ECE 445

SENIOR DESIGN LABORATORY

INDIVIDUAL PROGRESS REPORT

ClassroomClarity: A Portable Teacher Support Hub

Team #21

JESSE GRUBER
(jgruber5@illinois.edu)

TA: Aishee Mondal
Professor: Michael Oelze

April 2, 2025

Abstract

This document provides a detailed description of my contributions and progress toward ClassroomClarity, our Senior Design final project. The design process, testing of the designs, and current implementation are thoroughly defined.

Contents

1 Introduction

This section presents the scope of the project and the individual responsibilities assigned to this project.

1.1 Project Overview

ClassroomClarity is a device to assist professors in understanding how a class is perceiving lecture information and students in asking questions during lecture. The hub incorporates an LCD screen to display questions asked by students, a single LED to indicate when a question has been asked, LED arrays to illustrate overall class understanding, and a vibration motor to remind a professor that questions have been asked. Students are able to send questions to the hub and professors are able to configure the hub via Bluetooth.

1.2 Individual Contributions

Up to this point, the personal tasks of the project have involved the control subsystem. These tasks include control signal organization, microcontroller pin configurations, external component filter circuits, and processes for signal processing.

To further explain, the control signal organization provides a clear list of what signals will be needed to execute the inner-workings of the hub. The signals based in the control subsystem connect to the power subsystem to power the microcontroller and the feedback subsystem to light the LEDs, turn on the vibration motor, and control the LCD.

Next, the microcontroller pin configurations were established for functionality of the hub. These configurations include assigning a pin a function, labeling a pin as an input or output, and planning around the default purpose set by EspressIf.

Additionally, RC circuits were introduced where mechanical contacts are used. These allow for debouncing the signals from buttons and the rotary encoder. The circuits are also used to pull the buttons high as the buttons connect to ground upon pressing the button.

Lastly, the processes for managing the signals from the rotary encoder and to the LEDs was set. The rotary encoder uses quadrature encoding to describe rotation and the signals to the LED arrays require calculation to portray the class's general understanding of material.

Along with generally helping the team with bringing all aspects of the project together into one unit, I still need to complete the software such that each component of the feedback subsystem acts appropriately to incoming data.

2 Design

This section describes the design process in detail for the individual contributions to the project.

2.1 Debouncing and Pull-up Circuits

Since there are mechanical contacts used in the control subsystem, debouncing circuits must be introduced. These Physical debouncing circuits were assembled instead of using just software for increased signal integrity. RC circuits were used to perform the debouncing. Resistor and capacitor values were chosen using the RC time constant.

$$\tau = R \cdot C \tag{1}$$

In regards to the buttons, a 10 $k\Omega$ resistor and 1 μF capacitor were chosen to complete this task as shown in Fig. 1. These values result in a time constant of 10 ms. Two buttons, EN and GPIO0, are used for programming, and the clear button is used with the feedback subsystem to clear the displayed question from the LCD.

While also utilizing an RC circuit, the rotary encoder featured a different architecture. The rotary encoder is used in conjunction with the feedback subsystem to scroll through the questions from the students to be displayed on the LCD. The design shown in Fig. 2 was suggested by the manufacturer of the encoder [1]. A 10 kΩ resistor was used with a 0.1 μF capacitor to construct this RC circuit.

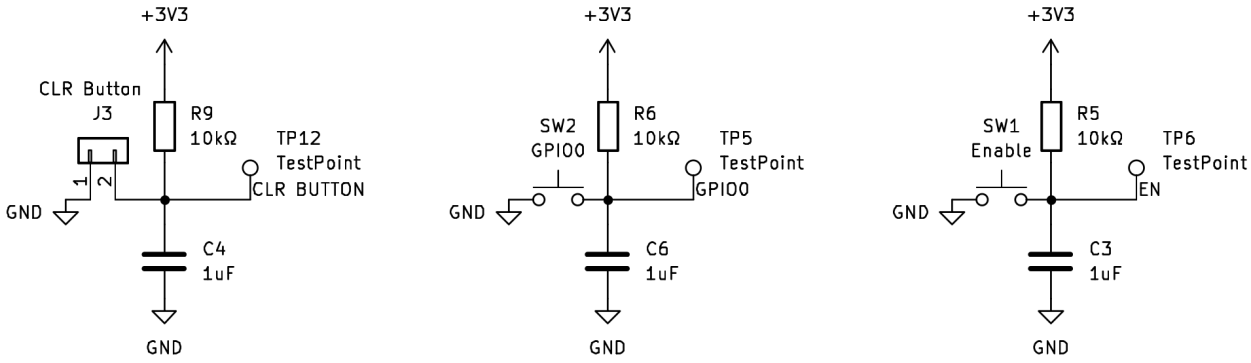

Figure 1: RC filter circuits for button debouncing

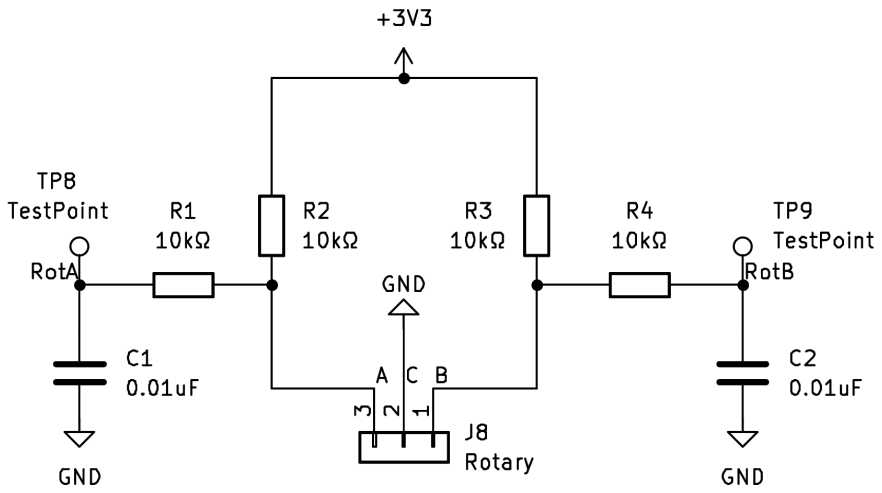

Figure 2: Rotary encoder debouncing circuit

LTSpice was used to confirm the functionality of the filters. Transient simulations were run with a switch emulating bounce from a button. The conditions of 10 kHz switching frequency and a 0.5 duty ratio were used. The switching frequency came from the worst-case scenario of a typical bounce occurring 10 to 100 times over 1 ms [2]. When going from unpressed to pressed and vice versa, the signal of the button as it was "bouncing" remained below the threshold of 0.8 V [3]. The simulation setups with results are shown in Fig. 9(a) and Fig. 10(a) for the button and rotary encoder, respectively.

Along with simulating a bounce event with the RC circuits, the time to surpass 2.64 V from 0 V was also observed. This simulation resulted in the capacitor value of 0.1 μF for the rotary encoder debounce circuit. Simulating the rotary encoder circuit with 1 μF resulted in an approximate response time of 32.26 ms to surpass 2.64 V as pictured in Fig. 11(a). However, it was desired that the response be quicker for robust processing of the signals. The 0.1 μF capacitor simulation in Fig. 12(a) resulted in a response time of 3.23 ms, which was considered adequate. On the other hand, the buttons surpassed 2.64 V in 16.12 ms with 1 μF as portrayed in Fig. 13(a). This response is acceptable as the button signals do not need to have a very high resolution. These buttons do not need to be as precise since the buttons are pressed once in a short interval of time.

Additionally, the circuits have changed since the Design Document was produced. Initially, the rotary encoder filter circuit was pulled up to 5 V. However, the voltage was corrected to 3.3 V to be within the range of the ESP32 input. Also, the 0.1 μF capacitor was removed from the button RC circuits as the 1 μF capacitor is sufficient with the 10 kΩ resistor for debouncing, and the clear button filter circuit took on the same design as the programming buttons for consistency.

2.2 ESP32 Pin Profile

A table was put together to describe the purpose of each pin in one place. The signal table in Appendix B outlines how each signal to/from the microcontroller will connect with each subsystem. Some constraints to pin configuration include acknowledging default pin assignments and adhering to strapping pins for critical microcontroller control signals. These strapping pins are listed in Fig. 3. Another feature of configuring microcontroller pins is the eFuses that may be burned. Since we do not need any of the attributes associated with eFuses, we did not burn any due to the process being irreversible.

Strapping Pin	Default Configuration	Bit Value
GPIO0	Weak pull-up	1
GPIO3	Floating	–
GPIO45	Weak pull-down	0
GPIO46	Weak pull-down	0

Figure 3: Default configuration of strapping pins [3]

2.3 Control System Processes

2.3.1 Rotary Encoder Signal Processing

The chosen rotary encoder uses quadrature encoding to describe the direction of rotation [1]. When the encoder is not being turned, both the signals of terminals A and B are high. When the rotary encoder is being turned, terminal B is low. Terminal A is high if there is counterclockwise rotation and low if there is clockwise rotation. It is important to note that terminal A will change from high and low or vice versa while terminal B remains low; therefore, only the first state of terminal A should be considered in defining the direction of rotation. This output characteristic is the result of the 90° phase shift between terminals A and B. Figure 4 provides pseudo-code for the approach to processing the rotary encoder signals.

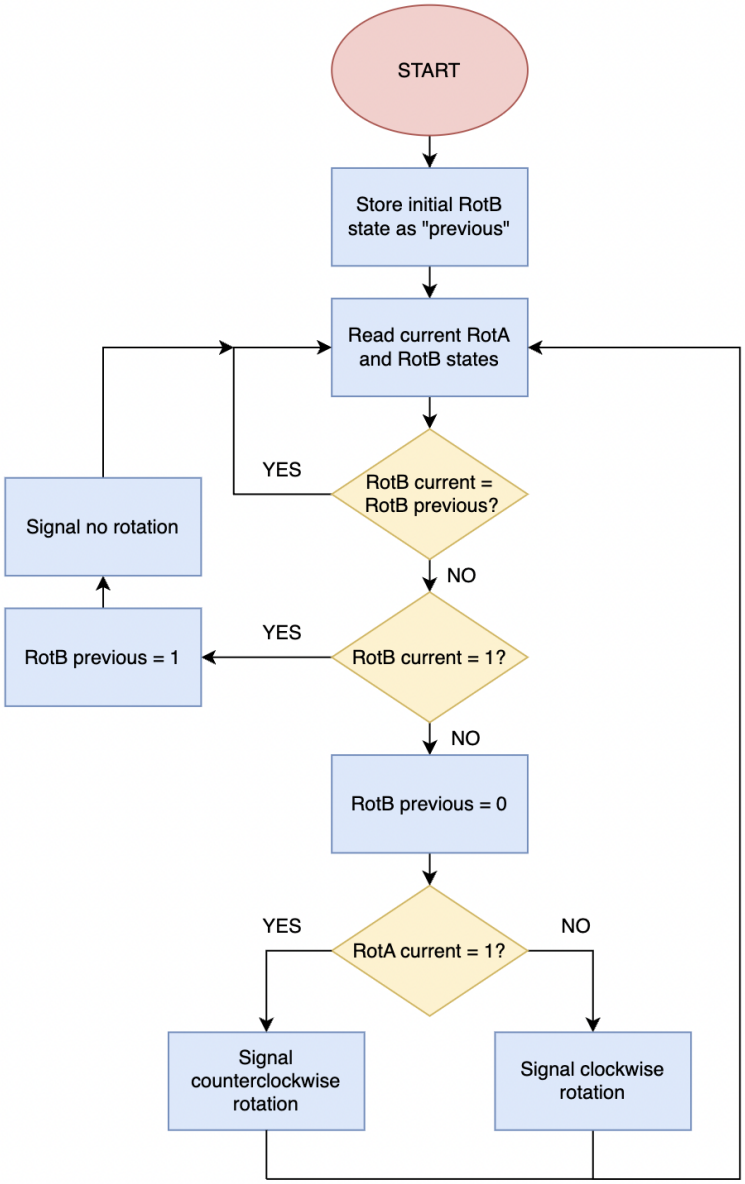

Figure 4: Flow chart of the rotary encoder signal processing procedure

2.3.2 LED Array Lighting

Figure 5 depicts the pseudo-code for the procedure in lighting the arrays of LEDs that correspond to student understanding level. The data used in this process will come from the Bluetooth data received from students. At this point, the method of determining the class size is not yet set. Two options, count the number of devices connected or set by the professor in the application, are being explored.

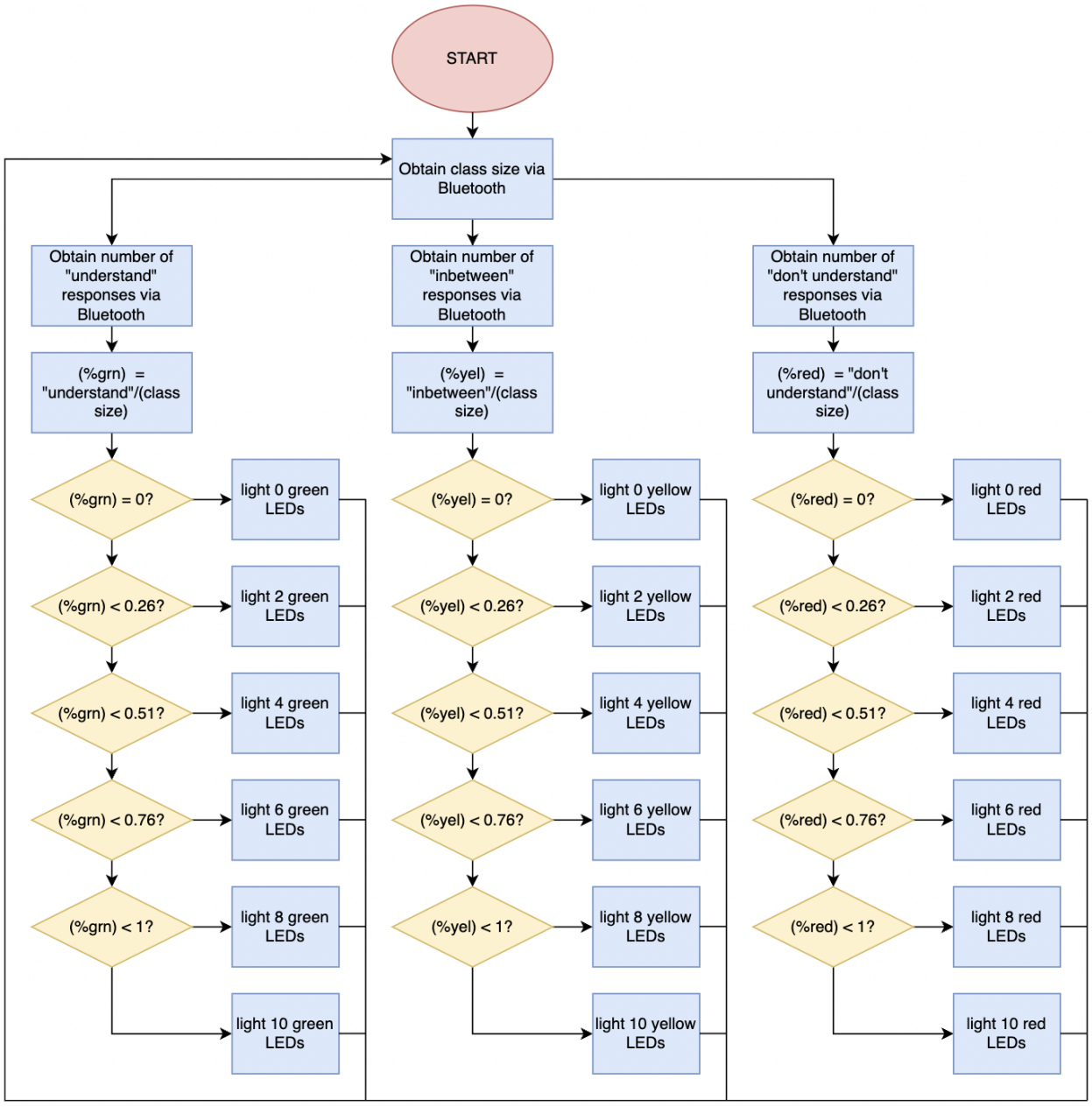

Figure 5: Flow chart of lighting LED arrays based on student level of understanding

3 Verification

This section provides test procedures and results to confirm the expected operation of the control system.

3.1 ESP32 Rotary Encoder Input

At this point, the rotary encoder has been tested while off the PCB and not connected to the microcontroller. The list indicates the requirement and the test procedure used to confirm the requirement.

- The rotary encoder processed signals must have a maximum between 2.5-3.6 V and a minimum between -0.3-0.8 V
    1. Probe processed signals RotA and RotB using an oscilloscope
    2. Begin turning the encoder and observe the phase shift of 90° between RotA and RotB
    3. Confirm the maximum voltage is between 2.5-3.6 V and the minimum voltage is between -0.3-0.8 V

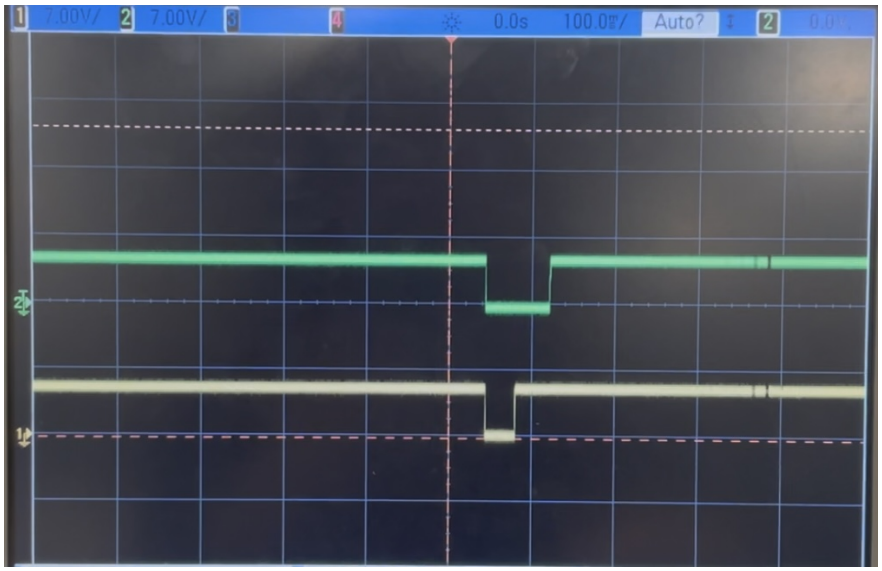

Figure 6: RotB (green) and RotA (yellow) traces of clockwise rotation

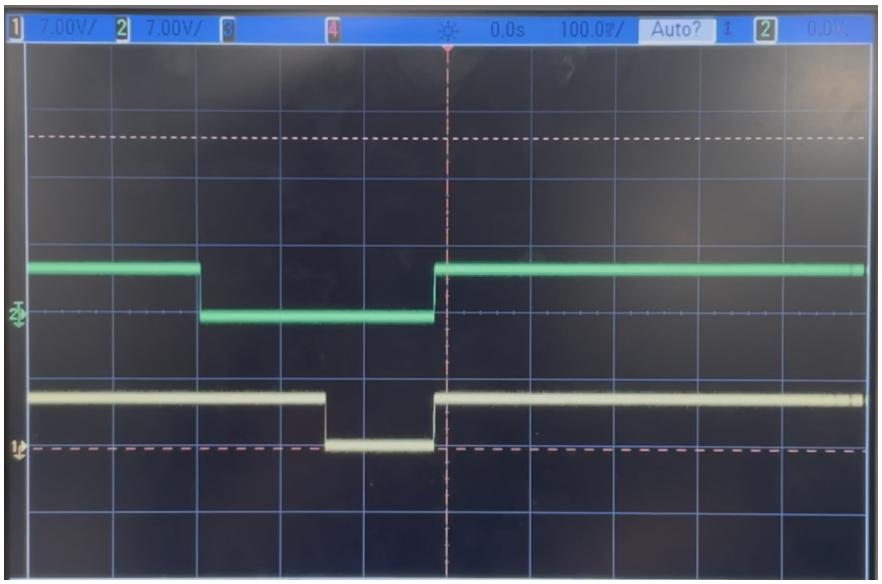

Figure 7: RotB (green) and RotA (yellow) traces of counterclockwise rotation

The rotary encoder must still be tested while on the custom PCB and connected to the microcontroller. However, the rotary encoder signals were successfully used to detect rotation during the Breadboard Demo and measured voltages in rated values while tested independently. Oscilloscope captures of clockwise and counterclockwise rotation signals are shown in Fig. 6 and Fig. 7, respectively. There was an initial issue in the code that would register clockwise and counterclockwise during one cycle but has since been resolved by fixing how terminal A's signal was processed.

3.2 ESP32 Button Input

This test has not yet been conducted, but the procedure to be used is outlined.

- The unpressed input voltage of each button GPIO must fall between 2.64-3.6 V, and the pressed input voltage must be between -0.3-0.8 V within one second of pressing the button
    1. Probe test points of each button using a multimeter
    2. Confirm the unpressed voltage is between 2.64-3.6 V
    3. Confirm the pressed voltage is between -0.3-0.8 V within a second of pressing the button

During the Breadboard Demo, the clear button successfully cleared questions off of the LCD. Additionally, the buttons used for programming were successful in setting the microcontroller to be programmed on the custom PCB. However, measurements have not been taken as of now.

3.3 ESP32 GPIO Output

This test has been partially completed.

- The low output voltage of each GPIO must be less than 0.33 V, and the high output voltage must be greater than 2.64 V
    1. Set a GPIO pin to low using Arduino IDE and program the microcontroller
    2. Probe the corresponding GPIO output pin on the microcontroller using a multimeter
    3. Confirm the low output voltage is less than 0.33 V
    4. Set a GPIO pin to high using Arduino IDE and program the microcontroller
    5. Probe the corresponding GPIO output pin on the microcontroller using a multimeter
    6. Confirm the high output voltage is greater than 2.64 V within a second of pressing the button

There are ongoing issues while trying to light an LED on the custom PCB. Because of this, the output voltage of GPIO pins was observed and within range. However, these measurements were not recorded, so the test will be repeated. This test is critical to the functioning of the feedback subsystem.

3.4 ESP32 SPI Output to LCD

This test still needs to be completed on the custom PCB.

- The LCD should display a maximum of 200 characters with a black background and white font
    1. In the microcontroller code, utilize the TFT_eSPI library included in the Arduino IDE to run the example TFT_Print_Test
    2. Upload the code to the microcontroller
    3. Confirm the demo's operation on the screen

The test was nearly completed during the progression to the Breadboard Demo. The LCD successfully ran the example code, but the number of characters that could be displayed was not confirmed.

4 Conclusion

This section outlines the timeline for the remaining work and a self-assessment.

4.1 Self-Assessment

I believe I have contributed sufficiently to the exceptional progress of our project. Along with my individual design and testing, I have contributed to group testing and debugging during our meetings. Our individual work has been coming together as a system very well thus far.

4.2 Remaining Work

The main issue to be tackled at this point is figuring out why the GPIO pins are not responding to being set high and low. Once this is completed, the feedback subsystem must be coded and tested and successfully react to data coming from Bluetooth. Prototype debugging and assembly must still be completed. Lastly, the final product will be completed, and the remaining documents and presentation must be finished.

4.3 Timeline

A majority of the tasks have been completed on time. We are slightly behind on the housing for the components and bringing the entire design into one working unit. The timeline of our goals for the rest of the semester is in Fig. 8.

Week of	Task	Group Member(s)
3/31	Fix GPIO driving issue	Jesse
	Feedback subsystems react properly to incoming data	Jesse
4/7	Edit and print final casing design	TBD
4/14	Final assembly	All
	Begin Final Report and Presentation	All
4/21	Mock Demo	All
	Final Presentation	All
4/28	Final Demo	All
	Complete Final Report	All
	Mock Presentation	All
5/5	Final Presentation	All

Figure 8: Projected timeline for completing remaining tasks

References

[1] Bourns. "PEC16 - 16 mm Incremental Encoder." (2023), [Online]. Available: https://www.bourns.com/docs/Product-Datasheets/pec16.pdf. (visited on 03/06/2025).

[2] J. Christofferson. "Switch Bounce and How to Deal with It," All About Circuits. (2015), [Online]. Available: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/#:~:text=%22When%20the%20switch%20is%20closed,any%20sort%20of%20bounce%20control:. (visited on 03/31/2025).

[3] Espressif Systems. "ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U Datasheet." (2021), [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf. (visited on 03/06/2025).

Appendix A LTSpice Simulation

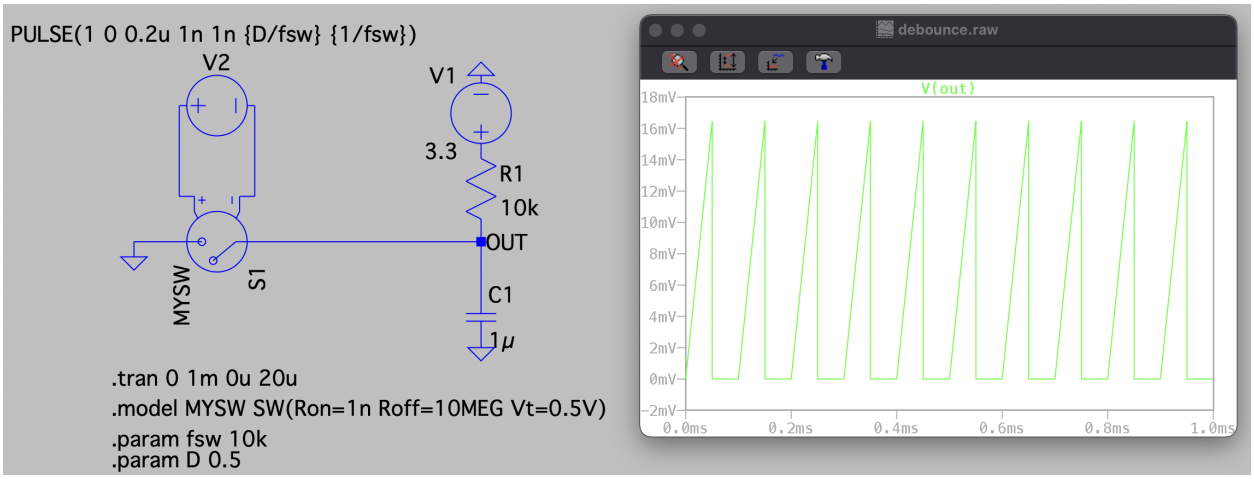

Figure 9: LTSpice simulation of button RC circuits during bounce

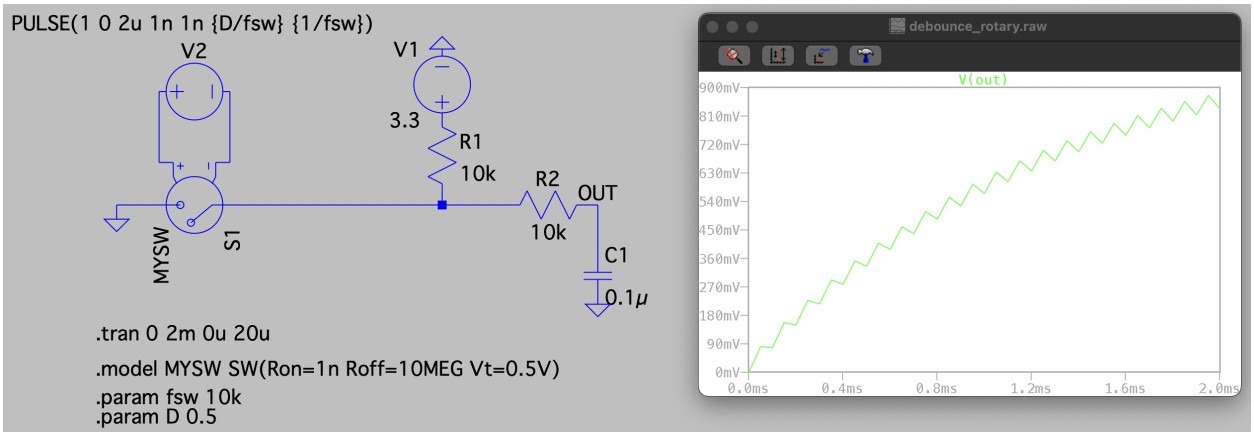

Figure 10: LTSpice simulation of rotary RC circuit during bounce

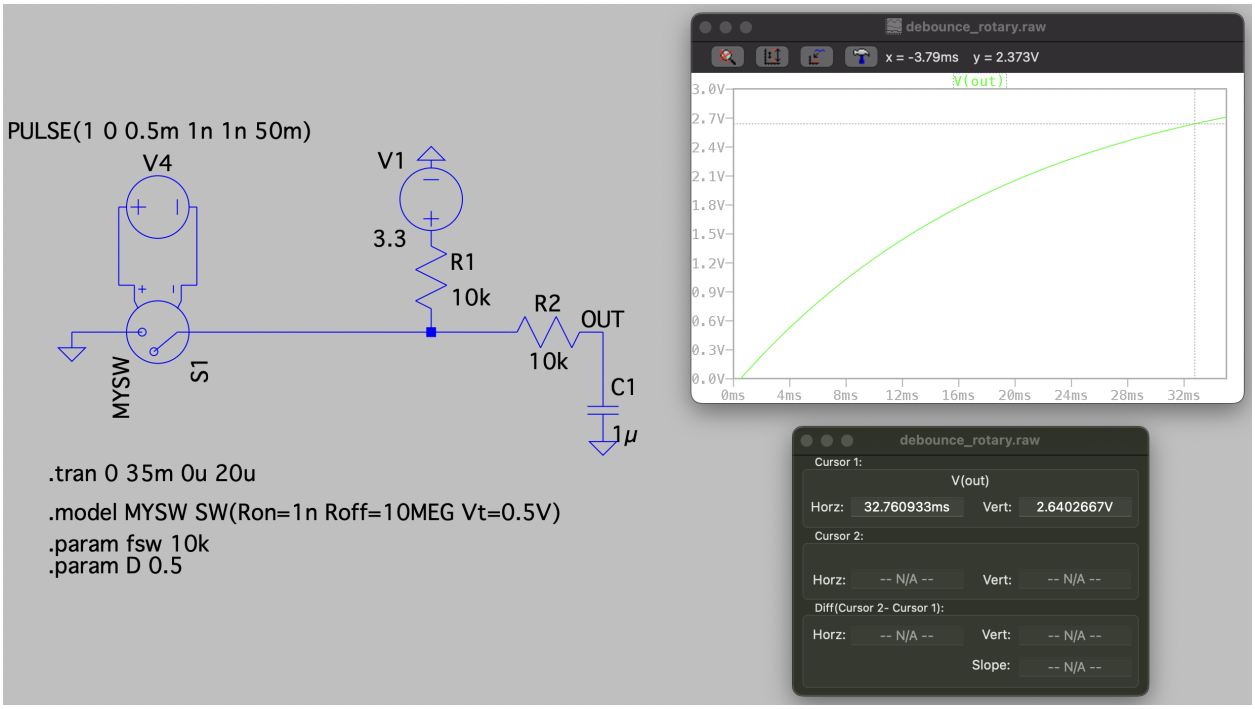

Figure 11: LTSpice simulation of rotary RC 1 μF circuit from 0 V to 2.64 V

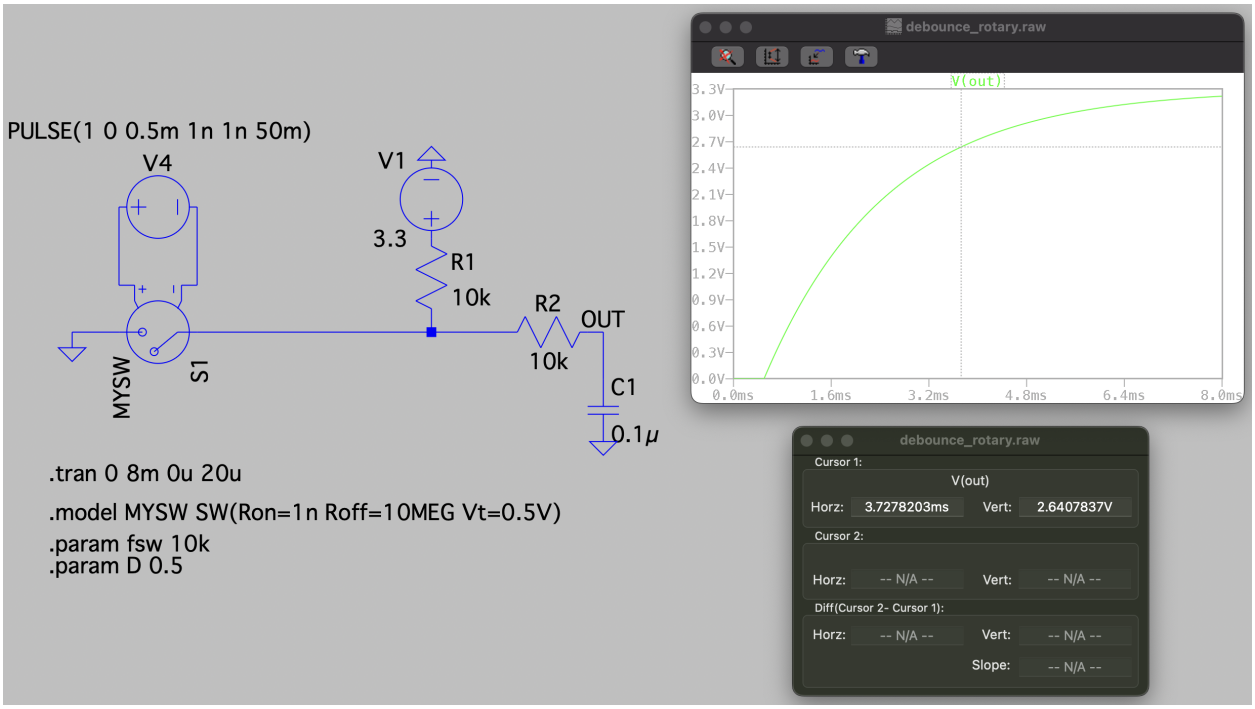

Figure 12: LTSpice simulation of rotary RC 0.1 μF circuit from 0 V to 2.64 V

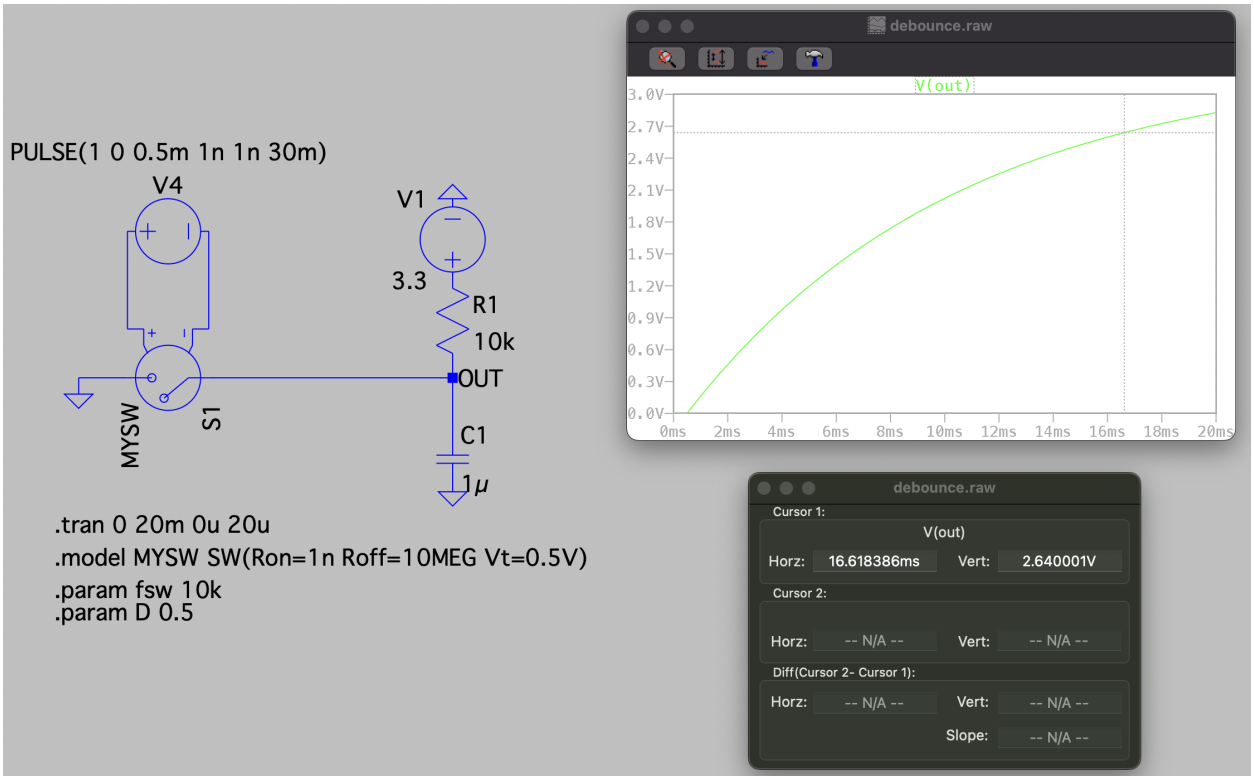

Figure 13: LTSpice simulation of buttons RC 1 μF circuit from 0 V to 2.64 V

Appendix B GPIO Pin Configurations

This figure is based on the pin definitions tabulated in the ESP32-S3-WROOM-1 datasheet [3].

Name	Pin Number	Function
GND	1	GND
3V3	2	Power supply
EN	3	Chip enable
IO4	4	Input; rotary encoder A
IO5	5	Input; rotary encoder B
IO6	6	Input; clear button
IO7	7	Output; vibration motor
IO15	8	Output; green LED set 1
IO16	9	Output; green LED set 2
IO17	10	Output; green LED set 3
IO18	11	Output; green LED set 4
IO8	12	Output; green LED set 5
IO19	13	USB_D-
IO20	14	USB_D+
IO3	15	Output; blue indicator LED
IO46	16	NC; programming strapping pin
IO9[1]	17	SPI; MISO
IO10[1]	18	SPI; SCK
IO11[1]	19	SPI; MOSI
IO12[1]	20	SPI; DC/RS
IO13[1]	21	SPI; RST
IO14[1]	22	SPI; CS
IO21	23	Unused
IO47	24	Unused

cont'd on next page

Name	Pin Number	Function
IO48	25	Unused
IO45	26	NC; programming strapping pin
IO0	27	Programming strapping pin
IO35	28	Output; red LED set 5
IO36	29	Output; red LED set 4
IO37	30	Output; red LED set 3
IO38	31	Output; red LED set 2
IO39[1]	32	Output; red LED set 1
IO40[1]	33	Output; yellow LED set 5
IO41[1]	34	Output; yellow LED set 4
IO42[1]	35	Output; yellow LED set 3
RXD0	36	UART receiving pin
TXD0	37	UART transmitting pin
IO2	38	Output; yellow LED set 2
IO1	39	Output; yellow LED set 1
GND	40	GND
EPAD	41	GND

[1]: Differs from default function