

1.0: Executive Summary.....	1
2.0: Project Description.....	2
2.1 Background: Robotics for Entertainment.....	2
2.2 Current Projects: Small Scale Open Source Animatronics.....	3
2.2.1 Spazzi: A Solenoid Powered Dancebot (BeatBots).....	3
2.2.2 Electric Parrot Project at MIT Media Lab (Massachusetts Institute of Technology).....	4
2.3 Motivation.....	4
2.4 Objectives & Goals.....	4
2.4.1 Short-Term Objectives Overview.....	5
2.4.2 What will CAPER do?.....	5
2.4.3 The CAPER Parrot Figure.....	5
2.4.4 The CAPER Input/Output Control Board (IOCB).....	5
2.4.5 The CAPER Voice Response Board (VRB).....	6
2.4.6 CAPER Operation Modes.....	6
2.4.7 Stretch Goals & Objectives.....	7
2.5 Constraints & Required Specifications.....	7
2.5.1 Constraints.....	7
2.5.2 Required Specifications.....	7
2.5.3 House of Quality.....	9
2.6 Physical Design.....	9
2.6.1 The Main Body (MB).....	10
2.6.2 Auxiliary Control Unit (ACU).....	11
2.7 Diagrams.....	11
2.8 Control Hardware.....	14
2.8.1 The Input/Output-and-Control Board (IOCB).....	15
2.8.2 The Voice Response Board (VRB).....	16
2.9 Software.....	17
2.9.1 Unified Conversation Mode (UCM).....	17
2.9.2 Conversation Module Overview:.....	17
Chapter 3.0: Project Research and Part Selection.....	18
3.1 Robotic Figure.....	18
3.1.1 Internal Skeleton.....	18
3.1.2 Outer Shell.....	20
3.2 Movement Characteristics.....	24
3.2.1 Motion Systems.....	24
3.3 Standing Hardware.....	28
3.4 Electrical Hardware.....	30
3.4.1 Input Output Control Board.....	30

3.4.2 Voice Response System Computation.....	36
3.4.3 Board Interfacing (cables vs wireless).....	39
3.5 Power Systems.....	39
3.5.1 Data Stream Voltage Control Hardware (Activating Circuit).....	40
3.5.2 VOX Circuit.....	41
3.5.3 Microphone, Amplifier, and Speaker.....	43
3.5.4 Microphone, Amplifier, and Speaker Selection.....	44
3.5.5 Electrical Wiring.....	47
3.6 Power Supply.....	47
3.6.1 Power Supply Selection.....	47
3.7 Communications.....	49
3.7.1 Communication Protocol.....	49
3.8 AI Technology.....	54
3.8.1 Deep Learning Framework.....	54
3.8.2 Model Architectures and Selection.....	55
3.9 Storage.....	62
Chapter 4.0: Design Constraints and Standards.....	64
4.1 Standards.....	64
4.1.1 IPC-2221: The Standard for Printed Circuit Board Design.....	65
4.1.2 ISO/IEC 30122-2:2017 – Information Technology, User Interfaces, & Voice Commands.....	66
4.1.3 IPC J-STD-001 Standard Soldering Requirements.....	66
4.1.4 ISO/IEC 42001 – Information Technology, Artificial Intelligence, & Management System.....	67
4.1.5 IEEE 801.11 – WIFI Standards.....	68
4.1.6 Table of Summary of Standards.....	69
4.2 Constraints.....	70
4.2.1 Economic Constraints.....	71
4.2.2 Time Constraints.....	72
4.2.3 Manufacturing Constraints.....	73
4.2.4 Durability.....	74
4.2.5 Input modes.....	74
4.2.6 Environmental Constraints & Sustainability.....	75
4.2.7 Social Constraints.....	75
4.2.8 Political & Ethical Constraints.....	75
4.2.9 Health & Safety Constraints.....	76
Chapter 5.0: Comparison of ChatGPT with other Similar Platforms.....	76
5.1 Benefits of ChatGPT.....	76
5.2 Downsides of ChatGPT.....	77
5.3 ChatGPT within our project.....	78
5.4 Example of Poor Performance: Inconsistent Writing.....	78

5.5 Example of Harmful Performance: Incorrect Information.....	78
5.6 Example of Good Performance: Formatting and Grammar Checking.....	79
Chapter 6.0: Hardware and Physical Design.....	79
6.1 CAPER Main Body.....	79
6.2 IOCB Circuitry.....	82
6.2.1 MCU Chip.....	82
6.3 Debugging Circuit Layout.....	85
6.4 VOX Circuit Layout.....	86
6.5 MIDI Optocoupler Layout.....	89
6.6 Voltage Regulator Layouts.....	90
6.7 Chassis Layout.....	92
Chapter 7.0: Software Design.....	95
7.1 IOCB Software Design.....	95
7.2 The Unified Conversation Module (UCM) Design.....	101
Chapter 8.0: PCB.....	107
8.1 PCB software.....	107
8.2 PCB planning.....	108
8.3 PCB Schematic.....	108
8.4 Initial Construction of the IOCB Prototype.....	110
Chapter 9.0: System Testing.....	111
9.1 IOCB Testing.....	111
9.2 The Next Steps.....	119
Chapter 10.0: Administrative Content.....	120
10.1 Finances.....	120
10.2 Bill of Materials.....	121
10.3 Distribution of Worktable.....	123
10.4 Project Milestones.....	124
Chapter 11.0: Conclusion.....	127

List of Figures and Tables

Figures

Figure 2.1: Design prototype	10
Figure 2.2: Main Block Diagram	11
Figure 2.3: Input Control Board Block Diagram	12
Figure 2.4: Output Control Board Block Diagram	12
Figure 2.5: Microcontroller Software Logic	13
Figure 2.6: Sensor and Conversational Module Data Pipeline	14
Figure 3.1: Walt Disney's Abraham Lincoln animatronic, courtesy of D23.com	19
Figure 3.2: The inner framework of a "Walt Disney's Enchanted Tiki Room" animatronic, courtesy of boingboing.net	22
Figures 3.3 & 3.4: Pneumatic cylinders(left) and solenoid valves (right), courtesy of frightprops.com	26
Figure 3.5: Power Supply Flow Chart	48
Figure 3.6: CAPER's conversational data pipeline	55
Figure 4.1 Hierarchy of IPC Design Specifications	65
Figure 4.2: Soldering Standards	67
Figures 6.0, 6.1 & 6.2: The elements of the Main Body	79
Figures 6.3 & 6.4: The metal tail link (left), and wooden mouth link (right)	80
Figure 6.5: CAPER's Main Body with all four movements activated	80
Figures 6.6 & 6.7: Resting wing position (left), activated wing position (right).	81
Figure 6.8: Complete Schematic of MCU and Immediate Connections	82
Figure 6.9: Schematic of Reset Circuit for MCU	84
Figure 6.10: Basic data flow diagram through FT232RL	84
Figure 6.11: Schematic of FT232FL Debugging Circuit	85
Figure 6.12: Old VOX Circuit Layout, courtesy of electroschematics.com	86
Figure 6.13: Modern op-amp VOX circuit, courtesy of freecircuitdiagram.com	86
Figure 6.14: Audio path through VOX circuit	87
Figure 6.15: Schematic of VOX Circuit	87
Figure 6.16: MIDI Optocoupler Data Path	89
Figure 6.17: Schematic of UART Optocoupler Circuit	89
Figure 6.18: Schematic of the +5V Regulator Using the 7805	90
Figure 6.19: Schematic of the -5V Regulator Using the 7905	91
Figure 6.20: Schematic of the 3.3V Regulator Using the LM1117	91
Figure 6.21: Initial Design for CAPER Boards' Chassis	92
Figure 6.22: The back left corner of the CAPER Boards' Chassis	92
Figure 6.23: CAPER Board Chassis Top View with general circuit location	93
Figure 7.1: IOCB Interrupt Hierarchy	95
Figure 7.2: Process within the Port 1 Function	96
Figure 7.3: Process within the Port 2 Function	97

Figure 7.4: Process within the UART (MIDI) Function	98
Figure 7.5: Publisher-Subscriber System Plan	101
Figure 7.6: Pin Monitor Flow Chart	102
Figure 8.1: Full IOCB Schematic	108
Figure 8.2: The complete prototype of the IOCB	109
Figure 9.0: LEDs switched on	111
Figure 9.1: LEDs switched off	111
Figure 9.2: The UART optocoupler for MIDI on a breadboard	112
Figures 9.3 and 9.4: The optocoupler output swinging high with no voltage at the input (left), and the output swinging low with 5V at the input (right)	112
Figure 9.5 Fairchild H11L1 datasheet excerpt. Full datasheet page in Appendix C	113
Figure 9.6: The 1995 Korg Trinity keyboard used to test the MIDI	113
Figures 9.7 and 9.8: Oscilloscope Readings after the first VOX circuit amplification stage	115
Figures 9.9 and 9.10: Readings from the second VOX circuit stage (the comparator)	115
Figures 9.11 and 9.12: The signal after exiting the rectifier and capacitor	116
Figure 9.13: The LED being triggered by the VOX circuit	117
Figure 9.14: The completed VOX circuit on a breadboard	117
Figure 9.15: Finished Main Body frame and solenoids, next to early concept art	118

Tables

Table 2.1: Specifications	8
Table 2.2: House of Quality	9
Table 3.1: Skeleton materials	19
Table 3.2: Skeleton Material Vendor Comparison	20
Table 3.3: Outer Shell materials	21
Table 3.4: Outer Shell Vendor Comparison	22
Table 3.5: Motion systems	24
Table 3.6: Motion Systems Part Comparison	27
Table 3.7: Chip specifications	30
Table 3.8: IOCB MCU Chip Comparison	32
Table 3.9: Debugging Circuit Comparison	35
Table 3.10: Hardware vs Server Offloading	38
Table 3.11: Voice Board comparison	38
Table 3.12: Transistors vs Relays	40
Table 3.13: Data Voltage Control Hardware Selection	40
Table 3.14: Op-amp Comparison	41
Table 3.15: Microphone Circuit Style Comparison	43
Table 3.16: Microphone Type Comparison	45
Table 3.17: Power Supply Comparison	48
Table 3.18: Communication Protocol	49

Table 3.19: MIDI Program Comparison	52
Table 3.20: Features comparison	55
Table 3.21: ASR vs LLM/SLM vs TTS	57
Table 3.22: tradeoff	59
Table 3.23: Model comparison	60
Table 3.24 Storage Device Comparison	63
Table 4.1 : Summary of Standards	69
Table 6.1: Pin Connections on MCU	82
Table 6.2: Capacitor Values	90
Table 7.0: Note & Movement Hex Values	99
Table 7.1: Complete MIDI Messages and Corresponding Movements	99
Table 7.2: Edge Sub-Module Attributes	100
Table 7.3: Topic Definitions	101
Table 7.4: Edge Sub-Modules Libraries Table	104
Table 7.5: Server Libraries Table	105
Table 10.1: Bill of Materials	120
Table 10.2: Distribution of Worktable	122
Table: 10.3: Senior Design I Project Milestones	124
Table 10.4: Senior Design II Project Milestones	125

Chapter 1.0: Executive Summary

When looking at the oncoming surge of new technologies like AI, and the fading out of others like animatronics; one may not expect to see any overlap between the two. Audio animatronics were once a thriving industry; a clever marriage between engineering and art, to create something sensational, and enjoyable by everyone. The peak of this technology occurred and died many years ago, and many newer technologies have developed since. The newest concept of the horizon is AI; the newest form of machine learning and computerized decision making that paints the illusion of sentient perception. Artificial Intelligence is viewed in many different ways by the public; extremely beneficial to our everyday life, a cheap job replacement for less complex work, or the end of mankind- none of which come without huge implications. When working with something so large and dynamic, it's easy for the public to grow suspicious and fear the changes. That's why we chose our project, known as CAPER (Compact Animated Parrot with Enhanced Responsiveness). So we can display Artificial Intelligence in a more lifelike and recreational manner by fusing it with a technology we all know and love; animatronics. By representing this new technology as something family-friendly, realistic, and for the purpose of entertainment; we can show the uses of AI in a non-harmful and more acceptable form. Simultaneously, we can revitalize what was once a booming industry, and give new life to a relic of the past.

Projects surrounding the integration of Artificial Intelligence into the animatronic world are more in demand than one would assume. In close proximity to where our group is based, both Disney and Universal Studios are currently in the works of large scale animatronic adaptations for their parks, attempting to create conversation-equipped AI to interact with park-goers. Other instances of Artificially intelligent animatronics can be seen in Disney's development of their Stuntronics program, creating robotic stunt doubles capable of performing aerobatics using electronic gyroscopes and laser vision (Sands, 2018).

What we are specifically interested in creating is a lifelike parrot with a base and realistic body, that has movement and response capabilities. Specifically, we are interested in giving it four fundamental movements (Beak, Body, Head, and Wings/Tail), while operating with an 80% accuracy rating in regards to response generation. To achieve this, we will design boards to monitor and produce the movements inside the bird, while actively listening to the user and generating artificial speech. These boards will serve as the neural center of the robotic bird, orchestrating its movements and interactions with users in real time. For our project, a focus has been placed on realism and fluidity. This project is entirely funded by our group members, so cost is also of the essence, but not enough to sacrifice efficiency.

In the following document, we will delve into the goals for the design of CAPER and then the intended requirements for our project to meet. We will then delve into the technical investigation and research our team did to learn about the technologies we plan to implement and the parts we will be constructing the device with. Then, after studying and confirming our project is in line with common industry standards and constraints relative to the parts we are using, we will go into depth on our design

process in both the hardware and software portion of the project. Then, we will detail all the schematics and block diagrams used in the design process when constructing CAPER. Finally, after detailing our plans for the testing of components and construction of the final product, we will include the budgeting and project marker information of CAPER.

Chapter 2.0: Project Description

Here, we'll briefly discuss the invention and development of animatronics and their predecessors. Along with this, we'll discuss the current state of Artificial Intelligence, and explain how the CAPER project utilizes these existing technologies.

2.1 Background: Robotics for Entertainment

The early forms of mechanically animated figures arose from the innovations of European clockmakers. Utilizing the same technology seen in clocks and music boxes, these clockmakers would design complex mechanical imitations of living creatures. These figures called “automatons” could carry out complex movements with repeatability and accuracy; performing such tasks as playing a musical instrument, writing, or drawing. Prime examples include the Prague Astronomical Clock, constructed in 1410 (Şengelen, 2022), or David Roentgen’s dulcimer-playing Marie Antionette automaton (Stein, 2016). By the 1700s, it was common to find such technology in a more domestic setting, such as cuckoo clocks or music boxes.

The journey towards contemporary robots began to unfold in the early 20th century. The automatons from centuries past operated solely mechanically, using springs, gears, and a complex series of camshafts and connecting rods (MacNeal & MacNeal, 2022). This was a huge limiting factor, as movements had to be physically “programmed” by the carving of each individual cam lobe. There wasn’t any method that allowed for efficient reprogramming of the movement patterns in these soon-to-be-obsolete machines. Electricity changed all this; programming individual movements was now easier than ever before, allowing engineers to program and reprogram with far less time and effort. One of the first significant milestones occurred at the 1939 New York World's Fair when attendees were introduced to Elektro and Sparko; an electrical man and dog duo created by Westinghouse. These figures, often hailed as the first modern robotic characters, demonstrated a remarkable set of capabilities using electricity, with Elektro capable of speaking over 700 words, and even smoking a cigar (Marsh, 2023).

Moving forward 25 years, Walt Disney and his company set out to develop this concept; electronically animating lifelike figures for entertainment purposes. Initially inspired by a small bird automaton, Disney realized that he could bring his animated characters to life, and display them at his Disneyland park for all to see (Staff, 2017). One of his initial forays was the “dancing man” figure in 1951; a small rod-manipulated puppet that would dance automatically using mechanical means. From this point onward, his company invested many years in developing this topic, and it wasn’t until 1961 that it was finally revealed to the public. Disney coined a new term for this technology, calling it the “audio-animatronic”; a term he would eventually copyright and trademark. The first

official audio-animatronics debuted in Disneyland in 1963, with the opening of “Walt Disney’s Enchanted Tiki Room”. This attraction featured many animated tropical birds, plants, and tikis, all using a combination of electric signals and pneumatic actuators. One of Disney’s most impressive animatronics debuted at the 1964 World’s Fair; a full-scale figure of Abraham Lincoln. Hereafter, animatronics became increasingly common at Disney Parks; including attractions such as the “Carousel of Progress”, “Pirates of the Caribbean”, “Haunted Mansion”, and many others (*The History of Animatronics*, n.d.).

The 1970s and 1980s saw the peak of animatronic technology, with animatronics appearing outside the walls of Disneyland and in other various entertainment venues. Two of the most famous chains that used animatronics were Chuck E. Cheese's Pizza Time Theater and Showbiz Pizza Place. Both locations featured their own renditions of animatronic bands at all of their locations (*The History of Animatronics*, n.d.). However, it was around this time when the popularity of animatronics began to slowly decline. This period, known as “The Video Game Crash of 1983”, directly resulted in the bankruptcy, merging, and even demise of many non-Disney entertainment venues (Beren, 2023). Aside from entertainment venues, it is important to note the significance of the film industry, and how pivotal its role was in the life of animatronics. The 1970s witnessed groundbreaking work, such as the pneumatic shark in “Jaws” (1975) and elements of the Xenomorph costume in “Alien” (1979). Rick Baker, a luminary in cinematic animatronics, significantly advanced creature effects with films like “An American Werewolf in London” (1981). For the film “Jurassic Park” (1993), Stan Winston Studios built the largest animatronic ever created; a 9-ton, 20-foot Tyrannosaurus-Rex nicknamed “Rexy”. This record was surpassed in 2001 with the even larger Spinosaurus, weighing in at 25,000 pounds (*Stan Winston School of Character Arts*, n.d.). Conversely, the “Jurassic Park” franchise also demonstrated the power of Computer Generated Graphics, and its ability to outright replace practical effects. Upon this discovery, the use of animatronic props in film also started to decline. Of course, this didn’t mark the end just yet. In 2022, Jim Henson’s Creature Shop recently constructed all the animatronics and mascots for the “Five Nights at Freddy’s” film, inspired by the immensely popular 2014 video game (Graves, 2023).

2.2 Current Projects: Small Scale Open Source Animatronics

This section discusses the current existing animatronic projects.

2.2.1 Spazzi: A Solenoid Powered Dancebot (BeatBots)

Spazzi, featured in Make: magazine, is a dancebot driven by solenoids and controlled by an Arduino microcontroller. It uses three solenoids to achieve eight distinct positions, enabling it to dance to music or inputs when commands are varied over time via software such as Max/MSP or Pure Data. This project demonstrates how limited movement repertoires can produce complex behaviors.

2.2.2 Electric Parrot Project at MIT Media Lab (Massachusetts Institute of Technology)

The Electric Parrot Project at MIT Media Lab was an endeavor aimed at designing a robot capable of engendering empathy through its interactions and behaviors. Active from January 2015 to August 2016, the project sought to explore the boundaries between technology and emotional connection by constructing a novel zoomorphic robot.

This robot was not just about simulating the physical appearance of a parrot but was designed to create its own life story by experiencing the world, being changed by these experiences, and communicating these experiences back to humans. The goal was to demonstrate that by giving the robot an implicit life story, it could invoke empathy in human interactions, potentially being used for empathy intervention.

2.3 Motivation

Just as automatons became obsolete and antiquated, traditional animatronics are soon going to follow suit. The next generation of entertainment-oriented robots will require the newest technology for any chance to succeed. The downside to this is the high cost of production and installation of modern animatronics. It would cost companies hundreds of thousands of dollars to upgrade their facilities to accommodate these new figures, and many companies cannot foot the bill. This is our motivation for CAPER; to create a low-cost, low-profile, robotic parrot, suitable for permanent installation in facilities with a smaller economic footprint.

2.4 Objectives & Goals

CAPER integrates modern AI and movement control systems for improved human interaction, being able to carry out simple conversations with the user. Voice activation and speech detection will prompt the AI program to generate appropriate movement and audio data in real-time. The raw electrical movement impulses will be translated into motion, and synchronized with the audio; it will look like you're *actually* talking to a parrot.

If a less advanced response mode is desired, CAPER will also feature more traditional control methods; real-time manual movement control, and prerecorded movement sequencing. While mainly focusing on improved human interaction, affordability and ease of use are also critical factors. We must consider the ramifications of overcomplication and expense; preventing installation in facilities with limited space or power resources. We don't ever want this to be the case. Developing user-friendly interfaces that don't require specialized knowledge, further widens the appeal of CAPER to include non-engineer users. Simplifying the design and using less material will reduce development, installation, and maintenance costs, making it a practical option for smaller venues, schools, and hobbyists. This approach not only widens the market but also allows for educational exploration into robotics.

Finally, the CAPER project is a response to the growing need for ethical and positive applications of robotics; aimed at moving away from potential negative uses like militarization and job displacement. We strive to take the next step in entertainment-based practical effects, and once again stimulate interest in the field of animatronics, using new technology.

2.4.1 Short-Term Objectives Overview

This section explains the short-term objective of CAPER.

2.4.2 What will CAPER do?

As mentioned before, CAPER will generate physical and auditory responses based on user voice input; but how will this be done? How will our design differ from what's being done by other companies? The answer lies in the physical design of both the robotic figure and controllers. The entire system of CAPER involves three main stages. Firstly, the generation of audio and movement response based on various user inputs. Secondly, the translation of raw data from the inputs into usable high-power voltage streams to animate the parrot. Lastly, the mechanical conversion of these voltage streams into movement. Therefore, we will have three respective subsystems: a voice response circuit board, an input/output movement board, and the parrot figure itself.

All of the low-level peripheral circuits we'll need will be consolidated into one of the three main subsystems. This "partial consolidation" allows for easy installation of each main system individually, and eliminates the worry of over-complicated maintenance and troubleshooting. These three systems are explained in greater depth below:

2.4.3 The CAPER Parrot Figure

CAPER will showcase four movements: beak opening/closing, vertical head tilt, vertical body tilt, and lateral tail/wing tilt. Each movement will be carried out electrically but without the use of ultra-precise position or speed monitoring. The binary "on/off" style of movement control makes programming more timely, efficient, and understandable, without jeopardizing the "realism" of the movement. The internal frame of the robot along with all the devices and physical extremities, will be completely concealed by the exterior "skin" of the parrot. The figure will stand upright, fixed atop a pedestal for permanent mounting. Wires will run out the bottom of the figure for connection to a separate control unit.

2.4.4 The CAPER Input/Output Control Board (IOCB)

This custom-designed PCB will contain a low-level microprocessing chip and all of the necessary peripheral circuits needed to animate CAPER. It will be enclosed in its own discrete enclosure separate from the figure, allowing for remote control. Most of the control board will be hidden, with only the necessary buttons, switches, and I/O ports visible from the outside. The IOCB will feature four distinct operation modes and will decipher two classes of inputs.

2.4.5 The CAPER Voice Response Board (VRB)

This pre-built edge computing development kit will have all the needed computing power to allow CAPER to give lifelike voice and movement responses to user interaction. Movement data will be sent to the IOCB, while audio will be sent to external speakers. The classes of input are Single Line Serial Communication over UART and Parallel Digital Inputs.

- **Single Line Serial Communication over UART:** Using MIDI protocol, data can be transmitted from an external controller in the form of a UART bitstream. Received messages will be decoded and translated by the IOCB. Therefore, all four of CAPER's movements can be controlled by a single serial communication channel.
- **Parallel Digital Inputs:** The IOCB will have four digital inputs corresponding to each of the four movements.

2.4.6 CAPER Operation Modes

The operation modes CAPER will employ are fully manual live operation, partial manual live operation, and pre-recorded sequence operation.

- **Fully-Manual Live Operation:** Using parallel digital inputs, CAPER can be manually operated using push buttons. CAPER features four movements, therefore there will be four buttons. Pressing a button will activate the corresponding joint movement, and depressing the button will return that joint to its resting state. In this mode, CAPER essentially behaves like an electric ventriloquist's dummy.
- **Partial-Manual Live Operation:** Using the parallel input for the mouth actuator, the user can plug a microphone into the IOCB and control CAPER's mouth with their voice. With an adjustable gain knob, the user can tune the sensitivity of the microphone for better accuracy. The remaining three movements can still be operated with the corresponding push buttons, or be moved automatically using the Partial-Manual mode. By flipping a selector switch mounted on the enclosure of the IOCB, the user can activate a subroutine within the IOCB's program, that randomizes the head, body, wing, and tail movements, allowing for complete hands-free operation of CAPER.
- **Pre-recorded Sequence Operation:** CAPER can be operated like a traditional animatronic using an external playback device. Using any easily attainable digital audio workstation (DAW), the user can pre-program a sequence of movements on their computer using MIDI, and transmit the data stream into CAPER's UART input. Once a sequence of movements has been recorded by the user, that exact sequence can be played back repeatedly. Most DAWs allow for simultaneous playback of MIDI streams and audio streams, meaning you can synchronize CAPER's movements with the audio, and play them both at the same time. (The audio can be sent to external speakers).

- **Automatic Live Operation:** CAPER can listen and respond using a synthetic voice to users with its speaker and move in time with the voice response. Neither the movement nor the content of the voice response given is pre-programmed in this operational mode.

2.4.7 Stretch Goals & Objectives

In future versions, CAPER could see its lifelike nature increase with the inclusion of sensors to capture more of its environment without substantially increasing the cost. The output of those sensors could be used to create responses to touch, lighting, and the presence of humans, making both voice and movement feedback more dynamic and unique. Another possible upgrade could be an improvement to the control system by adding battery support, allowing CAPER to be operated in a greater variety of environments. Further still, more movement vectors are also a possible upgrade. Coupling that with the new sensors could give CAPER an almost uncanny dose of realism to its movements, allowing for use not only as entertainment but also in education. It is even possible to make the figure user customizable!

2.5 Constraints & Required Specifications

This section briefly discusses the constraints and specifications. A deeper look can be found in Chapter 4.

2.5.1 Constraints

- Time – Senior Design II takes place in the Summer semester. This is about 4 weeks less to build a working prototype than the typical Fall/Spring semesters. This project cannot have every feature we want to include due to time constraints.
- Costs – \$1300 is the maximum estimated total, though prices can vary depending on the availability of parts and distributors.
- Quality – Balance between quality versus expenses. The goal is to keep CAPER as affordable as possible while providing the highest quality outputs.
- Legality – Intellectual property rights.

2.5.2 Required Specifications

These are the specifications chosen regarding current objectives and constraints. The highlighted specifications are the ones that were selected for the focus of the demonstration.

Table 2.1: Specifications

Hardware	
Movement capability	Solenoids should be able to move the joints in a way that makes the parrot life-like with fluid motions such as head tilt, mouth movements when speaking, wing flapping, etc. Head – 45 degrees Mouth – 50 degrees Body – 30 degrees Wings & Tail– 45 degrees each
Interactive modes	The selector switch will be used when the MIDI is inactive. The position of the switch will declare the mode the parrot is in. Turning the switch on would activate randomized movements for the head, body, and tail flap. Turning the switch off would activate total manual control using button inputs.
Dimensions	Around 1.5 ft tall. Maximum 2ft
Durability	Able to last at least two years with minimal maintenance
Weight	Maximum of 15 pounds.
Power Consumption	Maximum 40W
Cost Maximum	\$1300
Response time to user input in push buttons	2 seconds
Software	
Voice Recognition	The parrot should be able to recognize when its name is called and respond accordingly with 80% accuracy.
Audio Response	Parrot should respond in a realistic manner with intelligible speech and 80% accuracy. Parrot should be easily understandable by the user.

2.5.3 House of Quality

Table 2.2: House of Quality

Relationship Matrix	
↑↑	Strong Positive
↑	Positive
↓	Negative
↓↓	Strong Negative
	Not Correlated

		Customer Importance 1-5									Competitor Research			
			input modes	Response efficiency	usability	cost	response time	dimensions	power loss	durability	Competitor 1	Competitor 2		
			+	+	+	-	+	-	-	+				
Multiple ways to communicate	3	+	↑↑		↑	↓			↓					
conversational	4	+		↑↑		↓↓	↑↑							
natural movements	4	+				↓↓	↑↑	↓						
cost	4	-		↓↓		↑↑		↓	↑	↓↓				
realistic appearance	3	+		↑		↓↓		↑		↓				
Maintenance	3	-				↑				↑↑				
Ease of Installation	2	+			↑↑									
power consumption	3	-	↑		↑		↓		↑↑					
		Importance Rating	4	5	3	4	3	3	4	4				
		Targets	3 modes	80%	2hr	\$500	4sec	20lbs	50W	>2 years				

2.6 Physical Design

CAPER's design will incorporate two distinct physical structures for its implementation.

- The Main Body (MB) will be the parrot-shaped robot that contains all the various systems needed for movement
- The Auxiliary Control Unit (ACU) will be a box that contains all computation systems and be responsible for controlling the robot's movements. It will be connected to the MB by wire.

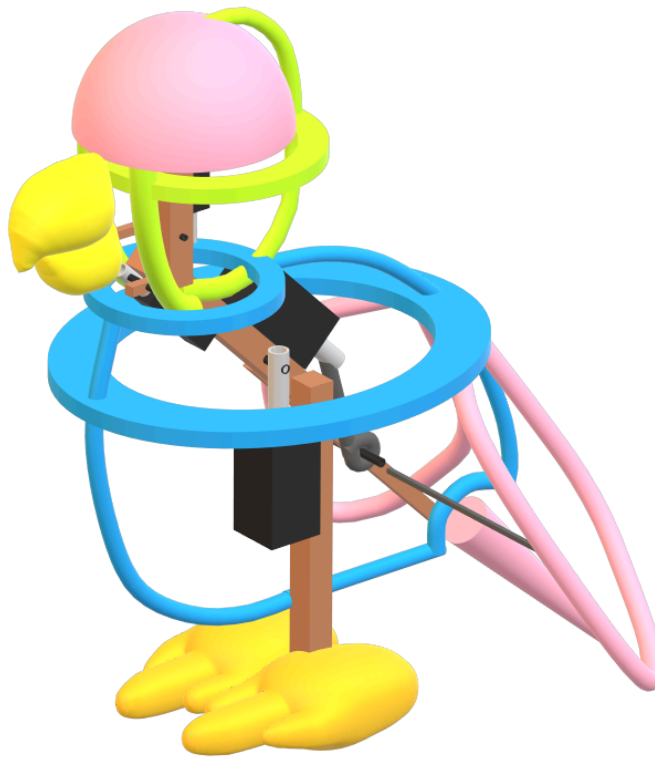


Figure 2.1: Design prototype

2.6.1 The Main Body (MB)

CAPER's cornerstone component group will be the robotic parrot apparatus that is operated by the control system, referred to as the Main Body (MB). The MB will contain all the systems needed to turn the signals from the control unit into movement.

The movements available to the MB are:

- Beak open/close
- Head tilt up/down
- Body tilt up/down
- Wing & Tail flap in/out

This central frame in the MB will support all the links, linkages, and electrical devices inside CAPER, and will resemble a 3-link, 2-revolute-joint robotic arm. The first link will extend vertically from the bottom pedestal, where the following two links will diagonally extend outward like a boom, giving CAPER the parrot-like posture. The gravitational force of the links' weight while in their resting positions will be counterbalanced by small coil springs; energizing the solenoid actuator will retract the link into its opposition position, countering the forces of the springs and gravity. Once fully assembled, the frame will be completely concealed by CAPER's skin; the outermost "parrot costume" constructed of malleable synthetic materials. Fully assembled, the figure will stand at approximately 1.5 feet (45.72cm) tall, fixed atop a pedestal for permanent mounting.

2.6.2 Auxiliary Control Unit (ACU)

The MB will not have the necessary internal spacing to accommodate for any of its control systems. The ACU will contain all PCBs needed for all of CAPER's modes, as well as its power supply.

The resulting container will need to be large enough to not only contain all devices but also accommodate for the plugging in of external devices. No mechanical stresses are expected here so the design will be made of cheaper 3D printed material.

2.7 Diagrams

Main Block Diagram

Everything within the red outline is housed in the PCB and all require an external power supply (see individual subsection diagrams)

- ❖ Blue = peripheral circuits
- ❖ Pink = external devices
- ❖ Green = microprocessor chip

****all components are to be acquired*

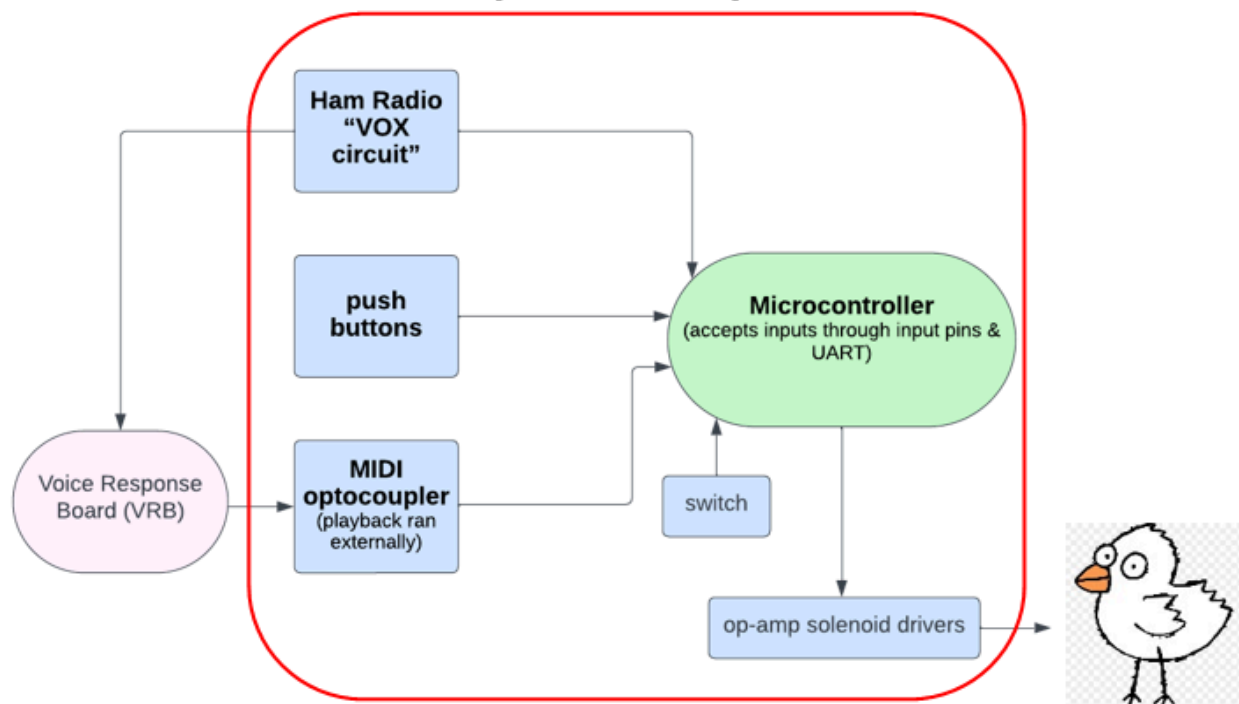


Figure 2.2: Main Block Diagram

Input Control Board Block Diagram

Blue = on PCB; requires power supply

Green = microprocessor inputs

Pink = external devices (may require separate power supply)

Blocks within red outline → VOX circuit (see main diagram)

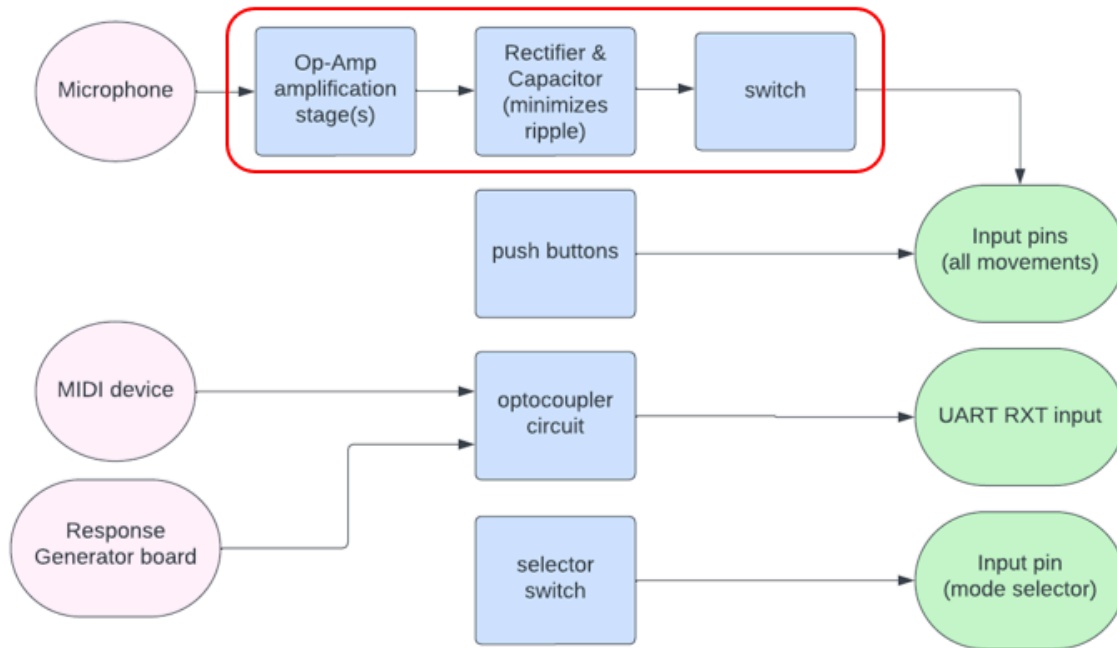


Figure 2.3: Input Control Board Block Diagram

Output Control Board Block Diagram

Blue = on PCB; requires power supply

Green = microprocessor outputs

Pink = external devices (may require separate power supply)

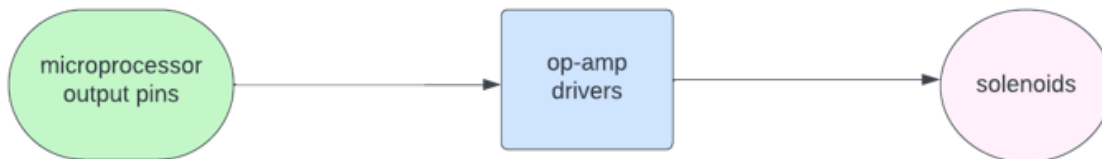


Figure 2.4: Output Control Board Block Diagram

Microcontroller Software Logic

- ❖ Blue = I/O pins as seen by the **code/program**
- ❖ Orange = hardware & external devices
- ❖ Green = pseudocode in the debugging program

****all components are to be acquired*

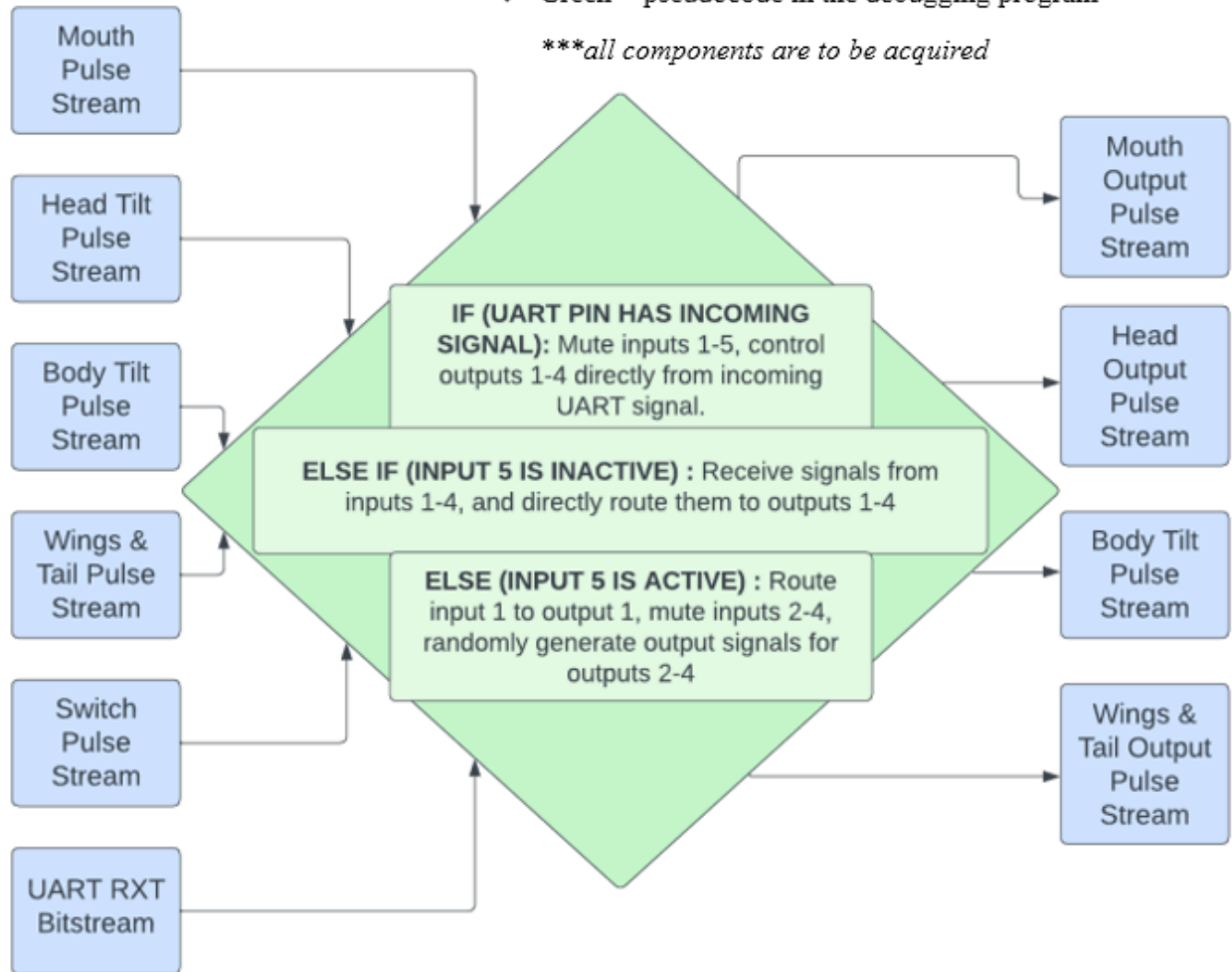


Figure 2.5: Microcontroller Software Logic

Sensor and Conversational Module Data pipeline

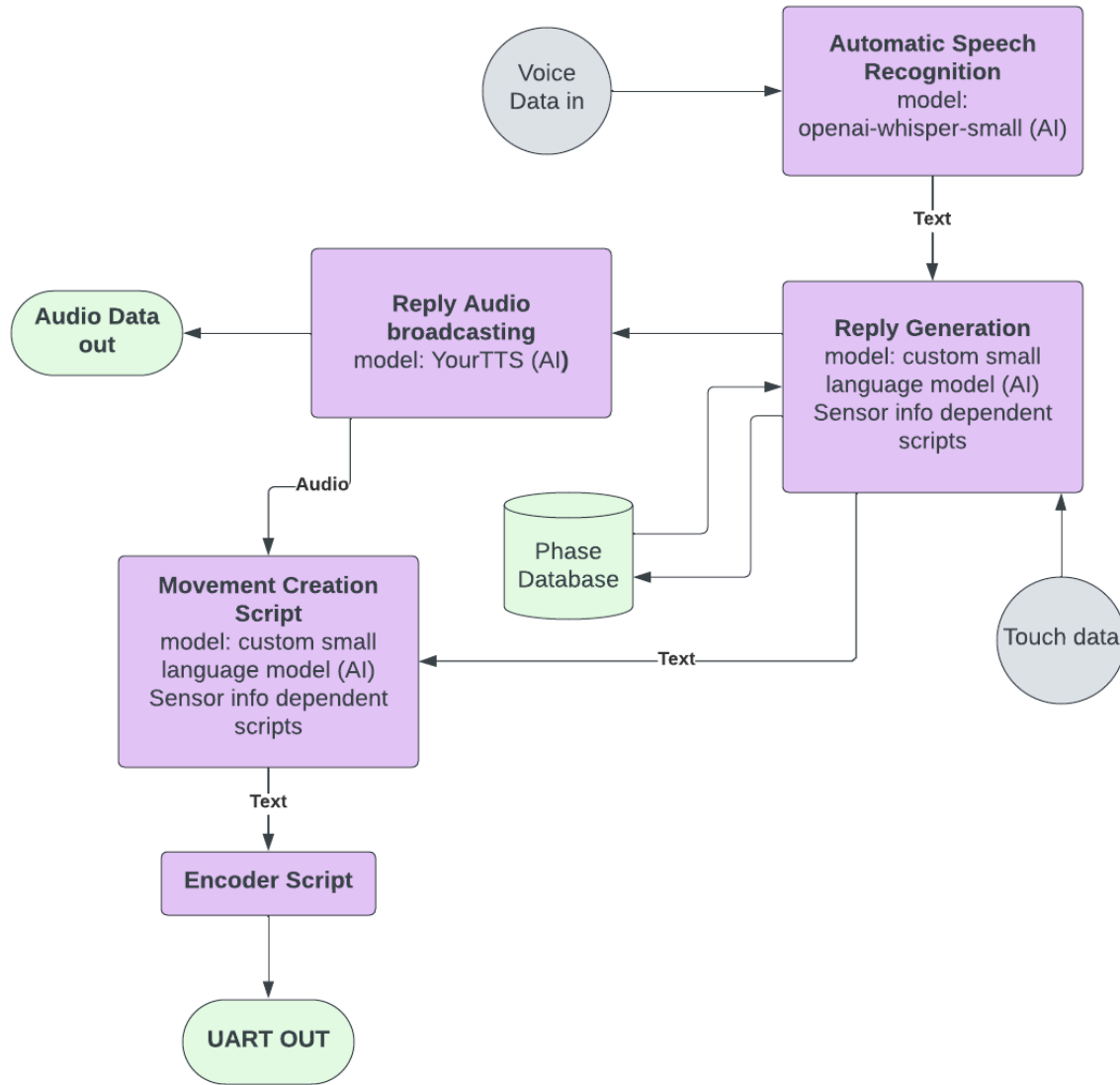


Figure 2.6: Sensor and Conversational Module Data Pipeline

2.8 Control Hardware

To achieve its comprehensive functionality, CAPER will necessitate the incorporation of multiple computational boards. The Input/Output-and-Control Board (IOCB) will be responsible for the critical operations of movement, power management/control, and input management. In contrast, the conversational capabilities, enabling CAPER to recognize spoken words and respond through its onboard speaker, will be facilitated by the Voice Response Board (VRB).

Communication across these boards is streamlined and unidirectional; UART will relay movement data to the main board for decoding, while audio data is transmitted directly to the speaker for immediate output. The entire control hardware suite will be powered by standard US wall voltage (120V@60Hz) via a cable to a standard receptacle wall outlet. Wall power will be bumped down and rectified to DC via an on-board power supply.

2.8.1 The Input/Output-and-Control Board (IOCB)

This board is to house enough I/O and computing power to drive CAPER's movements and allow it to respond to its environment in a lifelike way without compromising on response time. The IOCB will require an MCU that meets the following criteria: microprocessor, UART input stream processing, and power supply for digital pins.

1. **Microprocessor:** Feature a microprocessor strong enough for adequate data processing capabilities.
2. **UART Input Stream Processing:** Support processing of a UART input stream for voltage-stable serial communication.
3. **Power Supply for Digital Pins:** Provide power and be able to read from multiple digital pins with a voltage tolerance to accommodate digital signaling needs.

Power requirements for the boards' microelectronics cannot exceed what can be drawn from a wall outlet to ensure that CAPER can be operated from nearly anywhere. The board is to be housed in a separate case and connected with cables; this will make CAPER controllable from a distance like the animatronics of old. Basic decision logic for IOCB code in Figure 2.5

IOCB Peripheral Circuits:

Several distinct circuit groups will be housed on this board alongside the MCU, as seen in Figures 2.3, and 2.4 as the blue blocks:

- The Push Buttons: (4), Toggles all four respective movements manually
- Mode Selector Switch (1): Toggles automatic movements
- The Ham Radio "Vox" Circuit: Will generate pulses when voice is present
- The MIDI Optocoupler: An opto-isolator for voltage/ground stability for MIDI signal
- Bootstrap Loader: Allows for USB compatibility, debugging of MCU
- Power Supply/Regulators: Higher Voltage Power Supply, Low Voltage Regulators
- Actuator Relays

Push Buttons

Four physical push buttons will be connected to parallel pins on the IOCB. Pressing the button will transmit an electrical impulse and control one movement. Therefore there will be a mouth, head, body, and wing/tail button. Pressing each button will move the respective joint. Depressing the button will return the joint to its resting state.

Mode Selector Switch

A single, active-high switch, will be connected to its own pin on the IOCB. Operates similar to the push buttons; used for mode selection on the IOCB.

The Ham Radio “Vox” Circuit

A series of operational amplifiers and a bipolar junction transistor, wired to make a voice-activated switch, for hands-free mouth actuator movement and VRB voice detection. The design of this circuit will be similar to that of the Vox circuit commonly seen in ham radios. Audio from an external microphone will pass through multiple amplification stages and a half-wave rectifier. This remaining upper sideband signal can toggle a BJT switch connected to +Vcc. If working properly, the circuit will switch on whenever there's audio present.

MIDI Optocoupler

Transmitting over UART can be electrically unstable due to the separation of ground buses on the MIDI source and the IOCB. The use of an opto-isolator circuit links the electrical signals using light rather than physical connectivity, negating any voltage faults between the two grounds. Using a voltage source local IOCB, the MIDI bitstream will exit the opto-isolator as a stable digital signal, with all data bits uncorrupted.

Power Supplies and Voltage Regulators

The main voltage powering the entire assembly will be from external power supplies. Since CAPER's solenoid actuators require substantial current, they will be supplied directly from the high voltage supply. Lower voltages needed by the MCU and IOCB peripherals will be supplied from on-board low voltage regulators.

Actuator Relays

Relays will be used to receive low-power output signals from the IOCB and switch on to supply higher-power streams to the inside solenoid actuators.

2.8.2 The Voice Response Board (VRB)

The IOCB incorporates substantial computational resources but falls short of the specifications necessary for generating lifelike voice and movement responses. To address this, the Voice Response Board (VRB) will necessitate a chipset that offers superior processing speed, enhanced memory capacity, and greater I/O bandwidth.

These enhancements are critical for supporting the sophisticated systems essential for real-time conversational interactions. Key specifications for the VRB chipset include: power supply, memory, USB ports, clock speed, and data storage.

- **Power Supply for Digital Pins:** Must supply power to and enable reading from at least two digital pins. This feature is vital for interfacing with non-USB devices.
- **Memory:** A significant amount of RAM is required to facilitate the swift execution of scripts and AI-driven responses.
- **USB Ports:** At least two USB 3.0 ports are necessary for enhanced UART communication speeds.

- **Clock Speed:** Essential to support the intensive processing demands of voice interaction.
- **Data Storage:** Capability to support suitable storage space to accommodate extensive data and AI models.

The ability to produce realistic speech and interactive responses hinges on the selection of an AI model and the computational capabilities of the VRB. These outlined specifications represent the foundational requirements for the board's performance.

2.9 Software

CAPER's design requires a set of two distinct software packages. One to control the movements and hardware, and another for the robot's more advanced features. Both these systems can function independently and will be linked via serial.

2.9.1 Unified Conversation Mode (UCM)

The UCM will function as the core of CAPER's lifelike behavior, necessitating a design that prioritizes computing power and responsiveness. The software suite enabling the UCM will feature: response time, speech quality WER, response-relevant movement commands, and for a stretch goal, making CAPER have conversational speech.

1. **Response Time:** The UCM will process and respond to user voice inputs within 4 seconds, ensuring a smooth conversational flow.
2. **Speech Quality:** Responses will be clear and fully intelligible.
3. **Maximum Word Error Rate (WER):** The system aims for a maximum WER of 20% to ensure at least 80% accuracy in recognizing spoken words.
4. **Response-Relevant Movement Commands:** To add an element of expressiveness, responses will be accompanied by movement commands for CAPER, such as beak adjustments to mimic speech dynamics, adding thematic flair to interactions.
5. **Conversational (Stretch Goal):** As a stretch goal, the system will aim to generate responses that consider previous prompts, facilitating a more natural and engaging conversation.

2.9.2 Conversation Module Overview:

The UCM will require the following systems to function effectively: ASR, AI-based text completion, and TTS.

1. **Automatic Speech Recognition (ASR):** This program works to turn spoken word into text
2. **AI-based Text Completion:** This programs works to find ways to complete the input text data in a way that produces a response
3. **Text-to-Speech (TTS):** The final stage involves converting the generated text responses into spoken words.

Chapter 3.0: Project Research and Part Selection

This section goes over all of the types of technologies used within the project.

The format of this section is as follows:

1. A list of all technologies that can be reasonably used for a project of this scale.
2. For each listed technology a list of the pros and cons from using this said technology within the design.
3. The best choice out of the list along with further reasoning.

Each section and subsection will follow this pattern for ease of readability. In the case of hardware research, the charts and organization is with reference to Figure 2.2: Main Block Diagram. The subsequent diagrams further dissect what is already shown in the Main Diagram, therefore this section will cover those systems from a higher level of understanding.

3.1 Robotic Figure

CAPER's parrot-like figure will be what ultimately conveys the image of artificial life to the viewers. Aspects of physical durability and cosmetic realism are not to be overlooked when designing the body. In short, there are three main component groups to this figure: the internal frame, the outer skin, and the mounting hardware. Information regarding the automation of CAPER will be described in the following section for "Motion Systems". This section along with the sections regarding the frame, outer surface, standing hardware, and motion systems, are all describing systems noted by the little cartoon parrot figure in Figure 2.2: Main Block Diagram.

3.1.1 Internal Skeleton

In order for CAPER to carry out the 4 movements, a specialized central frame will be needed. This frame will support all of the movement hardware, as well as the outer "costume" layer. Due to an actual parrots' diagonal standing posture, our frame would have to retract to that position in its resting state, requiring at least 30 degrees of forward rotation. Likewise, the head of the parrot will remain in a vertical position, requiring a second joint with 45 degrees of rotation. When complete, the entire internal frame would need to resemble a 3-link, 2-revolute-joint robotic arm. Fixed atop the base (see Standing Hardware), the first link will extend vertically around 10 inches from the base. The second link will connect to the top of the first link using a single revolute joint and upward and forward, this link will be around 8 inches long. The final link which the head hardware will connect to will be around 4 inches long and extend vertically, canceling out the diagonal position of the second link.

Skeleton Materials

It is clear that a 3 link arm is the most suitable for application; as it allows for CAPER to carry out the 4 movements and maintain its parrot-like posture. There are many suitable materials to choose from for this application. The links would require a strong, and inflexible material; one that could support the weight of the body without bending or

breaking. The three main materials at our disposal are wood, metal, and 3D printed plastic. Each of these materials have their own weaknesses and strengths seen below:

Table 3.1: Skeleton materials

	Best	Middle	Worst
Cost	Wood	Plastic	Metal
Availability	Wood	Metal	Plastic
Durability	Metal	Wood	Plastic
Shapeability	Plastic	Wood	Metal

At first glance wood seems to be the safest bet from all the materials, it is strong, cheap and easy to work with; not requiring any overly expensive specialized tools. Metal is definitely the strongest material here, making it the most common choice in commercial applications. Figure 3.1 displays Disney's "A1" figure of President Lincoln; a great example of a metal-framed animatronic. Unfortunately, the main drawback of metal is the difficulty of shaping and constructing the pieces; it simply wouldn't be a practical choice given our budget. 3D-printed plastic is an interesting route that would give us the ability to design custom link shapes, but it requires more design time and is slower to manufacture.



Figure 3.1: Walt Disney's Abraham Lincoln animatronic, courtesy of D23.com

Skeleton Material Selection

Research concluded that wood is the most practical choice for CAPER's rigid internal frame. Of course, wood comes in many forms and varieties; all of which are readily available at any local hardware store. For CAPER, the choice was made to use $\frac{3}{4}$ inch and $\frac{1}{2}$ inch square dowels. The thicker $\frac{3}{4}$ inch dowel will be used for the 3 main supporting links, cut at 10, 8 and 4 inch lengths. The thinner $\frac{1}{2}$ inch dowel will be used for more of the animated properties including the mouth joint, the tail support, and

internal shaping when applicable. Joints between the links will be created using nuts, bolts, and washers.

The below table shows some available suppliers of wooden dowels. In the case of all three, the wood products are pressure treated, sanded, and have low defect/knot rates. None of them are pressure treated for outdoor use, but should hold up nicely with occasional exposure to the elements.

Table 3.2: Skeleton Material Vendor Comparison

	Waddell	Madison Mills	Woodpeckers
Wood Type	“Hardwood”	Poplar	Walnut
Sizes	36’ x 0.75”	36’ x 0.75”	36’ x 0.75”
Price	\$3.37	\$5.98	\$3.77

Choice: Waddell hardwood dowels

Reasoning: Waddell dowels suit the needs of the project perfectly. The actual type of wood used is described as “hardwood” which could vary from poplar, linden, or basswood depending on that store’s inventory. An alternative company, Madison Mill, sells the exact same dowels exclusively made from poplar wood. The more precise sourcing of this company doubles the price, making the Waddell wood more favorable to our project. In the case of Woodpeckers, the retail price and selection is similar to Waddell, but the product is only available through their online store; the delivery fees will more than double the price and add substantial shipping delay.

Expected Performance: The drilling of pilot holes for the revolute joints will minimize the occurrence of splits and splintering. Use of general purpose automotive grease will reduce friction and wear at these joints, and allow for more free movement when animated.

Stretch goal - 3D Printed Plastic: A possible optimization to the design would be to use a 3D printer to construct the links. With a standard cost around 20 dollars per kilogram (8 dollars a pound), this is a considerable alternative to wooden links. Pros of this method include the ability to create custom shapes without drilling or filing. However, the length of time required to design and produce the shapes is considerably more than the wood-working method. For this reason, the initial build of CAPER will use the wooden frame, with intent to eventually upgrade to a lighter plastic one.

3.1.2 Outer Shell

This flexible exterior “skin” will need to conceal all the inner workings and hardware, and give CAPER the parrot-like color and appearance. Ideally, we don’t want any seams between the different sections of CAPER, but we’ll also need the ability to easily remove and replace the skin as needed. Common materials used for animatronic skin are latex,

synthetic furs and feathers, and various rubbery plastics. Given the budget, and time constraints, other materials will also be considered including paper mache, felt/fabric, and styrofoam.

Note: We do not have to settle on any single material; CAPER will practically utilize all the available materials needed.

Table 3.3: Outer Shell materials

	Latex Rubber	Styrofoam / Foam	Paper Mache	Felt/ Fabric
Cost	\$70 a gallon	~\$3 per sq. ft.	<\$1 per sq. ft.	~\$0.50 sq.ft.
Cast Molding or Sculpting	Cast Molding	Sculpting	Cast Molding	Both (needs interior mold)
Will Have Visible Seams	No	Yes	Yes	No
Rigid or Flexible	Flexible	Moderately Rigid	Rigid and Fragile	Extremely Flexible
Requires Manual Color	Yes	Yes	Yes	No
Durable & Long Lasting	Yes	Moderately	No	Moderately
Suitable for Outdoor Use	Yes	Yes	No	Yes

At this stage, Latex Rubber can be safely ruled out, as it requires the most intensive and expensive preparation work. Paper mache may be suitable for some sections like the eyes or interior supports, but wouldn't look realistic on the outside. Materials like felt, foam and styrofoam would likely be our safest choice. Synthetic feathers, although not suitable for skin itself, would certainly work as a second layer outside the skin. Below is one of the animatronic robins from "Walt Disney's Enchanted Tiki Room" with the back panel removed. This figure's outer shell is quite rigid, being internally supported by an "airplane-like" metal cage.

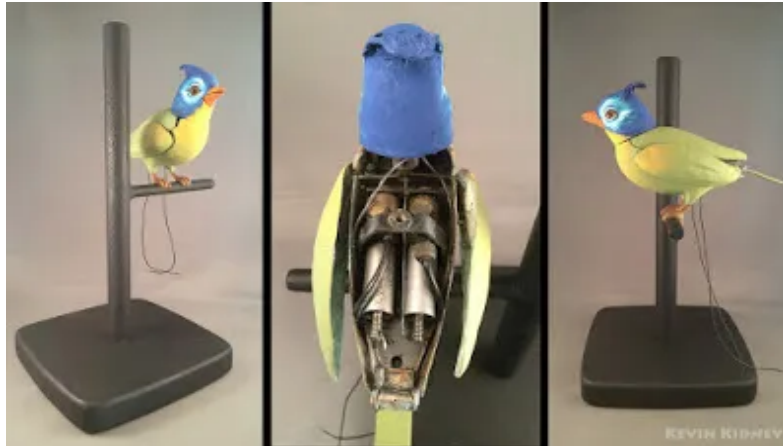


Figure 3.2: The inner framework of a “Walt Disney’s Enchanted Tiki Room” animatronic, courtesy of boingboing.net

Outer Shell Material Selection

The exterior surface of CAPER serves a dual function. This “skin” must conceal all internal components while providing a parrot-like appearance. Common materials for animatronic skins include latex, synthetic furs, feathers, and rubbery plastics. Due to budget, time, and experience constraints, our group is considering alternative materials such as paper mache, felt/fabric, and styrofoam. Our design will likely consider multiple materials when it comes to the body; one material to conceal the inner workings of the bird, and a purely aesthetic outer “skin” layer.

Our factors considered when selecting this material include ease of installation, cost, realism, and that said material does not interfere with the inner circuitry. For this portion of the design, we believe going forward with a combination of the options gives us the best chance at success. Foam provides a lightweight and flexible base, which allows for easy manipulation and shaping to fit CAPER’s contours. Felt offers a softer texture, mimicking the appearance of feathers while providing durability and less maintenance. Lastly, artificial feathers can be used towards the end of design to increase the realism of the exterior, enhancing CAPER’s aesthetic appeal. Comparing companies for these materials deals mainly with availability and cost, as there are thousands of companies that make identical craft materials; it would be virtually impossible to compare based on factors like texture and weave.

Table 3.4: Outer Shell Vendor Comparison

Foam Companies	Felt Companies	Faux Feather Companies
Foamorder <ul style="list-style-type: none"> High product availability ranging from bedding to hobbyist shapes 	Jo-Anns <ul style="list-style-type: none"> Over 70 felt kits and 30 sheet selections All primary and secondary colors 	Ebay suppliers <ul style="list-style-type: none"> Large assortment of unbranded suppliers ranging from \$4.50 to \$6

<ul style="list-style-type: none"> Commercial grade furniture foam options available Eco-friendly organic selections available 	<ul style="list-style-type: none"> Ranging from 1 to 6 feet dimensions per sheet Averaging \$11 to \$16 per yard. Many in-house brands 	<ul style="list-style-type: none"> per yard (yardage refers to length of package strip, not feather length) Color availability is inconsistent Many 2-day shipping options available
Foam Factory <ul style="list-style-type: none"> Residential and commercial product selections Wider range of hobby related foams and cushioning Polyethylene and polypropylene alternatives 	Michaels <ul style="list-style-type: none"> Several hundred felt selections, all ranging in size In-house and outsourced brands Assorted colors, available individually or in packs 6' by 3' sheets average around \$12 	AliExpress <ul style="list-style-type: none"> Many more selections available in more "standard" colors Different bird feather shapes available, including parrot feathers Prices are less than half of ebay Sold by the yard or in "bundles"

Choice: Michaels for felt and foam, AliExpress for feathers

Reasoning: The foam companies mentioned above had wide selections for bedding and commercial applications, but very little for small scale hobbyist projects. After searching around, distributors like Michaels carried a lot of in-house solutions for felts, fabrics, and even foams. In the case of JoAnn, there are quite a few selections for felt, but all tended to be more expensive than Michaels. Interestingly enough, faux feathers were quite expensive from these name brand retailers; off-brand distributors through AliExpress had the widest selection and lowest prices by far.

Expected Performance: Our selections were made with the intent to balance realism, flexibility, and affordability for CAPER's exterior. These materials will allow a seamless appearance while allowing for easy removal and replacement when necessary. With proper care and maintenance, these materials can withstand the rigors of use and effectively conceal caper's inner workings while still going for the realistic look we were going for.

3.2 Movement Characteristics

Although a very basic principle, choosing which four movements we want to have for CAPER is crucial for creating a lifelike appearance. It has already been mentioned that CAPER will have mouth, head, body, and tail movements, but we never clarified why these decisions were made. There are many possible movements we can create, and several different ways to achieve it. Given the fact that solenoid actuators are being used, we must orient the movements to be linear, rather than rotational. The first and perhaps most important movement is the mouth. CAPER's main feature is the ability to talk; not including an animated mouth would be pointless.

Going downward on the figure, the next movement will occur at the base of the head. We could either make a side to side rotation, or an up to down tilt movement. We chose the up and down movement both because of the solenoids limited stroke range, and the inability to control position. Also, "head bobbing" is a very common movement parrots do when seeking attention. Along with the head, the body tilt is also an important movement; posture is another form of body language in parrots, usually indicating when they're about to fly. The last movement we want to include lateral extension of the wings and tail. Since CAPER isn't going to fly, creating fully functional extendable wings is unnecessary. However, parrots often use their tail/wings to maintain balance on a perch; this is definitely a movement we can recreate, since it doesn't involve any drastic extension. Retraction of that solenoid will lift the tail slightly, and "puff" out the wings laterally.

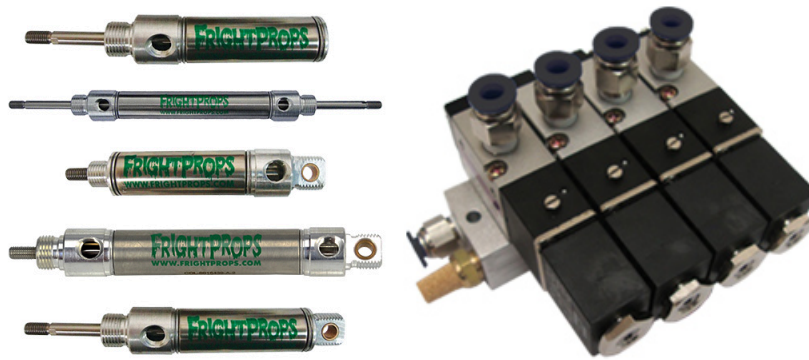
3.2.1 Motion Systems

Giving CAPER the ability to move is the most important mechanical aspect in this project; and there are many ways to do it. One of the most common methods of animating a figure is using compressed air (pneumatics). Solenoid actuated valves open and close moving the pneumatic ram back and forth from its extended position to its resting position. Fluidity and speed of the movements is controlled mechanically using various governors and counterbalancing springs on each joint. This is a very unreliable metric for controlling the movement, and any calculated values for "tightness" or "spring strength" may shift as parts wear out and age. Over time, these animatronics will lose their fluidity and stability, requiring routine maintenance; because of this, many modern robotics companies have made the switch to fully electric systems using servo motors. Movement and speed can be controlled electronically rather than mechanically, and movement sequence repeatability is increased dramatically. Other forms of movement include non-servo geared motors, gearless stepper motors, and solenoids.

Table 3.5: Motion systems

	Pneumatic s	Servos	Geared DC Motors	Solenoids	Stepper Motors
Cost	>\$300 total not	\$20.00 not including	<\$10.00	~\$15.00	\$13 for smaller

	including compressor or valve bank	controller			sized motors
Size	As small as 3 inch length	From “Micro” to “Jumbo”	Similar to servo	Similar to pneumatic	4cm wide square housing
Self-Contained (all inside CAPER)	No, requires external valve bank and air compressor	Yes, but will require additional control board	Yes	Yes	Yes, but would require additional control board
Suitable for Outdoor Use	Yes	Only the expensive ones	No	Yes	Only if outer shell is resistant (no)
Controllability	On/Off	Speed and Position	Approximate Speed and Direction	On/Off	Digital step counting, no real time position sensing
Movement	Linear Stroke	Limited or Cont. Rotation	Continuous Rotation	Linear Stroke	Continuous rotation
Power	From 200W to 4KW air compressor	~5W each, (20W total)	~0.2W each (~1W total).	~4W each (16W total)	72W each (288W total)



Figures 3.3 & 3.4: Pneumatic cylinders(left) and solenoid valves (right), courtesy of frightprops.com

Immediately we can rule out pneumatics and hydraulics. The requirement of an external compressor completely shatters our size and budget constraints; not to mention the need for a solenoid valve bank as seen in Figures 3.3 & 3.4. Geared DC motors can also be ruled out since there is no practical way to control the position without resorting to mechanical limitations; the small plastic gears in these motors are not well-suited for any stressful applications. Stepper motors are suitable for direct drive applications such as printer carriages and prismatic robotic arm joints. Since all of CAPER's movements are rotational, stepper motors have no practical use here. Servo motors and solenoids seem to be the most practical solutions for CAPER's movement, but both involve a drastic compromise. Servos boast the ability to control speed and position, but are expensive and difficult to implement. Solenoids are cheap and easy to implement but act in a binary fashion, with little to no position control. Given the constraints and goals for CAPER, it would seem solenoids are the better choice. Using servos would require an extra control board and significantly increase the processing requirements for the IOCB. This would be a worthwhile investment if CAPER performed extremely complex rotational movements, but this isn't the case. Solenoids allow for direct control from the custom IOCB board, and their binary nature allows for a far easier programming scheme with the VRB. The compromise in movement accuracy is far overshadowed by the gains of lower cost, near-silent operation, ease of control, and simplified controller design. Interestingly, the animatronic robin in Figure 3.3 uses a solenoid actuator for the beak; audio impulses from the show's audio would trigger the movement of the beak using a small magnet, while the tail, head, and wings were all air powered (Doctorow, 2016).

Motion Systems Part Selection

Solenoid configurations range greatly in voltage, current, pull strength, and stroke distance. Choosing appropriate sized solenoids at this stage is somewhat of a guessing game, as no exact numeric figures have been determined regarding the weight of the frame. The safest route to go is to select larger solenoids than what is required; it is better to have too much strength than not enough. Common solenoid actuator voltages are 12 & 24VDC, with current draws ranging from 0.5 to 1 ampere. Given Ohm's law, solenoids with higher voltage ratings can achieve similar pull strength to lower voltage

solenoids using proportionally less current. Using 24V solenoids instead of 12V will effectively halve our current requirement per unit of force, meaning we can install thinner electrical wires for each actuator. Even after narrowing down the voltage, there are still many different options regarding strength and stroke. Of the four movements CAPER has, the body tilt and tail extension will require more power than the mouth or head tilt. A future optimization would be to use smaller solenoids for those smaller movements.

Table 3.6: Motion Systems Part Comparison

Aexit	Features <ul style="list-style-type: none"> • Within voltage and current range as specified: 14.4W @24VDC • Suitable housing dimensions: 2.6" x 1" x 0.8" • Sufficient stroke length: 1.2" • 100g mass a piece. • Pull strength of approximately 35N • Spring loaded • Within budget range: \$12 to \$15 a piece
Ledex	<ul style="list-style-type: none"> • Well within power range at under 9 watts @ 24VDC • Housing dimensions of 2" x 1.5" x 1" • Smaller stroke length of 0.69" • 192g mass a piece • 124N pull force • Not Spring loaded
Uxcell	<ul style="list-style-type: none"> • Rated at 72W (when fully retracted) • Dimensions 1" x 0.75" x 0.66" • 10mm stroke • 10N force • Unspecified weight (smallest size of the selections) • Spring Loaded
RS Pro Linear	<ul style="list-style-type: none"> • 11W to 110W consumption from 10% to 100% DC supply • \$21 a piece • 2" x 0.75" x 0.75" • 12mm stroke length • Not spring loaded • 25N pull force • Corrosion resistant coating

Choice: "Aexit 24V, 0.6A, Push Type Open Frame Actuator Electric Solenoid"

Reasoning: The company RS Pro Linear manufactures a great variety of 24V solenoids with similar dimensions as the Aexit model shown above. Their solenoid with part # 1770114 would be a suitable part for the mouth and head tilt, having less stroke range

than the Aexit. Unfortunately, the price of these is significantly higher than any of the Aexit models. Similarly, the company Ledex produces a plethora of solenoids ranging in size and price. Here we have many more suitable replacement options, ranging from \$11 to \$35 a piece. Particularly, the model B17-L-154-B-3 is the most similar to our choice, priced at around \$16 each. All variables considered, the Aexit has the best all-around specifications for the price.

Expected Performance: These solenoids will have no trouble moving the joints of CAPER. They fit within the size, weight, and power constraints, and they have suitable stroke lengths. The lack of mounting hardware is the main downside, but all other solenoids seem to be at a similar disadvantage. Price and availability waver depending on the online listing, but they all have the same specifications. The lowest priced listing I found was for \$12.25 from Amazon with free shipping.

Surplus Counterparts: A unique opportunity presented itself in the quest for reasonably priced solenoids. I was able to obtain suitable counterparts from a surplus vendor for a fraction of the cost (\$2 to \$3 a piece). They are the exact same size, and have the exact same power and strength ratings as the Aexit solenoid model. Using these counterparts will reduce the amount we spend, and perform exactly the same. Despite this, these surplus counterparts will not be used in the final budget calculation. The company that produced these (Liberty Controls) is now defunct, and we cannot reliably list these as an available part. Budget calculations will use the Aexit solenoid model listed above, which is still in production today.

3.3 Standing Hardware

Ideally, CAPER should be able to be mounted to a wall, on top of a platform, or stand freely. This would require some sort of “base” that connects to the first link of the internal frame (at CAPER’s feet). The overall design of this base is certainly subject to debate, as many factors are at play including realism, installability, orientability, and overall feasibility. The initial idea was to use a heavy wooden block to mount CAPER on top of. The weight of the block would need to be significant enough so CAPER wouldn’t rock or fall over when moving, but also small enough to not be obstructive or bulky. For permanent installation, all additional hardware would be fixed to the base. Again, wood seems to be the more appropriate choice for this as it is cheap, strong, and dense enough. Metal and 3D printed plastic are good alternatives, and may be used in future redesigns.

Standing Hardware Selection

Mounting CAPER requires a stable and adaptable base that can easily support its weight and withstand its different movements without tipping or rocking. We essentially wish this to be a nonfactor in anything related to the parrot, just to serve as the stand. Our initial consideration was to use a heavy and shaped wooden block as it provides stability, cost-effectiveness, and is easily customizable.

Choice: Custom wooden block base from WoodCrafter

Reasoning: Various other companies offer custom wood shaping. However, when considering alternates for the base, where you get the wood from will not have a large impact. If wood does not work, we would pivot to 3D printed plastic bases or metal bases. UCF has their own 3D printer in the honors lab, which we would manually create ourselves if we went that route.

Pros:

- **Stability:** Wood provides the weight needed to support and stabilize CAPER, and can be shaped to provide support where the project is mounted.
- **Cost effectiveness:** Wooden blocks are relatively inexpensive compared to the other options listed, making them very budget friendly.
- **Customizability:** On top of the website we were looking at customizing it for us, wood can very easily be cut and shaped to fit different installation requirements and aesthetics.
- **Durability:** Treated, finished, and sealed wood can withstand environmental conditions as well as our alternatives and provide long lasting support.
- **Aesthetic:** As the goal of our project is a realistic and warming appearance, wood offers a natural scene that compliments CAPER's design.

Cons:

- **Design flexibility:** wood is not as flashy or vibrant as other materials could be, as well not easy to change once you have shaped it once.
- **Susceptible to damage:** if not properly maintained, wood is vulnerable to scratches, dents, and moisture damage.
- **Installation complexity:** Securing the hardware to the base may require drilling or other methods, while our alternatives could plan for a connection during the design of the support.
- **Size constraints:** If the wood is too large or heavy, it may become difficult for handling or transportation purposes. However, if it is too light, it won't support our project.

Secondary Choices: 3D printed plastic bases provide intricate design and lightweight properties but don't add to the overall aesthetic of the project. metal bases offer durability and design flexibility but may be more expensive.

Expected Performance: The wooden block base is expected to offer stability, cost-effectiveness, and adaptability for mounting CAPER. However, considerations should be made regarding transportation challenges, installation complexity, and potential maintenance requirements. Overall, the wooden block base is suitable for securely mounting CAPER while providing a natural and aesthetically pleasing appearance.

3.4 Electrical Hardware

Just like the motherboard in a personal computer, our selection of hardware components will be what brings CAPER to life. As mentioned before, there are two main control boards that comprise CAPER: the Input Output Control Board (IOCB), and Voltage Response Board (VRB). The IOCB will be a custom designed MCU board, containing the necessary power regulation, and MCU peripherals. The VRB board will be a pre-built, high powered, heavy computational board, will include the necessary peripherals, connections, and performance specifications needed to run the majority of our conversational pipeline software. Discussion of the software components for both boards will be discussed in the Software section of Chapter 3. Other hardware components external from the boards include the wiring, external power supply, solenoid relays, and audio interfaces (microphone, speaker, and audio amplifier). This section explores possible technologies for proving the necessary computing power for both boards, and the physical connectivity from the user to the CAPER figure.

3.4.1 Input Output Control Board

The heart of the IOCB, is the microcontroller unit that computes all of the received information. Whether it is the incoming MIDI signals, or the pulse streams from the buttons and VOX circuit; the IOCB's MCU will decode the incoming signals, and send the movement data streams to the voltage control hardware. This whole chip is noted from the large green “**Microcontroller**” block in **Figure 2.2**. Based on the research we've done, it is clear now that we need a chip that has sufficient I/O pins, UART capabilities, a usable debugging system, and a useful coding platform for programming. Although many microcontroller models exist, and within each model are many variations; the three most common options being the Texas Instruments MSP, Arduino, and Raspberry Pi. For the sake of comparison, we will consider the MSP430G2553, the Arduino ATMEGA328P, and the Raspberry Pi SC0914 chips. Each of these chips are well-regarded, commonly-available chip variations of their respective companies, and should each contain all of the necessary functionalities to control CAPER. The actual specifications are found below:

Table 3.7: Chip specifications

	MSP430G2553	Arduino ATMEGA328P	Raspberry Pi SC0914 (RP2040)
Speed	16 MHz	20 MHz	133MHz
Storage	16KB Flash	32KB Flash	264KB Flash
Pin Configurations	20 or 28 pins	21 or 28 pins	56 pins
UART	Yes	Yes	Yes
Debugging	Internal Bootstrap	debugWIRE	Done on PICO

Hardware	Loader using UART Bridge		board
Software	TI Code Composer Studio	Arduino IDE	Multiple IDEs can work for this

Clearly, each of these chips would work marvelously in theory; having more than the necessary qualifications to handle the processing. However, there are a few important factors to consider. In the case of the Raspberry Pi, the chip is meant to be used on the Raspberry Pi PICO microcontroller board. Programming the chip off the board is virtually impossible to do, and the package of the chip is much smaller than either of the other two. The IOCB is going to be a custom board specifically designed for CAPER; for this reason, we can safely rule out the Raspberry Pi models altogether. Other variations of the Raspberry Pi are very costly options, and are even more complicated.

Both the Arduino and MSP chips are viable options for the IOCB. Chip packages are easy to install on our board, they both have UART capability, and are able to be debugged off their boards. Speed, voltage, and memory specifications between the two are very similar as well. That being said, both chips will be considered as viable options for the IOCB. It is unfortunately too soon to know based on what is written on paper as to which is the superior chip for this application.

IOCB MCU Chip Selection

Perhaps the most important piece of hardware aside from the VRB board itself is the choice of MCU chip for the IOCB. This chip will take every input from every source and convert it into streams of output pulses for the relays. This includes the button and switch positions, the pulse from the VOX circuit, the incoming MIDI UART signal, and the debugging/bootloader circuit (differs from company to company). In short, the chip needs to have ample I/O pins, a compatible IDE for external programming, and enough capability to store and run the code without glitching or buffering. More specifically, CAPER requires sufficient clock speed, flash memory, RAM, programmable onboard timers (with interrupts), and usable firmware.

Upon researching the available options, we found two chip families that stood above the rest: Texas Instruments MSP430, and Arduino/Atmel ATmega. Both of these chips are available in similar packaging with similar features, and are commonly seen on their respective programming boards: the TI Launchpad and Arduino Uno. In the case of both chips, multiple versions exist of each; different packages, pin layouts, performance specifications, and firmware configurations are available within each family of MCUs. We've narrowed our choice to two chips from each family; one with a simple layout, and one with a complex layout. For both the TI and Atmel chips, the same tradeoff exists between the simple and complex layouts: the simple layouts are easier to test and install, while the complex layouts are more powerful. Below is a comparison table:

Table 3.8: ICOB MCU Chip Comparison

MSP430G2553	<p>Features:</p> <ul style="list-style-type: none"> ● Pin layout: 20, 28 and 32 pin packages available ● Package Size: 20-DIP ● Low power modes: (4), can shut off all clocks and timers ● Clock Modules: low frequency crystal and RC ● Timers: one 16-bit timer ● Serial Communication: UART, SPI, I2C, IrDA; automatic Baud-Rate detection ● Analog to Digital: supports 10 bit conversions ● Processing speed: up to 16MHz ● RAM: 512B, 16KB Flash RAM ● 16 bit architecture ● Voltage: 1.8 to 3.6 VDC
MSP430FR6989	<p>Features:</p> <ul style="list-style-type: none"> ● Pin layout: 80 or 100 pin ● Package Size: LQFP (PN or PZ) ● Low power modes: (3), shuts off all timers and clocks ● Clock Modules: Low & high frequency crystals, DCO, MODOSC, and external ● Timers; five 16-bit timers ● Serial Communication: UART, SPI, I2C, IrDA; automatic Baud-Rate detection ● Analog to Digital: 16 bit analog comparator, 12 bit ADC ● Processing speed: 16MHz ● RAM: 128KB FRAM, 2KB SRAM ● 16 bit architecture ● Voltage: 1.8 to 3.6 VDC ● Code security: 128-Bit or 256-Bit AES Security Encryption and Decryption Coprocessor ● Digital peripherals: 32-bit hardware multiplier, 3-channel internal direct memory access
Atmel ATMEGA328P-P U	<p>Features:</p> <ul style="list-style-type: none"> ● Pin layout: 28 pin (others available, but this layout is easiest to work with) ● Package Size: SPDIP ● Qtouch library: 64 sense channels; capacitive touch buttons, sliders, and wheels ● Clock Modules: Low frequency & full swing crystals, RC, external ● Timers: two 8-bit, one 16-bit, and six PWM channels ● Serial Communication: USART, SPI, and Phillips I2C

	<ul style="list-style-type: none"> • Analog to Digital: 6-channel, 10-bit (differs depending on package) • Processing speed: 20MHz (20 MIPS using single cycle instructions) • RAM: 32KB flash RAM, 2KB RAM • 8 bit architecture • Voltage: 1.8 to 5.5 VDC
ATMEGA2560	Features: <ul style="list-style-type: none"> • Pin layout: 100 pins • Package Size: TQFP or CGBA (25 pins per edge or 10x10 grid on bottom respectively) • Qtouch library: 64 sense channels; capacitive touch buttons, sliders, and wheels • Clock Modules: Low power crystal, LF crystal, Full swing crystal, internal RC, external • Timers: two 8-bit, four 16-bit, four 8-bit PWM channels & six-twelve PWM (2 to 16 bit) • Serial Communication: USART, SPI, 2-wire Byte Oriented serial interface • Six sleep modes, ADC noise reduction • Analog to Digital: 16-channel, 10-bit • Processing speed: 16 MHz (16 MIPS using single cycle instructions) • RAM: 256KB Flash RAM, 8KB RAM • 8 bit architecture • Voltage: 4.5 to 5.5VDC at 16MHz (as low as 2.7 for 8MHz) • JTAG compliance, extensive on-chip debugging support

Analysis of MSP Specifications: As expected, the FR6989 not only covers all the grounds of the G2553, but completely surpasses the performance in virtually every department. Along with this, the FR6989 chip features LCD display drivers, code encryption, and onboard direct memory access; features not included at all on the G2553. Debugging and programming the chips is done in similar fashion; using the internal bootstrap loader using the UART pins (an external UART bridge can facilitate this).

With all of the benefits to using the FR6989, the fundamental question appears: is it worth the extra time and effort of working with the FR6989's small and dense package to have all these great benefits? At first glance the answer is yes, but with more consideration, the G2553 becomes surprisingly more favorable. Ultimately, both chips have the necessary functionality to control CAPER easily. All the excessive functionality of the FR6989 chip is more than what is needed; in the end, most of that functionality will go unused. The 80 to 100 pins on the FR6989 is far more than what is needed on

our board; the smallest 20 pin layout would work just fine. The main deciding factor is how the G2553's larger scale is far easier to work with, especially in the testing phase. Though it wouldn't make too much of a difference once installed on the board, working with the chips off the board is a night and day comparison. The G2553 is much more friendly in experimental applications, even fitting nicely on a standard breadboard. This would allow us to easily take measurements and monitor the performances of the peripheral circuits in real time. We'll be able to make the necessary tweaks to not only make CAPER work, but work well. For this reason, the MSP430G2553 chip is the stronger candidate of the two TI selections; despite the far inferior performance ratings, it boasts a far superior level of usability and practicality. Perhaps we can consider implementing the FR6989 chip as a stretch goal; but for now, the MSP430G2553 is the primary choice from the Texas Instruments MSP family.

Analysis of ATMEGA Specifications: Similarly to the TI chips, both of these Arduino variations would have no trouble controlling CAPER. Interestingly, both of these chips are available in completely different packages, unlike the TI chips having only one package type with different pin amounts. The respective performance of the 328 and 2560 is the same across the board regardless of the package types. Considering this, the package on the 328 is exactly the same as the TI G2553; being compatible with standard breadboards. For this reason, the same exact logic applies; the simpler layout of the 328 chip is worth not having the extra features of the 2560 chip. In fact, the difference of performance between these two is far less than the two TI chips; the 328 even has a faster clock speed. The extra features of the 2560 will likely go unused, just as it is with the TI FR6989; and since the two are so similar, there is no point in considering the 2560 as a stretch goal upgrade. The ATMEGA328P-PU is the more logical choice between the two Arduino selections.

Choice: MSP430G2553

Reasoning: After comparing both companies' chips, a few new thoughts appear. Mainly, we can now safely rule out the FR6989 chip entirely, even as a stretch goal implementation. The ATMEGA328P performs similarly to the FR6989, all while being in a much more practical package. Despite this, we cannot yet rule out the G2553 chip. While it is true the 328 chip also outperforms the G2553 chip, there are still some tradeoffs. Benefits of choosing the 328 include having double the RAM, a 4MHz-faster clock speed, and two more timers. Downsides include the half-size word length and conflicts arising between multiple libraries. While these libraries can make tasks like establishing a MIDI connection much easier than the TI ever could be, there is a distinct lack of unification or hierarchy among them. When multiple of these libraries are used simultaneously; they can prevent each other from accessing the MCU's peripherals like the ADC converter or timer. It is because of this reason alone that the TI G2553 chip is the most favorable option. Having to code the entire program from the ground up gives us full control of every aspect of that code. It will take considerably more time to complete, but it eliminates the need to rely on someone else's work. This isn't a dealbreaker for the 328, as it is still a fantastic choice for an IOCB MCU chip. In

conclusion, the Texas Instruments MSP430G2553 is the best choice, and the Atmel ATMEGA328P-PU is a considerable alternative.

IOCB Debugging Circuit

USB connectivity to the IOCB is a crucial feature; the surface mount versions of the MSP430 will not be programmable on the MSP launchpad board, and will therefore need some external connectivity. There are really only three ways to do this: debug through the TI MSP-FET module, create the EZ-FET emulation circuit from the open-source schematics, or use a third party general-purpose UART to USB bridge.

The entire MSP chip family can be debugged using the MSP-FET module, though the cost is almost \$150 per unit. Alternatively, we can reconstruct the USB programming module seen on the MSP430G2ET launchpad; this circuit is readily available, built specifically for the MSP430, and complete with the .sch files. The only downside is the size and cost impact this will have; including this circuit on our board will increase its size by 50%, and would require many hours to implement. A considerable alternative is a common USB to UART bridge; this device is 15 times less expensive than the MSP-FET and is much smaller than the EZ-FET. Multiple of these circuits are available and all are compatible with the MSP chip for off-the-board debugging. Below are three fine examples of easily attainable, general purpose UART to USB bridges:

Table 3.9: Debugging Circuit Comparison

FTDI FT232BM Standard USB to Serial TTL	HiLetgo FT232RL Mini USB to TTL Serial Converter	WaveShare CP2102 USB UART Board
1 IC, several smaller components	Same as FTDI	1IC, different package but similar pin configuration
3.3V to 5.25V operating range	Same as FTDI	5V with onboard 3.3V converter
6 to 48MHz configurable clock	Same as FTDI	300b to 1Mb baud rate capabilities
500mW power dissipation	Same as FTDI	Same as FTDI

Choice: HiLetgo FT232RL Mini USB to TTL Serial Converter

Reasoning: This version of the UART bridge is exactly the same as the name brand FTDI counterpart and is 50% cheaper. It also has a simpler board layout than the CP2102, all while boasting similar performance. Using this with Code Composer Studio is possible after modifying the Post-Build instructions, allowing direct programming to the MCU straight from the computer. The FT232 can be directly added to the IOCB, and connect to the UART RX, TST, and RST pins of the MCU (Vcc and GND will be

unused). Connector pins may also be added to allow for external debugging using the Launchpad board.

Note: Usage of this device may also require use of pin 1.5; this shouldn't matter as the I/O pins are deactivated when debugging.

3.4.2 Voice Response System Computation

This section discusses the voice response system and the computation systems needed to implement it.

Pipeline Architecture

The voice response system consists of a voice-to-voice data pipeline that can interpret human speech and respond in a coherent and relevant way. As outlined above, this pipeline will consist of a variety of AI model architectures set up in series. The cost of this computational pipeline is expected to come nearly exclusively from the LLM, the piece responsible for accurate and relevant response text generation. This has the downstream effect of requiring a large amount of compute to run the whole pipeline without any noticeable delay.

In quantifiable terms, the amount of working memory any system running the pipeline would need is in excess of 10GB. This value is the result of comprehensive research into the state-of-the-art systems that are currently available for public use. Although a custom trained system would, theoretically, reduce the computational cost by being trained within constrained hardware, the time and complexity needed to successfully implement and train this system far outweigh the cost of the options outlined below.

There are multiple ways this could be resolved:

1. Acquire high-cost edge hardware for running the pipeline
2. Offload the LLM to a separate computation system and utilize wireless communication to maintain pipeline coherence.

High-Cost Edge Hardware

Purchasing hardware that matches the desired spec to run the whole pipeline locally is as simple as it sounds. Edge hardware consists of any computing system that is in close physical proximity to where the data it is processing is produced.

The most likely technology able to fill this role are SoCs (Systems-on-Chip) preferably with dedicated linear algebra compute cores.

The advantages of employing high-cost edge hardware are significant. Firstly, having a pipeline unified on a single device would limit the amount of supporting software required to link its modules together. Additionally, version control and cross-module compatibility issues would be dealt with easily, ensuring smooth operations. Another key benefit is that the entire pipeline would be able to function without the need for an internet connection, offering independence from network availability. Furthermore, all

data being created, processed, and stored in a single place would eliminate nearly all privacy concerns, a paramount advantage for sensitive applications. The issue of latency, especially when it comes to inter-module communication, would be all but eliminated, streamlining operations. Lastly, the ease of maintenance is a non-trivial benefit, simplifying the upkeep of the system.

However, the drawbacks of this method are considerable. The very high upfront monetary cost, resulting from the high computing power needed in a portable and power-efficient device, poses a significant barrier to entry. Moreover, the lack of upgradability is a serious limitation; once the software is optimized for the hardware, any architectural changes would need to consume the same or less compute to be viable, severely constraining future improvements. The goal of CAPER is to create something accessible. Thousands of dollars worth of computing hardware is not accessible.

Server Offloading

Server offloading consists of hosting the response generation LLM to a separate computer linked to the rest of the pipeline via an internet connection. Cloud provider choice notwithstanding, the cost of operating a cloud based pipeline is much lower than the cost of hosting everything on a single unified hardware platform.

This approach, although beneficial, comes with its own set of choices. Since CAPER is a senior design project, setting a system up to use proprietary software would represent the easiest option, but it would not be a good learning experience. Since an unspoken objective of this project is to foster design skills, the idea of running the server on the team's very own hardware while utilizing the team's very own security and server setup software, must be considered.

The benefits of server offloading are notable. Greater performance per dollar spent is achieved due to the increased cost efficiency of comparable off-the-shelf server hardware when compared to edge hardware. This approach also allows for any performance to be highly scalable, thanks to the ease of swapping server components out for more powerful ones, accommodating future growth. Additionally, this method presents a greater learning opportunity. Given that this is a senior design project, the importance of learning cannot be understated, making server offloading an attractive option for educational purposes.

However, there are drawbacks to consider. A dependence on stable internet communication for pipeline function reduces the portability of the system as a whole, potentially limiting its applicability in environments with unreliable network access. Additionally, the increase in pipeline complexity due to the myriad of supplemental security, redundancy, and communication software needed for it to function properly can complicate operations. Lastly, there is a potential for overengineering, which could lead to unnecessary complexity and increased costs, detracting from the project's aim of creating something accessible.

The following table summarizes the aforementioned arguments:

Table 3.10: Hardware vs Server Offloading

Aspect	High-Cost Edge Hardware	Server Offloading
Best performance to cost ratio	✗	✓
Best software scalability	✗	✓
Best hardware upgradability	✗	✓
Best pipeline simplicity	✓	✗
Best usage latency	✓	✗
Best educational opportunity	✗	✓

Although very much worth considering, the downsides of server offloading do not offset its monetary cost upside. Due to one of the project's aims being firmly cost-based, the best choice of pipeline architecture is one where the LLM is offloaded to a separate server, while the rest of the pipeline runs on the local VRB.

Voice Response Board Hardware Selection

The VRB's sole objective is to provide enough computation to transcribe text, transmit it to the offloaded LLM, process the response into movement commands and audio, and send said output to the appropriate peripherals. Based on the choices made in Section 3.9.2, the VRB will need 2GB of working memory at the very least and the necessary hardware to support PyTorch-based models. Based on the earlier choices, the choice of system for the compute for CAPER must have the following attributes:

Table 3.11: Voice Board comparison

Device	Price (USD)	CPU/GPU	RAM	Storage	Performance (GFLOPs)	Language
Nvidia Jetson Nano	\$149.99	Quad-core ARM A57 / 128-core NVIDIA Maxwell	2/4 GB	microSD	500	Cuda and OpenCL
Google	\$99.99	MediaTek	2 GB	8 GB	32 (32-bit)/64	OpenCL

Coral Dev Board Mini		8167s SoC / IMG PowerVR GE8300		eMMC	(16-bit)	
BeagleBone AI-64	\$187.50	Texas Instruments TDA4VM	4 GB	16 GB eMMC	160 (out-of-box) 8000 (requires models bit customized)	OpenCL

The best choice here is the Nvidia Jetson Nano due to its cuda support.

3.4.3 Board Interfacing (cables vs wireless)

Implementing UART communication forces our hand to use a wired medium for communication. Since the VRB and IOCB are going to be in close proximity to each other, this isn't a dealbreaker. A better stretch goal would be to make the connection between the VRB and language model wireless, or between the language model and the speaker. Though given our time and budget constraints, all of our efforts should be oriented to making CAPER move and respond in a lifelike manner. Making the VRB wireless would hardly make an impact on that issue. Information regarding this technology and the use of an optoisolator will be discussed in Part Selection. Despite being a major component in this system, there is virtually no reason to do a deep comparison, as there are no suitable alternatives to the already existing optoisolator. A MIDI system coupled with a microcontroller requires this optocoupler chip to be used in between to maintain stability between the two voltages.

3.5 Power Systems

Powering all of CAPER's subsystems will require multi-level voltage conditioning, with many different voltage and current limitations at play. Between the current draw of the solenoid actuators, or the instability of MCU chips with incorrect voltages, CAPER's power system will require attention to both capacity and accuracy. Information regarding power for the MIDI playback system and LLM network will be discussed later on, as those systems will be run (partially) externally, and will have their own power sources. Thankfully, there is an abundance of available power conditioning and voltage regulating devices at our disposal.

3.5.1 Data Stream Voltage Control Hardware (Activating Circuit)

Activating circuits are required to convert the low voltage data streams from the IOCB into high voltage power streams for the actuators. This can be achieved electrically, or electromechanically, i.e. transistors or relays. Two appropriate choices of each are the 2N2222 multi-purpose BJT, and the BESTEP 3.3V relay module. Actual part selection for these components may vary, see part selection for actual models. Below is a basic comparison of a transistors' ratings compared to the BESTEP relay module:

Table 3.12: Transistors vs Relays

	BJT (2N2222)	Relay (BESTEP)
Voltage Limits	5V (emitter to base). 60V (collector to base)	30V DC (triggers at exactly 3.3V)
Current Limits	800mA	10A
Switching Time	~150 microseconds	~10 milliseconds
Cost	\$0.40	~\$2.00

The use of a transistor switch seems to be an appropriate choice for the VOX circuit, but not so much for the CAPER actuators. Relays on the other hand, would work perfectly for this; they have a much higher power limit than typical general purpose BJTs. The difference in switching time is virtually negligible, as a 10ms delay won't even be perceived by the viewer.

Data Voltage Control Hardware Selection

The device immediately upstream from the solenoid actuators will be a bank of relays controlled by the IOCB MCU. These relays need to handle 24+ VDC, and be activated by voltages as small as 3.3VDC. Mechanical electrical relays were proven to be a better choice than BJTs alone for this application due to the power requirements of the solenoids downstream. However, mechanical solenoids are extremely large and expensive; we wouldn't be able to use them. The compromise is a combination between a transistor and optoisolator. These are readily available and intended for use with common microcontrollers including the Arduino Uno. Below is a comparison table of the three different brands:

Table 3.13: Data Voltage Control Hardware Selection

	BESTEP	HiLetgo	Teyleten
Price	<\$1	\$5.89	\$2
Config.	1 channel	1-8 channels	1 Channel
Power	10A 250VAC or 30VDC	10A 250VAC or 30VDC	10A 250VAC or 30VDC
Trigger V.	3.3V (5V options available)	5V or 12V (no 3.3V options)	3.3V
Connections	Raw Wire	Raw Wire	Raw Wire
Norm O/C	Open or Closed	Open or Closed	Open or Closed

Switch Delay	<20ms	*unspecified	<20ms
--------------	-------	--------------	-------

Choice: BESTEP (depending on price fluctuation)

Expected Performance: These affordable relay modules are perfect for activating the solenoids; their completely non-mechanical design allows for quick and silent control. Unfortunately, the non-adjustable voltage ratings require perfect level matching with the output pins. Insufficient voltage wouldn't be enough to activate the relay, and too much voltage will destroy the circuit instantly. Another issue is the single layer board layout that leaves the underside lands completely exposed and protruding out. Neither of these issues are unresolvable, but both will require appropriate precautionary measures to ensure safe, stable operation.

Possible Need for Alternate Sources: The production of these relays is difficult to keep track of; there are many rebrandings of the exact same relay from different suppliers. Each reiteration has the exact same specifications as is listed above for the BESTEP relays. It solely becomes a matter of price and availability at the time of purchase. The three brands mentioned above tend to be in stock more frequently, but other companies like ELEGOO, AEDIKO, JBTECH and SUNFOUNDER also produce them.

3.5.2 VOX Circuit:

Another section of CAPER that involves switching is the Ham Radio VOX Circuit; a simple circuit that converts microphone signal presence into a binary style pulse stream. Whenever considerable audio is present at the microphone, the VOX circuit will swing high; absence of sound will cause the circuit to swing low.

Op-Amp

Instead of using the relay modules, this circuit will utilize multiple op-amp stages instead. In total, four separate stages of op-amps will be required: two will convert the audio to a pulse stream, and the other two will slightly amplify the signal to be sent to the VRB for speech recognition. Here are some more details regarding the 4 channel op amp:

Table 3.14: Op-amp Comparison

TI TL084	Microchip Technology MCP6004
<ul style="list-style-type: none"> • Wide input voltage range: 4.5-40VDC • Operating temperature maximum 125 degrees Celsius • Fast slew rate of 13V/microsecond • +/-1.5V output swing 	<ul style="list-style-type: none"> • 6.5VDC maximum input across Vcc and Vdd • Operation max temperature 125 degrees Celsius • Slower slew rate 0.13V/microsecond

Op-amp Choice: TI TL084 4-Channel General Purpose Operational Amplifier

Reasoning: In terms of general purpose op-amps, this model certainly checks all the boxes. Having 4 stages in a single compact housing will allow us to construct the multi-stage circuits with a much smaller footprint. The only major downside to this chip is the requirement of two- distinct input supplies. Single rail op-amps do exist, and would eliminate the need for separate supplies. However, all the peripheral circuits on the IOCB will draw considerable current; more current than one single voltage regulator could safely supply without overheating. Given this, multiple voltage regulators will be needed at the same voltage levels; this makes the TL084 op-amp a more viable option for us. In the case of the Microchip technology alternative, specifications are quite inferior across the board, especially with the supply voltage ratings. These only allow a maximum of 6V across the input terminals, as opposed to the TL084's 40V range.

Transistor

On the pulse stream side, the signal exiting the second stage of the op-amp will enter the base of a NPN transistor. This will be what “switches” the pulse stream on and off when there is audio present. A trim potentiometer can be installed to adjust the sensitivity. Here are some more details regarding the transistor:

Transistor Choice: P2N2222A NPN Amplifier Transistor

Available companies: Central Semiconductor, CEL, Analog Devices, etc.

Features and Specifications:

- Emitter Collector Voltage: 40V
- Collector Base Voltage: 75V
- Emitter Base Voltage: 6V
- 600mA collector current rating
- <200ns turn off & turn on times
- Steady voltage to temperature relationship
- Current gain of 150 @room temperature & $I_c = 100\text{mA}$

Expected Performance: There was no point to making a pros and cons list, as the specifications are what is expected from a “general purpose transistor”. Given the adjustable nature of the preceding op-amp stages, multiple transistor models would work fine in this position. The 2N2222 transistor is well-suited for low voltage microcontroller switching applications; it would work fine in the VOX circuit. Since many different companies produce this transistor with the exact same specifications, there is nothing to compare except for price and package. The package likely chosen for this project will be the TO-92, with TO-18 and TO-39 as alternate choices. Similar widely produced models of NPN BJTs include the 2N3904, BC547, and 2N3055.

Main differences from 2N2222

- In the case of the 2N3055, that transistor is best suited for exceptionally high current applications; better for amplification rather than switching applications.
- The BC547 has a higher DC gain and a lower thermal resistance than the 2N2222.

- The 2N3904 behaves almost identically, but has a much lower current tolerance.

With the wide range of “general purpose” transistors, there’s no point comparing them; any of these BJT’s would get the job done, each having similar ratings and specifications. The 2N2222 is the perfect “middle of the road” choice, meeting all the necessary requirements but not completely surpassing them.

3.5.3 Microphone. Amplifier, and Speaker

Although not pictured, the microphone will plug into the VOX Circuit, as illustrated by the blue block in the upper left of Figure 2.2: Main Block Diagram. The microphone will be used for two main purposes; to capture audio in high fidelity for the language model to process, and to trigger the VOX circuit at a certain volume threshold.

The Microphone Circuit Style

Choosing a distinct microphone type is crucial for both systems to work properly, and there are many types to choose from. Common circuit styles include dynamic microphones, condenser microphones, and ribbon microphones, and they can have various “pickup patterns” including omnidirectional, cardioid, supercardioid, hypercardioid, and even bidirectional patterns like figure-eight. A brief comparison between the three types is seen below:

Table 3.15: Microphone Circuit Style Comparison

Dynamic	<ul style="list-style-type: none"> - Dynamic microphones are the most common circuits you can find in a handheld microphone. Essentially a reverse speaker, dynamic mics use coiled wire and a permanent magnet to generate electrical waves from an incoming sound source. They do not require an external phantom power supply, are very lightweight and portable, and relatively affordable compared to other mics (anywhere from \$50 to \$300).
Condenser	<ul style="list-style-type: none"> - Condensers (the British-English term for Capacitors), rely on a pair of capacitor plates to generate soundwaves based on the gap between them. Unlike Dynamics, Condenser microphones do require an external 70V phantom power supply to produce signal, but are usually more articulate in the 10kHz range and higher. Prices range from \$100 to \$4000+, not including the price of the power supply.
Ribbon	<ul style="list-style-type: none"> - Ribbon mics are one of the oldest microphone styles you can still purchase, originating in the 1920s. Active and passive versions both exist, but they CANNOT handle phantom power (unless you want to instantly destroy them). In fact, they are the most fragile style of microphone on this list, and expensive too, ranging as high as \$7000 dollars.

Based on our needs for CAPER, ribbon microphones check the least amount of boxes. Dynamics and condensers both seem like viable options, taking into account durability, versatility, and affordability. Ultimately, dynamics have the edge over condensers due to not needing a phantom power supply. We want as little noise as possible for the language model to accurately detect and translate speech. The presence of a phantom power supply (especially a cheaper one), would certainly have a negative impact on the noise floor. Likewise, omnidirectional pickup patterns are more affordable in dynamic configurations than in condensers. It is clear that we'll need to use an omnidirectional microphone for CAPER, as we want to detect noise from a close radius, not just directly in front of the mic.

The Amplifier Circuit Style

The variety of circuit styles for amplifiers is more straightforward, as our cost and size constraints limit us to small Class D circuits. These circuits are common in smaller bluetooth speakers, computer speakers, and even in smaller home theater/soundbar speakers. They are affordable, easy to produce and purchase, and are reliable circuits. Of course, a significant aspect of CAPER is to allow for user's choice in these departments. For instance; a large-scale commercial installation of CAPER would require a larger Class AB circuit with more wattage, and our board would be able to work with it. Thankfully, we aren't in a position that requires that much power, so a ~20W Class D amplifier with appropriately rated speakers would work perfectly.

3.5.4 Microphone, Amplifier, and Speaker Selection

The most important aspect of microphone and speaker selection isn't so much the brand and model, but rather the type. These devices should be treated with the same regard for specification as a mouse and keyboard would be to a computer; the choice is ultimately left to the user depending on the context of installation. Within the scope of our use, we're aiming for a 5 to 8 foot radius around the front of CAPER (give or take). For anyone standing within this zone, we want the microphone to audibly detect and receive their speech, and the speakers to clearly project the response audio back to them.

The preferable microphone configuration would be an omnidirectional dynamic microphone; capable of detecting sound from all directions without the need for an external phantom power supply. Interestingly, there is an easy hack to convert a cardioid-pattern unidirectional microphone into an omni using a simple and reversible modification. Cardioid dynamic microphones are designed with tuned ports on the rear of the basket behind the diaphragm; simply covering these ports up converts the cardioid pickup pattern into a spherical pickup pattern. This gives us many more options to choose from, as unidirectional microphones are usually more affordable. Utilizing some of the gain stages from the VOX circuit, the incoming signal will be buffered before being sent to the VRB.

As far as the speakers are concerned, installation on or inside CAPER would not be recommended as it will obscure the sound and likely not be able to fulfill all size, volume, and clarity requirements. Therefore, our focus turns to external speakers and

amplifiers to improve the quality and volume of CAPER's responses. Here are some common examples of dynamic microphones that would work:

Table 3.16: Microphone Type Comparison

Behringer SL84C	Shure SM58	Pyle PDMIC 58	Samson Q8X
<ul style="list-style-type: none"> • \$15 • Dynamic with cardioid pickup pattern • Sensitivity of 54 ± 2 dBV/Pa • 50-15kHz response 	<ul style="list-style-type: none"> • \$100 • Dynamic with cardioid pickup pattern • Sensitivity of -54.5 dBV/Pa • 50-15kHz response 	<ul style="list-style-type: none"> • \$22 • Dynamic with Cardioid pickup pattern • Sensitivity of $-54\text{dB} \pm 3$ dBV/Pa • 50-15kHz response 	<ul style="list-style-type: none"> • \$100 • Dynamic with Cardioid pickup pattern • -54dB dBV/Pa • 50-16kHz response

Microphone Choice: Behringer SL84C Dynamic Cardioid Microphone

Reasoning: The affordable Behringer SL84C is a great choice for a budget vocal microphone; it behaves almost as good as the SM58 at only a 5th of the price. Although it isn't omnidirectional out of the box, covering the tuned ports will convert it into one. The biggest downside to this model is the frequency pickup response; factors like proximity or angle have a great effect on the quality of the sound received. However, this is true with virtually any microphone, including ultra-expensive models. Considering AI platforms perform remarkably well with phone microphones a fraction of the size, there is no reason why this Behringer microphone wouldn't work. Provided there is little to no noise coming from neighboring circuits, the signal should arrive at the VRB clearly and with little distortion.

Speaker Choice: RIOWOIS Passive Bookshelf Speakers for Home Theater Surround Sound

Features:

- Cost effective
- Achieves all required specifications (size, clarity, volume)
- Matches wood aesthetic adding to realism for CAPER
- Comes in pair for extra range
- compact size of 4.3" in length and 6.6" in height
- 2.75-inch woofer and a 2-inch tweeter

Secondary Choices: Cambridge Audio Minx Min 12, Pyle Home 3, Logitech® Z150

Expected Performance: In this price range, variations from speaker to speaker most of the time aren't discernable in terms of quality or volume. The RIOWOIS speaker satisfies two important constraints for our project being cost efficient and adding to the overall aesthetic of CAPER. On top of that, they're sold in pairs which allows us for a greater total range of output. The independent tweeter and woofer work together to deliver more accurate and complete sound than a single full-range driver. They are not as nice as some speakers such as the Audio Minx Min 12, but in the grand scheme of things the quality is not worth the high price difference, and goes to show how customizable our project is. The speakers we have selected perform their desired role, add to the aesthetic, and keep the project expenses low, all while not taking away from CAPER as they're relatively small for a speaker.

Amplifier Choice: Lepai LP-2020AD

Features:

- Affordable at \$30 per unit, including 12 volt power supply
- 4 to 8 ohm speaker impedance
- Raw-wire speaker connections
- 20 Watts RMS total output
- RCA and 3.5mm stereo inputs
- Volume and bypassable treble and bass controls
- Sturdy aluminum enclosure
- <0.1% total harmonic distortion

Secondary Choices: Lepai LP808, Kinter MA170, Kinter K3118, Pyle PTAU45, Pyle PFA300

Reasoning: This compact class D stereo amplifier is a fantastic pairing with any of the passive speaker choices we mentioned above. Interestingly, this same amplifier circuit is available under several rebrandings and repackaging, with the Lepai counterparts being most favorable. The other variations of this amplifier from Kinter perform similarly, but all tend to have less wattage headroom. The 20-Watt-RMS output of this amplifier is capable of projecting the sound to the audience clearly and intelligibly, and the impedance range of 4-8 ohms allows virtually any speaker choice. Two parallel inputs are on the rear of this amplifier; a stereo 3.5mm female plug, and two mono RCA plugs. There are plenty of available splitters and connectors, so establishing connection between the VRB or the MIDI playback device is more than possible.

If we were to choose a set of active speakers (built-in amplifier), this device would not be necessary. However, given most affordable speakers are passive, this amplifier is the optimal choice to complete the system. On the plus-side, we can run any length of speaker wire between the amp and speakers, meaning we can adjust the volume and equalization remotely.

3.5.5 Electrical Wiring

The types of wire used between the systems will vary depending on the expected current draw of the downstream components. We have the choice between either solid-core or stranded wire to use for our systems. Solid wire is cheaper, less resistive, and stronger, but it is less flexible than stranded. For this reason, it would be logical to use stranded for virtually every portion of this system. Flexibility is the most critical factor, especially inside the CAPER figure, and the length of the wire isn't enough to justify the use of solid-core.

3.6 Power Supply

The goal for CAPER is to have a single, discrete power supply for the whole system, that all of the downstream voltage circuits can feed from using voltage regulators. For this, there are two choices of power supply type to choose from: linear or switching. Between the two, switching power supplies are more efficient and cost effective, whereas linear supplies are more stable. Either would work fine for supplying power to the IOCB given unlimited budget and low power requirements; but when considering these constraints, the switching power supply becomes more favorable. The solenoids can draw up to 16W of power when activated simultaneously; the higher current capacity of switching supplies will account for this and withstand the strain better than equally priced linear supplies. We wouldn't want any drastic voltage drops to occur for the IOCB, as sudden shifts can irreversibly damage the chip. This shouldn't be a problem for either circuit style.

3.6.1 Power Supply Selection

For the power supply, the selection process was quite simple. Due to the fact that the solenoids for CAPER require 24 volts to power, a 12 volt power supply was decided on. The solenoids can be connected between the +12VDC and -12VDC, creating a 24 volt potential. This way, we can run the voltage regulators off of only 12VDC, which is well under the maximum rating for them. The power supply will not only supply power to the solenoids, but all the various components in the circuits such as the 3.3V and 5V voltage regulators and transistors.

When deciding between switching versus linear supplies, the main decision was based on costs. The average cost of a switching supply is around 20 dollars while the linear supply sits at an average of 100 dollars. Because our project is self-funded, the obvious choice is sticking with the switching power supply. Besides the cost factor, switching supplies trump linear supplies in terms of efficiency. Linear power supplies tend to draw more power than the equivalent switcher, meaning that it will get hotter quicker as well, and thus the risk of overheating increases. A switching power supply is also smaller and lighter than a linear power supply, as well as allowing for a smaller overall circuit and increasing the efficiency of the AC to DC converter.. The only slight downside to the switching power supply is that it is somewhat noisier on their output. If the noise becomes too problematic, then filtering can be added afterwards. Aside from the solenoids, the power supply also directly powers the IOCB voltage regulators, as shown in Figure 3.5:

connectors on the chassis allow for live, neutral, and ground connections. This is a nice added feature for high current applications, though CAPER shouldn't come close to the 150W limit.

3.7 Communications

In essence, CAPER's VRB operates on two fronts, creating streams of audio and corresponding movement data when prompted by the language model. The audio data is already synthesized by this point, and can be amplified through speakers to the user. However, the movement data is still unprocessed; it still needs to be sent to the IOCB to be converted from low voltage digital message streams, high voltage analog streams for the actuators. This requires the implementation of a "communication protocol" and "language" between the two boards. Respectively, these are the digital equivalents of modulation, and encoding of an analog signal. The "language" we choose contains the information regarding actuator position and activation, and the communication protocol is a means of transmitting that language over a medium. Speaking of which, multiple mediums are to be considered for CAPER; i.e. wired vs. wireless, single-cycle vs pipelined. etc.

3.7.1 Communication Protocol

The three most common methods of sending digital data in an embedded system are UART, SPI, and I2C; each of these are viable options with most general purpose MCUs. Looking at Figure 2.2, this section and chart refers to the signal being sent from the Voice Response Board into the MIDI Optocoupler and into the Microcontroller.

Table 3.18: Communication Protocol

	UART	I2C	SPI
Synchronization	Asynchronous; requires transmitter and receiver to operate on same baud rate.	Synchronous; controlled by master serial clock, shared with slave.	Synchronous; master clock signal sent to slaves on separate lines.
Duplex	Half or Full-duplex	Full-duplex	Full-duplex
Slaves per Master	2 transceiving devices	128 slaves for 7 bit addressing, 1024 for 10 bits.	2-3 slaves per master
Directionality	Bidirectional; Tx to Rx for both devices	Bidirectional, two serial lines	Bidirectional; MOSI to MISO between master and slave(s)
Wires per	1 per direction (2	Two serial lines;	7 lines going into

Channel	in total)	all slaves connected	master, 4 going into slaves, all interconnected
Speed	460kbps	100kbps	>100MHz
Error Detection	Parity bits (parrot bits in this case 😊)	ACK/NACK bit	All hardware-based

Between the VRB and IOCB, the requirements are relatively lenient; factors like speed, processing power, and throughput are less relevant than factors like size and compatibility. For this reason, we can confidently rule out SPI. Despite its impressive speed, it requires many more connections and components than the other two. Given the need to control CAPER's four movements, implementing SPI would be extremely costly and inefficient. I2C and UART seem more practical; both have error detection, sufficient bitrates, and only require 1 wire per direction. That being said, UART ultimately has the edge over I2C. The requirement for synchronization of the signal would require the IOCB to share the clock with the VRB. This isn't a dealbreaker, but it certainly makes UART more favorable. Directionality doesn't matter since communication goes from the VRB to the IOCB and not the other way around, and there is no need for full-duplex operation. I2C would become more favorable if we wanted to control multiple robots using the same language model, or if we wanted to use servo motors; but since neither of these apply to CAPER, UART is easily the most efficient, practical, and simple way to establish communication between boards.

Communication "Language"

Visualizing the communication protocol as digital modulation, it is clear we need a form of encoding the data before transmitting. There are eight possible messages that can be sent through to CAPER; ON and OFF signals for each of the FOUR actuators. Assuming we use UART, we can only send information as bytes, meaning no more than 8 or 9 bits per message. It is entirely possible to create our own language to do this using single byte messages. Certain bits within the byte can control certain movements inside CAPER using "bit masking". This would definitely be the most practical way of controlling CAPER with virtually no downsides except one; compatibility. By creating our own language, we limit CAPER's ability to interface with any and all existing technologies. The solution to this is to use an already-existing language that has been standardized and established as a legitimate communication method. A language that comes to mind as the PERFECT candidate for this is MIDI; Musical Instrument Digital Interface.

Existing for over 40 years now, MIDI has consistently been the most popular way to digitally transmit information for synthesizer and digital pianos. There are standardized plugs, cables, and message formats that all musical instrument companies adhere to even today; and the best part about MIDI, it's a form of UART communication. This means that we can encode our messages using the MIDI protocol for CAPER, and be

able to control the movements over UART just as originally intended, except now we'll be using a standardized language many companies are already familiar with.

The only drawback to this is the message size; MIDI messages are three bytes long, slowing down our communication speed by a factor of three. This isn't a huge problem, as MIDI encodes identification bits to maintain message synchronization, and UART bitrates are more than enough to accommodate for the larger message size. Given that is the only major downside, MIDI opens many new doors for us that more than make up for this issue. Installing a female MIDI plug on the IOCB allows us to use any ordinary USB to MIDI cable to connect to the VRB. Another great feature is that MIDI doesn't require any drivers to use; you just plug it in and start transmitting. Perhaps the best feature of using MIDI is the countless available softwares we can use to record MIDI sequences and play them back. Essentially a digital player piano, these softwares can transmit patterns of MIDI notes with perfect timing and repeatability. By encoding each of CAPER's movements to a single note on the keyboard, we can program these notes into the software and play it back externally into the IOCB. This should also give us the ability to plug a digital piano into the IOCB and control CAPER directly, without using a computer.

Communications Equipment Selection

Communication between the VRB and IOCB is established via a UART transmission sent over a wire. Unlike I2C and SPI, MIDI devices do not require complete synchronization between the devices. Set with the same baud rate, the transmitting end sends the signal asynchronously to the receiving end, without having to share the same clock signal. Using this communication protocol, the data for toggling the actuators will be sent using the MIDI language. Each message is 3 bytes, containing information regarding which actuator is being toggled, and which position it is in (resting or extended). To establish this connection, the signal will exit through a standard USB to midi cable. This cable converts MIDI IN and OUT plugs into a single USB type A plug. In the case of CAPER, only the OUT plug will be used, as the IOCB only receives information from upstream MIDI devices. This cable would be the only device present between the VRB and IOCB, and the IOCB will be equipped with a MIDI female IN port for this cable to plug into. Using this MIDI input, a similar connection could be established with a MIDI playback device. For this, a standard PC can be used, and the MIDI data would be played back on any readily available musical workstation software. One of the biggest benefits of this system is how no drivers are needed; MIDI starts working as soon as connections are established. There is no need to make any unnecessary installations on either the VRB or the playback device. This would also allow us to plug in MIDI enabled digital pianos directly into CAPER. Since MIDI is completely standardized, the IOCB would not be able to distinguish the devices transmitting the MIDI signal; the VRB, playback software, or digital piano would appear to be the exact same device.

Despite the many benefits to using the MIDI/UART communication, it forces our hand to include a peripheral device to ensure a stable connection between the transmitting device and the IOCB, this device is called an optoisolator circuit. Using an optocoupler

chip and a few passive components, the optoisolator establishes an electrical connection without physically connecting the two devices. This produces the exact same data signal, only using a voltage source local to the IOCB rather than the upstream device. Several of these optocoupler chips are available, and many are compatible with microcontroller applications. Here is the information on the chip *specifically designed to work with the MSP*:

Choice: H11L1 Microprocessor Compatible Schmitt Trigger Optocoupler

Features:

- 16V capacity
- Up to 1 MHz data rate
- Guaranteed on/off switching threshold (as seen in datasheet)
- 100ns switching delay
- Operating temperature up to 100 degrees Celsius
- 6 volt reverse voltage limit
- 3A peak forward current
- 50mA output current

Reasoning: Given the 5V, 31,250 baud rate of MIDI signals, this optocoupler chip should have absolutely no issues operating between the IOCB and upstream devices. As seen in the datasheet, the optocoupler is subject to irreversible damage if negative voltage is run across it; protective diodes are put in place to prevent this. This is a similar situation to the relays mentioned earlier, as they have optocoupler chips on them as well. Once all the safety measures are in place, the whole optoisolator circuit will work perfectly with any MCU chip. Another great feature from this circuit is the ability to change the voltage. The 5V signal that MIDI is transmitted at can be changed just by varying the Vcc rating supplied to the optocoupler. The high 5V wave can be dropped to a more friendly 3.3V signal before entering the chip. This may not be a necessary step depending on the chip used, but it is a good feature to know about.

With regards to the MIDI transmitting devices, the VRB will require extensive custom programming to implement MIDI functionality. Thankfully, this isn't the case for simple playback; there are many PC compatible programs available to run and process MIDI data in real time. Depending on the choice, many of these digital audio workstations (DAWs) allow for simultaneous playback of MIDI and audio recordings, allowing CAPER to act out a preprogrammed sequence of movements synchronized with the audio. Some of these DAWs are open source, some require a one-time payment, others have an unlimited free trial. With respect to CAPER, the choice of DAW is completely arbitrary; as long as it is MIDI capable, it is all users' preference. Here are a few good choices that would work well:

Table 3.19: MIDI Program Comparison

<p>Reaper: Regarded as one of the best DAWs out there for music production, Reaper covers all the bases to get this project off the ground</p>	<p>Features:</p> <ul style="list-style-type: none"> • 64 bit audio processing • 128 channel MIDI capacity • Hundreds of MIDI after-effects • \$60 non-commercial license price • Compatible with all major plugins VST, VST3, etc.
<p>Studio One: Another great choice for audio editing, Presonus Studio One allows for intensive recording and editing of both MIDI and audio files.</p>	<p>Features:</p> <ul style="list-style-type: none"> • Easy to understand GUI • Built in “Beat Maker” MIDI tool • Compatible with all major plugins VST, VST3, etc. • One time license fee of \$99 for basic edition
<p>Waveform Free: The incredible features the other DAWs boast are relatively moot, as this project relies solely on MIDI capability. A completely open-source option should be able to get the job done just as well; Waveform Free is a fantastic example. The 12th edition of this software allows for multitrack audio and MIDI mixing; with an easy to understand GUI and window layout</p>	<p>Features:</p> <ul style="list-style-type: none"> • 3rd party plugin compatibility • Onboard MIDI libraries and editing tools • Allows for optional expansion packs @ \$50 each

Choice: Waveform Free

Reasoning: Given the scope of CAPER, there is no need to purchase any of the expansion packs for MIDI, nor audio editing. The fully free version of this software contains all of the vital features necessary; while perhaps not as powerful as the other paid options, it shouldn’t make a difference. This is easily the most favorable option out of the lot.

Alternatively: Using LMMS & Audacity together

Both of these DAWs are similarly, yet conversely disadvantaged, as they’re both only half-way capable of operating CAPER. Audacity allows for audio recording, but not MIDI. Linux Multimedia Studio (LMMS) allows for MIDI recording, but not audio. The compromise here is that Audacity still allows importing of already created MIDI files; meaning the audio can be recorded in Audacity, the MIDI can be sequenced in LMMS, then you can export the MIDI from LMMS and into audacity for simultaneous playback.

Of course, the Waveform Free Edition is more practical, it is still a good idea to have an alternate route. Despite the more tedious procedure, LMMS & Audacity used together should still be able to run CAPER at no cost.

3.8 AI Technology

This section explores CAPER's inbuilt conversational AI system. This section will not focus on hardware and instead try to establish which blend machine learning framework, models, and data pipeline architectures best fit the project's needs.

3.8.1 Deep Learning Framework

Deep learning frameworks are suites of software packages that streamline development, training, and inference processes of deep learning systems. As of time of writing, possible frameworks for CAPER include:

TensorFlow:

Developed by the Google Brain team, and released in 2015, TensorFlow is an open-source library for numerical computation and machine learning. It is renowned for its flexibility, comprehensive ecosystem with tools for researchers and developers, and strong support for production deployment. It offers both a low-level API for flexibility and a high-level API (Keras) for ease of use.

Models developed with this framework can be adapted to run on restricted computational resources using the TensorFlow Lite (TFLite) and TensorFlow Micro (TFLite Micro) toolkits.

PyTorch:

Developed by Facebook's AI Research lab (FAIR), and released in 2016, PyTorch is an open-source machine learning library based on the Torch software library. Its popularity for academic research and production software is a direct consequence of its ease of use, flexibility, and dynamic computational graph which allows for the dynamic memoization of states. The last feature being of particular importance for CAPER due to the need for embedded AI development.

The framework offers excellent GPU acceleration support and has, by all metrics, the best user experience when it comes to development and training of models. Models developed with this framework can be adapted to run on restricted computational resources using the PyTorch Mobile toolkit.

Microsoft Cognitive Toolkit (MCT):

Developed by Microsoft and released in 2016, the Microsoft Cognitive Toolkit is another respectable open-source deep-learning framework. Although its main design philosophy centers around making it easy to scale across multiple GPU-enabled servers, which is not the use case for CAPER, its support for Python and C++, and its focus on ease of training, make it an option worthy of examination.

Table 3.20: Features comparison

	Tensorflow	Pytorch	MCT
Has toolkit for embedded conversion	✓	✓	✗
Supports CUDA	✓	✓	✓
Supports CPU	✓	✓	✓
Supports OpenCL	✓	✓	✗
Supports OpenGL	✗	✓	✗
Best Ease of Use	✗	✓	✗

Although TensorFlow has more libraries for running its modules in constrained hardware resources scenarios, Pytorch's ease of use and non-negligible amount of edge hardware translation libraries make it the best choice of framework for CAPER's needs.

3.8.2 Model Architectures and Selection

The architecture of a model refers to how data within is structured, and how it uses said data to make predictions. Although this description implies a fair bit of flexibility when picking architectures for the various models needed for CAPER, a few types have already established themselves as best-in-class. This selection is what this subsection will be comparing.

As clarification, the general from of the CAPER conversational data pipeline looks like this:



Diagram Figure 3.6: CAPER's conversational data pipeline

Legend:

- **ASR:** Automatic Speech Recognition system. It turns audio data into more useful forms like text
- **LLM OR SLM:** Large Language Model or Small Language Model. These systems will take the text data and create a response in the form of text.
- **TTS:** Text To Speech. This system will take the text and create an audio signal corresponding to it.

Given that audio data is sequential by nature, this eliminates systems like regression or decision trees.

Here is a high-level breakdown of the systems that have the best shot at fulfilling CAPER's needs.

Recurrent Neural Networks (RNN)

- Function: This type of neural network was developed to process sequences of data by maintaining a 'memory' of previous inputs through internal states.
- Pros: Good for data that has temporal dependencies like text and speech data
- Cons: Forgets context in long-term dependencies scenarios
- Project Relevance (Expected Performance):
 - ASR (Good)
 - LLM/SLM (Mediocre)
 - TTS (Unproven)

Transformer-Based Systems (State-of-the-art)

- Function: This architecture was developed to make use of self-attention, allowing it to focus on the most important part of its inputs.
- Pros: Good for almost all types of data, as long as it can be vectorized.
- Cons: Very high computational cost for both training and inference.
- Project Relevance (Expected Performance):
 - ASR (Excellent)
 - LLM/SLM (Excellent)
 - TTS (Excellent)

Hidden Markov Models (HMMs)

- Function: This architecture was developed to make use of how hidden state-based Markov processes can predict sequential data.
- Pros: Effective when in sequential data scenarios.
- Cons: Performance degrades very quickly as input dimensionality increases.
- Project Relevance (Expected Performance):
 - ASR (Decent)
 - LLM/SLM (Null)
 - TTS (Null)

Long-Short Term Memory (LSTM)

- Function: This architecture takes the functionality of RNNs and makes them better at keeping track of long-term context within any given input
- Pros: Creates long-term dependencies making it better in applications where sequential input is larger.
- Cons:
 - Computationally expensive to train
 - Slower to train than transformers
- Project Relevance (Expected Performance):
 - ASR (Very good)

- LLM/SLM (Good)
- TTS (Very good)

Generative Adversarial Networks (GANs)

- Function: This architecture makes simultaneous use of two systems, the generator, which creates the output, and the discriminator, which rates the output, informing the generator how to adjust its next output for a better score.
- Pros: Capable of generating high quality output
- Cons: Unstable training
- Project Relevance (Expected Performance):
 - ASR (Decent)
 - LLM/SLM (Decent)
 - TTS (Good)

Convolutional Neural Networks (CNNs)

- Function: This architecture makes use of convolution to make input data easier to digest.
- Pros: Very capable in feature extraction tasks.
- Cons: Specialized for grid-like data, requires adaptation for use in the audio and text domain.
- Project Relevance (Expected Performance):
 - ASR (Good)
 - LLM/SLM (Null)
 - TTS (Good)

Although the transformer architecture is the best when it comes to quality and flexibility, CAPER does not have an unlimited computational budget. This table highlights the possible choices for each part of the pipeline.

Table 3.21: ASR vs LLM/SLM vs TTS

Pipeline Module	Criteria	Model Type
ASR	Highest Quality	Transformer
	Cheapest	HMM
	Best Cost-to-Performance	CNN for feat. extraction and transformer for modeling
LLM or SLM	Highest Quality	Transformer
	Cheapest	LSTM
	Best Cost-to-Performance	Transformer + LSTM

TTS	Highest Quality	Transformer
	Cheapest	GAN
	Best Cost-to-Performance	Transformer

CAPER's conversational system needs to give high quality responses, both in terms of the vocalization and response content. The best choices for each module based on this requirement are as follows: ASR, LLM, TTS.

ASR:

The best choice here is to use systems that make use of CNNs on the mel spectrogram of the audio to extract the features and make use of a transformer to classify each feature across time, creating the text output. As of time of writing, more research needs to be done to determine if the gradients can be fed into the response generator directly. Based on the team's testing, two models stand out when it comes to filling all the needs of CAPER's pipeline while fitting these criteria: Facebook's Wav2Vec2-Large-960h (Pytorch) and OpenAI's Whisper-Small. At first glance these models seem very similar as they both utilize the CNN for feature extraction from a mel spectrogram, they both utilize transformers as the basis for their language models (LMs), and are both multilingual. They do have differences, however.

Release Date: Wav2Vec was released by Facebook's AI Research (FAIR) team in September 2020, whereas OpenAI's Whisper-Small was released two years later in September 2022. This translates to Wav2Vec24-Large-960h having been used in a greater variety of environments leading to more data about its performance and fine-tuning.

Size on Disk: With its 244 million parameters Whisper-Small is slightly smaller than Wav2Vec24-Large-960h and its 317 million parameters despite being more recent. In terms of model storage this translates to a size on disk of 483.6MB for Whisper-Small's .pt file and 1262MB for Wav2Vec24-Large-960h along with all its supporting architecture. Compression is possible for both models to reduce storage as needed.

Size During Compute: With full 32 bit tensor precision, Whisper-Small requires 0.976GB of working memory for full function. In the same scenario, Wav2Vec2-Large-960h requires 1.484GB.

Performance: Wav2Vec24-Large-960h and Whisper-Small both have similar scores when it comes to their respective WERs (Word-Error-Rates). Whisper-Small has an error rate on Librispeech-clean of 3.54%. Wav2Vec2-Large-960h has an average WER of 2.925% across all tasks but beats out Whisper-Small in Librispeech clean at just 1.7%

The best decision for CAPER's voice recognition pipeline comes down to which of these models represents the best performance for the least amount of compute needed,

which will help minimize computation hardware cost and make the system as whole more energy efficient. Although Wav2Vec2-Large-960h holds the highest accuracy, even when accounting for dirty and noisy datasets when compared to Whisper-Small, it's high cost on disk and high compute cost do alot to sour that victory.

Table 3.22: tradeoff

Name/Value	Whisper-Small	Wav2Vec2-Large-960h
Size on disk (MB) [Minimize]	483.6	1262
Size during compute (GB) [Minimize]	0.976	1.484
WER (%) [Minimize]	3.54 (clean+dirty)	1.7 (clean) and 2.925 (clean+dirty)

Why

Whisper-Small is the best option:

- 291% smaller on disk
- 152% smaller during compute
- only 20% worse WER

LLM

The best choice here by far is to use chained pre-trained transformer blocks in a GPT configuration. Although using an LSTM to offload work from the transformer saves in compute, the added complexity to the pipeline offsets that advantage as data conversion will be necessary when it is swapped between architectures. GPT-based LLMs are the all-around best when it comes to text completion, making them a must when it comes to how lifelike CAPER seems.

The GPT LLM is designated to function as the central processing unit within CAPER's voice response pipeline. This system type, defined by its substantial parameter count and advanced deep learning frameworks, is engineered to accurately interpret and generate human language nuances. Its design facilitates a broad spectrum of functionalities, encompassing text synthesis, comprehension, and linguistic translation, thereby serving as a versatile core for CAPER's interactive capabilities.

Taking the choice to offload the system to a separate server makes it possible to run these computationally expensive systems without the need for extensive optimization. That does not make them free, however. Due to the nature of AI inference, the second best option to reduce CAPER's upfront and ongoing costs, aside from leveraging servers with an optimized speed-to-cost ratio, is to pick the smallest LLMs that still have reasonable performance in conversational settings.

As of time of writing, comparing LLMs is still in its infancy, meaning that accuracy of the metrics used to make our choices is not guaranteed. The team decided to use the Massive Multi-task Language Understanding (MMLU) score of a model as a benchmark to compare them by. The Massive Multi-task Language Understanding (MMLU) is a continually updated dataset whose main objective is to test the capabilities of LLMs in understanding and generating human-like responses across a diverse set of tasks.

The models under consideration for deployment include Mistral 7B Instruct, launched by Mistral in 2023, Meta's Llama 7B and Llama 2 7B, also introduced in 2023, OpenAI's GPT 3.5 Turbo. The 7B models are the only ones here that can be executed on self-hosted hardware, owing to their open-source code and model files. In contrast GPT 3.5 turbo is a colossal (175B parameters) closed source model. Its inclusion in the selection process is attributed to its cost-effectiveness for applications on externally managed servers, an approach not directly implemented by the team. Recall, however, that self-hosted server offloading was chosen due to it being a learning experience for this senior design project.

All of the open source models will be considered in their 4-bit quantized states. 4-bit quantization is the practice of reducing the size of the parameters that the model consists of, which, while having a very small impact on performance, will not cause any issues with the conversational abilities of said models. Due to their similar parameter count, all choices listed, besides GPT3.5 Turbo, have a similar on-disk size of ~13.5GB and a compute footprint of ~7GB, all results based on testing on local consumer hardware. One last thing to consider when choosing the model is the context size, a metric that defines how many word fragments the model can hold in its working memory. An increase in this metric means an increase in conversational retention and a better experience for the user. The table below compares all options based on the above listed criteria:

Table 3.23: Model comparison

Model Name	Size During Compute 4-bit quantized (GB)	Context size (tokens)	MMLU Score (%)	Open-Source
Mistral 7B Instruct	7GB	8192*	60.1	✓
Meta's Llama 2 7B	7GB	4096	45.3	✓
Meta's Llama 7B	7GB	4096	35.1	✓
OpenAI's ChatGPT 3.5 Turbo	N/A	16384	70.0	✗

Note: *Instruct models require some of their context window be taken up by a large system prompt, reducing amount used in conversation*

The best choice for this project is Mistral 7B due to its large context window, despite the need for a system prompt, and its high benchmark performance.

TTS:

This choice is rather difficult. Unlike ASRs or LLM/SLMs, the computational expense of the cheapest fully intelligible TTS vs the most expensive one varies by two orders of magnitude. As a result, the TTS will be implemented after all the other programs have had their necessary compute allocated. If there is a good amount remaining, a transformer-based TTS will be used, else, a simple voice synthesis engine will be used.

Transformer based TTS (Text-To-Speech) models come in the same level of variety as ASRs and LLMs. The options at our disposal are: Coqui's YourTTS released in 2023, SpeedySpeech released in 2020, and NVIDIA's Tacotron 2 + WaveGlow Vocoder. Unlike in previous sections, the amount of information related to each model is limited and exact parameter count for YourTTS and SpeedySpeech could not be found.

YourTTS:

- **Background:** This system was developed by the Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Sopra Banking Software, Defined.ai, Federal University of Technology, and Coqui; released in 2023
- **Architecture:** SOTA Transformer blocks.
- **Features:**
 - Voice replication
- **Pros:**
 - Voice replication and theoretically quality
- **Cons:**
 - No listed parameter count testing required

SpeedySpeech:

- **Background:** This system was developed by the Charles University; released in 2020
- **Architecture:** 1D Convolution-based blocks
- **Pros:**
 - Theoretically lower compute cost
- **Cons:**
 - No listed parameter count testing required

TacoTron2 +WaveGlow Vocoder:

- **Background:** Both TacoTron 2 and WaveGlow were created by the NVIDIA corporation; released in 2017 and 2018 respectively
- **TacoTron2 Architecture:** 1D Convolution-based blocks

WaveGlow Architecture: 1D Convolution-based blocks

- **Pros:**
 - Theoretically much lower compute cost
 - Active maintenance from developers
- **Cons:**
 - Worse fidelity
 - Older models

Based on these comparisons, the best choice is SpeedySpeech due to its theoretically higher speed and good balance between running cost and quality.

3.9 Storage

This section explores both the hardware and software choices regarding the storage of data for proper CAPER function. Here is a breakdown of the possible technologies for storing the voice response pipeline data, which, even at this stage of the design, is expected to be around 10 to 20 GB.

SD Card Storage:

- **Function:** Removable flash storage device used for storing data.
- **Pros:**
 - Easily swappable
 - widely compatible
 - Capacities up to several TB.
- **Cons:**
 - Slower compared to soldered storage options
 - High vulnerability to physical damage.

Soldered eMMC:

- **Function:** Embedded storage solution providing flash storage directly soldered onto the device's PCB.
- **Pros:**
 - Faster and more reliable than SD cards
 - Low cost per GB.
- **Cons:**
 - Limited maximum storage capacity
 - Not user-upgradable.

Soldered UFS:

- **Function:** Advanced flash storage technology soldered onto the device's PCB, offering higher data transfer speeds.
- **Pros:**
 - Faster data transfer rates compared to eMMC
 - Improved reliability and performance.
- **Cons:**
 - More expensive than eMMC per GB

- Requires newer hardware interfaces
- Not user-upgradable

Each storage method presents its own challenges and advantages when it comes to CAPER's systems. Given the current storage needs, estimate all options meet the data storage requirements at a reasonable price point. The only remaining concern, and the reason for the final choice, is the ease of use. Although soldered options present better cost per GB and higher speed for a comparable cost to SD card storage, it is the former that gives the best user experience when it comes to ease of use. SD card storage only needs the SD card slot soldered on, and allows for the easiest storage upgrades. All these factors lead us to pick SD card storage as our method of choice for storing bulk data like model parameters and our OS.

Storage Device Selection

Considering the storage for the firmware will be local to the selected chip. The only external storage devices needed are the SD cards needed to store the models that will be run locally.

All options are compared in this table:

Table 3.24 Storage Device Comparison

Device	Capacity (GB)	Price (\$)	Read/Write Speed (MB/s)
Lexar 128GB Micro SD Card, microSDXC	128	14.41	100/30
SanDisk Ultra 32GB UHS-I/Class 10 Micro SDHC Memory Card	32	9.26	100/48
SAMSUNG EVO Select Micro SD-Memory-Card	256	22.99	130/130

Based on this comparison we can comfortably pick the Samsung Evo Select option due to its higher rated speeds and greater capacity while only being \$8.58 more.

Chapter 4.0: Design Constraints and Standards

This section discusses design constraints and standards relative to CAPER.

4.1 Standards

Standards are an integral part of engineering design and provide crucial guidelines for the development and production of many technical components. Following standards can help increase safety, efficiency, productivity, and reliability in any projects taken on in the engineering field. Many standards are typically voluntary, but choosing to follow them can have direct impacts on the success of your project and its replicability. Without a set way of going about a problem, any improvements or replications could prove rather difficult, as well as the general understanding behind your design can be lost or not reach as large an audience. Following standards typically streamlines the production process and makes technologies much easier to adopt worldwide.

Standards are just as diverse as they are plentiful, as they can encompass many different professions and classifications. The American National Standards Institute (ANSI) categorizes standards into five main categories: voluntary, de facto, consortia, regulatory, and corporate standards. Voluntary standards, as implied by the name, are optional standards that typically pertain to performance-based, certification, and management system standards. De facto standards are standards based on widely used practices within the industry, for example, all keyboards following the layout of QWERTY. Other standards include Consortia standards, which aim to allow companies to work together to pool resources and limit participation. Regulatory standards ensure uniformity and efficiency and can include permits, licenses, certifications, etc. Lastly, we have corporate standards, which are specific to the company and help facilitate internal organization and communication.

In the context of our project, following standards will help us cut down on issues, maximize performance, and create a replicable project. From communication protocols between varying devices and Printed Circuit Board standards, following these rules can increase the desirability and ease of construction for our parrot. As well, complying with safety standards can minimize risks and ensure we follow the regulatory constraints. By incorporating these established and accepted standards into our design and selection process, we will be able to create a more reliable and standardized product that is in line with industry benchmarks for safety, quality, and performance.

4.1.1 IPC-2221: The Standard for Printed Circuit Board Design

This Standard serves the purpose of providing information on generic requirements meant to aid in the design of Printed Circuit Boards. This standard remains rather broad to be able to be applied to a broad spectrum of designs that can use a span of materials (metal, glass, ceramics, etc.) to provide the structure for mounting and combining electronic, electromechanical, and mechanical components. This standard itself uses lots of documentation from other standards available on IPC's website aimed at targeting more specific areas of PCB and component design. Variations of this standard are readily available once the user has decided what material to use that contains a

deeper look into that specific technology. The hierarchy for this set of specifications (2220-2229) can be found below.

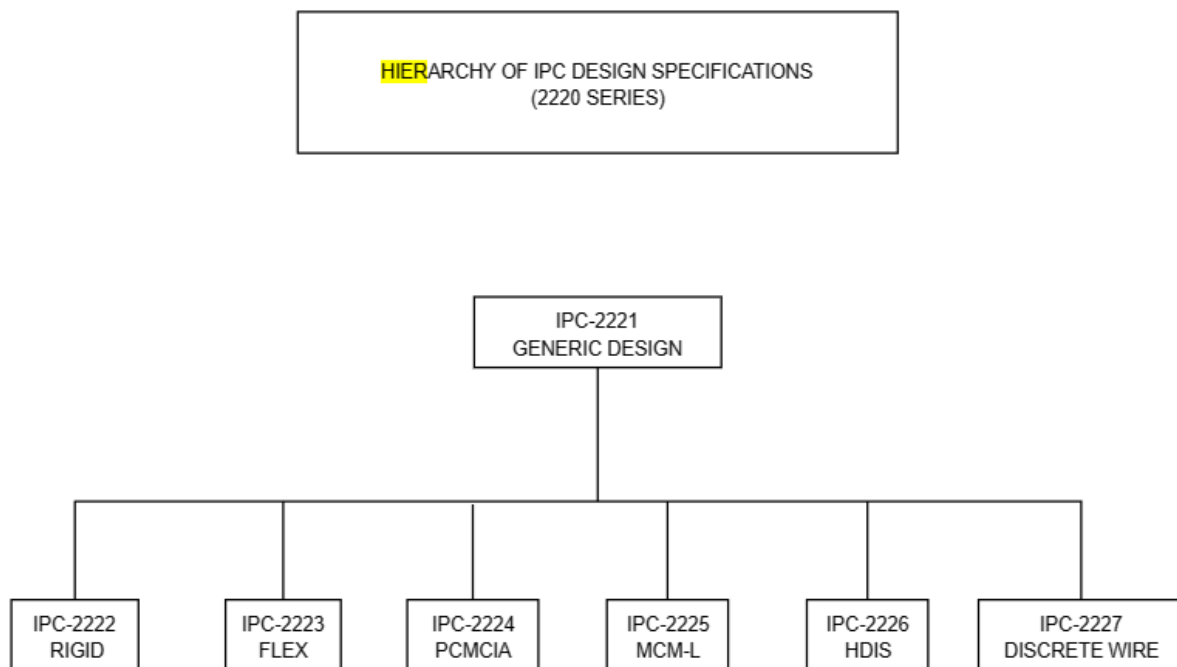


Figure 4.1 Hierarchy of IPC Design Specifications

Following PCB standards is critical for ensuring the reliability and performance of the C.A.P.E.R project. By following established guidelines, we can mitigate risks, optimize functionality, and streamline the development process. By following established practices and recommendations our project will be replicable and up to industry standard.

This standard contains many subsections, many of which will be applicable to our project. Material selection will assist us in meeting our project requirements as considering factors such as mechanical strength and conductivity can increase our project's durability and compatibility with the parrot's operational environment. Material and component selection directly affects our ability to reach our desired constraints and our project's efficiency goals. Board size is also a factor we must pay close attention to, as it must fit within our parrot yet still must be large enough to reach our performance goals. As well at this point in our project planning, we are unsure if we will be using one or multiple boards, so placement is of large importance.

IPC-2221 continues to touch on major considerations we must make when designing our PCB. Considering the proximity of all our components/boards within such a small space, Electrical and Thermal properties must be closely considered. Following the sections on Interconnections and Assembly issues can help us avoid errors and ensure electrical continuity. Lastly, keeping comprehensive documentation throughout this

entire process is incredibly important to the traceability, troubleshooting, and future revisions of our board.

IPC-2221 is an overarching document that covers various standards when it comes to designing a Printed Circuit Board. It is material-dependent and covers many different designs, however, will allow us to shrink down the standards applicable to us once we have completed research and chosen parts for our project going forward and even assist in helping us choose components before any construction. By following IPC-2221, we can avoid setbacks and malfunctions when designing our PCB while maximizing efficiency and retaining our traceability.

4.1.2 ISO/IEC 30122-2:2017 – Information Technology, User Interfaces, & Voice Commands

ISO (The International Organization for Standardization) and IEC (The International Electrotechnical Commission) came together to form a worldwide standardization for projects/items utilizing voice commands and responses. Our project will be utilizing voice recognition and the ability to process this speech and reply accurately, so this standard directly affects said project. The standard outlines the importance of accuracy when recognizing voice commands and provides testing criteria to ensure accuracy in speech recognition programs. Important aspects of a standardized speech recognition program include being able to differentiate between similar words, understanding different accents, and recognizing the difference between plurals. It states that any specific action should have an easy-to-pronounce command. Lastly, this standard outlines a test that effectively standardizes your voice-responsive project. This includes using a test group of individuals from different accents and ethnicities to ensure the project is capable of working no matter the user's background. The test also includes unexpected commands, which given that our project will be able to respond at will, is something we will also include. We have listed one of our constraints as our project having an accurate response given the command given to it, which will make adhering to this standard important for the long-term function of the CAPER project as a proper transmission of the command is needed to form a proper response.

4.1.3 IPC J-STD-001 Standard Soldering Requirements

IPC-J-STD-001 is a standard developed by the Association of Connecting Electronics Industries (IPC) that aims to assist in electronic assembly manufacturing. It helps to set requirements for materials, fabrication, and quality assurance to ensure that all electronic assemblies are reliable and perform as desired. This standard is regularly updated to keep up with evolving industry practices and improving quality standards.

This standard emphasizes material selection and goes into depth about criteria for solders and cleaning agents. It also provides guidelines for inspecting your work, workmanship standards, and quality assurance measures. Such inspecting guidelines include solder fillet height and component alignment. The cleanliness of soldered equipment and following the manufacturer's instructions on heating/cooling rates is of the utmost importance. Quality soldering practices and instructions for any member

involved in the Soldering process can be found in the standards documentation. The illustration below displays the general difference between good and poor soldering:

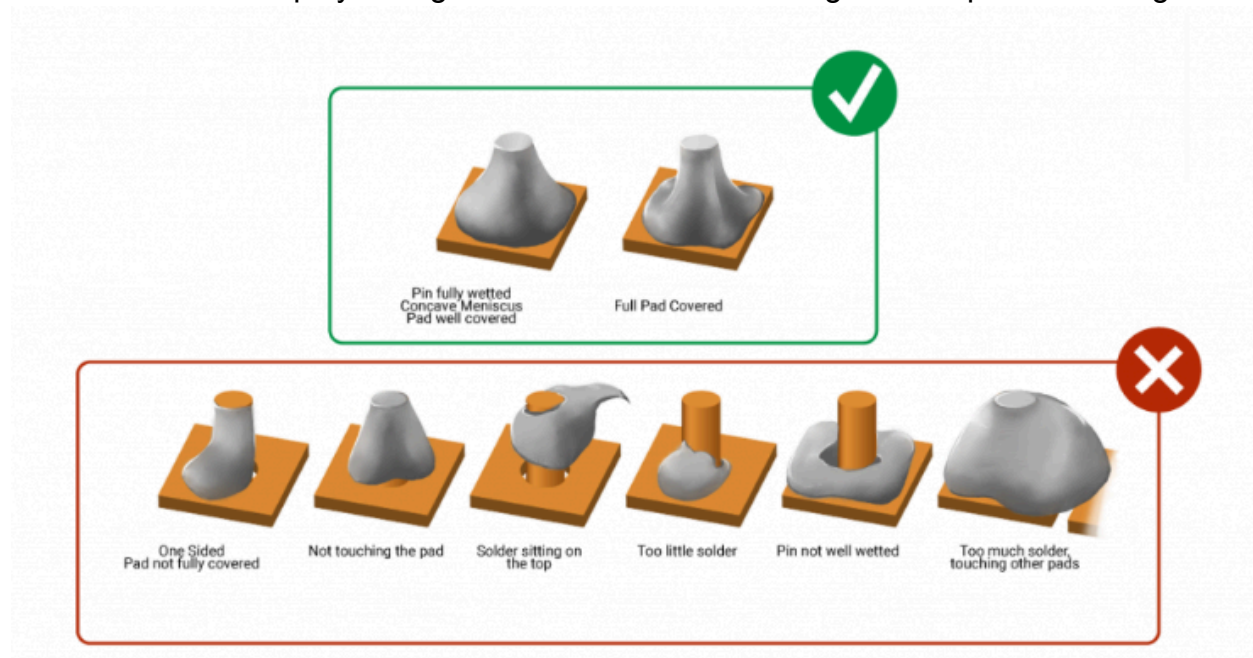


Figure 4.2: Soldering Standards

4.1.4 ISO/IEC 42001 – Information Technology, Artificial Intelligence, & Management System

ISO (The International Organization for Standardization) and IEC (The International Electrotechnical Commission) again came together with the goal of standardizing Artificial Intelligence within a variety of programs. As an increasingly popular technology, Artificial Intelligence (AI) is likely to continue to increase as a growing economic force. However, many in the public are afraid of the rapid growth and possibilities presented by Artificial Intelligence without regulation. This goes hand in hand with our project, in that our goal is to make AI seem more human and natural and show off the positive aspects and helpful contributions it can make. This document is very broad but aims to aid organizations in utilizing, developing, and monitoring AI-based products and services.

One of the most impressive properties of AI remains one of its biggest reasons to be feared, which lies in its decision-making ability due to machine learning. Machine learning allows a program to learn and change the bound and development it was initially given, sometimes in the wrong direction. It takes a special kind of management to be able to overview and restrict this to keep the project on course. This document lays out requirements for a management system to keep the project from deviating too far from its original purpose. Key objectives include clear goals, a defined structure, and a clear understanding of the goals of the program.

This standard is very recent (2023) and admittedly does not have all the answers at the moment. They encourage organizations to make use of accepted frameworks and previous standards, along with changing field practices, to fully utilize this technology. AI

regulation has been a controversial topic in the public lately with many prominent figures calling for more rules and regulations. The end goal is to find that fine line between responsible and ethical contributions while allowing for innovation and freedom.

Our program will be utilizing AI in the shape of an ASR (Automatic Speech Recognition system), LLM (Large language model), and TTS (Text-to-speech program). It is our responsibility to follow the standards listed to ensure our program understands what we are asking of it and responds in that fashion. The goal of our project is to react realistically and humanely via carrying out conversation with our AI feature, which means we must use AI responsibly to ensure our project acts ethically and reliably. Considering our bird will be regularly conversing with the public, it is of the utmost importance we clearly define its responses with a main goal and rules to ensure it does not speak out of turn, as that would be an ethics issue. The document also touches on the fact that implementing AI shouldn't alter previous processes and structures. To follow this, we will be able to run the mobility function of our bird independent of the AI, as well as not changing the physical look of the bird. Our software must accurately understand the words spoken to it and properly respond to said input to be a successful AI model and uphold this standard.

4.1.5 IEEE 801.11 – WIFI Standards

IEEE 802.11 is a set of WiFi standards designated by the IEEE. The standard and amendments provide the basis for wireless network products using the Wi-Fi brand and are the world's most widely used wireless computer networking standards. This standard is prevalent in almost every home and office network to allow various devices with each other and connect to the internet without connecting wires. This standard is often revised with the newest version of the standard being what the market aligns its goals with. This standard also delves into bandwidth, modulation techniques, and the range necessary to utilize these networks. This standard utilizes various frequencies including 2.4 GHz, 5 GHz, 6 GHz, and 60 GHz. The most commonly used generation of Wi-Fi is currently generation 5 (802.11ac), while generation 6 is still being rolled out. Generation 6 reduces interference and increases available network capacity, however isn't widely used yet. Generation 7 is in the final stages of development and is likely to be implemented in late 2024. With our leading option for the voice recognition feature of our board being server-based, Wi-Fi implementation is an important feature to C.A.P.E.R.

4.1.6 Table of Summary of Standards

The Summary of Standards (Table #) serves as a basic summary of the standards that we dove into more in-depth above, and specifically outlines where and how they'll be used within our specific project. By observing the table above, we can see that the majority of our standards lie in the construction of the board, and ensuring the correctness of our voice response function. Considering our main goal is to create a realistic prototype, both of these sections are vital to our project. Following these standards has two significant benefits in regard to our project. The first is that it will allow us to more easily meet our constraints and increase the overall efficiency of our

project. Second hand, it will ensure the functionality of C.A.P.E.R. and increase the net safety and durability of the project. The most important standard listed is the PCB design rules, as most of the other rules are just a combination of rules for different parts of the project, while the PCB design is vital to all other functions and is a large focus of our group. Here is a summary of those standards:

Table 4.1 : Summary of Standards

IPC-2221: The Standard for Printed Circuit Board Design	<p>Created by IPC.</p> <p>General standard that encompasses all basic PCB design processes.</p> <p>Delves into topics such as material selection, placement, conductivity, etc.</p> <p>Contains many subsections for specialized cases.</p> <p>Will be followed in regards to our PCB design</p>
ISO/IEC 30122-2:2017 – IT, User interfaces, and Voice commands	<p>Issued by ISO&IEC</p> <p>Specific methods of response and voice commands.</p> <p>Specifies how to test voice commands and responses of projects.</p> <p>Includes requirements specifying inclusion and edge cases.</p> <p>Lists test phrases and test requirements for different voices/phrases.</p> <p>Will be used in testing parrots response efficiency</p>
IPC J-STD-001 Standard Soldering Requirements	<p>Issued by IPC.</p> <p>Defines recommended materials and quality of said materials.</p> <p>Explains soldering methods.</p> <p>Notes common Soldering errors and visual representations of said errors.</p> <p>Will be used in board design</p>

ISO/IEC 42001-IT, AI, Management system	<p>Issued by ISO&IEC.</p> <p>Insists on the importance of defining clear goals for your project to ensure it's operating within its bounds.</p> <p>Recommends test cases to ensure the response is as expected per given input.</p> <p>Will be used to ensure accuracy of parrots voice recognition capabilities</p>
IEEE 802.11 - Wi-Fi Standards	<p>Created by IEEE.</p> <p>Allows all devices to connect to Wi-Fi and interact with each other.</p> <p>Specifies range, modulating technique, bandwidth, etc.</p> <p>Regularly replaced when a new generation of wifi becomes adopted.</p> <p>Shows specifications for every generation ever produced and their operating ranges.</p> <p>Will be used to connect to the server to run voice response functions.</p>

4.2 Constraints

Project constraints are limiting factors for projects that can impact quality, delivery, and overall project success. Understanding the practical design restrictions that must be followed is another element that makes a design successful. Recognizing the constraints early on helps in setting realistic goals, timelines, and budgets. Because projects often rarely go as planned, understanding these constraints provides us flexibility on the chance something goes wrong. In this section of the post, we will address several potential roadblocks that may arise when attempting to use the product. There are many constraints to consider, and we have chosen to focus on economics, time, durability, input modes, environmental, social, political, and health & safety, with emphasis on economics and time. These are relevant constraints that are important to discuss.

4.2.1 Economic Constraints

A difficult challenge in particular to take note of resides in costs and funding. To start with, this robot is solely funded by the team members since this is unsponsored. This project relies on the financial contributions of the team to bring this parrot to life, which

can be difficult considering the expenses and that every member is a working college student. We must secure enough funds and manage our budget efficiently. While we have our budget at 1300 dollars, we would much rather prefer to stay much lower than that amount. The ultimate goal would be under 1000 dollars. As much as we would like to make the robot as cheap as possible, we do not want to skimp on quality. Our team is striving to minimize expenses without compromising on the scope of work.

Because our team plans on diving deep into the more intricate software capabilities of CAPER with its lifelike behaviors such as legible speech quality, conversational responsiveness, and response-relevant movement commands such as beak movement while speaking, we will likely end up opting for a pricier, but higher quality voice response board than originally planned in order to handle the AI and software computations.

Our main investment is in the voice response board. We must invest in this board since it is a critical part of bringing life to the robot with motions and vocal capabilities. The least expensive option would be the NVIDIA Jetson Nano developer kit at about 150 dollars on Amazon as of March 2024. As much as we would like to go the cheapest route, it is unlikely this is feasible due to the board not having enough power to produce the results we want when it comes to running the LLM and similar softwares. It is simply improbable that this option would suffice for our project's requirements. However, a way to go around that is to implement cloud computing so that the heavy computation can be offloaded on a separate server.

When comparing this to our most expensive option, the NVIDIA Jetson AGX Orin developer kit at about 2000 dollars. The price spike from the least to the most expensive is astronomical as we have to consider what we truly need for the scope of this project. The middle-ground approach would be most viable as it is most able to account for both quality and costs to stay within budget while also supporting machine learning. The prices of these microcomputers depend on the type of GPU, size of memory, number of ports, power supply, and more. However, we are considering using a wifi adapter to offload heavy computation to a more powerful system. This detailed analysis will aid in reaching a decision that aligns with our project's goals and our wallet's happiness.

Other economic constraints that we encountered, though are not as heavily impacting our wallets individually, are other parts of the robot. By each component, it does not seem too bad with the internal skeleton of the bird costing around 30 dollars, the power supply costing 20 dollars, and so on, but when considering the amount of components that go into this project, everything very quickly adds up. Everything added together outside of the voice response boards brings the cost to around 350 dollars alone. Now, considering the middle-range cost of the voice response board sitting at around 500 dollars, if we were to go down the middle of the road for every component, we would be sitting at a pretty 850 dollars- well below the goal budget. If we were to go with the most expensive option, our bird would likely be well over 3000 dollars; way over the allowed budget, considering the most expensive Jerson board itself is 2000 dollars.

Another important investment is in the durability of the robot. The material used to build CAPER must be sturdy enough to withstand prolonged use and therefore must be a higher quality material such as wood, 3D printed plastic, or metal. Every electrical component such as wires, buttons, switches, microcontroller boards, and more will have multiple copies of each. This is necessary because the quality of these components is not the best in the market, so it is probable that some components come defective. There is also the issue of accidentally destroying or frying components during the building and experimental phase. This again, adds up. However, since these components are so cheap, we likely will not hit over \$100 in total.

Another economic constraint we encountered was the microphone selection. Although these parts are more reasonable in price, between 10 and 200 dollars, the range is still quite high and needs a more thorough look at exactly what we need and what we can get away with in terms of price and quality. The average wireless microphone averages at 100 dollars. A good omnidirectional microphone is necessary for picking up full, clear, and easily understandable speech to reduce response delay from the parrot. It is important so that the program can pick up clear, comprehensible speech. We could just as easily go the cheapest route at 10 dollars off of Amazon, but the chances of the mic not working properly or picking up every single word increases significantly. The constraint regarding this microphone is that the choice will be made based on what is leftover of the budget. It is likely we will end up choosing something around the 20 dollar range. Despite facing budget constraints, we are actively finding the necessary pieces required to successfully create a well-functioning parrot. Although we cannot use the best items on the market, we can source good enough components to work with the robot. Overall, we feel confident enough to fulfill our end goal: to have a fully functioning, voice-responsive life-like parrot with basic head, beak, and body movements.

4.2.2 Time Constraints

Time management is an essential life skill in all aspects of existing, especially when major projects are involved. How effectively the time is organized is essential to the success of completing this project. In order to truly be successful, we understand that every member must set aside an ample amount of time to work on the project both individually and together. If even one person falls behind, not only will it stress the other members out, but it could jeopardize the whole project, cause us to drop a letter grade, or worst of all, we could even fail the semester completely. That is why it is so important to work together as a team and organize our time effectively to achieve the best results possible. Collaboration with an emphasis on clear communication and task allocation is a requirement to maximize productivity.

It is important to note that every member of this group has a busy and demanding schedule. Meeting and accommodating everyone's schedule is a massive time constraint. Every member is taking multiple classes in addition to Senior Design one, as well as taking on a part-time job on the side. These additional everyday tasks need to be taken into account when balancing time. Understanding that everyone has different schedules and providing the proper flexibility when necessary enables each member to

give their all. Careful planning, task delegation, deadline setting, and coordinating timelines and meetings are imperative to effective group organization. A couple of the members, for instance, work all day on specific days of the week, and thus those days are typically reserved for individual work such as research or writing their own assigned sections.

Despite the differing schedules, we aim to meet at least three times a week to discuss any updates, questions, or issues. A supportive environment is crucial when a team member is having difficulties as a result of an unforeseen event. To guarantee work completion, the team will support the struggling member. Poor organization of time could result in failing a course, having to start over, and postponing graduation for at least a semester. Understanding these risks decreases the likelihood of failure, promotes preventative measures, and highlights how crucial it is to complete the project on time.

Another aspect to take into account is the time frame given to complete the project once Senior Design 2 begins. Unlike typical Senior Design Projects, our second semester takes place over Summer 2024, which is a 12-week semester instead of the typical 16-week semester that takes place in the Fall or Spring semesters. There are many components we would like to include in the robot parrot but are unable to due to time constraints such as customization, wireless communication, and touch recognition. They are our stretch goals, but not the main goals of the simple, life-like movement (head tilt, beak movements, etc) and voice recognition and response to keep this project achievable within the allotted time.

4.2.3 Manufacturing Constraints

The manufacturing constraints for this project are not a large issue since we are not using anything niche. Everything we are planning on using is easily accessible online. However, due to this shortened semester, shipment time becomes significantly more important. Standard shipping will be chosen to avoid extra shipping costs, so we must consider how long it could take. Some components could easily take at least a couple of weeks, and if shipping internationally, it could even become a month. Time will be taken to decide whether To combat these concerns, the aim will be to order the components before the end of the spring semester so everything arrives and we can begin the building process promptly.

4.2.4 Durability

Durability is a critical constraint for the CAPER project, ensuring that the animatronic parrot can withstand prolonged use and potential environmental stresses. The exterior "skin" of the parrot must be tough enough to resist damage from handling and impacts while maintaining its appeal. Internally, components must be securely mounted to withstand mechanical vibrations and shocks without losing functionality. Additionally, electrical connections and wiring must be insulated and protected to prevent damage from moisture, dust, and other environmental factors. Every component must be internally tuned for both resilience and performance to comply with CAPER project requirements.

To efficiently absorb and disperse the energy from vibrations and shocks, this involves mounting all mechanical components securely. An animatronic parrot's service life is extended and its functioning is preserved thanks to its mechanical resilience, which also protects the fragile internal components. Internal damage could result in expensive repairs or, worse, a total system failure, thus this is an important consideration. Adherence to durability standards ensures that CAPER remains operational and visually appealing over its lifespan, minimizing the need for frequent repairs or replacements. To guarantee that the animatronic parrot continues to be robust and visually appealing, careful consideration must be given to the material choice for its skin, striking a balance between aesthetics and durability.

4.2.5 Input modes:

The CAPER project aims to offer multiple input modes to accommodate different user preferences and interaction styles, enhancing its versatility and usability. These input modes may include voice commands, manual controls, pre-recorded sequences, and automatic responses to environmental stimuli. Exploring the range of input possibilities further, voice commands provide a natural means of engagement that enables hands-free control and communication between the user and the animatronic parrot. For users looking for an approachable and organic way to connect, this mode is especially helpful.

When direct physical engagement is necessary or for some applications, manual controls, on the other hand, give a tactile experience with precision and responsiveness that some users prefer. Furthermore, compatibility with external devices and protocols is necessary to support diverse input methods, such as MIDI for instrument synchronization or UART for serial communication with external controllers. CAPER exhibits a dedication to accessibility, flexibility, and creativity by accepting a broad range of input modes. This strategy allows CAPER to be more flexible and adaptable to a wide range of situations and uses, in addition to increasing its appeal to a wider audience. Because of its adaptable interaction features, CAPER can be used for a variety of purposes, including education, entertainment, and interactive exhibits. This maximizes its usefulness and impact across a range of contexts.

4.2.6 Environmental Constraints & Sustainability

It is important to consider how rising technology affects the natural environment, though, because CAPER is simply a 1-1.5 foot tall robot that utilizes minimal power, it is unlikely to have a massive effect on the surrounding environment. The larger concern here is how the surrounding environment impacts this robot. CAPER will be a portable robot, but due to time constraints, the parrot will be water-resistant rather than waterproof. This means that CAPER will not be able to withstand exposure to rain. The resistance, however, will allow it to handle mild forms of liquid spillage. The focus of the parrot is more on its durability since it is portable so if it happens to be dropped or bumped into something, it will be okay and still be able to function accordingly. The electrical components will stay within the parrot, so as long as the exterior does not crumple from external circumstances (rain, wind, fall damage, etc.), then everything should function

properly. In order to reduce waste, components that are pre-owned and can be used, such as resistors and capacitors provided by the senior design lab, will be used.

Due to how this project is, sustainability is more of a personal constraint since it is not absolutely required, but would hopefully not leave any lasting effects on the environment. CAPER is built to last, so with proper care and maintenance, it should be decades before replacement. Despite the likely minimal effects on the environment, we must acknowledge that there is waste coming from this project. The materials utilized in this project might, depending on how they are made or disposed of in the end, have an adverse effect on the environment. How these processes occur makes this a possibility. Extras of each component will likely be used, and then thrown away due to trial and error during the prototype testing phase of the project. Not only from the electrical components but the mechanical ones as well. Building the skeleton also requires trial and error in getting measurements correct, which could result in wasted metals, plastic, and wood.

4.2.7 Social Constraints

The goal of CAPER is entertainment. The constraints imposed by society are primarily related to cost and availability. At the very least, high-quality robots often cost thousands of dollars, and the finest models can cost hundreds of thousands of dollars each. Ours offers something reasonably priced that smaller companies could use for their customers' entertainment. Despite having fewer options for functions than its rivals, this robot would easily prevail in terms of cost and usability.

4.2.8 Political & Ethical Constraints

Whether or not we are "stealing" or "copying" the ideas of another organization is one of the political limitations that we have considered. One major political constraint that will be well-researched is the possibility that our efforts will unintentionally violate the creative domains of other businesses. We understand that we must proceed with caution to guarantee that our initiatives stay properly inside the bounds of moral and legal guidelines. In addition to protecting our interests, we also progress the field as a whole when we carefully navigate intellectual property rights. There are no ethical constraints with this project since this is purely made for entertainment purposes and has no way of harming anyone.

4.2.9 Health & Safety Constraints

The health and safety constraints of this project are minimal. All of the electrical components are tucked away within the robot itself, so the chances of a shock happening are next to none. The edge-case scenario would be leaving CAPER out in the Florida summer heat, and then running every possible function simultaneously. In this scenario, there is a possibility that CAPER could overheat and blow up. Though again, warnings would be clearly stated in the instruction manual if this product were to be sold, so the possibility of any explosions is essentially none. Ensuring that when developing and creating, the gadget is assembled correctly and that safe power connections are used reduces the concerns of any potential accidents.

Chapter 5.0: Comparison of ChatGPT with other Similar Platforms

ChatGPT has recently exploded onto the scene as a know-all-do-all platform that is supposed to be widely versed in a plethora of subjects. Many believe platforms similar to chatGPT are going to harm the job market and begin replacing most jobs that aren't field. However, after further study into the applications of chatGPT, it may not be as versatile or implementable as many think. As four students in the computer/electrical engineering field, we've had more exposure than the typical person to AI's capabilities and have some great anecdotal experience. The appeal of ChatGPT is that it scrolls through and "reads" all of the websites you would've originally individually googled, read through, and taken notes on in a fraction of the time. In the following sections, we will cover the good, the bad, and our personal experiences with ChatGPT and similar platforms.

5.1 Benefits of ChatGPT

Some people swear by ChatGPT, others can't stand it, but a fair evaluation can prove its benefits and weaknesses and show they are very situational. When it comes to what ChatGPT is good at, the most benefit is provided when brainstorming, coding, or asking for general overviews of topics. For example, when beginning senior design, our group simply fed our interests and strengths to ChatGPT and asked it to produce a list of relevant projects and roles for our group. This provides a base outline, and then we held a meeting to individually review each idea and decide what modifications we would make and what roles we could assign. This is where ChatGPT shines - idea generation. Asking it for specifics and to decide everything for you is something it cannot do - every team is different and every situation has parameters that a machine cannot entirely prepare for. Without extreme documentation, many details will be left out of ChatGPT's consideration, and even at that it's not a perfect design yet. However using it to generate ideas that your group can then further refine it is great for. There are many amazing projects all over that would take a long time to individually research and find, but ChatGPT can do that in seconds.

That then brings us to another benefit of ChatGPT, the speed. For mundane and simple tasks, it can save you a large amount of time to learn something trivial or find general solutions/ideas for your situation. An example of this is that when we we're doing our Senior design project, none of us had any website design experience. The website as well did not have any affect on our grade for this course, so it was considered trivial. ChatGPT gives us a good layout and is very helpful when it comes to modifications to the website, when all it takes is a simple request. This shaves off a ton of time and increases the efficiency of our team by allowing us to focus on our project with the saved time. When taking the time to google something yourself, you have to find the right article, find the spot in that article that's giving you the information that's relevant to your task, then figure out how to implement it. This can take hours depending on your success rate, when ChatGPT may be able to find it and implement it for you in a matter of minutes.

5.2 Downsides of ChatGPT

However great ChatGPT is in the above scenarios, it would be foolish to not recognize its shortcomings. As mentioned previously, ChatGPT does not do everything for you and will not know the ins and outs of your specific situation. Even with a very in depth debrief, any AI system will not be able to perfectly replicate the situation you are in or account for all the factors that are present in the real world. Just because there's a successful recording of a similar experiment online elsewhere, your experiment will have different parameters, technology, components, and geographical factors that make whatever ChatGPT feeds to you slightly inaccurate.

One of the other issues is that when telling ChatGPT it's solution is incorrect, it is not adept at fixing its solution. It is still running based on the data available and will likely go against what it initially believes to try and give you a desired answer. Thus again, asking for a specific solution is not helpful. The majority of ChatGPT's responses come from scholarly articles, firmly published websites, and easily accessible data across the internet. Therefore, when using a search browser yourself such as Google Chrome, you have access to things such as Reddit or Quora which provide anecdotal yet informal information that may be more helpful to your situation. On those websites you can garner real responses from people with a lot more experience than an online platform does. This often can result in a better look into the topic or common issues in the field that go unnoticed by something sweeping the web in large.

Other issues show themselves in the form of safety issues and situations that require real time interactions. When we're dealing with issues with circuit work, debugging, or other real time instances, feeding the situation to ChatGPT so it understands every parameter of what we're going through proves rather difficult. The process of testing our system, asking ChatGPT why it didn't work, understanding and implementing it's (hopefully) right solution would take a lot longer than troubleshooting it as a group. As well, ChatGPT does not consider many variables that go unaccounted for like built in resistance or any impurities in components we use. This can lead to safety concerns within the group and can ruin components within our project, introducing a longer setback time.

5.3 ChatGPT within our project

As far as ChatGPT's assistance goes into our project, it has been helpful. As touched on above, we used it in our project idea brainstorming phase and as well for the construction/editing of our website. Many would argue that this is harmful as it robs us of the skills we could've gained such as web design, but by simply inspecting the code and actually understanding the changes the application is making, I'm able to learn the information just as fast by seeing it in practice. As we venture to the actual construction of our project, the influence of ChatGPT on our project will dwindle as we utilize chat boards and anecdotal experience to try and understand the issues within our very specific project. We may run our issues by ChatGPT on the off chance it has something helpful to offer, but that won't be our first resource. In summary, ChatGPT is much more helpful when it comes to idea generation and working in the more broad spectrum of

things, but once things become more detailed and situation-oriented, its appeal dwindles.

5.4 Example of Poor Performance: Inconsistent Writing

We attempted to use ChatGPT within our project to write the introduction to our Divide and Conquer 10-page document. It was instructed to write the draft for an introductory paragraph, using the preliminary description of the project as a guide. Since the system cannot keep track of a lot at once, the strategy was to keep feeding the previous content back into the system to expand it. The resulting text was inconsistent with the one it previously wrote causing the writing to drift from topic to topic. This was not acceptable, and the whole introductory section had to be redone by hand.

5.5 Example of Harmful Performance: Incorrect Information

One of the only times ChatGPT was potentially harmful to the project was within its preliminary stages. Specifically when it came to finding examples of similar systems to CAPER that were recently completed or still in development. ChatGPT has web search functionality, allowing it to access pages, read their content and relay it to the user. The idea behind using it for this was to automate both search and summarisation. The results were unequivocally harmful. ChatGPT proceeded to provide:

- Hallucinated links and attributions
- Completely false information
- Misattributed links (real like but that had notion to do with the project it was citing)

The erroneous nature of the results was only determined when we tried to check the provided links for additional detail.

Afterwards, all research was done without the use of GPT-based AIs.

5.6 Example of Good Performance: Formatting and Grammar Checking

Where these systems lack actual intelligence when it comes to context outside of their training, they more than make up for it in their ability to do simple tasks like formatting and grammar checking.

Within our project, ChatGPT is only used for putting difficult ideas into words, and helping overcome writer's block by providing outlines for paragraphs. Its content is not copied verbatim, as it often makes mistakes when it comes to the project's goals and our ideas, making its help superficial at most.

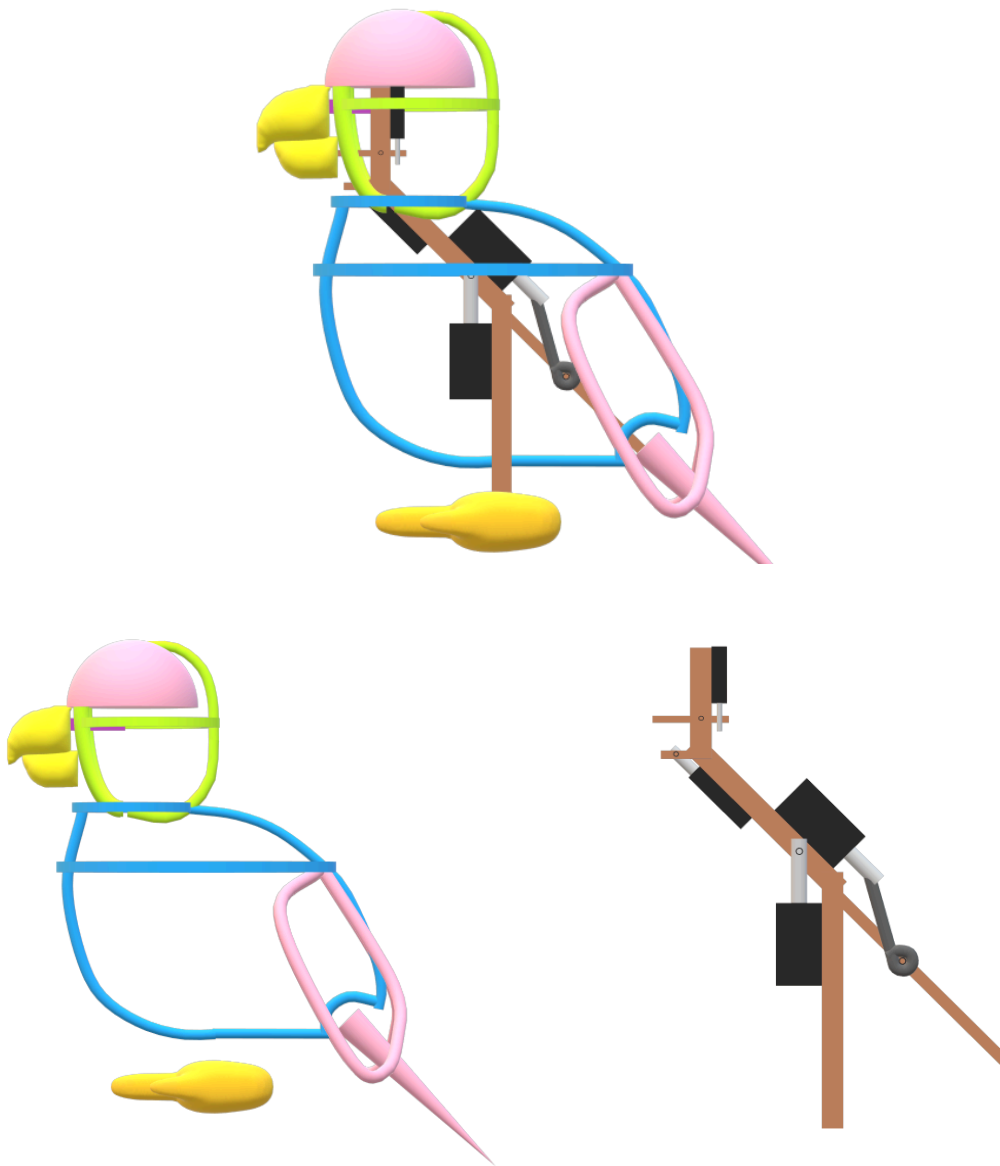
ChatGPT's good performance extends to grammar checking as well. Making sure that whatever is being written follows a technical writing style and helping ensure cohesion between parts written by different team members.

Chapter 6.0: Hardware and Physical Design

This section will further elaborate on the physical implementation of the materials and components selected in Chapter 3. This includes the construction of the main body, the circuitry of the IOCB, and the exterior layout of the chassis that will house the majority of CAPER's external electronics.

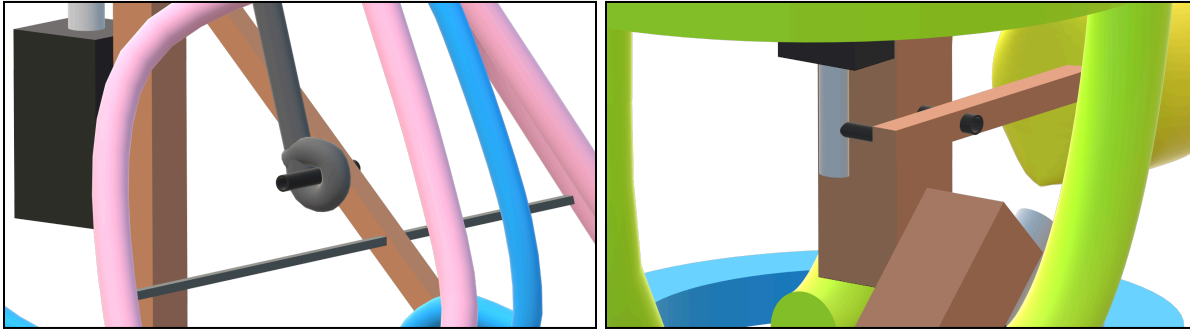
6.1 CAPER Main Body

The below illustrations display the CAPER figure in its resting state. The wooden central frame shown in the lower right supports all of the mechanical devices, as well as the internal shaping framework seen on the lower left. Combining these two elements together results in the complete figure shown at the top.



Figures 6.0, 6.1 & 6.2: The elements of the Main Body

Once upholstered in felt and feathers, all the internal workings will be completely concealed. Here, we can begin to display how each of the aforementioned materials are used. The pink hemispherical “cranium” at the top would likely be made of foam or styrofoam, same goes for the beak. The green, blue, and pink beams (skull, midsection & wings) would be constructed from rubber tubing or thin metal rods, depending on the desired rigidity of that section. Rigidity becomes especially important when creating the mechanical linkages between the solenoids and the frame. All four movements require sturdy links to prevent structural instability. This can be achieved with thin wooden dowels or metal rods. Examples of both are shown in Figures 6.3 and 6.4:



Figures 6.3 & 6.4: The metal tail link (left), and wooden mouth link (right).

Animating one movement at a time is easy enough, but simultaneous movements cannot conflict with each other. For instance; position of the head cannot block or limit the range of the mouth. Likewise, the orientation of the wings and tail need to move proportionally with reference to the rest of the midsection. This all requires clever placement and attachment of the internal shaping segments, so that they move freely with respect to the closest neighboring links. Additionally, placement of the solenoids and linkages must also take into account the surrounding obstacles. This way; even with all four solenoids activated, there is no interference between one body section and the other. Below is an illustration showing what this would look like:

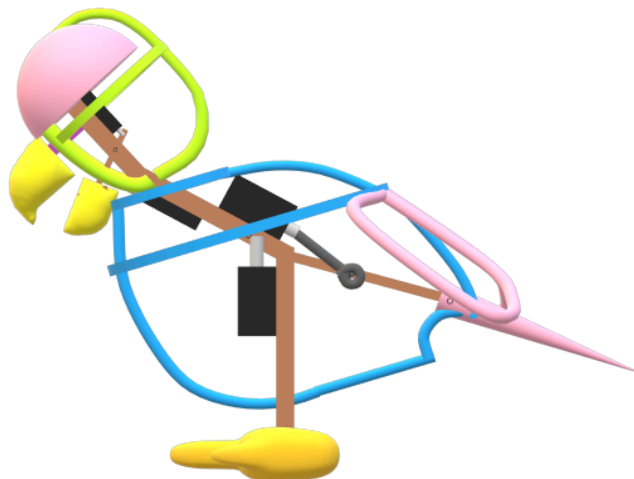
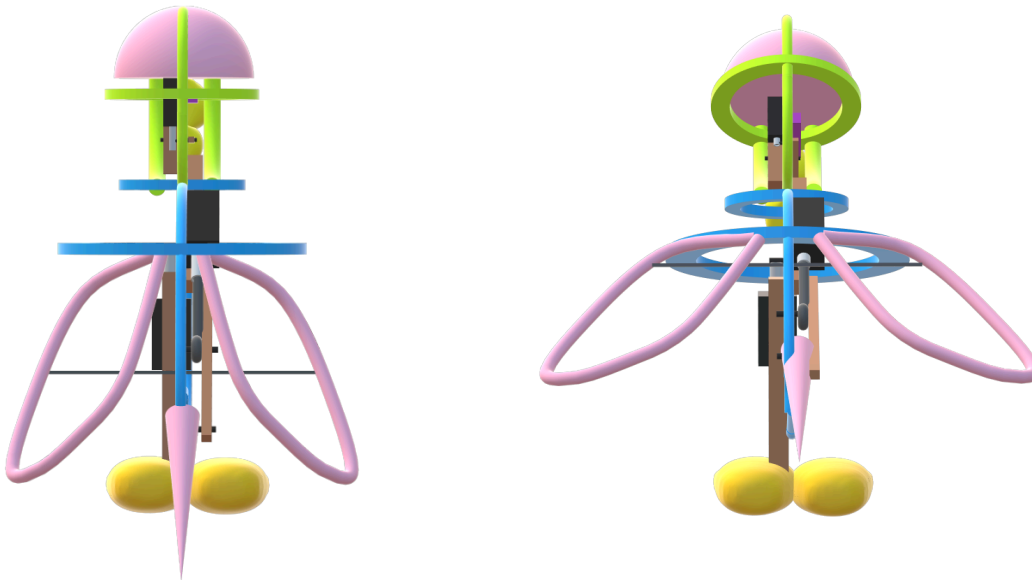


Figure 6.5: CAPER's Main Body with all four movements activated

The mouth solenoid is the top-most actuator, positioned at the rear of the top link. When it retracts, it pulls the beak downward with a “see-saw-like” connecting rod. The head solenoid, located in the front of the second link, pulls the entire head/mouth assembly forward. Similarly, the tail solenoid is located on the back of the second link, pulling the tail rearward and the wings out laterally. The body solenoid is located at the top of the bottommost link, tilting the entire body assembly forward when activated. A better understanding of the lateral wing movement can be obtained from a rear-facing point of view as seen below:



Figures 6.6 & 6.7: Resting wing position (left), activated wing position (right).

Even with the slight lack of dimensionality and position control of CAPER’s movements, the overall range of motion is impressive. Once fully animated and connected to the controller, the Main Body will be able to convey the realism it was designed to convey.

6.2 IOCB Circuitry

With regards to electronics, circuit design, and board layout, the IOCB will be where we direct the bulk of our attention. In total, there are seven circuit groups that are all integrated in the custom IOCB board. Those groups are: the MCU & reset connection, debugging circuit, VOX circuit, optocoupler circuit, and the three voltage regulators.

6.2.1 MCU Chip

The central chip on the IOCB; the MSP430G2553IPW20, is what will be used on our board. The 20-pin surface-mount package of this chip will be used, as the IOCB will be entirely surface-mount. Below is the full schematic diagram of the MCU with all the immediate connections and solder pads. Directly below the schematic is a chart describing each of the 20 pins on the chip, and their use in the circuit (if any):

Note: The pin layout on the schematic refers to the pin numbers as designated by the package layout. The chart numbers the pins as seen by the software.

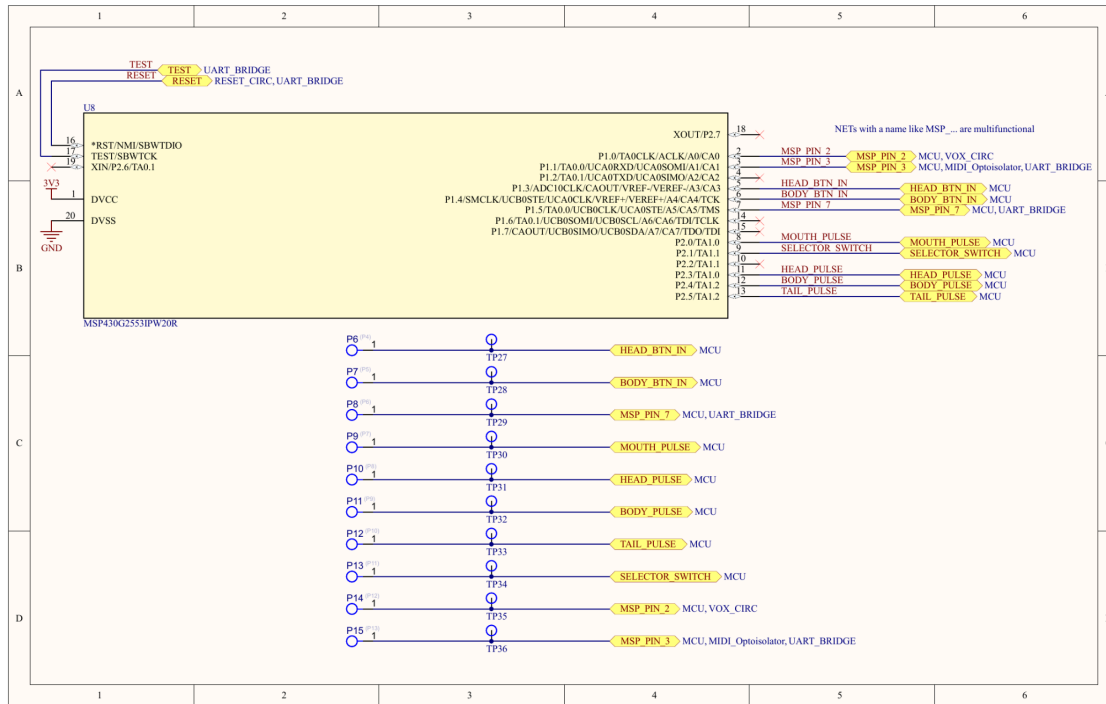


Figure 6.8: Complete Schematic of MCU and Immediate Connections

Table 6.1: Pin Connections on MCU

Pin	Functionality	Pin	Functionality
DVCC	3.3V supply	DVSS	GND
P1.0 I/O	Mouth & VOX Input	P2.6	(unused)
P1.1 UART RX	MIDI & Debugging	P2.7	(unused)
P1.2 UART TX	(unused)	TEST	Debugging
P1.3 I/O	Head Pulse Input	RESET	Debugging & Reset
P1.4 I/O	Body Pulse Input	P1.7	(unused)
P1.5 I/O	Tail & Debugging	P1.6	(unused)
P2.0 I/O	Mouth Pulse Output	P2.5	Tail Pulse Output
P2.1 I/O	Selector Switch Input	P2.4	Body Pulse Output
P2.2	(unused)	P2.3	Head Pulse Output

Power Pins (DVCC & DVSS): The MSP430 operates at 3.3VDC, supplied from an onboard voltage regulator. The DVSS pin will share the same ground as all IOCB peripheral circuits, and the upstream power supply.

Parallel Pulse Input Pins (1.0, 1.3, 1.4, &1.5): These four inputs control the mouth, head, body, and tail actuators respectively. They are all configured in an active-low setting, with the internal pull-up resistors activated. The spring loaded buttons will be mounted off-board, on an external chassis that houses the IOCB; four solder pads will be placed on the board for the four buttons. Pressing a button will short that input to ground, switching the input on. Releasing the button will open the pin from ground, switching that input off again. This allows for the inputs to be held for any desired duration. The output of the VOX circuit will also connect to pin 1.0, when the voice-activated partial live operation mode is active. Both the push-button and the VOX circuit can control the mouth at all times.

Selector Switch Pin (2.1): This selector switch is used whenever the user wants to switch from fully manual operation, to partial-manual operation. This pin is also configured as active-low with the built-in pull-up resistor enabled. The switch will also be mounted off board, connected via a solder-pad. When the switch is shorted to ground, the four input pins are all active. Opening the switch from ground mutes the inputs from the head, body, and tail pins (1.3, 1.4, & 1.5), leaving only the mouth pin active (1.0). Switching to partial-manual mode will also activate a subroutine in the code to randomly generate outputs for the head, body, and tail. Information on this code will be described in Chapter 7, in the IOCB Software section.

Output Pins (2.0, 2.3, 2.4, &2.5): These four respective output pins go to the mouth, head, body, and tail relays located off the board. They are configured in the code to switch between 0 VDC and 3.3 VDC for off and on. These outputs will be active in all operating modes of the IOCB.

MIDI UART Pin (1.1): One of the two uses of the UART RXT pin (1.1) is to receive the MIDI data for controlling the movements. The upstream UART transmitting device (Jetson NANO or MIDI playback device), will send the stream of data to the IOCB's onboard optoisolator. The voltage-stable data leaving this optoisolator will then enter pin 1.1. The voltage received will be a digital pulse oscillating between 0 and 3.3 V. Information regarding the construction of this optocoupler circuit will be described below.

Debugging Pins (1.1, 1.5, TEST, & RESET): The test and reset pins, along with both the receiving and transmitting UART pins on the MSP, will be needed to interface with the onboard FT232-RL UART bridge. Through this bridge, the necessary debugging data can be sent from Code Composer Studio, directly to the MCU, using the installed Bootstrap Loader. Pin 1.5 is also used as one of the inputs for the Tail actuator button. This input will become deactivated when debugging occurs, and the pin will interface with the FT232 until the debugging is complete. Information regarding the construction of this debugging circuit is described below.

Reset Pin: A second function of this Reset pin is to reset the MCU with every power-on. This is accomplished by bridging a 10k resistor between the RST pin to +Vcc (3.3V), and connecting a 100nF capacitor between RST and GND. Below is the schematic of the Reset Circuit:

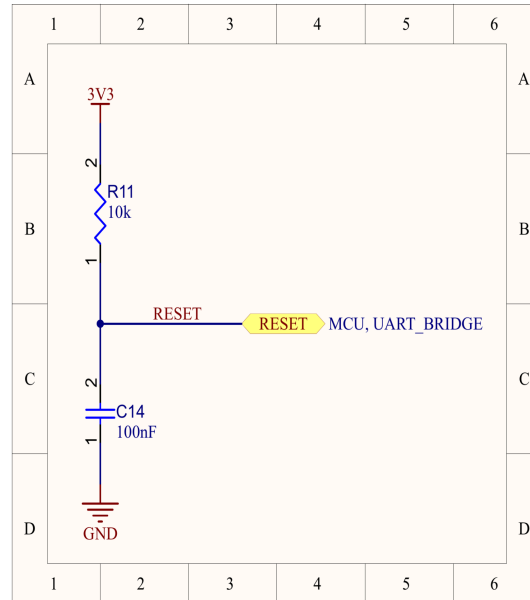


Figure 6.9: Schematic of Reset Circuit for MCU

6.3 Debugging Circuit Layout

The debugger circuit falls between the debugging computer and the MCU. The FT232 chip requires 5 VDC to work, which can be sourced from the USB connection instead of the onboard regulators. This way, the debugging chip is only working when USB is plugged in. The MSP430 will receive its power from the onboard regulators, so it is on all the time. The basic data transferring connections between the debugger and the MCU are seen in the following diagram:

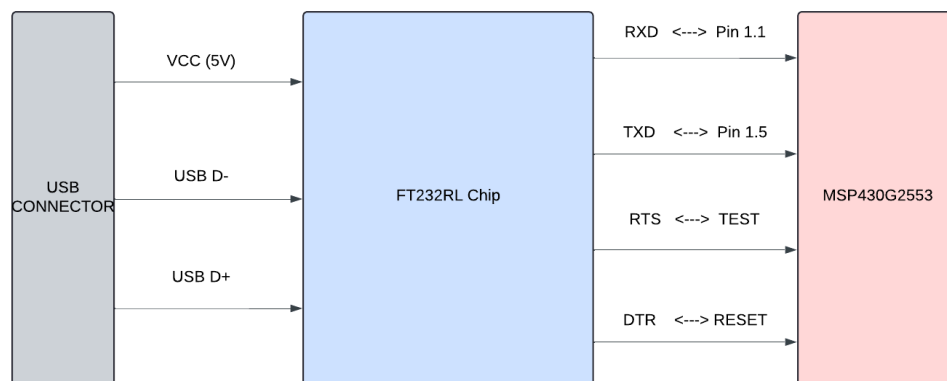


Figure 6.10: Basic data flow diagram through FT232RL

These are the necessary connections needed to transfer information to the chip over UART, not including power and ground connections (please refer to complete schematic

for this). The FT232 communicates solely through the D- and D+ lines. From this, it generates the four outputs to be connected to the MSP. As seen above; RXD, TXD, RTS, and DTR on the FT232, go to MSP pins 1.1, 1.5, TEST, and RESET respectively. Here is the full schematic of the UART Bridge Debugging circuit:

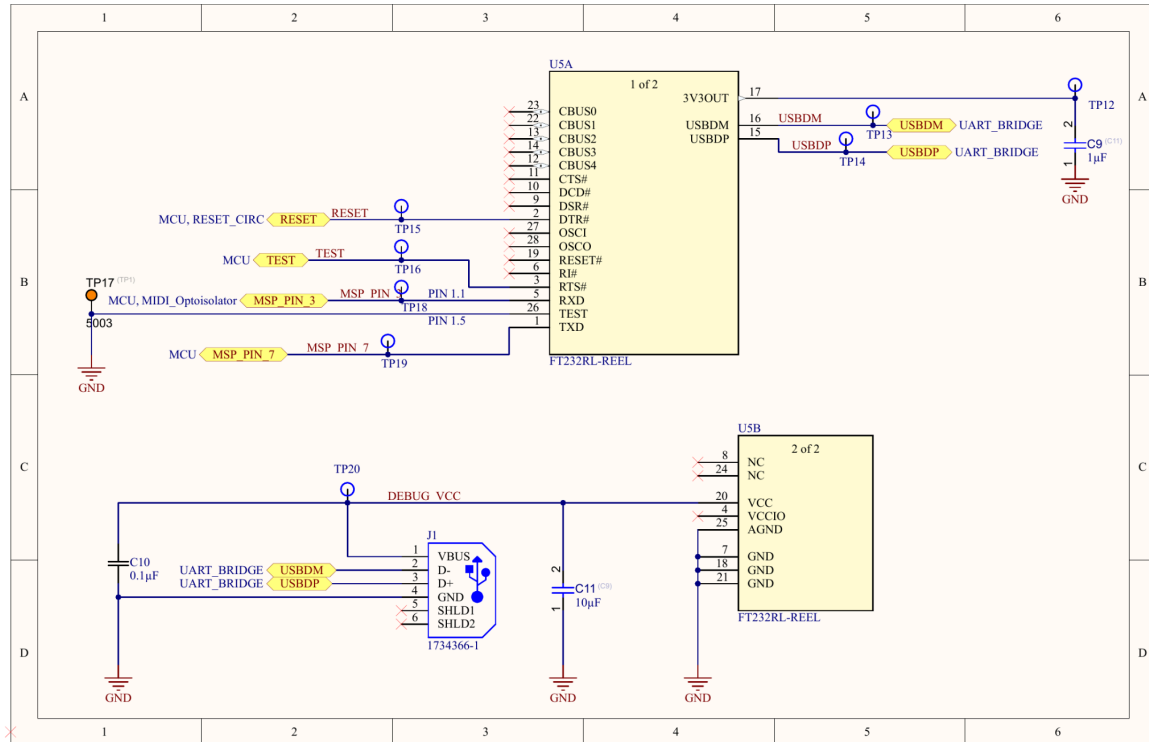


Figure 6.11: Schematic of FT232FL Debugging Circuit

6.4 VOX Circuit Layout

The VOX circuit is named after a standard functionality seen in ham radios; the ability to transmit using voice activation instead of physically pressing the PTT button. It is a piece of technology that has existed for more than 50 years, with early renditions of the circuit using BJTs. Figure 6.12 displays one of these older circuits; a PNP and darlington NPN transistor are paired with a mechanical relay to do the switching. These older designs are suboptimal, as possible current overloads could occur from the relay (Aspencore, 2015). Newer circuit designs that use operational amplifiers are less prone to these issues, such as the one in Figure 6.13. Using two single channel op amps, the signal enters as an aperiodic audio signal, and transforms into a clean, usable, active-low pulse wave. It is that exact circuit in Figure 6.13 that served as the basis for our design, albeit with several component modifications. Once completed, the circuit will match the schematic below in Figure 6.15, the audio signal will follow the conversion path as seen in Figure 6.14.

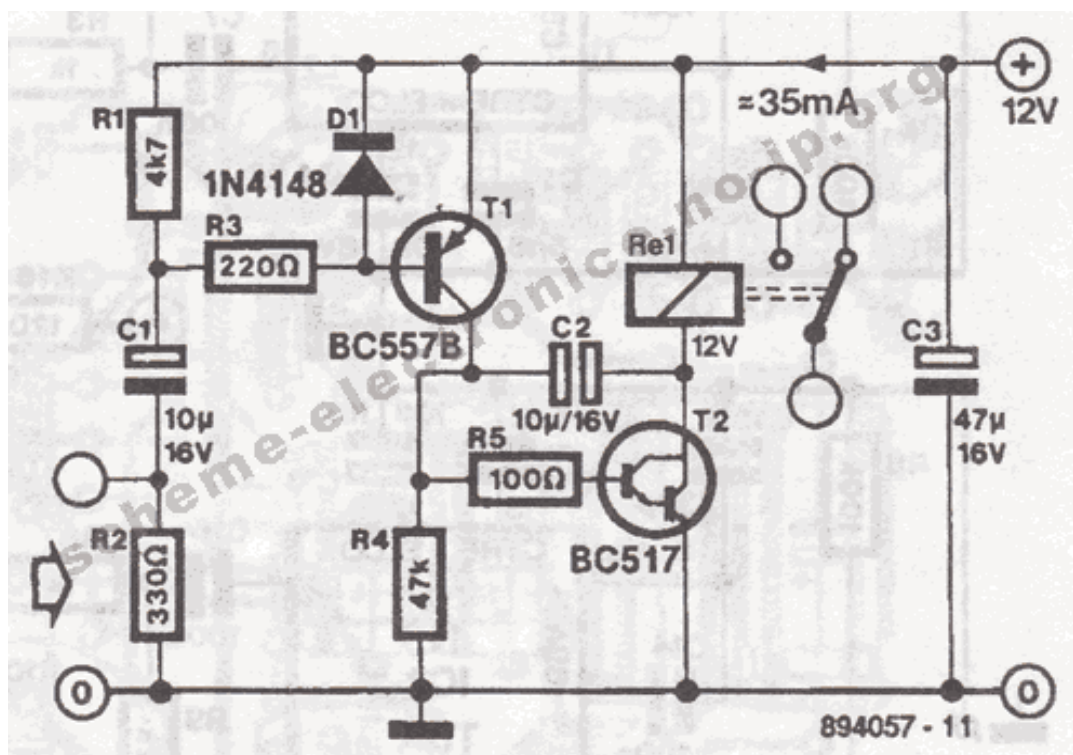


Figure 6.12: Old VOX Circuit Layout, courtesy of electroschematics.com

Vcc = +9 to +14VDC

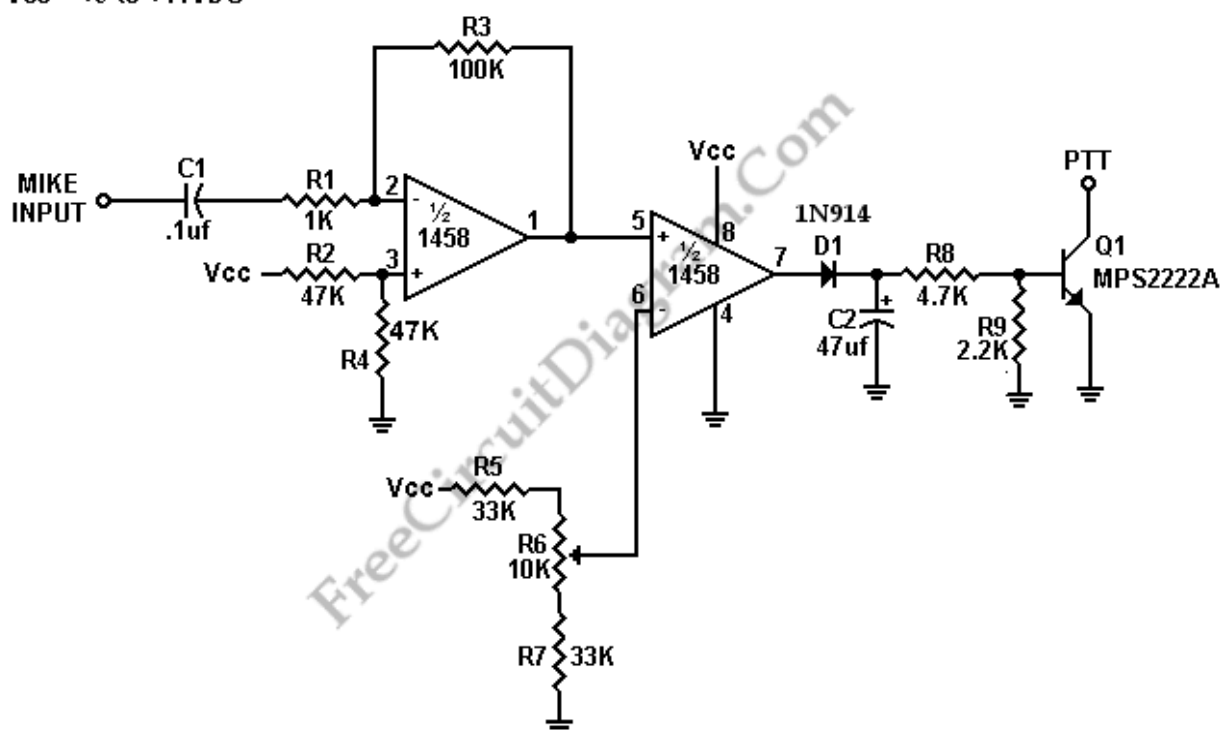


Figure 6.13: Modern op-amp VOX circuit, courtesy of freecircuitdiagram.com

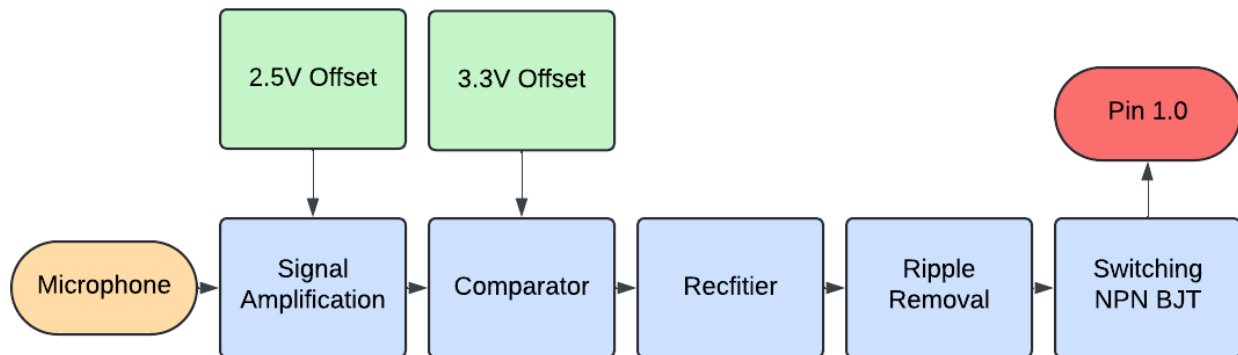


Figure 6.14: Audio path through VOX circuit

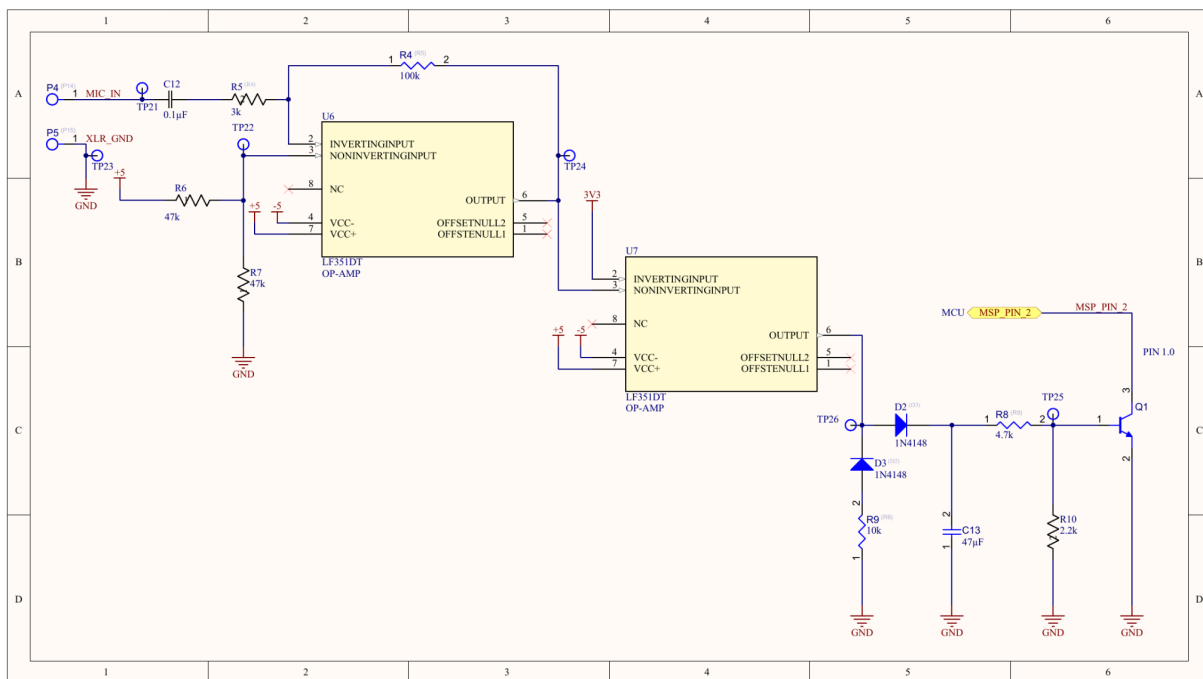


Figure 6.15: Schematic of VOX Circuit

Stage 1: Amplification: Immediately exiting the dynamic microphone, the audio signal ranges from only 2 to 4mV in amplitude. To make the signal more manageable, it enters an inverting op-amp stage. The resistors in this stage are selected to yield an approximate gain of 30x. Inverting the audio waveform won't affect the performance, as the phase is completely irrelevant to the performance. The important factor is that the non-inverting input is left available for a DC offset. A DC voltage of around +2.5V

($V_{cc}/2$) is sent into the non-inverting input, positively shifting the output signal. This is a more convenient signal placement for the following stage.

Stage 2: Comparator: The now-shifted waveform enters the non-inverting input of the second op-amp. At the inverting input, a DC reference point is set to 3.3V. Since there are no feedback resistors in this stage, the output will vary between $+V_{cc}$ and $-V_{cc}$ depending on the value at the non-inverting input. Since the input signal rests at 2.5V, it only needs to rise 0.8 volts to trigger the comparator to swing high. The frequency of the output wave will still match the microphone signal, but now the amplitudes are fixed at $\pm V_{cc}$.

Stage 3: Half Wave Rectifier: The signal exits the comparator as a variable-frequency square wave with a peak to peak amplitude of 10V (-5 to +5). Two diodes are placed here; a reverse-biased diode shunted to ground, and a forward-biased diode in series with the output. When the comparator stage is swung low, the signal flows through the shunted diode coming from ground. A current limiting resistor is placed here to protect both the diode and the op amp. This configuration also prevents damage to the forward biased diode, preventing it from ever reaching breakdown voltage. When the comparator stage is swung high, the shunted diode is shut off and the signal flows through the series diode and across a shunted capacitor. This capacitor holds a momentary charge between the peaks of the square wave, removing the ripple. When the frequency of the signal is high enough, the square wave peaks will be combined to a single pulse.

Stage 4: BJT: The signal is now a pulse wave that varies between 0V and 5V, that switches on whenever there is speech present at the microphone. This signal enters the final stage of the VOX circuit; the NPN BJT. The pulse wave enters a small voltage divider, lowering it to approximately 1.6V. This enters the base of the transistor, where the emitter is connected to ground, and the collector is connected to pin 1.0 on the MCU (the mouth trigger). With the pulse wave at 0V, the transistor is cut-off, and no voltage can pass between the collector and emitter. When the pulse wave is 1.6V, the transistor enters saturation, and current passes through. The MCU's pull-up resistor for pin 1.0 will prevent any excess current from damaging the MCU or the transistor.

6.5 MIDI Optocoupler Layout

The H11L1 optocoupler chip is used in our design to isolate the incoming MIDI signal from the MCU. Operating this chip requires three connections to the MIDI plug, one to V_{cc} , one to ground, and one to the RX pin on the MCU. The passive components required for this circuit are a protective diode, a few filtering capacitors, and a few current limiting resistors. Ignoring those protective elements, the basic signal path is illustrated in the diagram below:

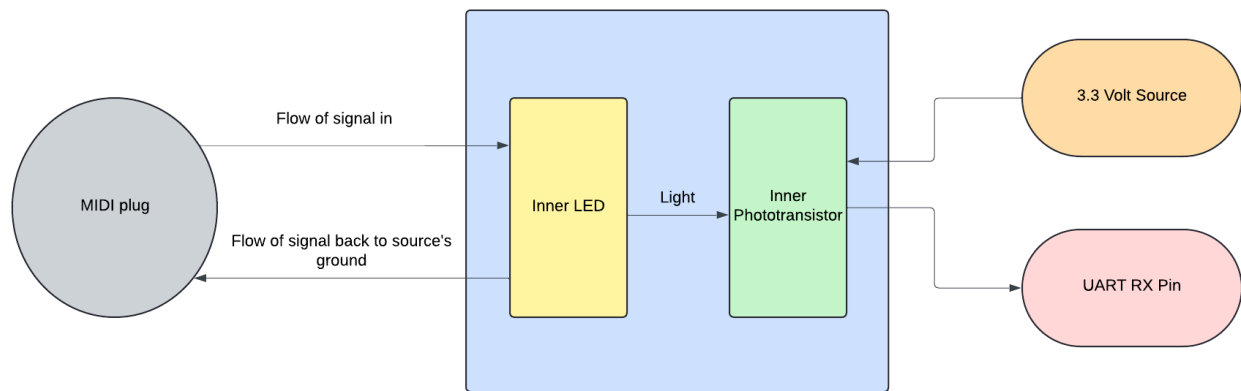


Figure 6.16: MIDI Optocoupler Data Path

The blue square is the H11L1 chip; complete with the LED and phototransistor inside. The MIDI plug itself will be mounted off the board due to its size, with connections made to the board via wiring and solder pads. Below shows the full schematic diagram of the MIDI Optocoupler circuit:

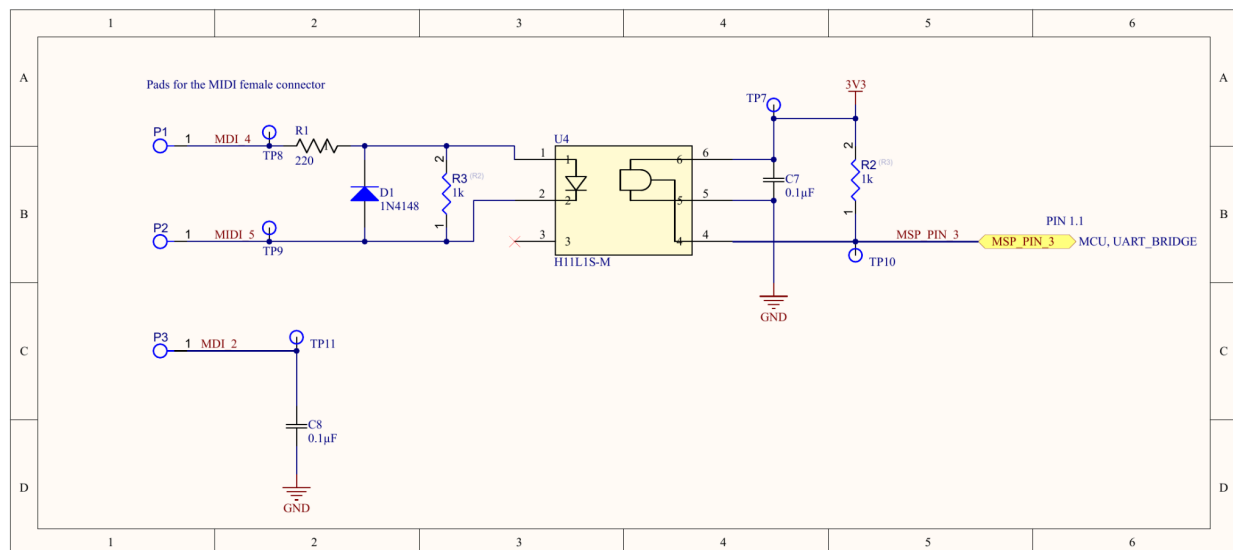


Figure 6.17: Schematic of UART Optocoupler Circuit

6.6 Voltage Regulator Layouts

All voltage regulator circuits on the board have the same basic construction. The voltage chip converts the input voltage to the rated output voltage using pulse width modulation. Two capacitors are shunted across the chip to ground; one at the input, and one at the output. The input capacitor prevents any AC impurities from entering the chip and damaging it, while the output capacitor minimizes ripple. With all the various peripheral circuits on the IOCB, voltages of +3.3V, +5V, and -5V will be needed for full functionality. Respectively, the LM1117, 7805, and 7905 chips will be used for the

voltages. Below is a chart with the recommended capacitor values as mentioned in the chips datasheets:

Table 6.2: Capacitor Values

Chip	Input Shunt Capacitor	Output Shunt Capacitor
LM1117T-3.3	10uF	10uF
7805	0.33uF	0.1uF
7905	2.2uF	1uF

In the case of the 7905; since it is a negative voltage regulator, it requires a negative input voltage with reference to ground. Both the LM1117 and 7805 require positive input voltages. Below are the three schematics for the LM1117, 7805, and 7905 voltage regulator circuits:

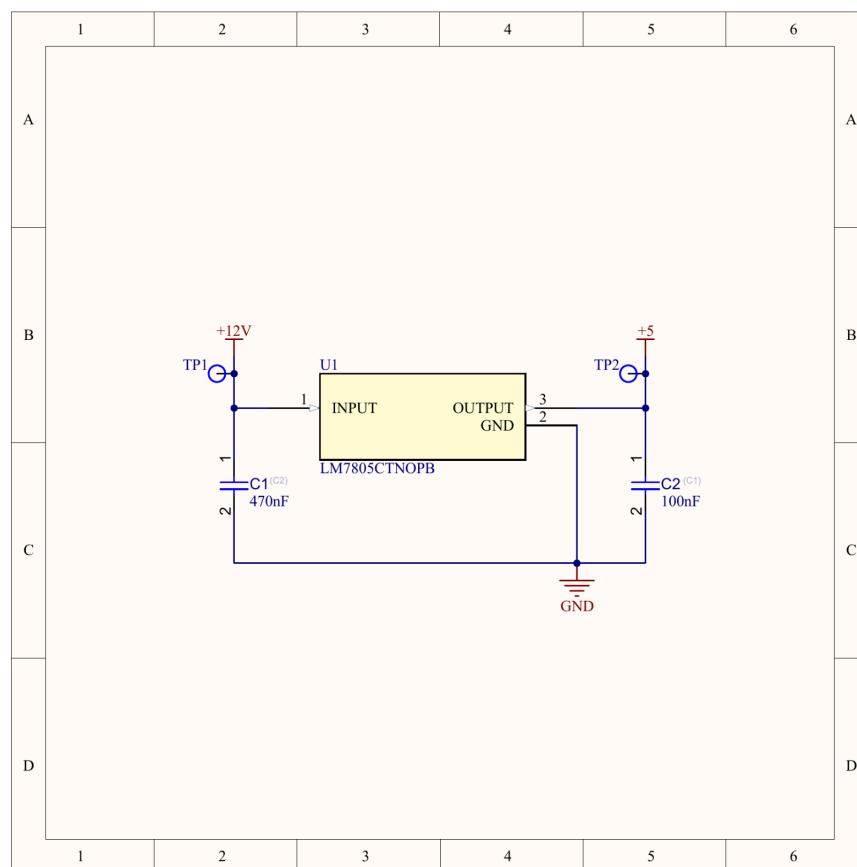


Figure 6.18: Schematic of the +5V Regulator Using the 7805

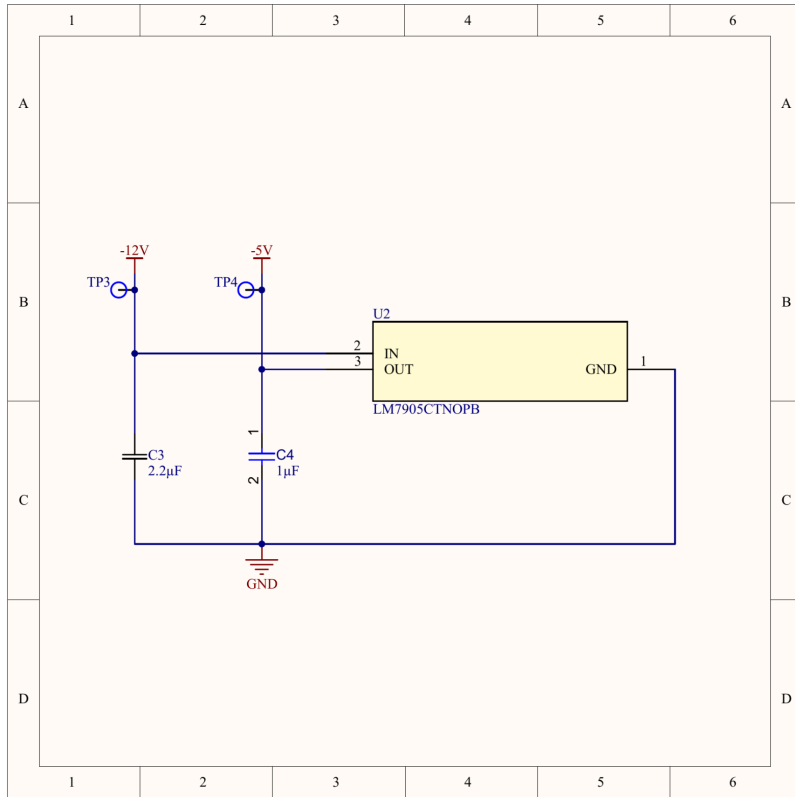


Figure 6.19: Schematic of the -5V Regulator Using the 7905

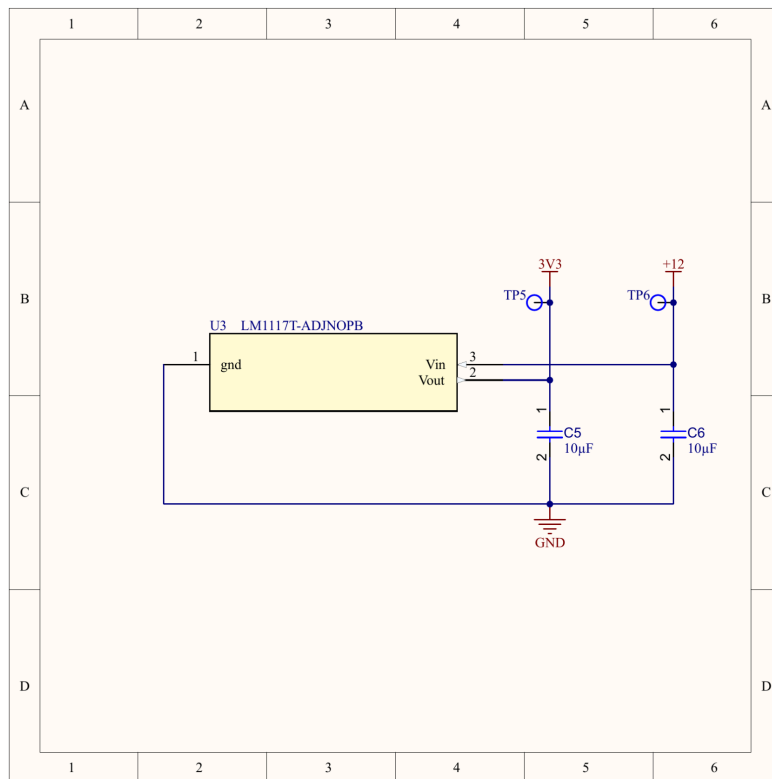


Figure 6.20: Schematic of the 3.3V Regulator Using the LM1117

6.7 Chassis Layout

To conveniently store the IOCB, VRB, actuator relays, and DC power supply in one location, a suitable enclosure or chassis will be constructed. This chassis will serve not only as a protective element for the circuit boards, but also as a nice support structure for the boards and their immediate neighboring components. The exact shape and layout of the chassis is somewhat arbitrary, as many different layouts could meet the necessary requirements. Here is a mockup of a potential design for the chassis, shown at an angle from the top right corner:



Figure 6.21: Initial Design for CAPER Boards' Chassis

The most used ports and switches are located near the front of the enclosure, for efficient, immediate access. The four spring-loaded push-buttons are located in the top right corner, and are color coded for easy distinguishability. Ports and plugs shared between the IOCB and VRB are placed next to each other, eliminating the need for ultra long and expensive adapters and cables. Spatially, the chassis is laid out flat, with dimensional resemblance to that of a VCR or DVD player. This provides better stability upon a countertop, as well as superior heat distribution. With this layout, there is still quite a lot of unused space on the top and sides. Despite this, the volume on the inside is quite efficiently spent. The less-frequently used ports are located on the rear, shown in the following figure:



Figure 6.22: The back left corner of the CAPER Boards' Chassis

From this point of view, the power plug, power switch, and four actuator terminals are seen. The terminals installed are similar to that of raw wire connections for bookshelf

speakers. The black terminals for each actuator are wired directly to -12VDC, coming straight from the power supply. The red terminals alternate between open and +12V, for when the actuators are resting and activated respectively. The solenoids inside the CAPER main body will have ground connections soldered directly to their frames for voltage stability. This ground line can then be fixed directly to the chassis via a screw terminal (not shown in the image). Placement of all these terminals and switches on the rear, correspond strongly to the location of the circuit groups on the inside. The same principle applies for the location of all the front-mounted ports, buttons, and switches; their placement on the outside directly relates to the internal layout. Being installed in this relatively flat chassis, the four internal circuit-groups do not overlap or stack on each other. They are spaced out evenly, taking into account heat, the datapath and power flow. This general layout of the four circuit groups is shown below in Figure 6.23:

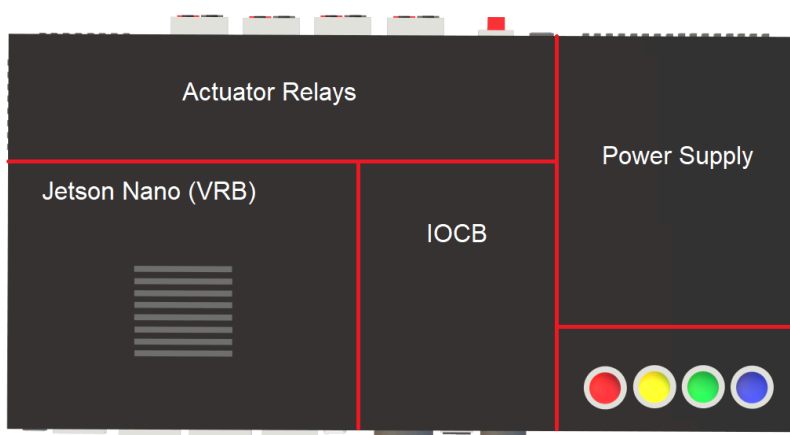


Figure 6.23: CAPER Board Chassis Top View with general circuit location

As briefly mentioned above, the biggest reason for choosing this layout is for proper heat dissipation. Every circuit-group within this enclosure produces a considerable amount of heat, requiring logical design choices to be made to ensure proper cooling. This is especially the case for the VRB and power supply, which will produce the most heat out of all the circuits. For this reason, those circuits are placed along the outer perimeter of the enclosure, providing direct access to side air vents. There is a top mounted vent above the Jetson Nano, right where the CPU heatsink is located. Performance testing and analysis will ultimately dictate whether or not the current ventilation arrangement is sufficient. For example, the power supply section only has rear and right-side vents in this figure, but top mounted vents may also be installed if more cooling is needed. The IOCB will produce significantly less heat than any of the other components; having a much smaller current draw. Placing this circuit in-between the others shouldn't cause any immediate cooling issues. If overheating ever becomes an issue with the IOCB, a top mounted vent would solve the problem. Similarly with the relays, they should never reach a point under normal operating circumstances where they overheat and malfunction. The current from the downstream solenoids will always be within the relays' tolerance, and they'll get plenty of airflow by the left rear and side vents. Installation of cooling fans is another way to better manage the heat; this is often a standard feature on various Jetson Nano board configurations.

Chapter 7: Software Design

This chapter outlines the software design implemented throughout CAPER's computational hardware. The systems explored are for the IOCB, VRB, and offboard server configuration.

7.1 IOCB Software Design

The C-language program on the Input/Output Control Board, will recognize the different signals seen at the input pins, and generate the appropriate signals on the output pins. This full code can be found in Appendix D under "IOCB Code". Referring back to Figure 2.5, there are six inputs and four outputs that need to be managed by this program:

Inputs

- Mouth Pulse Stream
- Head Pulse Stream
- Body Pulse Stream
- Wings and Tail Pulse Stream
- UART MIDI Datastream
- Selector Switch Pulse Stream

Outputs

- Mouth Output Pulse Stream
- Head Output Pulse Stream
- Body Output Pulse Stream
- Wings and Tail Output Pulse Stream

Each of the four outputs will be active regardless of the operating mode the IOCB is in. Managing the signals at the input requires the program to continuously poll and monitor their values, reacting to the changes as soon as they occur. This is achieved using Global Interrupts within the chip; essentially a digital "mailbox flag" that alerts the code when an input is received. In order to receive and interpret these interrupts, the Main Function has to perform several initializations. This is done in the following order:

1. **Define the "char" variables** needed to store the incoming MIDI bytes.
2. **Implicitly declare** the external functions the Main will call to.
3. **Run the external "setUp()" function.** This disables the watchdog timer, sets the submaster-clock to the specified baud rate (32,150), configures the UART RX pin, and enables the UART interrupt; returns to Main when finished.
4. **Define the four push-button input pins** using the Port 1 Direction register. Set them to be active-low, and enable the internal pull up resistor. Enable the Port 1 Interrupt Flag and clear it for all inputs.
5. **Define the selector switch input pin** using the Port 2 Direction register. Set as active-low with internal pull-up resistor enabled. Enable the Port 2 Interrupt Flag and clear it for the selector switch pin.
6. **Set up the four outputs** using the Port 2 Direction Register, set their outputs to be OFF initially.
7. **Enable the global interrupts**, and enter an infinite "while" loop.

The use of a low power mode is possible, since only the submaster clock is used. This isn't a huge concern, as the difference it would make is perceivably irrelevant. With the solenoids that consume 16W each; we wouldn't notice a few microwatt-hours difference. Instead, the code sits in the "while" loop indefinitely, until an interrupt flag is raised, and the respective function is called. Once the called function is completed, the program returns back into the Main, and resumes the infinite "while" loop. The structure of the three interrupt functions is seen below:

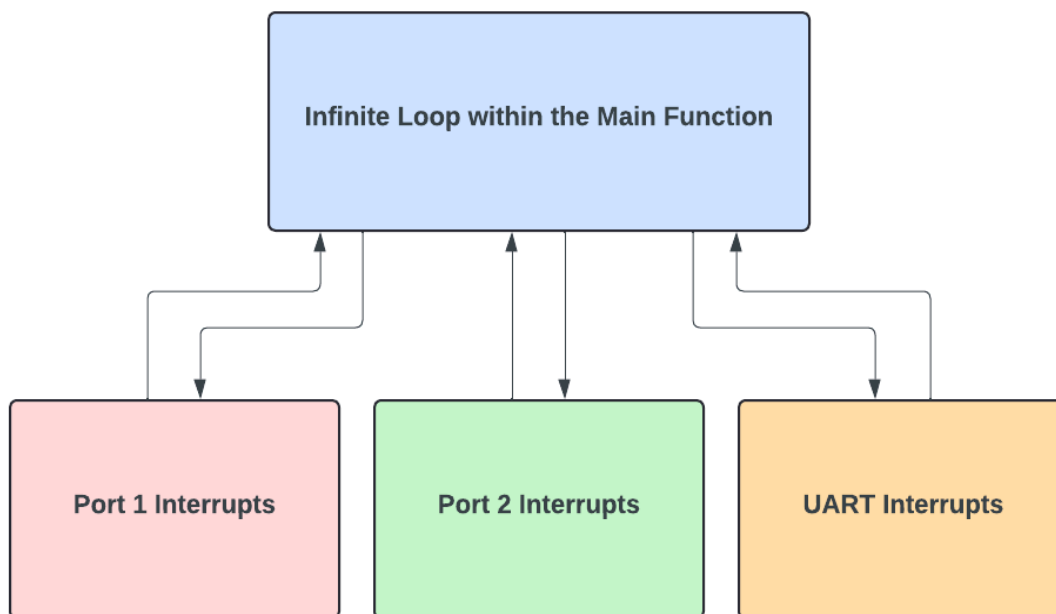


Figure 7.1: IOCB Interrupt Hierarchy

The organization of these interrupts aligns almost perfectly with the four operating modes. Port 1 handles everything for Full-Manual operation; toggling the four outputs when any of the respective push buttons are pressed. Port 2 handles the Partial-Manual Operation; toggling the head, body, and tail automatically, and toggling the mouth when its button is pressed. The UART function works for modes 3 & 4: Pre-Recorded Sequence, and Automatic Live Operation. In that case, the only difference is which upstream device is generating the MIDI. Pre-Recorded sequences would be played back from a PC, and the Automatic Operation from the VRB. Either way, the behavior of the IOCB is identical for both of these modes; it wouldn't be able to tell the difference.

Organizing the interrupt functions this way gives equal hierarchy across the board. It then becomes a matter of which data enters the chip first. For instance, flipping the selector switch would trigger the Port 2 Interrupt, and direct the program into the Port 2 function. While in this function, interrupts from both the MIDI and Port 1 buttons are ignored. The program remains in this function, and only exits after the switch is opened. Once returned to the Main Function, the process repeats and the program waits for the next flag to raise. To ensure consistent, problem-free operation, the interrupt flags are appropriately cleared and monitored by the three interrupt functions. This way, no

actuators get stuck open or closed, and the program will always return to the Main Function once the input is processed. The structures of these functions can be seen below in Figures 7.2, 7.3, & 7.4.

Port 1 Interrupt Function: Fully Manual Operation

The Port 1 function flag is raised when one or more of the four input buttons are pressed, and the Port 2 and MIDI flags are lowered. The process begins by running a small delay, around 25000 clock cycles. This prevents any flawed data caused by button bouncing from making it through. Assuming the button was actually pressed, the program quickly checks the status of all four inputs. It checks the Port 1 Input Register by masking the bit of that specific movement. If the button is pressed, the corresponding output is turned on; if the button is not pressed, the output is turned off, and the corresponding Port 1 interrupt flag is cleared. It will check all four inputs every time, even if only one button was pressed. It checks them sequentially from mouth, head, body, to tail; meaning if all four buttons were to be pressed simultaneously, they would turn on in the order the function checks them. This would cause only a few microseconds of delay, and wouldn't be perceivable by the user. Also, the status of each button is independent from the other; any possible combination of the four buttons can be toggled without issue. After all four functions are checked, the program returns to the Main Function. Figure 7.2 illustrates this process:

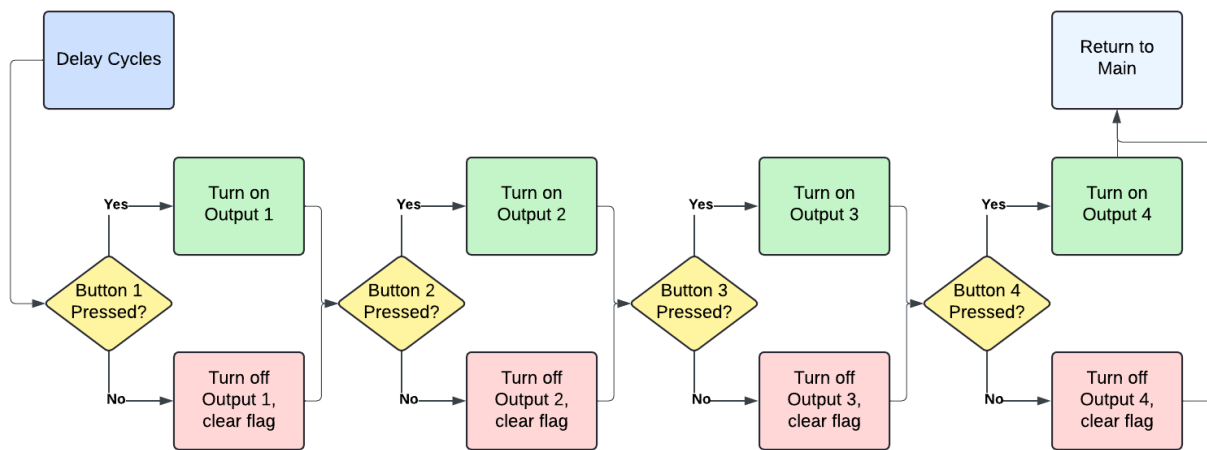


Figure 7.2: Process within the Port 1 Function

The four interrupt flag bits in the Port 1 Interrupt Register are left raised until their corresponding buttons are depressed. Meaning if the user were to press and hold a button down, the Port 1 function would be called repeatedly until the user releases. When this occurs, both the Port 2 and MIDI flags are ignored. The program wouldn't acknowledge either of those raised flags until all four buttons are depressed. Once all four buttons are depressed, the Port 1 function will then clear all four flag bits and the program will return to the Main function to wait for the next interrupt. In the case of the Port 2 Interrupt, there is only one input assigned to that register; the selector switch. If the selector switch is turned on, this sends the function to the Port 2 Interrupt Function.

Port 2 Interrupt Function: Partial Manual Operation

When the selector switch is turned on, the Port 2 Function is called. Counter variables are defined and initialized for the head, body, and tail. These are the three movements that will be automatically toggled by the program, leaving only the mouth to be manually controlled. The program enters a “while” loop that terminates whenever the switch is released. For every iteration of this loop, a 25,000 clock cycle delay is introduced to slow the process down slightly. The mouth input is checked in the exact same way as the Port 1 Function; if the Port 1 Input register shows the mouth button was pressed, the output is toggled on, or else the output is left off. This also allows the user to use the VOX circuit, as it toggles the mouth input exactly the same way as the physical button. The input buttons for the remaining three movements are completely ignored by this function. Instead, the function relies on the values of the counter variables to toggle them. When each variable reaches a certain predefined value, that respective output is turned on. When each variable reaches another predefined value, that respective output is turned off and the variable is re-initialized at 0. Setting each of these predefined values to different random numbers will then create the illusion of unplanned random movement of the three joints. Figure 7.3 illustrates this process:

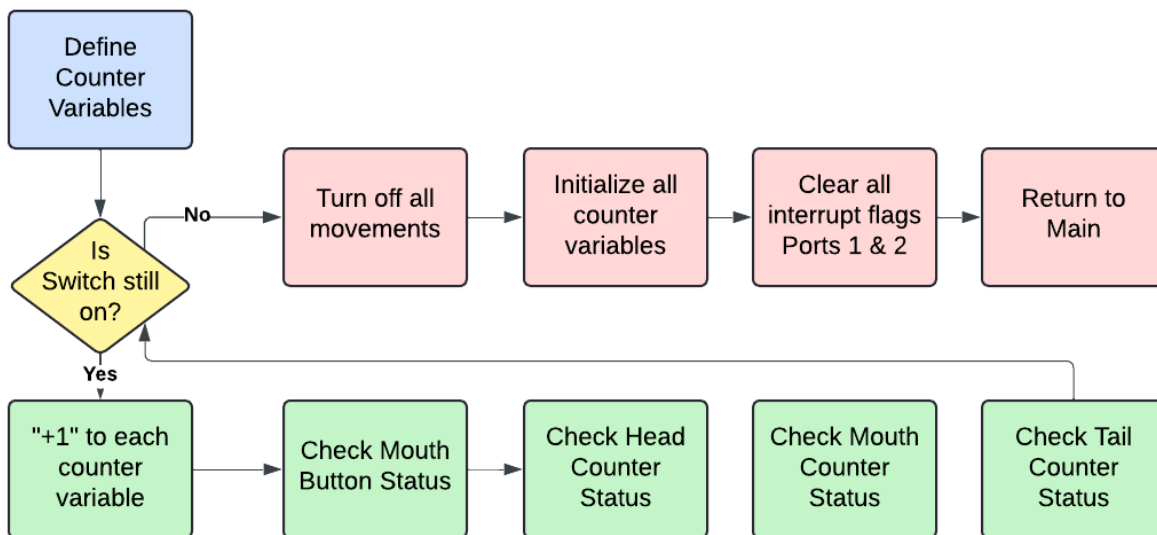


Figure 7.3: Process within the Port 2 Function

When the switch is turned off, the program exits the “while” loop, turns off all the outputs, initializes the variables, clears flags for Ports 1 & 2, and returns to the Main Function. Clearing the Port 1 Flags prevents the program from immediately entering the Port 1 Function, for any buttons that may have been pressed while in Port 2. Both the Port 1 and MIDI interrupt flags are completely disregarded while in the Port 2 Function. Although this concept applies to all three functions, the Port 2 Function is the most strict, as the only interrupt function that has a “while” loop. That loop will be active for as long as the switch is toggled, which could be for as long as the user wants. Port 1 and MIDI behave similarly, in that they only ignore the flags for as long as the user holds down a movement. As soon as the button is released, or the “release note” MIDI

message is sent, the program resumes its idle state. The MIDI function responds the quickest out of the three, as there is no need to use the “_delay_cycles()” operation at any point. As soon as the message is sent, the program immediately responds and acts accordingly.

UART Interrupt Function: Pre-Recorded Sequences and Fully Live Operation

Both the Port 1 & 2 functions are self constrained; they perform all the checking and switching and immediately return to the Main Function. The UART function is the only one that calls out to other external functions; all of which are implicitly declared at the top of the program code above the Main Function. As soon as the UART flag is raised, the interrupt sends the program to the UART Interrupt Function, which immediately calls to the “Handle MIDI” function to decipher the UART RX Buffer. Every MIDI message consists of three bytes: the status byte, the note data byte, and the control data byte, sent in that order. MIDI is standardized so that all status bytes have an MSB of ‘1’, and all data bytes have an MSB of ‘0’. When the three byte message is sent, the message is sent one byte at a time, and the program has to figure out what each byte is. Once all three bytes are received, the program reads and divides the message into four variables: “status”, “channel”, “note”, and “velocity”. Channel and velocity don’t have any effect on the movements, so they are ignored. The status and note variables decide whether a movement is on or off, and which movement it corresponds to respectively. It is the value of this status byte that dictates which function the program calls to next. Figure 7.4 illustrates this process:

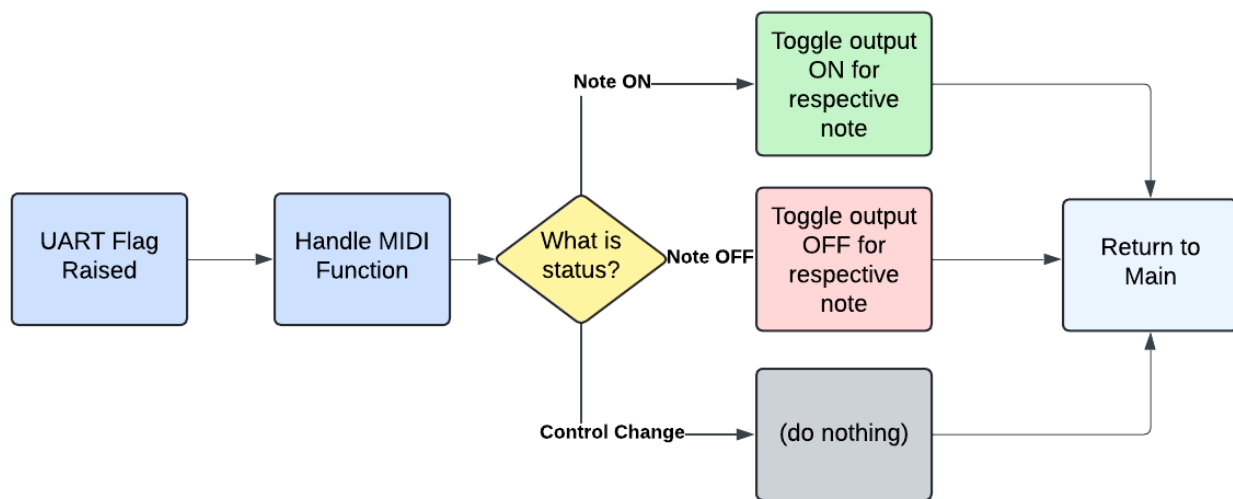


Figure 7.4: Process within the UART (MIDI) Function

Status bytes can have three distinct hex values: “0x90” means “NOTE ON”, “0x80” means “NOTE OFF”, and “0xB0” means “CONTROL CHANGE”. The zero in each of those messages means that the channel is selected as “CHANNEL 0”, but as mentioned before, this is completely irrelevant to CAPER’s operation. For each of the three possible outcomes, there are three external functions that can be called: “handle note on”, “handle note off”, and “handle control change”.

The “handle note on” and “handle note off” functions can toggle any of the four movements, depending on the value of the “note” variable. The four movements of CAPER were assigned to notes C4, D4, E4, and F4 on the standard piano, as seen in the chart below:

Table 7.0: Note & Movement Hex Values

Note	Movement	Note variable hex value
C4	Mouth	0x3C
D4	Head	0x3E
E4	Body	0x40
F4	Tail	0x41

Logically, putting together the status and data bytes yields the following messages: (bit values noted as a “#” are irrelevant and ignored by the program”)

Table 7.1: Complete MIDI Messages and Corresponding Movements

Message	Action
0x9# 0x3C 0x##	MOUTH OPEN
0x8# 0x3C 0x##	MOUTH CLOSE
0x9# 0x3E 0x##	HEAD DOWN
0x8# 0x3E 0x##	HEAD UP
0x9# 0x40 0x##	BODY DOWN
0x8# 0x40 0x##	BODY UPRIGHT
0x9# 0x41 0x##	TAIL UP
0x8# 0x41 0x##	TAIL DOWN

Those are the eight possible messages that can trigger movement. Notes that aren’t C4, D4, E4, or F4 would be completely ignored by the program. All messages beginning with “0x90” would get sent to the “handle note on” function, and all messages beginning with “0x80” would get sent to the “handle note off” function. If a status byte of “0xB0” was ever sent, the message would get sent to the “handle control change” function, where nothing would happen. A control change status byte would never be sent from the VRB or any upstream MIDI device, but may accidentally be sent while testing

CAPER with a digital piano (as I have done before). For this reason, the “handle control change” function is still left in the program to prevent any possible synchronization issues. Once all three bytes are processed, the program returns to the main function for the next awaited input to be received.

7.2 The Unified Conversation Module (UCM) Design

The UCM design is split into two main sections:

The local section, or Edge Module (EM) is meant to be run entirely on the Jeston Nano. It is composed of python 3.10.0 code that cleans up incoming audio, converts it into text via OpenAI’s Whisper-Small, and converts the text output of the offboard response generator into audio for the speaker to broadcast out as sound.

The offboard section, or Cloud Module (CM) consists of all the code that enables the efficient operation of the Mistral 7b Instruct LLM. This includes the initialization code for the model and the server setup code for allowing the local and offboard sections to communicate safely. The following subsections will explore the design of each of these sections in detail: the Edge Sub-module, and the Server Sub-Module.

Edge Sub-Module:

As outlined earlier the sub-module’s purpose is to host all the voice components of the UCM.

The system will be entirely coded in Python 3.10.0 using PEP8, PEP484, and PEP257 standards for increased readability and scalability. The code will have the following attributes:

Table 7.2: Edge Sub-Module Attributes

Attribute Name	Attribute Value
Programming Language	Python 3.6.9
Programming Paradigm	Procedural Programming
Coding Standards	PEP8, PEP484, and PEP257
Number of sub-modules (.py files)	5
System Style	Publisher-Subscriber

The entire edge sub-module is designed to react with minimal overhead by utilizing the MQTT framework, coordinated by the Mosquitto MQTT broker, to run multiple scripts of both the subscriber and publisher types. The publisher subscriber system is illustrated below:

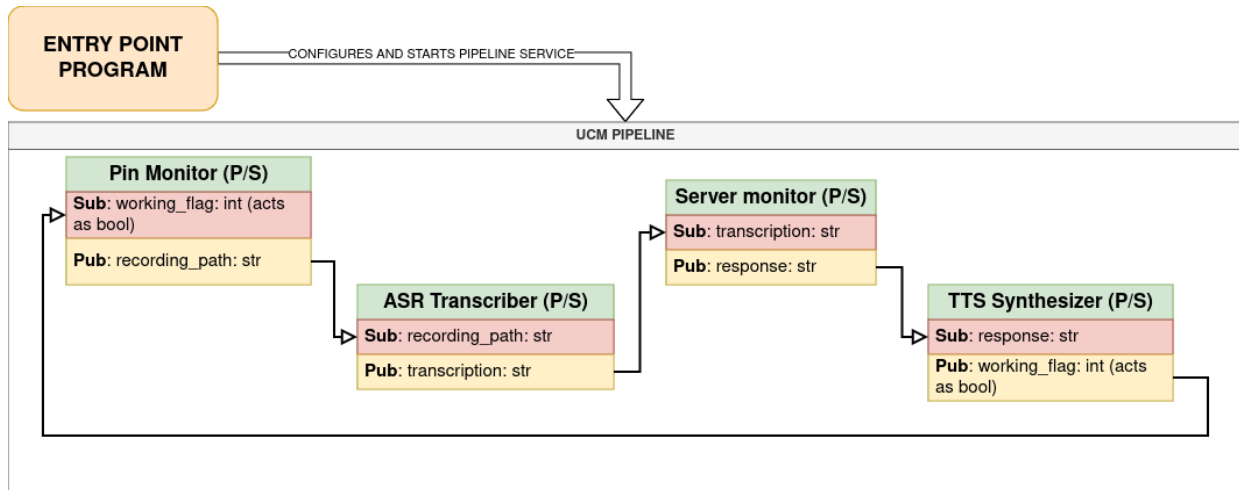


Figure 7.5: Publisher-Subscriber System Plan

The system consists of 5 scripts each with a specific role within the pipeline. A list of each along with an overview and operational sequence of events can be found below.

1. Global Configuration (**global_config.py**)

This header manages and provides configuration parameters to other components within the sub-module, ensuring consistent settings across the platform. Topic definitions within the global configuration.

Table 7.3: Topic Definitions

Topic Designator	Topic Value
pin_monitor_P	/pin/path
pin_monitor_S	/tts/flag
asr_P	/llm/input
asr_S	/pin/path
llm_P	/tts/input
llm_S	/llm/input
tts_P	/tts/flag
tts_S	/tts/input

2. Pin Monitor (**pin_monitor_PS.py**)

This script monitors and controls pin states on Jetson Nano, to control audio recording. The operational flow is as follows:

1. The script subscribes to the topic containing the flag (**pin_monitor_S**) indicating pipeline status.
2. It prepares to publish its output file path data.
3. The script continuously checks for a rising edge on Jetson Nano PIN 7 via the Jetson.GPIO library.
4. Once a rising edge is detected, the script checks if the pipeline is ready to both transcribe and broadcast, based on a flag sent by the last component of the pipeline (tts_synthesizer).
5. If the pipeline is confirmed ready, the script starts recording until a PIN 7 falling edge is detected.
6. The recorded audio is then analyzed to determine if the signal is long enough to contain speech.
7. Once all conditions are met, the path to the audio file is published on **pin_monitor_P** topic for the transcription system to receive.

Its behavior is illustrated below:

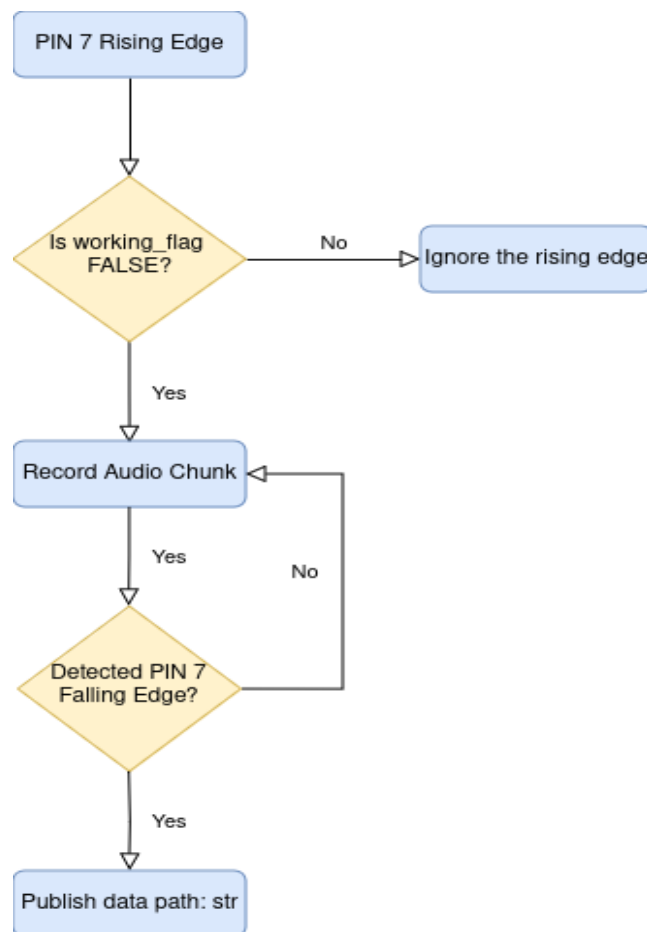


Figure 7.6: Pin Monitor Flow Chart

3. Automatic Speech Recognition Transcriber (**asr_transcriber_PS.py**)

This script converts spoken audio from users into text for processing by the large language model. The operational flow is as follows:

1. The script subscribes itself to the file path topic (**asr_S**).
2. It prepares itself to publish the resulting text.
3. Initializes OpenAI's Whisper-Small model.
4. Receives the path for the audio file.
5. Performs sampling checks to ensure the audio is at the required 16kHz.
6. Trims and resamples the audio as needed.
7. Transcribes the audio into text of type string.
8. Publishes the transcribed text to the **asr_P** topic.

4. Large Language Model Generator (**llm_generator_PS.py**)

The script sends the data to the server for onboard processing. Once a response is received, the json is decoded and the text is published to the tts synthesizer.

1. Subscribes to **llm_S** topic for text input.
2. Sends the data to the server for processing via the small API.
3. Waits for the server to respond with the processed data.
4. Decodes the received JSON from the server.
5. Extracts the text from the decoded JSON.
6. Publishes the extracted text to the **llm_P** topic, directing it to the TTS synthesizer.

5. Text-to-Speech Synthesizer (**tts_broadcaster_SP.py**)

This script converts text responses into spoken audio, which can be played back to the user. In addition it creates the robotic movement commands and translates them into MIDI for transmission via the Jetson Nano's USB port to the IOCB (input-output control board) The operational flow is as follows:

1. Subscribes to **tts_S** topic for LLM response input
2. Connects to the Jetson Nano's UART system.
3. Initializes SpeedySpeech via the TTS.api library
4. Converts the received text into spoken audio.
5. Converts the .wav audio to mel spectrogram (mel_spec).
6. Creates movement commands based on random numbers derived from the mel_spec, where volume intensity increases movement
7. Translates the movement commands into MIDI.
8. Plays back the generated audio.
9. Sends the MIDI commands via the Jetson Nano's USB port to the IOCB.
10. Publishes False to the **tts_P** topic indicating that the pipeline is no longer working

A breakdown of all libraries used in the edge portion of the UCM is outline below:

Table 7.4: Edge Sub-Modules Libraries Table

Module	Associated Libraries
Common Across All Modules	paho-mqtt, os, sys, json
Pin Monitor (pin_monitor_PS.py)	Jetson.GPIO, threading, sounddevice
Automatic Speech Recognition Transcriber (asr_transcriber_PS.py)	whisper, librosa, soundfile
Large Language Model Generator (llm_generator_PS.py)	requests, flask (optional)
Text-to-Speech Synthesizer (tts_broadcaster_SP.py)	TTS, librosa, mido, serial
General Purpose	numpy, scipy, logging

Server Sub-Module

The role of the server sub-module is to create an API entry point for the edge sub-module to access. This API is configured locally using Flask and hosted using Gunicorn.

The server will be built by a single python file that configures the server and loads the LLM for local use.

System Sequence of Events:

The following list describes the series of events for when the script is used as a source app in Gunicorn.

1. Model and Tokenizer Initialization:
 - a. Loads a pre-trained causal language model (AutoModelForCausalLM) from a specified local path and configuration.
 - b. Initializes a tokenizer (AutoTokenizer) that matches the model's expected input format.
2. Pipeline Creation:
 - a. Creates a text-generation pipeline (pipe) object that combines the loaded model and tokenizer to facilitate the generation of text based on input prompts.
3. Flask App Initialization:
 - a. Instantiates a Flask application object to define endpoints and start a web server.

4. Basic Authentication Setup:
 - a. Defines a dictionary (users) containing username-password pairs for authentication.
 - b. Sets up an authentication mechanism using HTTPBasicAuth to secure endpoints.
5. Global Variables:
 - a. Declares and initializes conversation_history to store the history of interactions.
6. Function to Append to History:
 - a. Defines append_to_history to update conversation_history with user inputs and model responses.
7. Function to Count Tokens:
 - a. Defines pre_tokenize that converts text to tokens using the tokenizer, printing and returning the token count.
8. Interactive Prompt Function (Local Testing):
 - a. It starts by prepending user input with stored conversation history for context.
 - b. Generates responses using the text-generation pipeline
 - c. Updates conversation history with the new exchange.
 - d. Strips history from response
 - e. returns stripped response as a string
9. Verify Password Function:
 - a. Defines a function to verify username and password for authenticated API access.
10. API Endpoint for Text Generation:
 - a. Defines a /generate endpoint in the Flask app that requires basic authentication.
11. Function for JSON conversion
 - a. Calls the prompt function with the appropriate str input
 - b. Converts the result into JSON and returns it
12. Run Flask Application:
 - a. Starts the Flask server with debug mode enabled on all network interfaces and a specified port.

The table below outlines all modules used:

Library	Purpose
flask	Web framework to create and manage web server, routes, and endpoints.
flask.request	Used to handle incoming request data (e.g., JSON from API calls).
flask.jsonify	Converts Python dictionaries to JSON responses for API outputs.

<code>flask_httpauth</code>	Provides mechanisms for securing Flask routes with authentication.
<code>transformers.AutoModelForCausalLM</code>	Part of the Hugging Face Transformers library, used to load pre-trained causal language models.
<code>transformers.AutoTokenizer</code>	Also from Hugging Face Transformers, used for tokenizing texts to model-appropriate input formats.
<code>transformers.pipeline</code>	Simplifies the inference process, providing an easy-to-use interface for applying pre-trained models.

Table 7.5: Server Libraries Table

Chapter 8.0: PCB

For Senior Design, one of the main requirements for the project is to create a printed custom board (PCB) that makes a significant contribution to the functionality of our robot. Luckily, the professors informed us that the design does not need to be created directly from scratch. Rather, existing software and hardware components can be used to create our design.

The PCB is crucial to the success of our project since it distributes electrical power to CAPER. As such, to guarantee the production of a working PCB, appropriate design decisions and developing processes must be followed. For design decisions, every circuit in the complete schematics was analyzed and explained in detail in Chapter 6, which also covered the functions of each circuit. Overall, this section describes the PCB software we used, PCB planning, the PCB schematic, and the initial construction of the ICOB prototype.

8.1 PCB software

When it came to choosing what software to be used for PCB design, we quickly narrowed it down to two options: Altium and Eagle. This is because both of these come with a student license, meaning we could easily obtain access to the student versions of the software for one year if using EAGLE or six months if using Altium. However, due to EAGLE having limitations such as board size and the number of layers allowed, we ultimately decided on Altium.

Altium

Altium is the #1 choice for electronics design, allowing users to create everything from schematic to simulations to final design all in one interface. Its functionality allows the software to cater to the needs of complex and intricate projects. Due to the vast amount

of features that Altium possesses, the regular license being \$355 a month or \$11,970 for a perpetual license, larger companies tend to use this software.

Despite the hefty price tag, luckily for us students, there exists an educational license that is accessible through providing proof of academic enrollment, such as, signing up using a student email. This allows us to utilize the PCB building and designing features of what normally would be exceptionally expensive software, at no cost whatsoever for a period of six months. This is also the one case where taking Senior Design II is more beneficial in the Summer semester rather than the Fall semester. Signing up for the license in March, the expiratory period would be in September. This means that the license will be active through half of Senior Design I and all of Senior Design II.

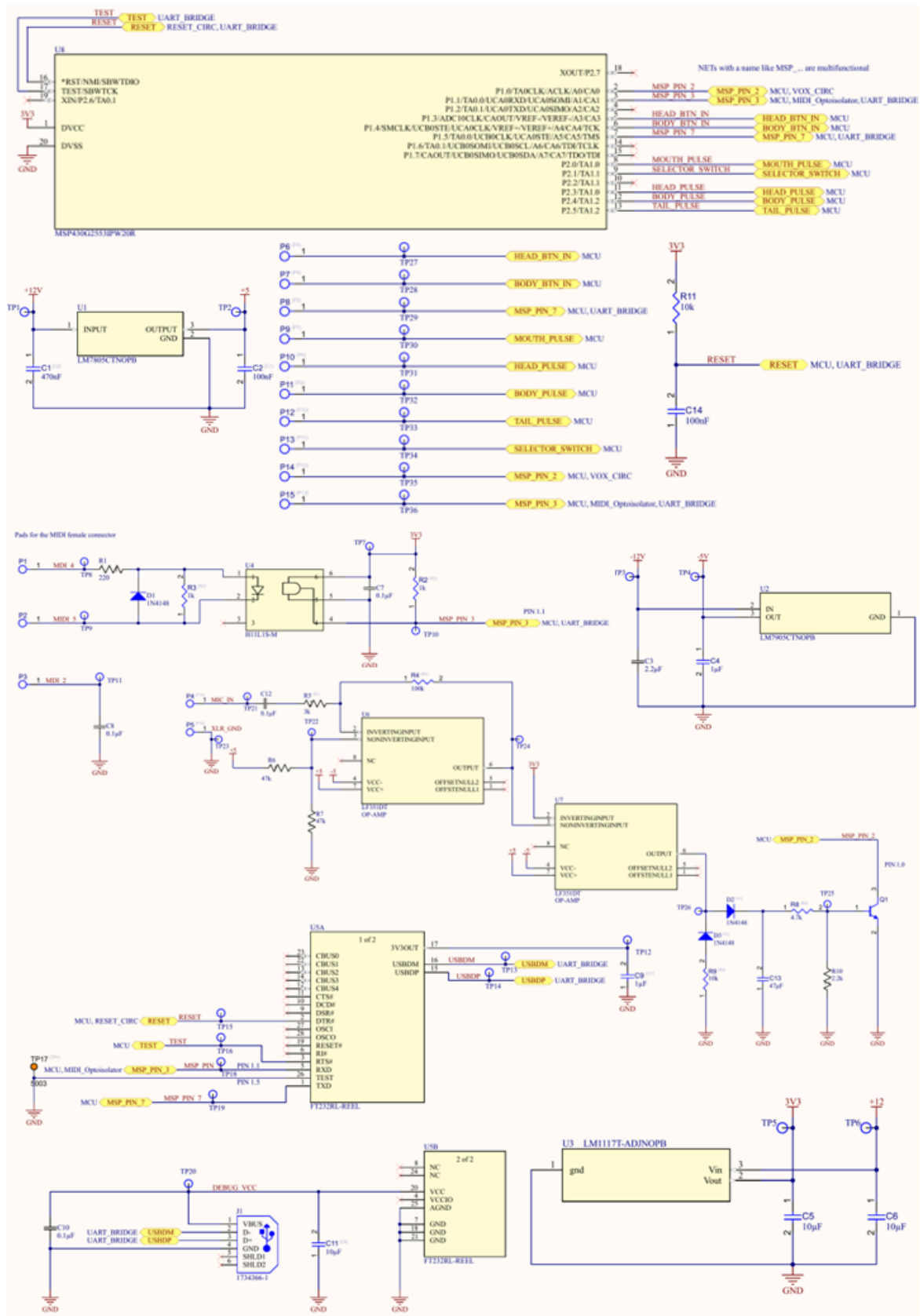
Not only does Altium offer a user-friendly interface for both layout and schematic design, it also comes with an excellent management system. Altium efficiently allows users to export all necessary files to get a quote and printed board. It is also easy to make changes to BOMs without having to recreate them every time. Its popularity in the industry also means there are extensive resources for troubleshooting. What brings Altium above the rest of the potential PCB softwares is the ability to 3D render our board, meaning that we will be able to have an idea of how the board will look in real life as we will have the ability to rotate the model to check if everything is as designed. We hope to achieve a fully functional and completed PCB design early into Senior Design II.

8.2 PCB planning

In order to create a reliable design, it is important to follow proper component and placement and grouping. Component placement should aim to produce a board that is simple to route, ideally with the fewest number of possible layer transitions. The design must also meet component placement requirements and adhere to design guidelines. Though balancing these points can be challenging, a straightforward procedure can assist a board designer in positioning components that satisfy these specifications. Because of mechanical enclosure limitations or simply because of their size, components frequently have to be installed in certain places. Prior to moving on to the rest of the plan, it is advisable to arrange these elements first and ensure they are secure in place. Parts like CPUs or high pin count integrated circuits typically need to be connected to other parts of the design so grouping specific components makes trace routing simpler. Overall, minimizing the amount of crossing nets will make it easier to implement since each intersection requires vias for layer transitioning. It is possible to get around this through creative placement by rotating and grouping components.

8.3 PCB Schematic

Several pre-existing circuits were put together to create our PCB. Refer to the datasheets in the Appendix for where the circuits were referenced from and chapter 6 for a deeper look on these circuits. Below is the finished board schematic as of Senior Design I. The full schematic below is blurry due to attempting to fit eight circuit groups into one drawing. Necessary adjustments will be made throughout the testing process during Senior Design II in order to have a full functioning PCB layout.



8.4 Initial Construction of the IOCB Prototype

The prototype for the IOCB consists of the MSP430G2ET Launchpad board, and the breadboards for the push buttons, the selector switch, the UART optocoupler, and the VOX circuit. The Launchpad board has all the necessary peripherals installed on it to use the MSP chip, as well as to power the optocoupler. Using jumper cables, we connected the pins from the Launchpad over to the respective breadboards. In total, there were 9 connections to the buttons, 2 connections to the optocoupler, and 1 connection to the VOX circuit. This does not include the ground connections shared across the breadboards. Below is a photograph of the entire circuit, hooked up and placed upon a clipboard:

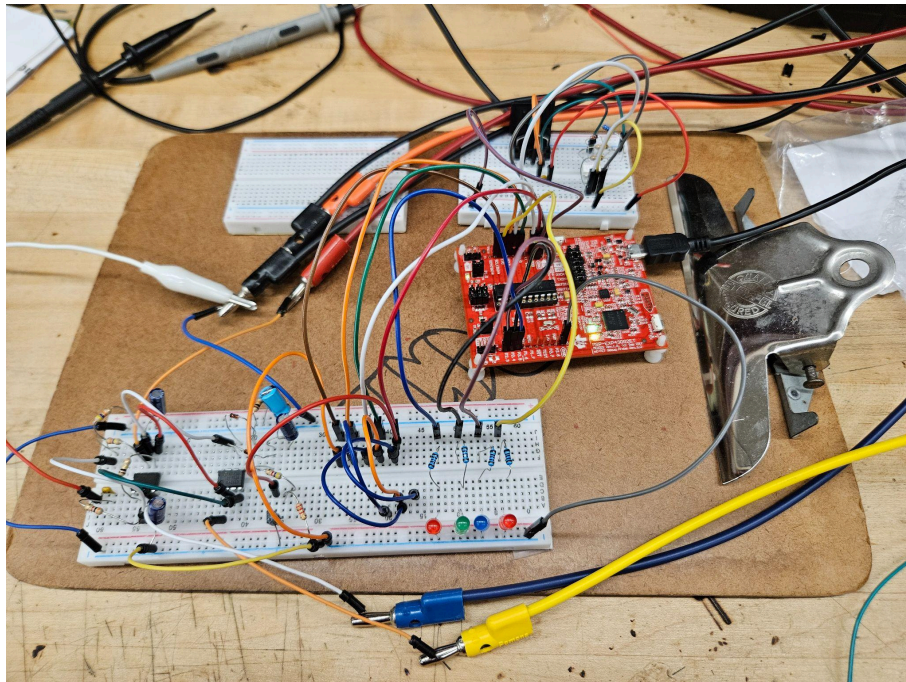


Figure 8.2: The complete prototype of the IOCB

The various banana plug cables seen in the photograph are from a power supply located off-screen. This photo was taken when testing the VOX circuit with the rest of the board. Information on how this was done, along with how the other peripheral circuits were tested will be discussed in the next chapter. There, all the results from the various conducted tests will be displayed, along with the future plans for the official construction of CAPER.

Chapter 9.0: System Testing

This chapter discusses the testing of the IOCB, as well as our future testing plans for Senior Design II.

9.1 IOCB Testing

The initial build of the IOCB required prototyping several of the onboard peripherals, specifically the UART Optocoupler, and the VOX circuit. Of all the circuits, these two were easily the most important circuits to build, as their designs were never proven to work in this configuration, and CAPER couldn't work without them. Given this, it was imperative that the VOX circuit and UART Optocoupler were built first, and tested extensively to ensure their compatibility with the MCU. Once solid designs were developed, the schematic drafting process began. Of course, before any of this was possible, we had to get the Launchpad board up and running.

Using the MSP430G2ET Launchpad

Working with the Launchpad as a matter of knowing how to program, as all of the necessary power and debugging hardware was already included on the board. This version of the MSP Launchpad conveniently has the exact same chip that is used on the IOCB: the MSP430G2553. This version of the chip is the 20pin, through-hole package called the "IN20". The version on the IOCB will be the surface-mount version, called the "IPW20"; the functionality of both chips is identical. Ideally, we can perfect our programs on the easy-to-handle Launchpad, and then install them onto the IOCB, knowing exactly what the outcome will be. If all the kinks are worked out now in the software, the programming of the IOCB would only have to happen once.

Blinking the LEDs

One of the very first programs you are introduced with when using the MSP is the "blink the LED" program. This program actually comes standard with Texas Instruments Code Composer Studio, as one of the default projects you can build. The underlying principles inside that simple project, resonate strongly with the final IOCB code. The program generates different outputs based on certain input stimuli. It was just a matter of finding the correct pins to use, and allocating them as such. It was eventually decided to have the four inputs all on Port 1, and the four outputs on Port 2. The selector switch input was also placed on Port 2, originally because of the lack of pin availability on Port 1. This didn't seem like a monumental decision at the time, but it proved to be one of the best decisions made in the course of this project (more on this later).

Initially, a simple program was created that would operate the four outputs according to the position of the four inputs. This was essentially the first mode: Full Manual operation. The four LEDs were connected to the four outputs on Port 2, each with a 1 kiloOhm current resistor to limit the current. In place of push buttons, basic breadboard jumper cables were used. Connecting the jumper to the input pin and shorting it to ground behaves exactly like a button would in active-low configuration. With the internal pull-up resistors activated, there was absolutely no possibility of damaging the chip from current overload. After debugging the chip it worked perfectly; each of the four jumper wires toggled their respective output LEDs when shorted to ground. The following

pictures display this behavior in action. The first picture shows all the jumper wires connected to ground, and all four LEDs are turned on. The second photo shows all the jumper wires disconnected from ground, and all the LEDs are off. For Figure 9.1 the four jumper wires were plugged into random, unused nodes on the breadboard for the sake of cable management. Since those nodes are completely unused, plugging the jumpers into them is exactly the same as leaving them open; it just looks neater. You can see those same four jumpers plugged into the ground rail in Figure 9.0, thus triggering the LEDs.

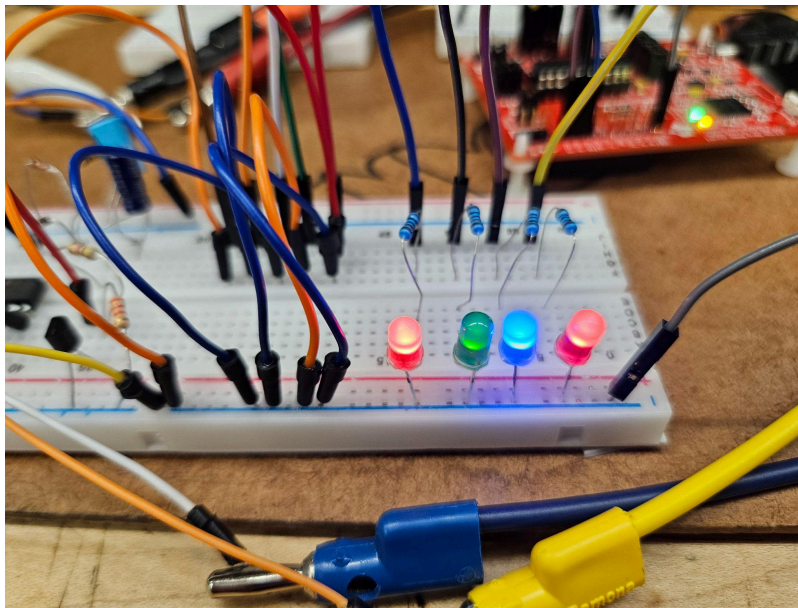


Figure 9.0: LEDs switched on

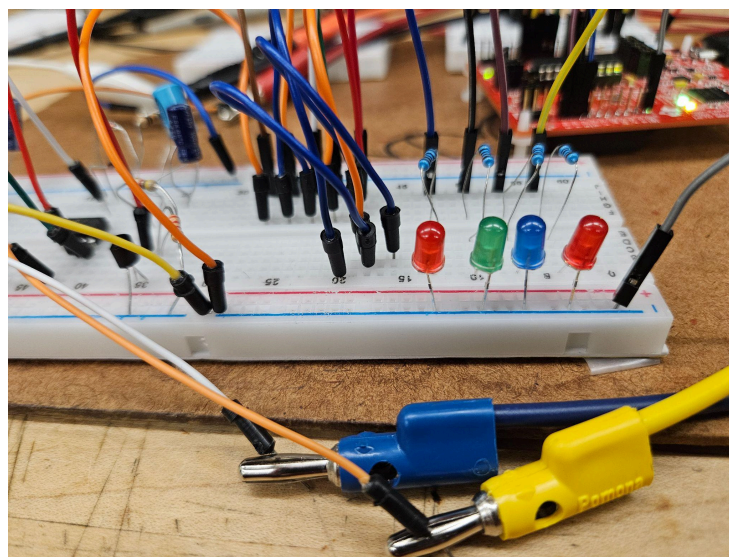


Figure 9.1: LEDs switched off

Once the pin locations were figured out, it was time to get the Partial Manual Mode up and running. This meant using the single Port 2 input as a selector switch. Within the

main function of the code, a nested “while” loop was placed to check if the Port 2 switch was ever closed. Closing this switch would cause the program to enter the nested “while” loop and stay there until the switch is released. Using counter variables, the nested “while” loop would automatically toggle the body, head, and tail; leaving the mouth to still be controlled manually. This function worked perfectly from the beginning.

The UART Optocoupler

The next piece of hardware created was the optocoupler circuit that would be used for MIDI. The premise of this circuit is that data would enter as a 5 volt pulse wave, and exit as a 3.3v pulse wave. This circuit was built on a small breadboard seen in the photo below:

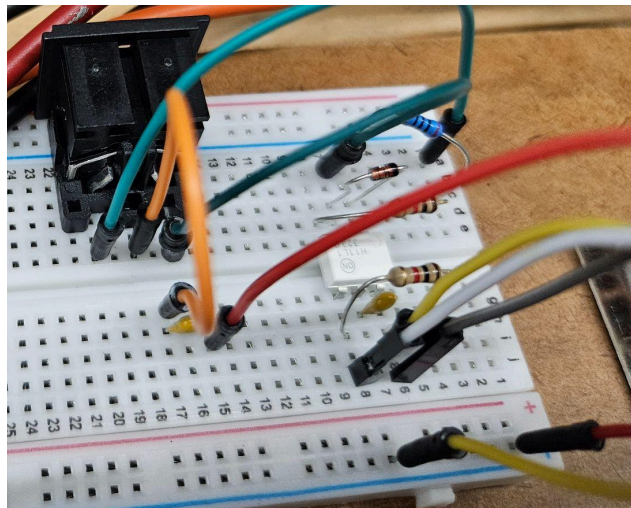
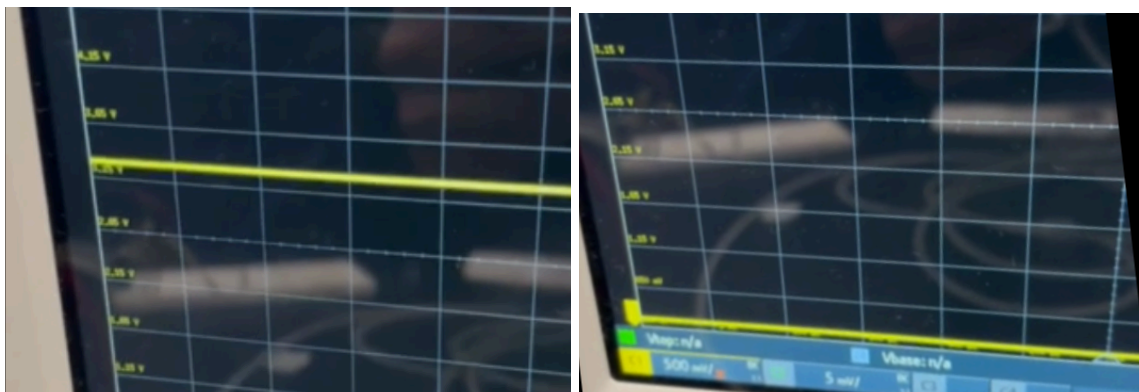


Figure 9.2: The UART optocoupler for MIDI on a breadboard

If this circuit was behaving correctly, placing a 5 volt signal at the input pins, should cause a 3.3V signal to show up at the output. Trying this with a DC power supply showed me that the optoisolator was working fine; only problem was it was behaving precisely the opposite from what was expected. Applying 5V at the input caused the output to swing low to 0V. Removing the 5V caused the output to swing high at 3.3V. This output was monitored on an oscilloscope, the displays of which are shown below:



Figures 9.3 and 9.4: The optocoupler output swinging high with no voltage at the input (left), and the output swinging low with 5V at the input (right)

After looking at some of the datasheets, it turns out that that is exactly how the chip is supposed to behave. One of the datasheets even had a truth table explicitly defining the inverting behavior:

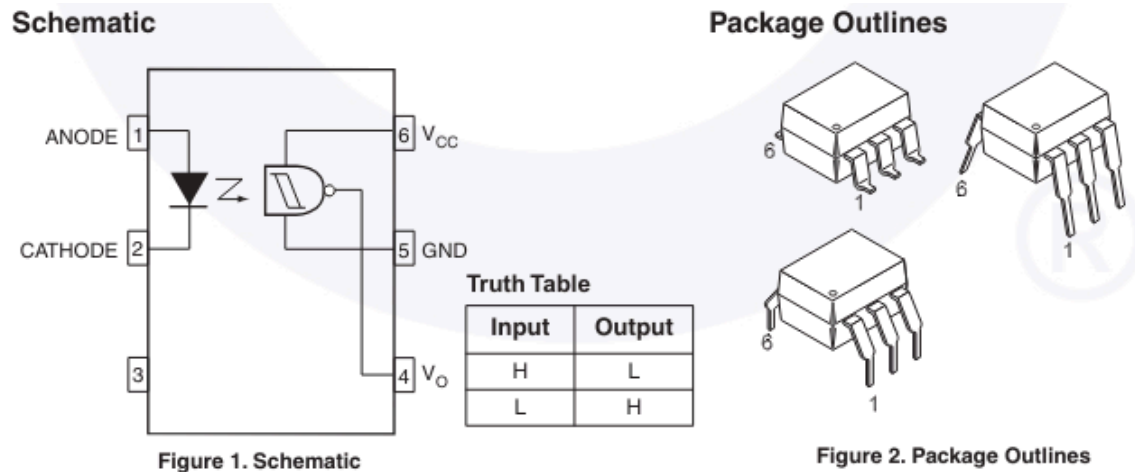


Figure 9.5 Fairchild H11L1 datasheet excerpt. Full datasheet page in Appendix C.

Given this insight, it was time to test the circuit using an actual MIDI signal. This meant creating the MIDI program first.

Testing the MIDI Functionality

Ruede(2014) compiled a set of code together that served as the skeleton for the MIDI function for CAPER. This original code can be found in Appendix D. “I think i got a pretty good solution. After 2 days of searching in the Internet, I finally found a fitting example, getting me to understand the handling of the MIDI message” (Ruede, 2014). After extensively modifying the code to fit the specifications of CAPER, it was time to test the code using an actual MIDI signal. For this, a Korg Trinity v3 DRS workstation keyboard was used. Despite being from 1995, it had full MIDI functionality, via a set of ports on the rear. A male-to-male MIDI cable was plugged from the output of the keyboard to the input of the optocoupler. Below is a photograph of the exact Korg Trinity that was used:



Figure 9.6: The 1995 Korg Trinity keyboard used to test the MIDI

With the heavily modified MIDI reading program, pressing the keys turned the respective LEDs on. This proved that the UART configuration was working perfectly. The only issue is that the LEDs wouldn't turn off when the notes were released. A few tweaks were made to the logical operators in the "handle MIDI" function and that cleared up the issue perfectly. Pressing the four keys would independently toggle their respective LEDs without any issues. The next step was to create a single program that toggled the outputs using the buttons and MIDI at the same time. This will be the code that gets installed on the IOCB during the actual construction of CAPER.

The Complete IOCB Program

Getting two completely different programs to work together can be difficult, as there usually is no way of establishing hierarchy among the different functionalities. This can cause issues when two or more program functions cancel each other out and override each other. This is exactly what happened during the first attempt of the complete code. The functions in the infinite "while" loop of the first program, were directly pasted into the infinite "while" loop of the MIDI program. This is where the program counter sits indefinitely, while waiting for the MIDI interrupt flag to be raised. Placing the code for the push buttons into this "while" loop completely disabled the program's ability to attend to the MIDI buffer. It was clear that this "while" loop needed to remain completely empty, and the Port 1 & 2 interrupt vectors needed to be used.

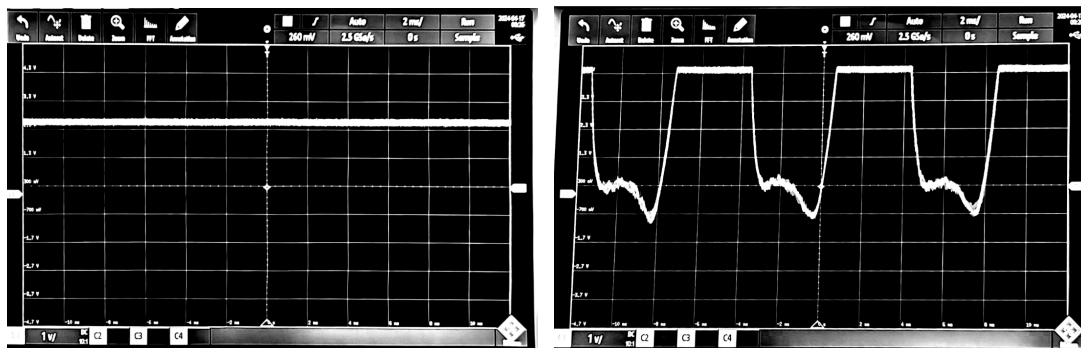
As mentioned briefly at the beginning of this chapter, choosing a Port 2 pin to be used as the selector switch input was the best possible decision. This gave the selector switch its own interrupt function, allowing for complete separation between Fully Manual, and Partial Manual operation. After just a few tweaks regarding the interrupt flag registers, the program was working absolutely flawlessly. The push buttons worked just like before, the selector switch activated the automatic movements, and the MIDI input triggered the LEDs. The program never crashed, glitched, or malfunctioned in any way that would require a system reset. The full operation of the IOCB program is explained in Chapter 7, and is included in its entirety in Appendix D.

Testing the VOX Circuit

The method of testing the VOX circuit involved building it one stage at a time, and testing the waveform at every stage. It was imperative that appropriate values were chosen for all the resistors, as the final version of this circuit will not be adjustable. Another major concern was the possibility of overcurrent malfunctions, specifically with the BJT or MSP. For this reason, we set the Vcc for both op amps to be +/-5V, with the intention of possibly raising the amplitude if needed. Ultimately, the circuit worked just fine with +/-5V and no change was needed.

The first stage of this circuit is an inverting op-amp stage, with an offset. Using the equation: **Gain = R_f / R_i** , original values included a 100k Ohm feedback resistor, and a 10k Ohm input resistor. This yielded a gain of around 10, which was quite insufficient. The output of the microphone peaked at around 20mV; which even at a gain of 10, only peaked around 200mV (this was while practically screaming into the microphone). The series resistor was eventually lowered to 3k Ohms, yielding a gain of around 33.

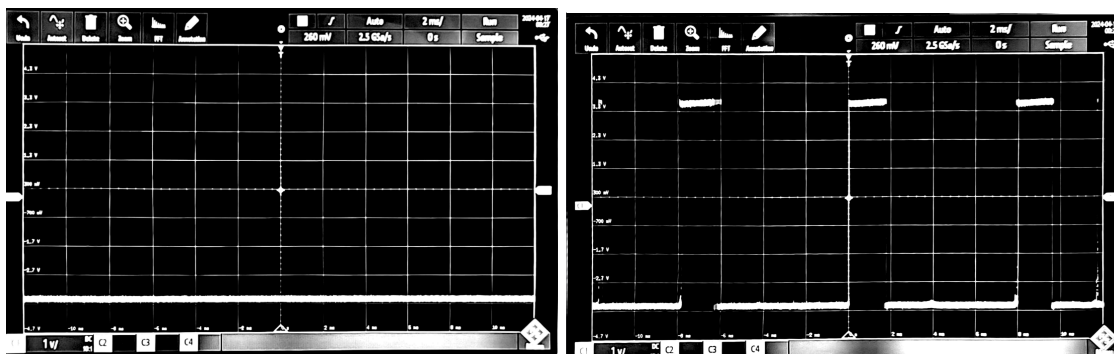
Combining this with the 2.5V offset, provided a much more usable signal for the next stage. The coupling capacitor placed before this stage removed all of the low frequency noise, providing a very clean signal with, and without speech present. This can be seen clearly from the following two oscilloscope readings in Figures 9.7 and 9.8:



Figures 9.7 and 9.8: Oscilloscope Readings after the first VOX circuit amplification stage

In Figure 9.7, the signal sits right at 2.5V, with no speech present at the microphone. The waveform is almost completely flat, with virtually no detectable noise. Speaking into the microphone yields a waveform similar to the one seen in Figure 9.8. Using a moderately loud speaking voice, a 5.5 to 6 V peak-to-peak audio wave was generated by the amplifier. The upper halves of this wave would be cut-off due to clipping, and the lower halves dropped as low as -800mV. Since fidelity isn't an issue, the clipping wouldn't make a difference to the performance; there was no need to raise the headroom of this stage.

The second stage of the VOX circuit was constructed; the output of the first stage was directed into the non-inverting input of the second, with a 3.3V reference voltage set at the inverting input. Without adding a feedback resistor, this created an "infinite" gain, essentially creating a comparator. When given the same Vcc as the first stage, this comparator would swing high to +5V, and swing low to -5V depending on the value seen at the non-inverting input. It would switch between these two values almost instantly, the only limiting factor being the op-amp's slew rate of 14V/microsecond. With the incoming audio wave set to idle at 2.5 volts, it was expected that normal talking volume would produce a pulse wave with 30 to 40% duty cycle. The oscilloscope readings in Figures 9.9 and 9.10 prove this to be true.

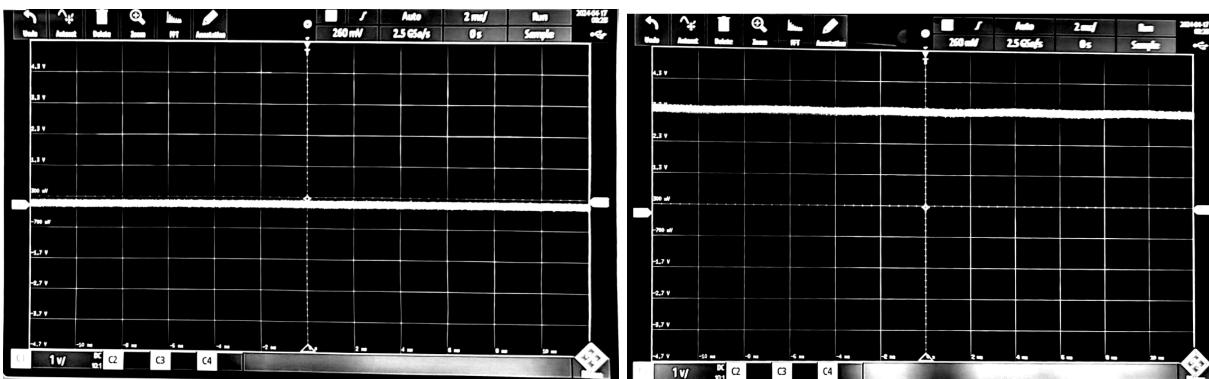


Figures 9.9 and 9.10: Readings from the second VOX circuit stage (the comparator).

These waveforms behave almost exactly as expected based on the design criteria. The comparator stays swung low when no audio is present, as seen in Figure 9.9. The comparator will swing high whenever the input voltage goes above 3.3V providing a 20% duty cycle pulse wave seen in Figure 9.10. The signal varies directly between -5 and +5 volts without any distortion or disfigurement. The duty cycle is slightly lower than originally expected, but this shouldn't affect the performance or accuracy. All things considered, the microphone sensitivity shouldn't be too high, or else it could trigger false-positives from random noises around the room. Given that possibility, the reference voltage is kept at 3.3V.

For the rectifier stage, only the positive halves of the cycle are needed; any voltage below 0V can be disregarded. For this reason a forward-biased diode is placed following the comparator; only allowing positive voltages to pass through. According to the datasheet for the 1N4148 diode, it can handle a reverse bias voltage of 70V before breakdown occurs. This means the rectifier could've been left as a half wave, completely disregarding the negative voltage. A second, reverse-biased diode was added anyway, to ensure no problems occur at the comparator op-amp. This reverse biased diode was connected to ground, in series with a 10k resistor. This keeps the op-amp under a load and allows current-flow at all times; positive or negative.

Immediately following the forward-biased diode was a capacitor shunted to ground. This capacitor would hold a momentary charge in between the peaks of the pulse wave. With the correct value, it would turn that series of individual peaks into one solid DC pulse. The oscilloscope printouts in Figures 9.11 and 9.12 show this behavior:



Figures 9.11 and 9.12: The signal after exiting the rectifier and capacitor

Figure 9.11 shows the rectifier output with no speech present. Instead of being a -5V DC signal, it is set right at 0V. Figure 9.12 shows the rectifier with speech present. The 5V pulse wave is converted to a DC signal at around 3.3V (this is purely a coincidence), with minimum ripple (around 1mV). This signal goes through a voltage divider dropping it down to around 1.6V, and sending it into the base of a BJT, where the collector would connect to the Mouth Pulse pin (3.3V), and the emitter goes to ground. At this point, no more oscilloscope readings were taken; the VOX circuit was connected to the mouth pin and tested with the rest of the IOCB powered on. The photograph in Figure 9.13 shows the now-working VOX circuit in action. It is obvious that the mouth LED is triggered by

the VOX circuit and not the buttons, because the jumper wires are disconnected from ground (LEDs switched off). Funny enough, they're located in the exact same unused spot, as seen in Figure 9.1

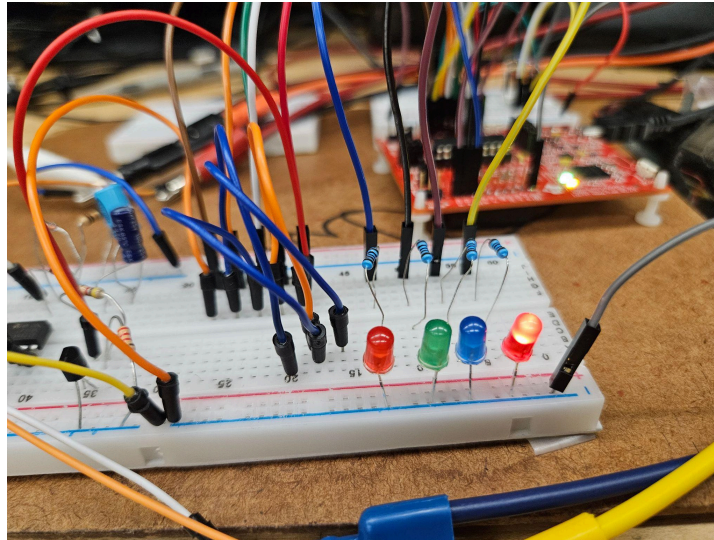


Figure 9.13: The LED being triggered by the VOX circuit

The VOX circuit performed beautifully; a fully-functional, voice-activated switch for the mouth. The original jumper wire for the mouth still worked, even with the new VOX circuit hooked up. Using both at the same time caused no issue, as both the VOX circuit and jumper wire were just parallel pathways to ground. This is good to figure out now, as there can't be any ground loops or faults when the IOCB is actually built. Since no problems occurred, no corrective action was taken. The VOX circuit in its entirety can be seen below in Figure 9.14:

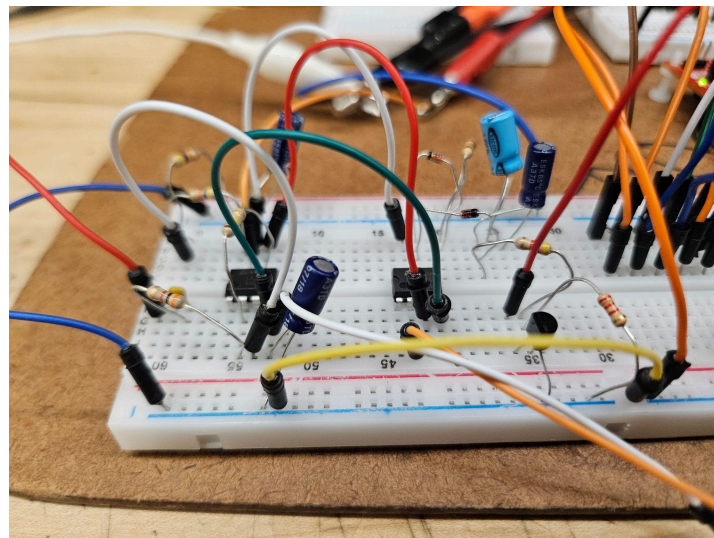


Figure 9.14: The completed VOX circuit on a breadboard

9.2 The Next Steps

Going into Senior Design 2, the plan is to slowly move away from the Launchpad and breadboards and to finally create the custom-made board. The schematic designs are already completed; now it's just a matter of creating a logical board layout to house these circuits. The testing phase gave us the insight we needed to safely move ahead and assemble the board. Now that we have the proper waveforms and values, we can more easily detect problems and fix them before irreversible damage occurs.

Once the work order is sent out to get the board made, we can begin to focus on the secondary aspects of the project. This includes designing the chassis for the boards, constructing and decorating the parrot figure, and setting up the miscellaneous external circuits: the relays, the speakers, the amplifier, the MIDI playback etc. Parts of the final parrot figure are already assembled including the 3 main links of the body frame. They are pictured below alongside the four solenoid actuators:

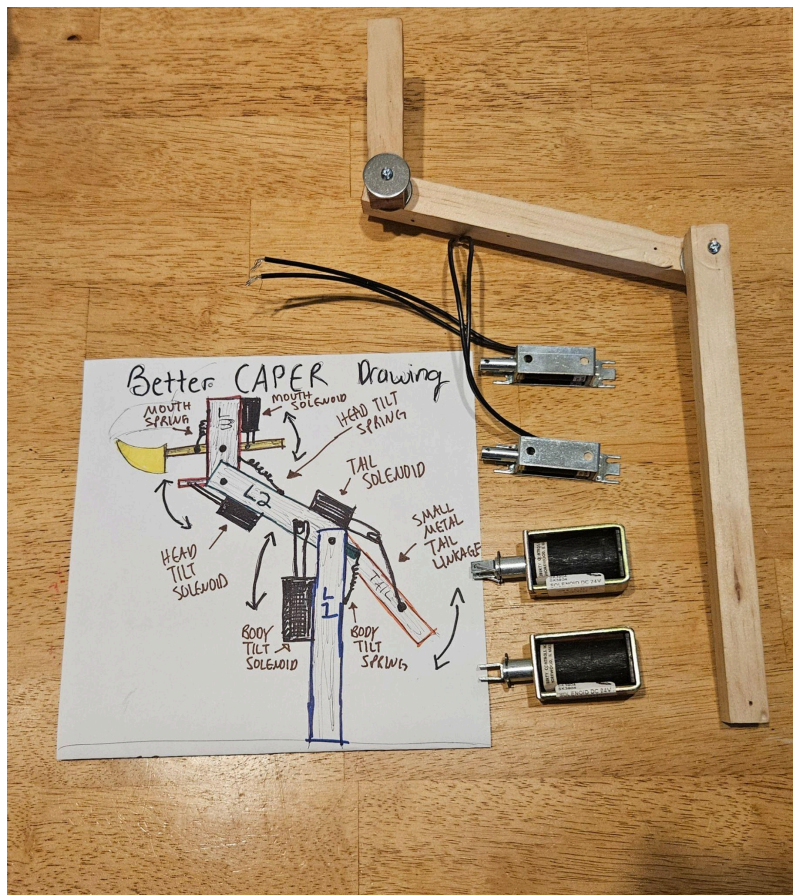


Figure 9.15: Finished Main Body frame and solenoids, next to early concept art

Despite the main visual attraction of CAPER being the parrot figure, most of the work throughout senior design 2 will be perfecting the software and electronics. Mechanically, CAPER is quite simple; once mounted to a sturdy base, the figure will go together much quicker.

We'll need to prepare for the worst; expecting good results, but planning for whatever unexpected mishaps may occur. Components that are easily tarnishable or fragile will be purchased in bulk, as prices are low and lead-times are high. Any large-scale changes that need to be made will need to be done as early as possible. This way, we'll have time to make new schematics, and send out an order for an updated board. Given the extensive planning we've done so far, this shouldn't happen; but it is better to be safe than sorry.

Another key goal is optimization. This applies more to the VRB than it does to the IOCB, as the IOCB will likely never be reprogrammed when finished. Any tweaks or adjustments that need to be made will likely be done once everything is working and put together. By the end of Senior Design 2, we will not only have an animatronic parrot that works; we'll have an animatronic parrot that works well.

The last part of our project will be integrating everything together. Once the boards are programmed and optimized, the parrot figure is built and decorated, and all the exterior circuitry is up and running, final connections and installations will occur. This will happen once all possible improvements have been implemented. We don't want to reprogram any of the boards once they're installed in the enclosure, and all the parrot's mechanisms must work perfectly before the outer shell is installed. In the end, we'll have a fully working system; capable of performing all the predetermined actions.

Chapter 10.0: Administrative Content

This section discusses the financial aspects of CAPER and the timeline followed to complete the project. We covered every administrative concern related to our project in this area. topics include material cost budgeting, time management, goal deadlines, and the components necessary for the prototype.

10.1 Finances

This project is not sponsored by anyone, and everything is being paid out of pocket, split equally between the four team members. Initial estimates are in the table below. Since not every component is concretely decided on, the estimated costs could possibly increase later on in the project once all components are selected. Some items may be obtained for free from the campus labs, which would alleviate some costs. All costs were rounded up due to the possibility of needing multiple of each component for testing. The maximum budget is about twenty percent more than the estimated budget, assuming that the most expensive option for the voice response board is chosen, so there is wiggle room for trial and error. Still, it is preferable to keep this project as affordable as possible and stay well below \$1300.

- Goal budget: below \$1300
- Estimated total from Divide and Conquer document: \$1055

10.2 Bill of Materials

The bill of materials contains the name of the required components, manufacturers, distributors, part numbers, quantities, and unit costs. Once we created the BOM and reviewed all of the costs, the tentative costs seem to be much lower than we initially anticipated. This is amazing in terms of costs and allows for more trial and error in regards to possible inconveniences. All important components were included, though, it is important to consider that this bill of materials does not include any duplicates. Therefore, if any components were to fail for any reason, the costs will increase and be greater than what is listed here. Many costs were alleviated due to thinking outside of the box and ultimately deciding to go with a less expensive VRB (NVIDIA's Jetson Nano) while using an outside source as the wifi adapter to offload computing to a separate server. With this, even the increased costs from the testing phase should keep us well under \$1000!

Table 10.1: Bill of Materials

Component	Manufacturer	Distributor	Part #	Quantity	Unit costs
volt regulator	Texas Instruments	DigiKey	LM7805 CT	1	\$1.87
volt regulator	Texas Instruments	DigiKey	LM7905 CT	1	\$1.61
volt regulator	Texas Instruments	DigiKey	LM1117 T-ADJ	1	\$1.19
microcontroller	Texas Instruments	DigiKey	MSP430G2553 IPW20 R	1	\$2.81
optoisolator	Isocom Components 2004 LTD	DigiKey	H11L1S MIS-ND	1	\$1.04
USB UART IC	FTDI Chip	FTDI Chip	FT232RL	1	\$4.95
op amps	STMicroelectronics	DigiKey	LF351DT	2	\$1.06
hardwood dowels	Waddell	Home Depot	N/A	1	\$3.37

Felt	Creatology	Michaels	N/A	1	\$6.49
Foam	Juvo Plus Inc.	Michaels	N/A	1	\$20.99
Feathers	N/A	Aliexpress	400 pieces	1	\$2.99
Solenoids	Aexit	Amazon	f190412 ae0593 47	4	\$12.25
8x8 wooden block base	WoodCrafter	Woodcrafter	N/A	1	\$12.50
Nvidia Jetson Nano	NVIDIA	Newarrk	900-134 48-0020 -000	1	\$156.88
relay modules	BESTEP	Amazon (pack of 5; 4 required)	SRD-03 VDC-SL -C	1	\$9.99
microphone	Behringer	Amazon	SL 84C	1	\$14.90
speakers	RIOWOIS	Amazon	DS6500 M	1	\$36.99
amplifier	Lepai	Parts Express	LP-202 0AD	1	\$30.00
power supply	MEAN WELL	Amazon	LRS-15 0-12	1	\$16.00
SD memory card	SAMSUNG	Amazon	EVO select 256GB	1	\$22.99
wifi adapter	TP-Link	Amazon	N150	1	\$10.99

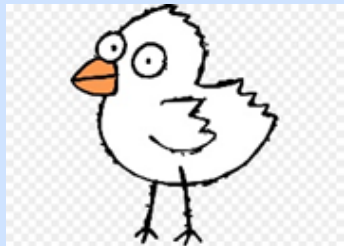
Total estimated costs (not including duplicates other than the minimum required amounts): \$428.67

This is our current final bill of materials as of Senior Design I, Spring 2024. It is our current layout of components we intend to use in the finished project, however, we will continue to update it throughout Senior Design II to reflect any significant adjustments or additions of new materials.

10.3 Distribution of Worktable

The table below shows how the tasks have been divided between the members. Each member has an overlapping task to promote collaboration and teamwork. All members are aware of and involved in every aspect of the project even without having the assigned task.

Table 10.2: Distribution of Worktable

Task	Primary	Secondary
AI implementation	Paco	Kellen
Sensors	Kellen	Sarah
Embedded systems	Kellen	Billy
Circuits	Billy	Sarah
PCB design	Sarah	Paco
Organization	Sarah	
documentation	Paco	
Construction	Kellen	
Project lead	Billy	

10.4 Project Milestones

This team formed over winter break and began brainstorming just before the start of the spring semester in preparation for Senior Design I. For the milestones, the cells in blue are in reference to project documentation and the cells in pink are in reference to project design. Every milestone for Spring 2024 is included, along with the anticipated completion date. Along with the expected duration, a start and finish date are displayed. We plan to order our PCB early to have extra time for testing and assembly. This will hopefully offset some of the time lost from having a shorter Senior Design II semester.

Table: 10.3: Senior Design I Project Milestones

Task	Description	Anticipated competition date	Duration
Senior Design I Documentation			
Discuss project ideas	Member meeting to discuss project ideas	Already completed prior to Senior design I	1 week
Choose Project	Member meeting to finalize project choice → robot parrot	Already completed prior to Senior design I	1 week
Advisor/ Reviewer selection	Choose and email reviewers → Dr. Piotr Kulik, Dr. Vikram Kapoor, Dr. Matthew Gerber	2/9/24	2 weeks
Work on Divide & Conquer	Member meetings regularly to work on the paper together + discuss future aspects. Sections have been divided between members	2/2/24	2 weeks
Chan Discussion	Discuss the project + specifications with Dr. Chan	2/8/24	30-minute Zoom call

Update divide & conquer	Update the D&C according to Dr. Chan's suggestions, then adding the document to the website.	2/15/24	1 week
All component selection	Tentative BOM + decide on all main components needed for the hardware	3/3/24	4 weeks
Additional Research	Finalize research and sources for the final report	3/3/24	4 weeks
60 page milestone	The halfway point of the report	3/27/24	4 weeks
Chan Discussion #2	Discuss the project with Dr. Chan	4/2/24	30-minute zoom call
Update 60 page document	Update the D&C according to Dr. Chan's suggestions, then adding the document to the website.	4/9/24	1 week
System Design	Making the overall schematic of the project	3/27/24	4 weeks
Finalize and review SD1 document	Ensuring completion of all requirements and making necessary adjustments.	4/21/24	4 weeks
Final Document	Completion of the 120-page final document.	4/23/24	4 weeks
Breadboard Prototype(s)	Completion of the breadboard prototype	4/23/24	4 weeks

PCB design & order materials	Completion of PCB design + ordering all materials; complete BOM	4/30/24	3 weeks
------------------------------	---	---------	---------

The significant milestones for Summer 2024 are included in the table below, but as of yet, there are no deadlines due to the lack of information regarding these requirements. This table will be updated in Senior Design 2 accordingly to match the given requisites.

Table 10.4: Senior Design II Project Milestones

Senior Design II Documentation			
PCB testing and redesign	Assembling PCB + testing all parts to ensure they work properly. Adjustments will be made as necessary to ensure proper completion and function of CAPER.	TBD	TBD
Integration	Test and integrate individual systems	TBD	TBD
Finalize documentation	Completed documentation of the entire project. Adjustments are made as necessary.	TBD	TBD
Finalize prototype	Ensure the project is fully functional with all intended components involved.	TBD	TBD
Final project PowerPoint	Complete final presentation PowerPoint + practice final presentation	TBD	TBD

Final presentation	Final 10 minute presentation demonstrating CAPER's voice recognition, audio response, and response time to push button inputs.	TBD	TBD
--------------------	--	-----	-----

Chapter 11.0: Conclusion

The introduction of Artificial Intelligence to the field of robotics and animatronics provides an endless stream of possibilities as to what's next. This uncertainty and raw power strikes fear into lots of people, calling for extreme amounts of AI regulation to ensure it doesn't lead us to our own destruction. While these worries have merit, AI should be looked at as a tool capable of assisting with humanity's biggest struggles, not as something to bring on more issues. Instead of dismissing AI out of fear for what it could do, our group set out to show the positive and recreational effect AI can have on the everyday world in the hopes of changing public opinion to be more accepting of Artificial Intelligence.

In an attempt to highlight our thought process and approach, this document has covered our project from the beginning brainstorming stages to the creation of the prototype of CAPER's end abilities. We delved into the background of animatronics and the boom of AI, and how that led to the current state of the field of development behind integrating the two. We analyzed concerns/problems in the perception of the field of AI, and used this to create our mission statement and basis for the project. Once this mission statement was declared, we set out to list out a reasonable approach to finding a solution to said statement and the plan of action to reach said solution.

This led to the brainstorm page, and us thinking of what specific actions and visible aspects we would like to include to reach our goal. This stage of the process was very imaginative, and ultimately came down to what we as a group felt best communicated our intended statement. After discussion, we decided on a Parrot as they are a mascot typically used for recreation/thought of fondly, and one often used for speech based projects. We laid out goals as a realistic design, fluid movements, and voice response capability implemented via AI.

The next step of our project was to lay out specific constraints such as degrees of movement, response accuracy, and response efficiency to increase the realistic nature of our Parrot. This led to a whole barrage of research determining what parts allowed for the most freedom of design, which would be most budget friendly, and which allowed for the easiest installation for the project to function flawlessly. As well, our group delved into research on specific standards that would affect our construction of CAPER, and as well took a much more in depth look at constraints now that we had the physical parts in

front of us. After an in depth review of multiple parts from multiple lenders, we chose those best fitted to assist us in the designing of the hardware/software of CAPER that met the requirements outlined by the standards we found online.

The hardware and software were then outlined in chapter 6 and 7, and with these schematics and blueprints in place, we turned our focus to the construction of the entire project. Trying to view CAPER as one cohesive unit, we looked into the relation between the software and hardware as well as what points would be useful for testing. With this in place, our group began our construction of the prototype, by breadboarding the circuits and beginning our PCB construction. As we submit this document we will be finalizing our 3 minute video showcasing the capabilities of caper in prototype stage, outlining the main functions we hope to implement before coming together to build the physical project in its entirety.

This document is subject to change, and will likely be revised as our group ventures into Senior Design 2 and the actual construction of the CAPER project. Many unforeseen issues are sure to arise, and parts will need to be replaced. Nonetheless, our group has a formidable document detailing our entire thought, brainstorming, and design process that will allow us to pinpoint sections with errors and address them later. Our research up to this point has allowed us to enter the prototyping stage of CAPER, and then the construction stage. Throughout this process in Senior Design 1 we have learned irreplaceable skills in teamwork, research, and design settings. It is through the use of these skills we were able to turn a few thoughts on a project and a few strangers just 3 months ago into a fully fledged and capable project, and a true team.

With CAPER, we aim to demonstrate that AI is just as welcoming as it is powerful. Through its friendly aesthetic and impactful capabilities, CAPER embodies the potential of AI to enrich areas of our lives while reminding us of its profound influence. As we continue to explore the frontiers of human-robot interaction, let CAPER serve as a symbol of the improvements that can be made in the simplest parts of life, and not the end-all-be-all that the media portrays it to be.

APPENDIX A: Bibliography

1. Graves, S. (2023, October 23). How Five Nights at Freddy's brought its creepy animatronics to life. *Gizmodo*.
<https://gizmodo.com/five-nights-at-freddys-interview-puppets-jim-henson-1850943504>
2. Marsh, A. (2023, January 10). Elektro the Moto-Man had the biggest brain at the 1939 World's Fair. *IEEE Spectrum*.
<https://spectrum.ieee.org/elektro-the-motoman-had-the-biggest-brain-at-the-1939-worlds-fair>
3. Şengelen, Ö. (2022, October 31). *A legendary clock in Prague: 600 years of history and fabled tales*. Daily Sabah.
<https://www.dailysabah.com/life/travel/a-legendary-clock-in-prague-600-years-of-history-and-fabled-tales>
4. Stein, S. (2016, March 18). *Visiting the Automaton of Marie Antoinette*. The Paris Review.
<https://www.theparisreview.org/blog/2016/03/18/automaton/>
5. MacNeal, D., & MacNeal, D. (2022, July 25). *Automata: The odd magic of living machines*. Art of Play.
<https://www.artofplay.com/blogs/stories/automatons-the-odd-magic-of-living-machines>
6. *The history of animatronics*. (n.d.).
<https://roborobotics.com/Animatronics/history-of-animatronics.html>
7. Staff, R. (2017, May 7). *The singing bird that inspired Walt Disney*. M.S. Rau.
<https://rauantiques.com/blogs/canvases-carats-and-curiosities/singing-bird-inspired-walt-disney>
8. Beren, D. (2023, July 31). *What was the video game crash of 1983 and why did it happen?* History-Computer.
<https://history-computer.com/what-was-the-video-game-crash-of-1983-and-why-did-it-happen/>
9. *Stan Winston School of Character Arts*. (n.d.).
<https://www.stanwinstonschool.com/> Massachusetts Institute of Technology. (n.d.). *Electric Parrot - Overview*. MIT Media Lab.
<https://www.media.mit.edu/projects/electric-parrot/overview/>
10. Doctorow, C. (2016, August 15). *What's inside a tiki bird?* Boing Boing.
<https://boingboing.net/2016/08/13/whats-inside-a-tiki-bird.html>

11. Aspengore. (2015, October 20). *Vox Circuit*. ElectroSchematics.com.
<https://www.electroschematics.com/vox-circuit-schematic/>
12. Sands, S. (2018, November 14). *Disney's stuntronics are getting more intelligent...and superhuman*. AllEars.Net.
<https://allears.net/2018/11/14/disneys-stuntronics-are-getting-more-intelligent-and-superhuman/>
13. Vandenuecker, D. (1992). MIDI tutorial for programmers.
<https://www.cs.cmu.edu/~music/cmsip/readings/MIDI%20tutorial%20for%20programmers.html>
14. Adamczak, B. (2020a, December 30). *MIDI protocol overview*. YouTube.
<https://www.youtube.com/watch?v=49Oqa4D3Afk>
15. Khatri, D. (2020, October 12). *Programming MSP430G2553 through BSL | MSP430 on a breadboard | Uart Bridge Programmer*. YouTube.
<https://www.youtube.com/watch?v=Zg41y4pgCIA>
16. Ruede, A. (2014, May 29). *How to MIDI-in aka how to store 3 bytes properly (UART)*.
How to MIDI-in aka how to store 3 bytes properly (UART) - MSP low-power microcontroller forum - MSP low-power microcontrollers - TI E2E support forums.
<https://e2e.ti.com/support/microcontrollers/msp-low-power-microcontrollers-group/msp430/f/msp-low-power-microcontroller-forum/584083/msp430fr4133-can-i-run-a-single-voiced-midi-through-my-device>

APPENDIX B: Copyright Requests

Figure 3.1:

Link: <https://d23.com/a-to-z/audio-animatronics/>

Request:

Hello!

I'm an electrical engineering student enrolled at the University of Central Florida, and I'm requesting permission to use a photograph of the Abraham Lincoln animatronic taken from the D23 A to Z page on Audio-Animatronics. Me and my group members are building an animatronic for our Senior Design class, and would like to use the photo in our report. Below is a link to the exact page, where the photo in question is located:

Link: <https://d23.com/a-to-z/audio-animatronics/>



Audio-Animatronics - D23

Audio-Animatronics - When Walt Disney found an antique mechanical singing bird in a shop in New Orleans when he was there on vacation, he was intrigued.

d23.com

We'd really like to use the photo, as the achievements of WED enterprises were a huge inspiration for our project to begin with. Proper credit will be given to the source of this photo in our document's bibliography.

Thanks in advance,

William Genevrino
College of Engineering, University of Central Florida

Figure 3.2:

Link: <https://boingboing.net/2016/08/13/whats-inside-a-tiki-bird.html>

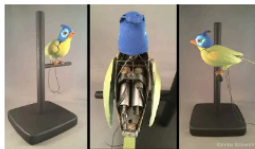
Request:

Hello!

I'm an electrical engineering student at the University of Central Florida, and I would like your permission to use a photo from your website. We are typing a report about animatronics for our senior design class, and we'd like to include the photo of a bird animatronic from the article "What's inside a Tiki Bird" by Cory Doctorow. It is the very first photo you see at the top of the page; just under the title. Below is a link to the page for your convenience.

Link:

<https://boingboing.net/2016/08/13/whats-inside-a-tiki-bird.html>



What's inside a Tiki Bird?

Kevin Kidney owns a couple of audio-animatronic birds from the Enchanted Tiki Room, the first Disney showcase for robotic animals, still running and glorious today — he's decided to make...

boingboing.net

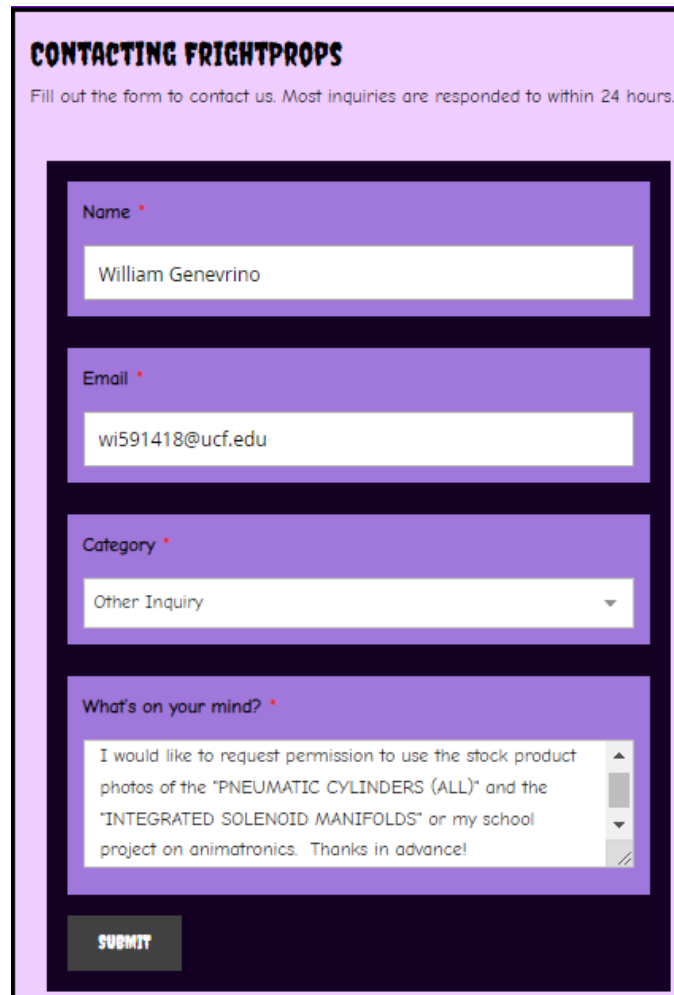
If you could grant us permission to use the photo, that would be wonderful. The Enchanted Tiki Room was a huge inspiration for this project of ours.

Thanks in advance,
William Genevrino
College of Engineering, University of Central Florida.

Figures 3.3 and 3.4

Link: <https://www.frightprops.com/pneumatics/solenoid-valves.html>

Request:



CONTACTING FRIGHTPROPS

Fill out the form to contact us. Most inquiries are responded to within 24 hours.

Name *

William Genevrino

Email *

wi591418@ucf.edu

Category *

Other Inquiry

What's on your mind? *

I would like to request permission to use the stock product photos of the "PNEUMATIC CYLINDERS (ALL)" and the "INTEGRATED SOLENOID MANIFOLDS" or my school project on animatronics. Thanks in advance!

SUBMIT

Figure 6.9

Link: <https://www.electroschematics.com/vox-circuit-schematic/>

Request:

Your Message(Tell us how we can help)*

I would like to request permission to use the photo in the article about Ham Radio VOX circuits for my school report. Here is the link to that article:
<https://www.electroschematics.com/vox-circuit-schematic/>
Thanks in advance|

Figure 6.10

Link: <https://freecircuitdiagram.com/2266-basic-vox-circuit-controls-ptt/>

Request:

Photo Use Permission for School Report

Your Message

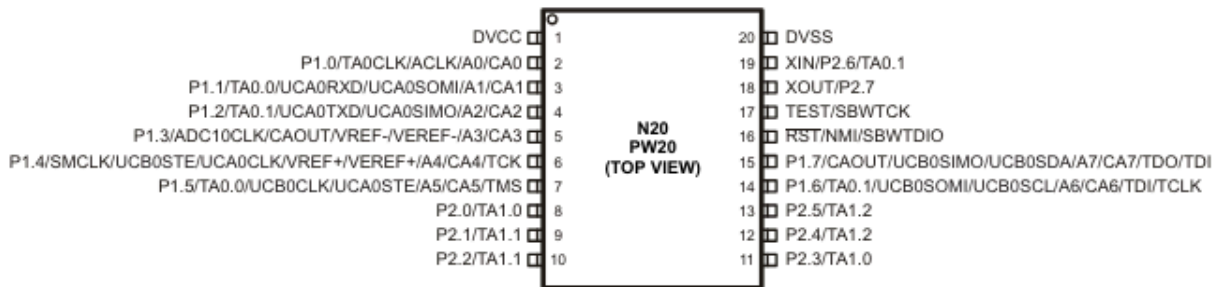
I would like to request permission to use the photo of the simple VOX circuit schematic for my school report. Part of our project uses a very similar circuit, and we'd really like to display yours as reference in our report. Here is the link to the page:

<https://freecircuitdiagram.com/2266-basic-vox-circuit-controls-ptt/>

Thanks in advance!

APPENDIX C: Datasheets and Pin Layouts

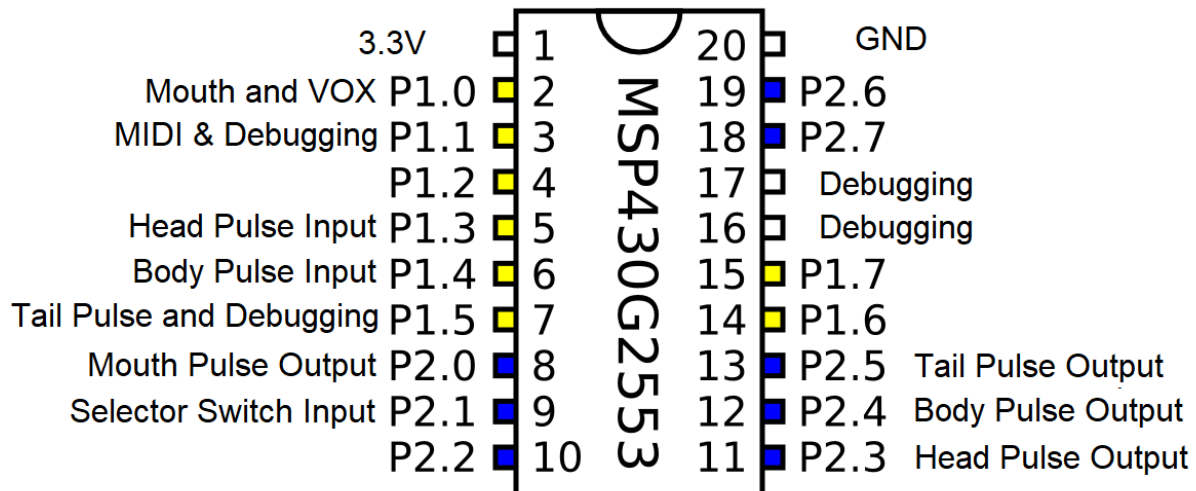
Pin Layout from Datasheet



NOTE: ADC10 is available on MSP430G2x53 devices only.

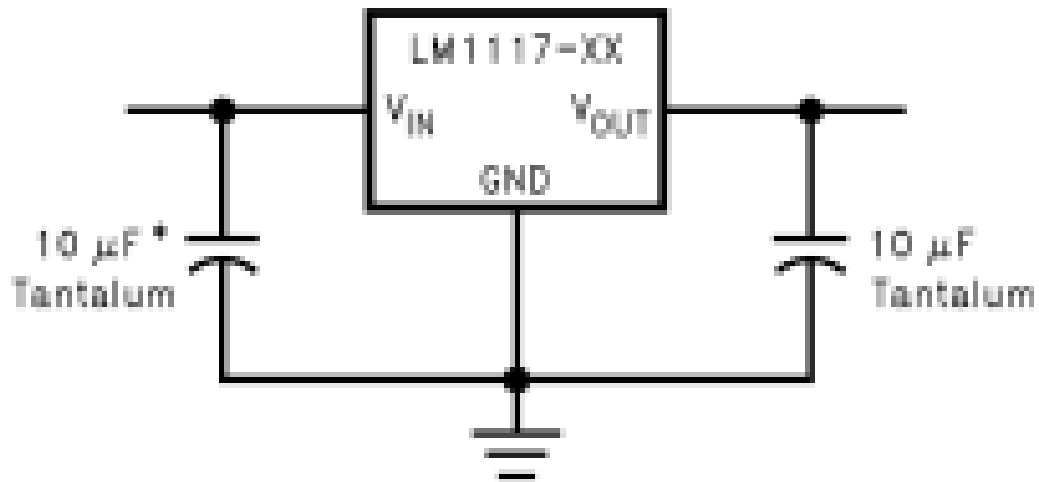
MSP430G2553 with labels

MSP430G2553IPW20
(any 20 pin package)



Pins 1.6, 1.7, 2.2, 2.6, & 2.7 are unused

LM1117 Typical Configuration and Maximum Ratings

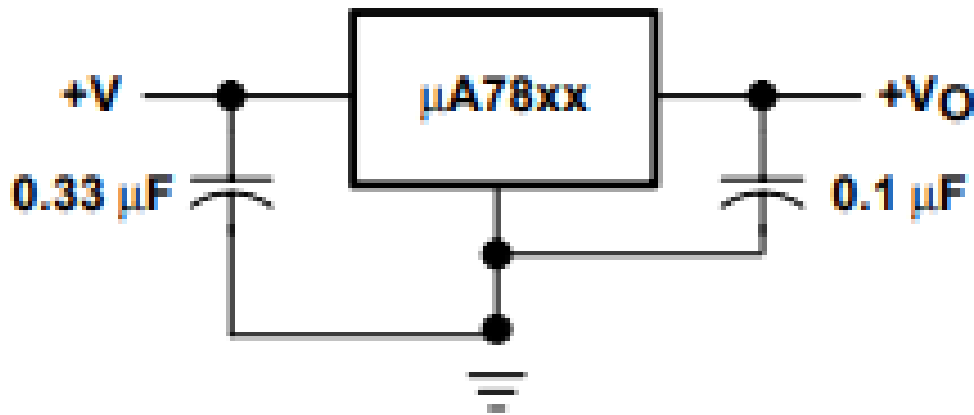


*Use of this capacitor is optional if within small distance from voltage source

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Input Voltage (Note 1)	V_{in}	20	V
Output Short Circuit Duration (Notes 2 and 3)	—	Infinite	—
Power Dissipation and Thermal Characteristics Case 318H (SOT-223) Power Dissipation (Note 2) Thermal Resistance, Junction-to-Ambient, Minimum Size Pad Thermal Resistance, Junction-to-Case	P_D $R_{\theta JA}$ $R_{\theta JC}$	Internally Limited 160 15	W $^{\circ}\text{C}/\text{W}$ $^{\circ}\text{C}/\text{W}$
Maximum Die Junction Temperature Range	T_J	-55 to 150	$^{\circ}\text{C}$
Storage Temperature Range	T_{stg}	-65 to 150	$^{\circ}\text{C}$
Operating Ambient Temperature Range LM1117 LM1117I	T_A	0 to +125 -40 to +125	$^{\circ}\text{C}$

7805 Typical Configuration and Maximum Ratings



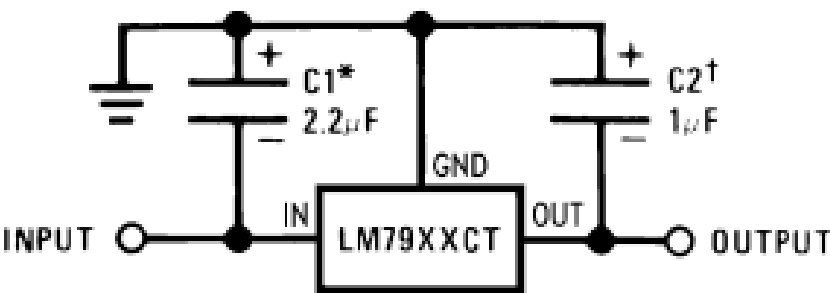
SLVS056J – MAY 1976 – REVISED MAY 2003

electrical characteristics at specified virtual junction temperature, $V_I = 14\text{ V}$, $I_O = 500\text{ mA}$ (unless otherwise noted)

PARAMETER	TEST CONDITIONS	T_J^\dagger	μA7808C			UNIT
			MIN	TYP	MAX	
Output voltage	$I_O = 5\text{ mA to }1\text{ A}$, $P_D \leq 15\text{ W}$, $V_I = 10.5\text{ V to }23\text{ V}$	25°C	7.7	8	8.3	V
		$0^\circ\text{C to }125^\circ\text{C}$	7.6		8.4	
Input voltage regulation	$V_I = 10.5\text{ V to }25\text{ V}$	25°C		6	160	mV
	$V_I = 11\text{ V to }17\text{ V}$			2	80	
Ripple rejection	$V_I = 11.5\text{ V to }21.5\text{ V}$, $f = 120\text{ Hz}$	$0^\circ\text{C to }125^\circ\text{C}$	55	72		dB
Output voltage regulation	$I_O = 5\text{ mA to }1.5\text{ A}$	25°C		12	160	mV
	$I_O = 250\text{ mA to }750\text{ mA}$			4	80	
Output resistance	$f = 1\text{ kHz}$	$0^\circ\text{C to }125^\circ\text{C}$		0.016		Ω
Temperature coefficient of output voltage	$I_O = 5\text{ mA}$	$0^\circ\text{C to }125^\circ\text{C}$		-0.8		$\text{mV}/^\circ\text{C}$
Output noise voltage	$f = 10\text{ Hz to }100\text{ kHz}$	25°C		52		μV
Dropout voltage	$I_O = 1\text{ A}$	25°C		2		V
Bias current		25°C		4.3	8	mA
Bias current change	$V_I = 10.5\text{ V to }25\text{ V}$	$0^\circ\text{C to }125^\circ\text{C}$			1	mA
	$I_O = 5\text{ mA to }1\text{ A}$				0.5	
Short-circuit output current		25°C		450		mA
Peak output current		25°C		2.2		A

[†] Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a $0.33\text{-}\mu\text{F}$ capacitor across the input and a $0.1\text{-}\mu\text{F}$ capacitor across the output.

7905 Typical Configuration and Maximum Ratings



ABSOLUTE MAXIMUM RATINGS⁽¹⁾

Input Voltage	
(V _o = -5V)	-25V
(V _o = -12V and -15V)	-35V
Input-Output Differential	
(V _o = -5V)	25V
(V _o = -12V and -15V)	30V
Power Dissipation ⁽²⁾	Internally Limited
Operating Junction Temperature Range	0°C to +125°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	230°C

- (1) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is intended to be functional, but do not ensure Specific Performance limits. For ensured specifications and test conditions, see the Electrical Characteristics.
- (2) Refer to [DESIGN CONSIDERATIONS](#) for details.

FT232RL Pinout

3.1 28-LD SSOP Package

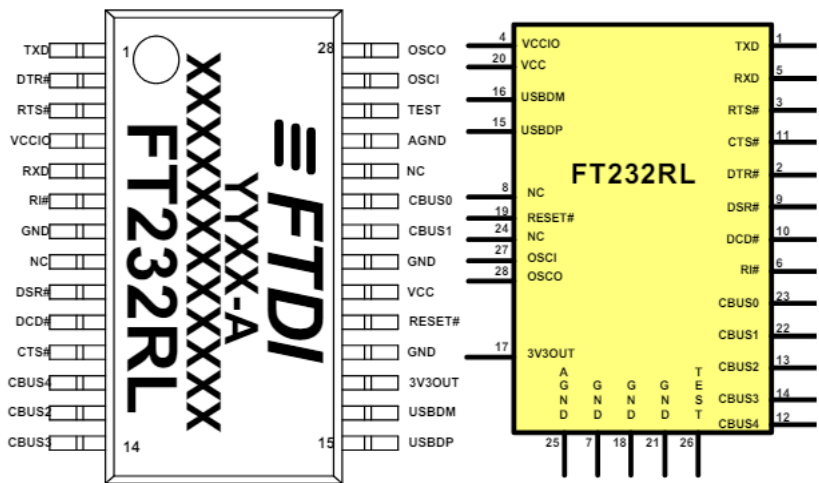


Figure 3.1 SSOP Package Pin Out and Schematic Symbol

FT232RL USB Powered Configuration from FDTI Datasheet



6 USB Power Configurations

The following sections illustrate possible USB power configurations for the FT232R. The illustrations have omitted pin numbers for ease of understanding since the pins differ between the FT232RL and FT232RQ package options.

All USB power configurations illustrated apply to both package options for the FT232R device. Please refer to Section 3 for the package option pin-out and signal descriptions.

6.1 USB Bus Powered Configuration

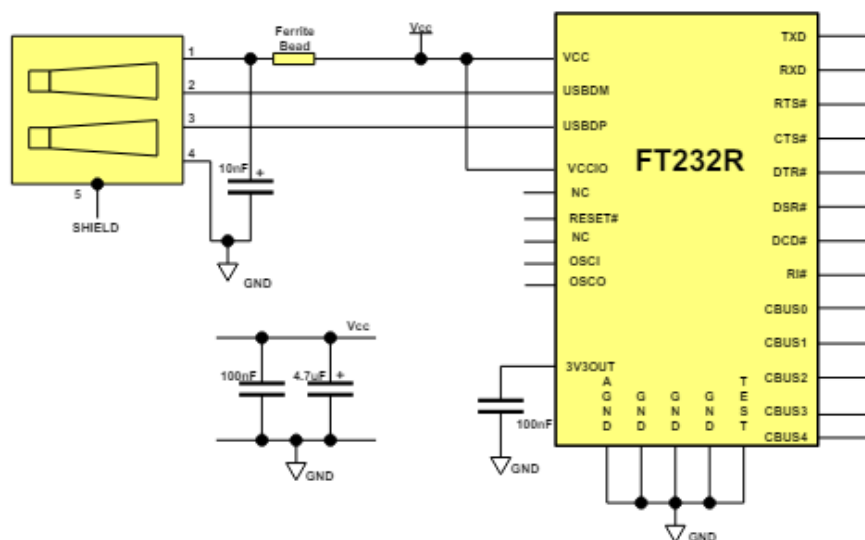


Figure 6.1 Bus Powered Configuration

Figure 6.1 Illustrates the FT232R in a typical USB bus powered design configuration. A USB bus powered device gets its power from the USB bus. Basic rules for USB bus power devices are as follows –

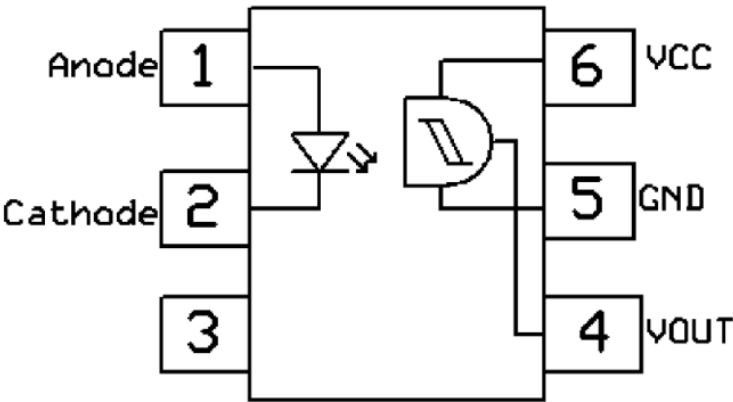
- i) On plug-in to USB, the device should draw no more current than 100mA.
- ii) In USB Suspend mode the device should draw no more than 2.5mA.
- iii) A bus powered high power USB device (one that draws more than 100mA) should use one of the CBUS pins configured as PWREN# and use it to keep the current below 100mA on plug-in and 2.5mA on USB suspend.
- iv) A device that consumes more than 100mA cannot be plugged into a USB bus powered hub.
- v) No device can draw more than 500mA from the USB bus.

The power descriptors in the internal EEPROM of the FT232R should be programmed to match the current drawn by the device.

A ferrite bead is connected in series with the USB power supply to reduce EMI noise from the FT232R and associated circuitry being radiated down the USB cable to the USB host. The value of the Ferrite Bead depends on the total current drawn by the application. A suitable range of Ferrite Beads is available from Steward (www.steward.com), for example Steward Part # MI0805K400R-10.

Note: If using PWREN# (available using the CBUS) the pin should be pulled to VCCIO using a 10kΩ resistor.

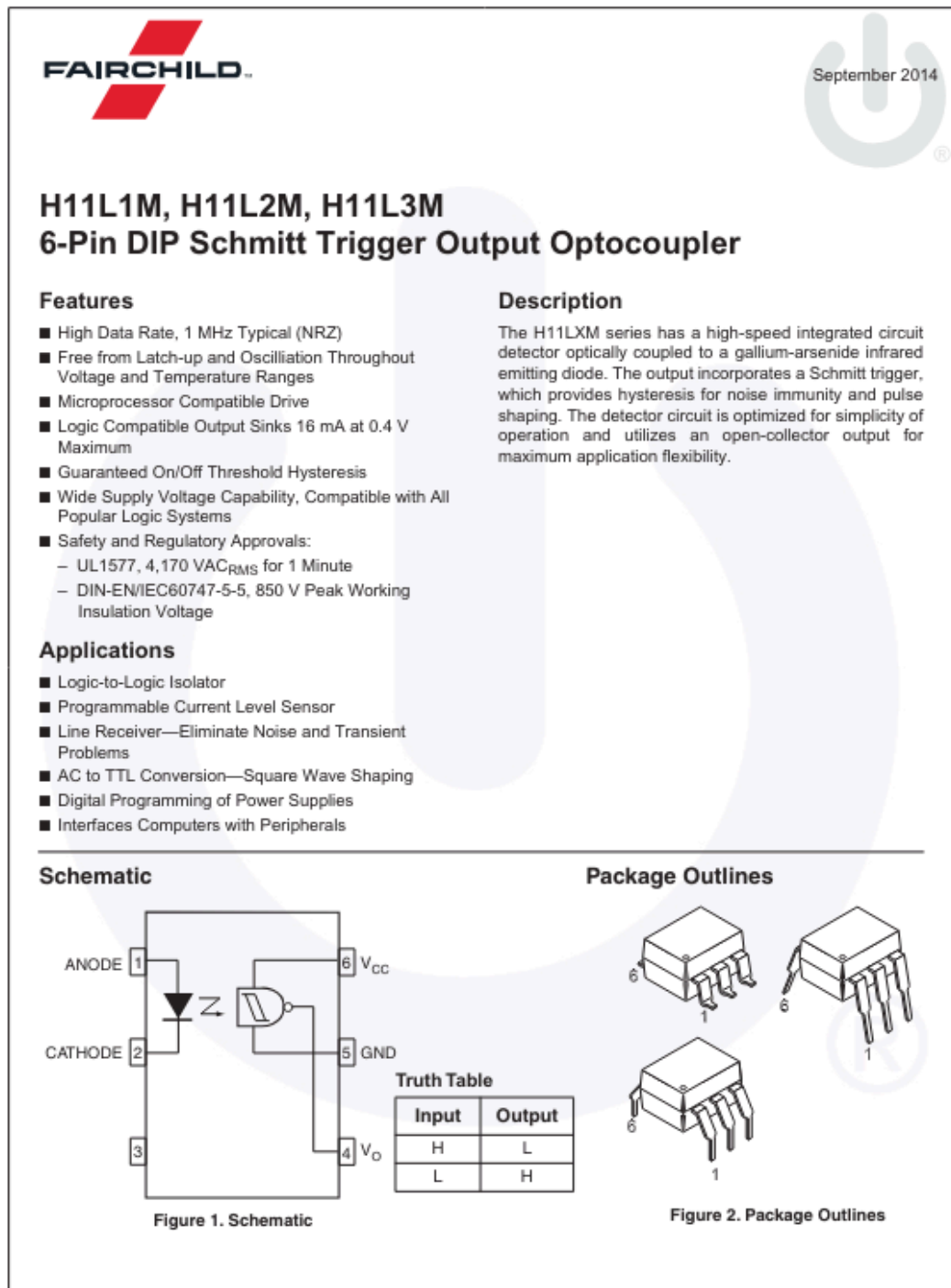
H11L1 Optocoupler Layout and Specifications



Absolute Maximum Rating

Symbol	Parameter	Rating	Units
		H11L1 H11L2 H11L3	
T _{STG}	Storage Temperature	-55 ~ +150	°C
T _{OPR}	Operating Temperature	-40 ~ +100	°C
T _{SOL}	Lead Solder Temperature	260	°C
V _{ISO}	Isolation voltage	5000	VRMS
EMITTER			
I _F	Continuous Forward Current	60	mA
I _{PF}	Peak Forward Current (300us pulse, ≤1 μs P.W)	1	A
V _R	Reverse Voltage	6	V
P _D	Power Dissipation	100	mW
DETECTOR			
V _O	Output Voltage	0 to 16	V
V _{cc}	Supply Voltage	3 to 16	V
I _O	Output Current	50	mA
P _D	Power Dissipation	150	mW

H11L1 Optocoupler datasheet with truth table from Fairchild



APPENDIX D: Programming Code

IOCB Code

```
#include <msp430.h>

unsigned char channel;
unsigned char status;
unsigned char note;
unsigned char velocity;
unsigned char byteNr;
unsigned char myChannel;

void setUp();
void handleMIDI(unsigned char RxByte);
void handleNoteOn(unsigned char channel, unsigned char note, unsigned char
velocity);
void handleNoteOff(unsigned char channel, unsigned char note, unsigned char
velocity);
void handleControlChange(unsigned char channel, unsigned char controller,
unsigned char value);

int main(void){
    // MOVEMENT PIN CHART

    //OUTPUTS
    //MOUTH = 2.0
    //HEAD = 2.3
    //BODY = 2.4
    //TAIL = 2.5

    //INPUTS
    //SELECTOR = 2.1

    //MOUTH = 1.0
    //HEAD = 1.3
    //BODY = 1.4
    //TAIL = 1.5

    //P1DIR |= BIT0 +BIT6;    //Declare PIN0 AND PIN1 OF PORT 1 AS
OUTPUT
    //P1OUT &= ~(BIT0 +BIT6); //set output as low at first

    //P1DIR &= ~BIT3;    // set 1.3 as input
    //P1REN |= BIT3;    //enable pull resistor
    //P1OUT |= BIT3;    //set as pull up
    setUp();
    //set up the inputs
```

```

        P1DIR &= ~(BIT0 + BIT3 + BIT4 + BIT5 ); // set 1.0 1.3 1.4 1.5 as
inputs
        P1REN |= (BIT0+ BIT3 + BIT4 + BIT5 ); // enable pull up
resistors
        P1OUT |= (BIT0+ BIT3 + BIT4 + BIT5); //set as pull up
        P1IE |= (BIT0+ BIT3 + BIT4 + BIT5); // enable interrupt bits for
four button inputs
        P1IFG &= ~(BIT0+ BIT3 + BIT4 + BIT5); //clear interrupt flags

        P2DIR &= ~BIT1; // set 2.1 as input
        P2REN |= BIT1; // enable pull up resistors
        P2OUT |= BIT1; // set as pull up
        P2IE |= BIT1; // enable interrupt bit for 2.1
        P2IFG &= ~BIT1; // clear interrupt flag for bit 2.1

        //setting up outputs
        P2DIR |= BIT0 + BIT3 + BIT4 + BIT5 ; //2.0 2.3 2.4 & 2.5 are
outputs, all other 2's inputs
        P2OUT &= ~(BIT0+ BIT3 + BIT4 + BIT5 ); //set all as low initially

//enable global interrupts
_enable_interrupts();

while(1){}
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{ _delay_cycles(25000);
    if(P1IN & BIT0){ //manually control mouth
        P2OUT &= ~BIT0; // keep off
        P1IFG &= ~BIT0; //clear interrupt flag for 1.0
    }
    else{
        P2OUT |= BIT0; //turn on bit
    }

    if(P1IN & BIT3){ //manually control head
        P2OUT &= ~BIT3;
        P1IFG &= ~BIT3; //clear flag for 1.3
    }
    else{
        P2OUT |= BIT3;
    }
}

```

```

    }

    if(P1IN & BIT4){        //manually control body
        P2OUT &= ~BIT4;
        P1IFG &= ~BIT4;    // clear flag for 1.4
    }
    else{
        P2OUT |= BIT4;
    }

    if(P1IN & BIT5){        //manually control tail & wings
        P2OUT &= ~BIT5;
        P1IFG &= ~BIT5;    // clear flag for 1.5
    }

    else{
        P2OUT |= BIT5;
    }

}

#pragma vector=PORT2_VECTOR
__interrupt void Port_2(void)
{
    int head = 0; //counter variables for the automatic movements
    int tail = 0;
    int body = 0;
    P2OUT &= ~(BIT0 + BIT3 + BIT4 + BIT5);
    while(!(P2IN & BIT1)){ //when selector switch is toggled
        _delay_cycles(25000);
        body = body + 1;
        head = head + 1;
        tail = tail + 1;

        if(P1IN & BIT0){        //manually control mouth
            P2OUT &= ~BIT0;
        }
        else{
            P2OUT |= BIT0;
        }

        if(head == 50){
            P2OUT ^= BIT3;

```

```

        head = 0;
    }

    if(body == 100){
        P2OUT ^= BIT4;
        body = 0;
    }

    if(tail == 200){
        P2OUT ^= BIT5;
        tail = 0;
    }
}

P2OUT &= ~(BIT0 + BIT3 + BIT4 + BIT5); //reset all movements before resuming
manual control
body = tail = head = 0; //reset counter variables
P2IFG &= ~BIT1; //clear flag for 2.1
P1IFG &= ~(BIT0+ BIT3 + BIT4 + BIT5); //clear flags for all button inputs
}

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void){
    handleMIDI(UCA0RXBUF);
}

void setUp(){

    WDTCTL = WDTPW + WDTHOLD;                // Stop WDT

    /* Use Calibration values for 1MHz Clock DCO*/
    DCOCTL = 0;
    BCSCTL1 = CALBC1_1MHZ;                    // set DCO to 1 MHz
    http://bennthomsen.wordpress.com/ti-msp430-launchpad/msp430g2553-hardware-uart/
    DCOCTL = CALDCO_1MHZ;                    // set DCO to 1 MHz

    /* Configure Pin P1.1 RXD (Secondary peripheral module function is
selected) */
    P1SEL = BIT1;
    P1SEL2 = BIT1;

    /* Place UCA0 in Reset mode to be configured */
    UCA0CTL1 = UCSWRST;
    /* Configure Clock*/

```

```

UCA0CTL1 |= UCSSEL_2;           // Use SMCLK
UCA0BR0 = 32;                   // Set Baud rate to 31250 with
1MHz Clock
UCA0BR1 = 0;                     // Set Baud rate to 31250 with
1MHz Clock
UCA0MCTL = UCBRS0;               // Modulation UCBRSx = 1

/* Take UCA0 out of reset */
UCA0CTL1 &= ~UCSWRST;

/* Enable USCI_A0 RX interrupt */
IE2 |= UCA0RXIE;
}

void handleMIDI(unsigned char RxByte){
    if(RxByte & BIT7){           // check if status Byste
        status = RxByte & 0xF0;   // get status
        channel = RxByte & 0x00;  // get channel
        byteNr = 1;
    }
    else{                         // if not status byte, it is a data byte
        if(byteNr == 1){          // check if its first staufs byte
            note = RxByte;        // get note value
            byteNr = 2;           // after 1st data byte comes 2nd data byte
        }
        else{
            velocity = RxByte;    // get velocity value
        }
    }

    if(status == 0x90){           // check if status is noteOn
        handleNoteOn(channel,note,velocity);
    }
    else if(status == 0x80){
        handleNoteOff(channel,note,velocity);
    }
    else if(status == 0xB0){
        handleControlChange(channel,note,velocity);
    }

    note = 0;
}

void handleNoteOn(unsigned char channel, unsigned char note, unsigned char
velocity){
    if(channel == myChannel){
        // i dont think i need this
    }
}

```

```

        if(note == 0x3C){
            P2OUT |= BIT0;
        }
        if(note == 0x3E){
            P2OUT |= BIT3;
        }
        if(note == 0x40){
            P2OUT |= BIT4;
        }
        if(note == 0x41){
            P2OUT |= BIT5;
        }
    }

void handleNoteOff(unsigned char channel, unsigned char note, unsigned char
velocity){
    if(channel == myChannel){
        // i dont think i need this
    }

    if(note == 0x3C){
        P2OUT &= ~BIT0;
    }
    if(note == 0x3E){
        P2OUT &= ~BIT3;
    }
    if(note == 0x40){
        P2OUT &= ~BIT4;
    }
    if(note == 0x41){
        P2OUT &= ~BIT5;
    }
}

void handleControlChange(unsigned char channel, unsigned char controller,
unsigned char value){
    if(channel == myChannel){
        // do nothing because control change is unused
    }
}

```

Alexander Ruede's MIDI code skeleton

```

unsigned char channel;
unsigned char status;
unsigned char note;
unsigned char velocity;
unsigned char byteNr;
unsigned char myChannel;

void setUp();
void handleMIDI(unsigned char RxByte);
void handleNoteOn(unsigned char channel, unsigned char note, unsigned char
velocity);
void handleNoteOff(unsigned char channel, unsigned char note, unsigned char
velocity);
void handleControlChange(unsigned char channel, unsigned char controller,
unsigned char value);

int main(void){

    setUp();

    __bis_SR_register(LPM0_bits + GIE);          // Enter LPM0, interrupts enabled
}

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void){
    handleMIDI(UCA0RXBUF);
}

void setUp(){

    WDTCTL = WDTPW + WDTHOLD;                    // Stop WDT

    /* Use Calibration values for 1MHz Clock DCO*/
    DCOCTL = 0;
    BCSCCTL1 = CALBC1_1MHZ;                      // set DCO to 1 MHz
http://bennthomsen.wordpress.com/ti-msp430-launchpad/msp430g2553-hardware-uart/
    DCOCTL = CALDCO_1MHZ;                        // set DCO to 1 MHz

    /* Configure Pin P1.1 RXD (Secondary peripheral module function is
selected) */
    P1SEL = BIT1;
    P1SEL2 = BIT1;

    /* Place UCA0 in Reset mode to be configured */
    UCA0CTL1 = UCSWRST;

    /* Configure Clock*/

```

```

    UCA0CTL1 |= UCSSEL_2;                // Use SMCLK
    UCA0BR0 = 32;                        // Set Baud rate to 31250 with
1MHz Clock
    UCA0BR1 = 0;                        // Set Baud rate to 31250 with
1MHz Clock
    UCA0MCTL = UCBRS0;                  // Modulation UCBRSx = 1

    /* Take UCA0 out of reset */
    UCA0CTL1 &= ~UCSWRST;

    /* Enable USCI_A0 RX interrupt */
    IE2 |= UCA0RXIE;
}

void handleMIDI(unsigned char RxByte){
    if(RxByte & BIT7){                  // check if status Byste
        status = RxByte & 0xF0;        // get status
        channel = RxByte & 0x0F;       // get channel
        byteNr = 1;
    }
    else{                               // if not status byte, it is a data byte
        if(byteNr == 1){                // check if its first staufs byte
            note = RxByte;              // get note value
            byteNr = 2;                 // after 1st data byte comes 2nd data byte
        }
        else{
            velocity = RxByte;          // get velocity value
        }
    }

    if(status & 0x80){                  // check if status is noteOn
        handleNoteOn(channel,note,velocity);
    }
    else if(status & 0x90){
        handleNoteOff(channel,note,velocity);
    }
    else if(status & 0xB0){
        handleControlChange(channel,note,velocity);
    }
}

void handleNoteOn(unsigned char channel, unsigned char note, unsigned char
velocity){
    if(channel == myChannel){
        // send to DAC!!!
    }
}

```

```
void handleNoteOff(unsigned char channel, unsigned char note, unsigned char
velocity){
    if(channel == myChannel){
        // send to DAC!!!
    }
}

void handleControlChange(unsigned char channel, unsigned char controller,
unsigned char value){
    if(channel == myChannel){
        // do something
    }
}
```