Compact Animated Parrot with Enhanced Responsiveness (C.A.P.E.R.)

Kellen Danielsen, Sarah Tse, Paco-Jaleel Balthazar, and William Genevrino

University of Central Florida, Department of Computer and Electrical Engineering, Orlando, Florida, 32816 USA

***Abstract* — In an effort to intertwine an older technology (animatronics) with an emerging technology (artificial intelligence), C.A.P.E.R. invites a different perspective to what an animatronic can be. C.A.P.E.R. comes with realistic, lifelike movements to imitate that of a real parrot while utilizing an artificial intelligence model to have realistic and fast-paced conversation with its users. C.A.P.E.R. has multiple input modes for different methods of communication and can be modeled to fit any project, independent of whatever frame chosen to be modeled.**

***Index Terms* — Animatronics, Context awareness, Computational intelligence, Microphones, Voice mail, Robot control, Robot learning**

I. Introduction

C.A.P.E.R. is a combination of two different systems operating together, and a completed, high-level electrical and computer engineering senior design project. The project, implemented almost entirely on a printed circuit board, is a realistic looking parrot with a smooth, fluid motion system, as well as artificial intelligence capabilities allowing the bird to listen and respond to a user's input in a correct and concise manner. C.A.P.E.R as well comes with name recognition, and multiple input modes, these including movements operated by buttons or automating most movements but the beak being responsive to audio input. The uniqueness of C.A.P.E.R. is that it is not defined by its frame, but could be applied to any sort of system or animatronic, making it very desirable for theme parks, shows, etc.

The expectations for the senior design team members was to research safety standards relevant to A.I. implementation and printed circuit board design, then design the circuit, LLM, and TTS models that would enable our project to meet the standards and engineering specifications we laid out in Senior Design. The result is a fluid, realistic, A.I. capable animatronic.

II. System Components

Our movement and response systems are best represented in terms of their components. This section will provide a semi-technical introduction to all the major components utilized in C.A.P.E.R.'s design, which will be elaborated on further later.

A. Input-Output Control Board

All of the low-level computing required to animate C.A.P.E.R. is performed by our custom "Input-Output Control Board" (IOCB). This board houses the MCU, low-power voltage regulators, along with all our custom peripheral circuits allowing for multi-board connectivity, and microphone connectivity for voice activation. The various signals entering this board will be processed and exit as a series of digital pulse-streams for the movements.

B. Microcontroller

At the center of our custom "input-output control board (IOCB), sits the Texas Instruments MSP430G2553IN20 Microcontroller chip. The master clock runs at 16MHz, there are 16 KB of flash, 512B of SRAM, UART/SPI/I2C connectivity, and programmable GPIO pins. Using the 20-pin version allows full parallel connectivity between all of the peripheral circuits to the four outputs without the need for a shift register. Operating at 3.3V allows for easy connectivity and compatibility to external boards, and the Ultra-Low Power Consumption rating (230uA) allows for extremely efficient continuous operation.

C. Bird Figure, Solenoids, and Relay Board

The parrot figure itself features four movements: mouth open/close, head-tilt, body-tilt, and tail/wings upward-lift. These movements are carried out through the use of 24V electrical solenoids fixed to the wooden support frame inside C.A.P.E.R. Through the use of springs, connecting rods, and various mechanical linkages, the solenoids quietly and smoothly move their respective parts to their activated positions. Despite being rated at 24VDC, this voltage is stepped down to 12V to reduce power consumption, and heat loss. With this, the figure can require up to 72W peak power to actuate all movements simultaneously. To switch these solenoids, the 4 channel ARDUINO relay board is used. This board receives the low-power 3.3V electrical impulses from the MCU, and converts it to higher-power 12V streams for the solenoids.

D. Power Supply

The Input-Output Control Board, Relay board, and solenoids, all receive power from the MEAN WELL LRS-350-12 DC power supply. Providing a maximum current of 29A, this switching power supply has no trouble powering everything connected to it. It has a built-in cooling fan, an indicator LED, and surge protection for up to 300VAC.

E. Voice Response Board (VRB): Jetson Nano

The voice response board consists of a Jetson Nano Development kit paired with a USB microphone, and a USB wifi dongle. This hardware is powered by a 5V 2A external power supply feeding it through its power jack.
The Nano was chosen as it has the necessary compute and IO to accommodate for the Unified Conversational Module's (UCM) data requirements. The wifi connection is intended solely for sending and receiving audio over ssh to a separate server that runs everything.

F. Server Hardware

Although the server hardware was not explicitly chosen with this project in mind, its specifications are relevant to the design constraints of response speed and accuracy.
The server is a port forwarded ssh server hosted on an Ubuntu 22.04 Desktop equipped with a AMD Ryzen 7 Pro 3700 CPU and a NVIDIA GeForce RTX 3060 GPU (12GB)

G. UART/MIDI

To establish communication between the IOCB and VRB, UART messages will be sent using the MIDI language. The asynchronous design of UART prevents the need for a slave-master wiring configuration, and all messages can be transmitted on a single serial channel. An optoisolator circuit is included on the IOCB to localize the grounds between the devices. The information within these UART messages is encoded using MIDI (Musical Instrument Digital Interface); a digital language optimized for synthesizers and music-production softwares. Being commercially available since 1983, MIDI is ubiquitous and standardized, allowing for full connectivity with countless devices and accessories. In the case of C.A.P.E.R. if gives us a fantastic set of "blueprints" to establish multi-device connectivity with a high degree of standardization, eliminating the need for proprietary, and virtually incompatible communication protocols.

III. System concept

To properly showcase our design as a complete system, a flowchart is used below.

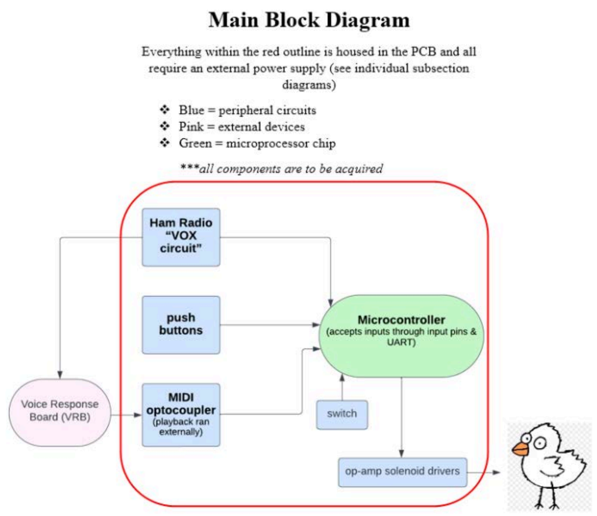

Main Block Diagram
Everything within the red outline is housed in the PCB and all require an external power supply (see individual subsection diagrams)
❖ Blue = peripheral circuits
❖ Pink = external devices
❖ Green = microprocessor chip
***all components are to be acquired
Ham Radio "VOX circuit"
push buttons
MIDI optocoupler (playback ran externally)
Voice Response Board (VRB)
Microcontroller (accepts inputs through input pins & UART)
switch
op-amp solenoid drivers

Fig. 1. Complete system flowchart dictating the functionality and path of our system.

The pink block represents the Voice Response Board, the blue blocks represent the peripheral circuits, and the green block represents the Input-Output Control Board. All of the blocks within the red outline will be connected at all times, the VRB will be connected via USB, allowing for optional disconnection. The Ham Radio VOX Circuit is one of the custom peripherals on the IOCB, allowing for voice-activated mouth operation. NOTE: This is a completely different functionality than the LLM voice response, and is performed completely by the IOCB MCU. All of the LLM functionality will be done by the VRB, and sent to the IOCB as a MIDI message through the optocoupler. The MIDI optocoupler is another peripheral on the IOC, using the H11L1 optoisolator chip. The push buttons and selector switch will be soldered to the board using low-gauge insulated wires, and the relay board will be located externally, and connected via insulated wires as well. All of these aforementioned devices will be housed alongside the power supply, in our custom chassis, allowing for sufficient airflow and spatial locality.

As stated earlier, our project has two distinct goals: To perform fluid, realistic motion with multiple input modes and to have an AI model able to intake speech and respond accordingly. The following sections will cover the hardware and software implementations that make both of those goals possible.

A. Hardware Concepts

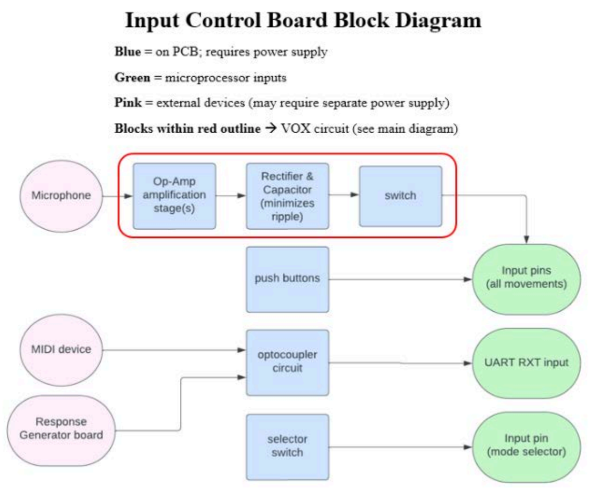

Input Control Board Block Diagram
Blue = on PCB; requires power supply
Green = microprocessor inputs
Pink = external devices (may require separate power supply)
Blocks within red outline → VOX circuit (see main diagram)
Microphone
Op-Amp amplification stage(s)
Rectifier & Capacitor (minimizes ripple)
switch
push buttons
Input pins (all movements)
MIDI device
optocoupler circuit
UART RXT input
Response Generator board
selector switch
Input pin (mode selector)

Fig. 2. Input Output Control Board block diagram showcasing the different inputs and how they're processed.

This figure is a more zoomed-in display of the left half of Figure 1. The pink blocks represent devices separate from the IOCB, the blue blocks are expanded stages of the peripherals, and the green blocks are the input pins on the MCU.

On the top row, you see the signal path for the voice-activated mouth movements. Using a handheld dynamic microphone, audio signal is sent into the VOX circuit (everything within the red outline). The VOX circuit amplifies the signal to a more usable level, rectifies the signal to remove any negative voltages, and triggers a MOSFET switch to activate the mouth through one of the GPIO pins. These GPIO pins are also used by the push buttons and selector switch.

The push buttons and selector switch allow for full or partial manual control of the four movements. Pressing any of the four color-coded push buttons will send voltage to the respective GPIO pin, and trigger the corresponding movement. The selector switch operates on the same principle but mutes the inputs from the head, body, and tail buttons. This will leave the mouth fully manual to be controlled either by the mouth push button, or the VOX circuit (both share the same exact GPIO pin).

The VRB (or any external MIDI playback device), will connect to the optocoupler peripheral circuit. The signal exiting this circuit will retain all the same digital information but will be more voltage-stable and prevent any ground faults from occurring. This signal is then sent into the MCU's UART module for processing. All four movements can be controlled using this single MIDI channel.

B. Software Concepts

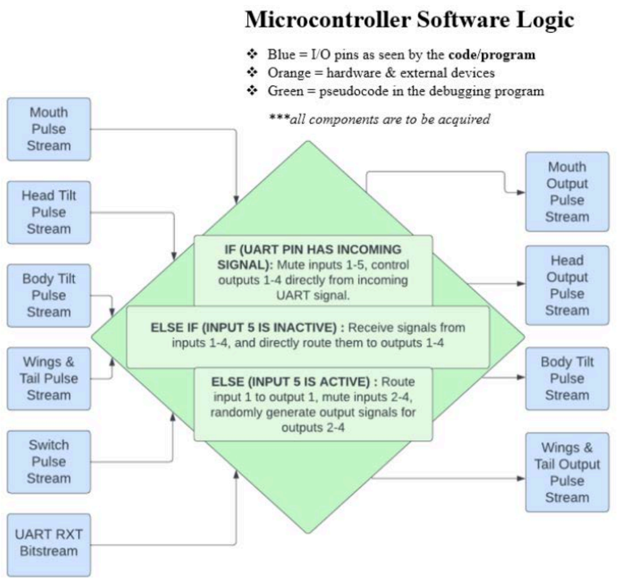

Microcontroller Software Logic
❖ Blue = I/O pins as seen by the code/program
❖ Orange = hardware & external devices
❖ Green = pseudocode in the debugging program
***all components are to be acquired
Mouth Pulse Stream
Head Tilt Pulse Stream
Body Tilt Pulse Stream
Wings & Tail Pulse Stream
Switch Pulse Stream
UART RXT Bitstream
IF (UART PIN HAS INCOMING SIGNAL): Mute inputs 1-5, control outputs 1-4 directly from incoming UART signal.
ELSE IF (INPUT 5 IS INACTIVE) : Receive signals from inputs 1-4, and directly route them to outputs 1-4
ELSE (INPUT 5 IS ACTIVE) : Route input 1 to output 1, mute inputs 2-4, randomly generate output signals for outputs 2-4
Mouth Output Pulse Stream
Head Output Pulse Stream
Body Tilt Pulse Stream
Wings & Tail Output Pulse Stream

Fig 3. IOCB Microcontroller block diagram showing the incoming streams and how they're processed.

The green diamond represents the main function, which responds to changes in input register bits to perform respective operations. The blue boxes represent the bits in the Port registers; these may be set as input or output depending on the bit. Altogether, there are 6 individual inputs and 4 individual outputs controlled by the MCU. Management and hierarchy of these signals are done on a first-come, first-serve basis; meaning the program will process whatever input arrives first, and ignore all other inputs until the current process is completed. This is achieved using interrupts and utilizing the various Ports available on this MCU.

The four push-buttons are all located on Port 1, the Selector switch is on Port 2, and the MIDI has its own set of interrupts. The four outputs are also on Port 2, but are configured as outputs in the code, and are therefore ignored by the program when scanning for inputs. Pressing any of the four push buttons will trigger the Port 1 interrupt. The program will scan the Port 1 Input register to see which button(s) are pressed, it will then activate the corresponding output bits. The program will repeatedly scan until all buttons are released; then the interrupt flag will be lowered and the function will wait until the next interrupt occurs.

In the case of the Selector Switch on Port 2. flipping this switch will raise the Port 2 Interrupt Flag and the program will enter the Port 2 Function. It will remain in this function until the switch is released. While in this function, the head, body, and tail movements will be automated by the code. All three of these buttons will be ignored while in this mode, but the mouth button will be left completely manual. This allows the user to use either the button or the VOX circuit to control the mouth. Once the selector switch is turned off, both the Port 1 and Port 2 interrupt flags will be cleared, and the program will wait until the next event occurs.

An incoming MIDI message will trigger a UART interrupt. NOTE: This will only happen if the incoming UART message is the correct baud rate (31250). Once this happens, the MIDI Handling function will be called to decipher the MIDI message. The chart below shows the eight possible messages the program can read, any message different from these eight will be ignored.

Meaning	MIDI Message (Hex)
Mouth Open	`0x90 0x3C 0x00`
Mouth Close	`0x80 0x3C 0x00`
Head Activate	`0x90 0x3E 0x00`
Head Deactivate	`0x80 0x3E 0x00`
Body Activate	`0x90 0x40 0x00`
Body Deactivate	`0x80 0x40 0x00`
Tail Up	`0x90 0x41 0x00`
Tail Down	`0x80 0x41 0x00`

Table 1. The eight possible MIDI messages

Depending on the message, the function will (de)activate the corresponding movement. In between messages, the program will return to the Main function until the next interrupt occurs.

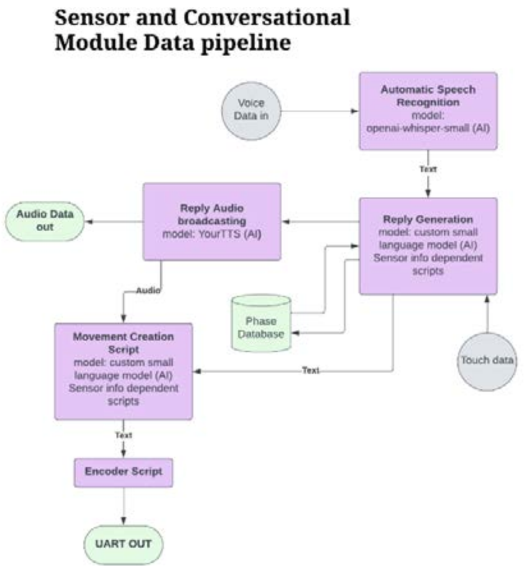

Sensor and Conversational Module Data pipeline
Voice Data in
Automatic Speech Recognition model: openai-whisper-small (AI)
Text
Reply Generation model: custom small language model (AI) Sensor info dependent scripts
Reply Audio broadcasting model: YourTTS (AI)
Audio Data out
Audio
Phase Database
Touch data
Text
Movement Creation Script model: custom small language model (AI) Sensor info dependent scripts
Text
Encoder Script
UART OUT

Fig 4. Block diagram representing how the A.I. processes incoming speech.

IV. HARDWARE DESIGN

Altogether, the IOCB consists of 6 circuits: The MCU and its surrounding components, a +5VDC regulator, a -5VDC regulator, a +3.3VDC regulator, the VOX circuit, and the MIDI optocoupler. Now that the functionality of those circuits was described, here is how each of them are constructed.

A. MCU and Surrounding Components

As mentioned earlier, the IOCB uses the 20 pin version of the MSP430G2553. Of these pins, four are used for the push buttons, one is used for the selector switch, four are used for the outputs, one is used for MIDI, two are used for debugging and two for +Vcc/GND. The 5 connections two the buttons and selector switch go to a set of headers that will be soldered to, the footprint for these headers is seen below:

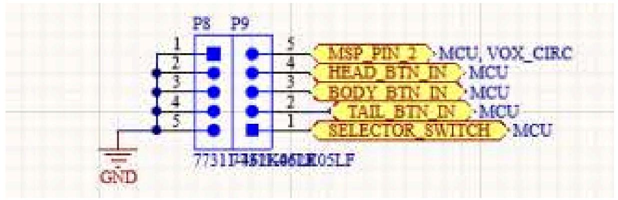


Fig. 5 : The Button and Selector Switch Headers

For each of the 5 connections, there is one header pin going to the MCU, and one header pin going to ground

(10 pins total). This configuration is due to the buttons and switch being set up as active-low in the program. The MSP430 has internal pull-up resistors to limit the current, and shorting the respective pin to ground will trigger the respective bit in the Port's input register. The VOX circuit connects to the same pin as the mouth push button, and the optocoupler connects to the UART RX pin.

The 3.3VDC supply has a 10uF filter capacitor shunted to ground, and the GND terminal connects to the ground plane using a via on the board.

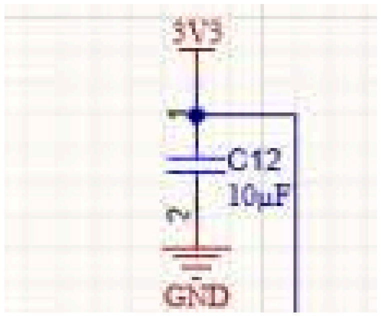


Fig. 6 : The +3.3VDC connection to the MCU

For ease of debugging of this circuit, a pair of header pins allow direct access to the MCU's TEST and RESET pins. In the case of the RESET pin, a pull-up resistor to 3.3VDC is included, and a RESET triggering capacitor is shunted to ground. This capacitor is set to 1nF, to allow for the debugging pulse wave to enter the MCU unaltered. Its main purpose is to momentarily short the reset pin to GND with every startup, resetting the code and preventing any glitches. The MCU can be debugged using the FT232 or CP2102 UART Bridges, or the MSP430G2ET development kit (ezFET). These connections are displayed in the following figure:

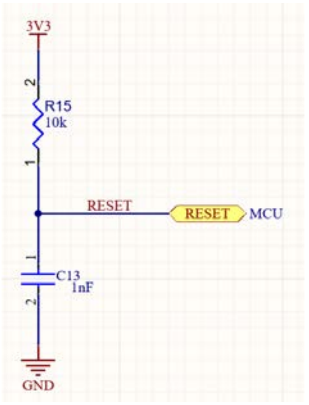


Fig. 7 :The RESET pin pull-up resistor and capacitor

As for the output pulses, they are sent to a set of male headers for direct connection to the off-board relay module.

B. Voltage Regulators

All three voltage regulators on the board use the following configuration:

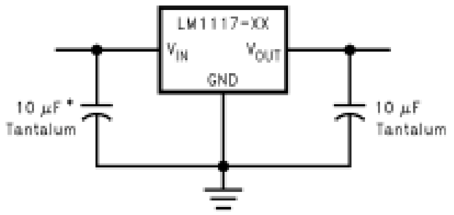


Fig. 8: LM1117-XX "Typical Application" circuit seen in the datasheet, courtesy of Texas Instruments.

For each voltage regulator, there are a pair of capacitors; one shunting the input to ground, and one shunting the output to ground. These filter caps prevent any AC noise from entering the circuit, as well as to smooth out the output voltage leaving the regulators. Their values differ based on what is specified in their datasheets but vary between 2.2uF and 22uF.

For both the +3.3VDC and -5VDC regulators (LM1117-3.3 and LM7905 respectively), the circuits they're powering require very little current. For this reason, we can get away with using linear regulators. For the +5VDC regulator, we needed to use a switching regulator: the VXO7805. This regulator is responsible for powering the external relay board, which can require as much as 300 mA of current. Along with this, protective diodes are also installed in the following configuration:

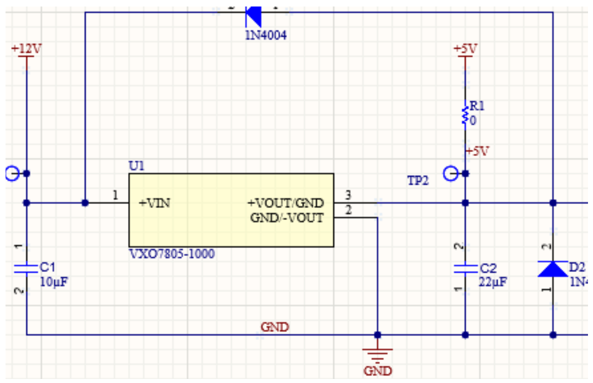


Fig. 9: VXO7805 with filter caps and protective diodes

One diode connects from the output to the input, and the other connects from the ground to the output. These diodes are used to protect the regulator from any possible inrush current that can occur in the relays when energized. Since this is the only regulator with a substantial load, it is the only regulator circuit that has these diodes.

NOTE: The -5VDC regulator requires a negative input voltage. For this, a standalone 12VDC power supply brick is used (with reverse polarity); whereas the +3.3VDC and +5VDC regulators get their supplies from the main power supply.

C. *VOX Circuit*

As briefly illustrated in the red outline in Figure 2, the VOX circuit consists mainly of an amplification stage, a rectification stage, and a switch. More specifically, the audio signal goes through an inverting amplifier with an offset, then into a comparator with an offset, through a half-wave rectifying diode, a ripple-reducing capacitor, a current limiting/load resistor, then into the base of a BS170 MOSFET. Put this all together, and you have a functioning voice-activated switch!

The audio enters the dynamic microphone and creates an analog electrical signal. At this stage the signal is less than 10mV peak-to-peak, which is far too low to reasonably work with. The signal passes through a coupling capacitor and enters the first stage: the inverting op-amp. Since the audio fidelity and phase doesn't matter, an inverting amplification stage is used. This keeps the non-inverting input available for a 1.65V reference voltage to offset the output (Vcc/2). The gain of this stage is 200; equal to the ratio of the feedback resistor to the input resistor (200k and 1k respectively). The audio leaves this stage and enters the comparator.

The comparator takes the signal leaving the amplification stage, and removes a bulk of the negative peaks. This prevents excessive reverse biasing of the diode. Of course the diode isn't even close to its breakdown voltage at any time; either way, the comparator is a good protective measure to take.

After exiting the comparator, the signal is almost completely positive, although not entirely. Despite not having a negative supply voltage, the comparator often swings as low as -300mVDC; enough to instantaneously discharge the capacitor following it. To completely remove the remainder of this negative voltage, a half wave rectifier is used. The figure below shows this post-comparator circuit in its entirety:

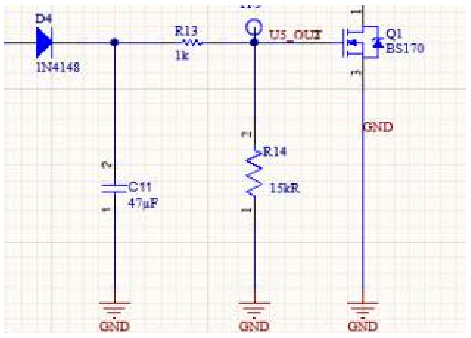


Fig. 10: Post-comparator circuitry

The 47uF capacitor stores charge temporarily and reduces the ripple of the pulse wave to a constant DC stream. All-in-all, this means whenever audio is present at the input, the voltage swings high, whenever audio is NOT present, the voltage goes to zero.

This DC pulse enters the gate of an n-channel MOSFET, with the drain connected to the push-button pin, and the source connected to ground. Whenever speech is present at the microphone, the MOSFET is switched on, and the pin is shorted to GND, activating the mouth movement. Lack of audio shuts the MOSFET off, deactivating the mouth movement.

D. *MIDI Optocoupler*

The final and perhaps most important peripheral circuit next to the MCU itself is the MIDI optocoupler. Through this circuit, all communication with the VRB is possible. It uses the H11L1 optoisolator chip to connect the incoming +5VDC MIDI signal to the MCU safely, and step the voltage down to a more manageable 3.3VDC signal. The H11L1 basic layout is as follows:

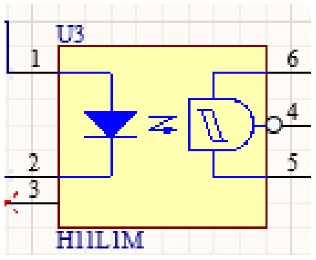


Fig. 11: The H11L1 zoomed in

The signal entering the optoisolator passes the information to the MCU without physical connectivity; protecting the input device and the IOCB from each other. An interesting feature about this optocoupler is how it inverts the data signal entering it. If the input signal swings high, the output swings low and vice versa. Despite this, the MCU has no trouble reading the MIDI message through this device.

V. SOFTWARE DESIGN

A. *Unified Conversation Module (UCM)*

The UCM's software stack includes two primary libraries, Whisper.cpp and LLama.cpp, utilized across two hardware systems and two programming languages. It comprises two main components: the edge block, built in Python, which handles recording audio, sending it to the server for inference, receiving synthesized audio responses, and generating movement commands for CAPER based on the audio content; and the server-side inference block, built in both C++ and Python, which processes the input from the edge system and synthesizes response audio for the edge system to use in movement command generation.

The edge block is local to the Jetson Nano Dev kit, and the server block is local to the Desktop server. An ssh connection to the server is required for the edge block to function.

Once the connection is established the main thread of the edge block behaves as follows:

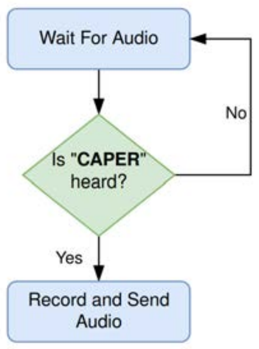


Fig 5. Main Thread For Audio Sending.

Once the input is sent, a loop in another thread takes care of wait for the response it looks as follows:

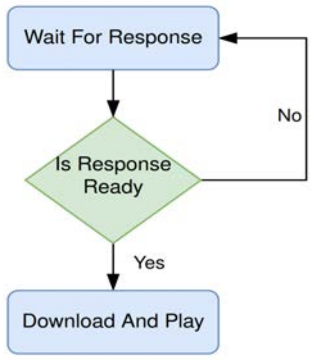


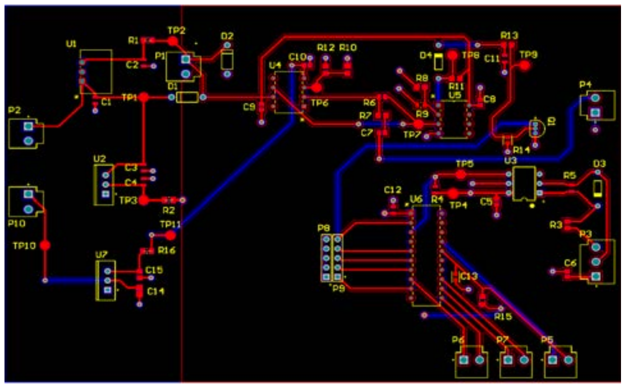
Fig 6. Server Response Retrieval and Playback Thread

The server block runs three models side by side: OpenAI's Whisper Small for speech-to-text, Mistral's 7B Instruct LLM for accurate response prediction, and SpeedySpeech for speech synthesis. Response audio is synthesized, it is retrieved by the client via a targeted SCP (Secure CoPy) execution.

The UCM is integral to providing CAPER with the capability to quickly and accurately respond to user audio input, and its design meets all requirements for this project's success.

VI. BOARD DESIGN

Our project is operated largely from two boards, the Jetson Nano and our very own printed circuit board(PCB). The PCB houses our voltage regulator circuits, as well as the MIDI opto-isolator, Vox Circuit, and MCU. This PCB was designed in Altium and manufactured via JLCPCB. The components on the board were soldered on by our design team to ensure accuracy and to allow for testing of the components/individual circuits before integrating the system. Our board utilizes both a ground plane and 3.3 voltage plane, the former for the ease of grounding and the latter to shortcut routing dilemmas.



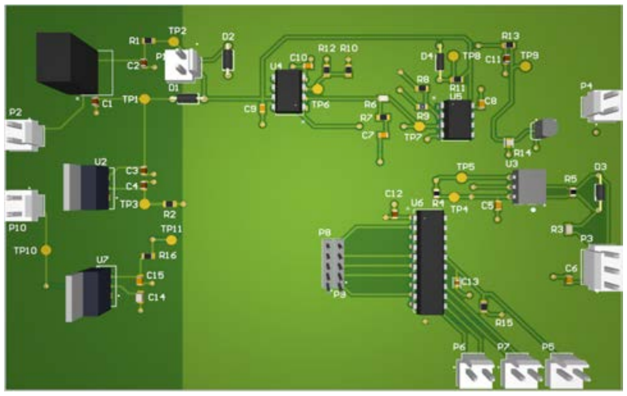


Fig. 5 & 6: Showcases the routing and connections made on our board as well as the planes in both 2D(top) and 3D(bottom)

VII. Conclusion

The preceding systems comprise our Senior Design project and allow our project to display all of the engineering specifications within the guidelines of the project. Our team seamlessly integrated the hardware and software components of our project and was able to troubleshoot where problems arose. As a result of in-depth research, documentation, and application of the skills we have acquired over our time at UCF, we have successfully created a prototype for C.A.P.E.R.

VIII. Biography

Paco-Jaleel Balthazar
Paco is a graduating senior at the University of Central Florida studying Computer Engineering. After graduating, he plans to work in the FPGA field.

William Genevrino
William is a graduating senior at the University of Central Florida studying Electrical Engineering. After graduating, he plans to get a job within the field of Power Distribution.

Kellen Danielsen
Kellen is a graduating senior at the University of Central Florida studying Electrical Engineering. After graduating, he plans to work for SV Microwave in South Florida as an Electrical Engineer.

Sarah Tse
Sarah is a graduating senior at the University of Central Florida studying Electrical Engineering. After graduating, she plans to get a job in Power Distribution.

Acknowledgement

The authors wish to acknowledge the assistance and support of Dr. Vikram Kapoor, Dr. Matthew Gerber, and Dr. Piotr Kulik for serving as our reviewers for this final project.

We would also like to thank Dr. Chung-yong Chan, Arthur Weeks, and Lei Wei for serving as our advisors and assisting in the troubleshooting of our project.

Lastly, we would like to thank and acknowledge all of the professors, students, and UCF as a whole for assisting us throughout the last four years and putting us in a position to create such a project. We are forever thankful.

References

[1] What is MIDI (Musical Instrument Digital Interface)? - Definition from WhatIs.com," *WhatIs.com*. https://www.techtarget.com/whatis/definition/MIDI-Musical-Instrument-Digital-Interface January 2023.

[2] Microelectronics: Circuit Analysis and Design - Donald A. Neamen - Fourth edition 2010