ECE167 Sensing and Sensor Technology

Lab2

James Huang

Collaborator: AI

Checkoff TA: various time/various TA

Commit: 2710ff0b6ae412e91c4545e09556a24e45d36d2a

Table of contents:

1. Project Introduction 1
2. System Block Diagram 2-5
3. Hardware Components 6-8
4. Software Components 9-12
5. Testing and Result 13-18
6. Conclusion 19

1. Project Introduction

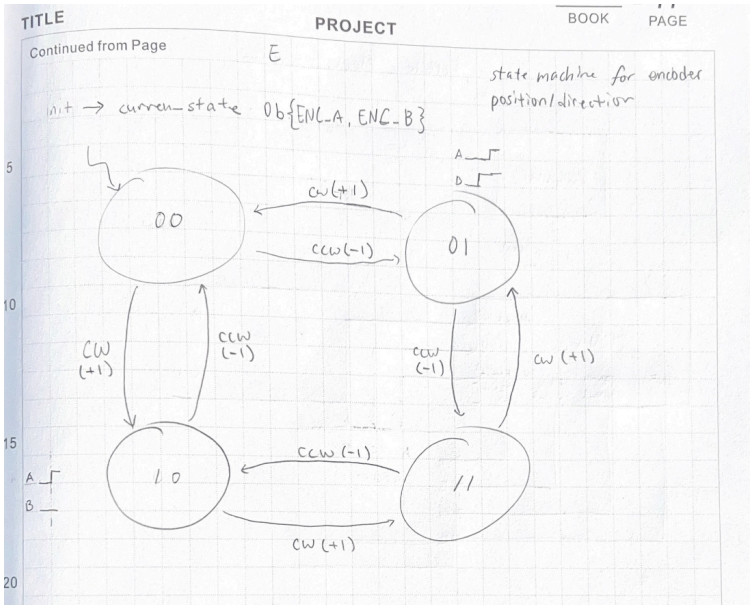
The first part of Lab2 is to set up the encoder with our MCU. The header file is already given with needed functions and initialization code. The corresponding source file is to be created. The encoder should produce a positive or negative degree each time it is turned, either rolled over or continued after 360 degrees. The on board LED should change color corresponding to the color wheel given in the lab document.

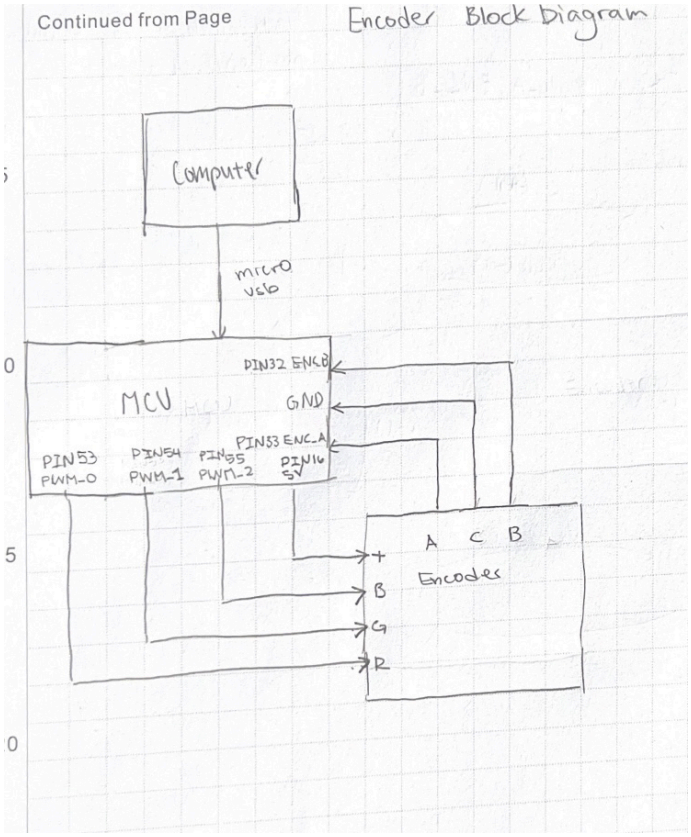
The second part is to set up the ultrasonic sensor, which detects distance in millimeters. This change in distance reading should be accurate and converted to change in tone output with the speaker. In addition, a least square regression should be used for the distance reading, implemented first in MATLAB and then in C code.

The third part is to set up the capacitive touch sensor. First, a simple RC circuit (touch sensor is our capacitor) to see the change when we touch the sensor on an oscilloscope. Second, a capacitive bridge and instrumental amplifier is constructed to see the difference when we touch the sensor. The Instrumental amplifier is made up of 3 op-amps (MCP6004). Third, a relaxation oscillator is a build from LM555 Timer IC, in astable mode. This circuit will produce a square wave with 1-5KHz frequency with correct resistor values. The touch sensor will be hooked up to change the frequency of the output as we touch the sensor. Lastly, this changing output square wave will be an input to the microcontroller, and based on the change, it will determine is the touch sensor is activated.

2. System Block Diagram

State machine for the encoder:

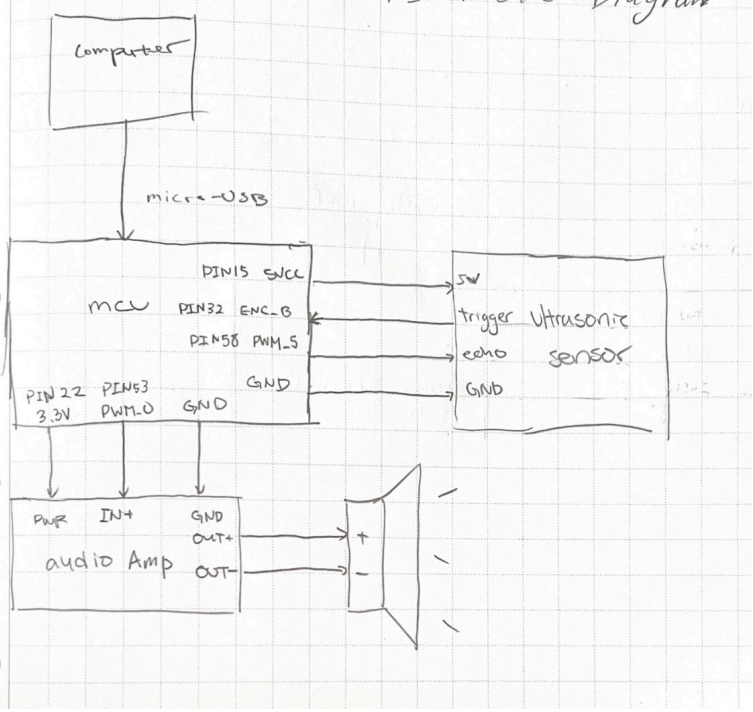




PROJECT

Continued from Page

PING Block Diagram



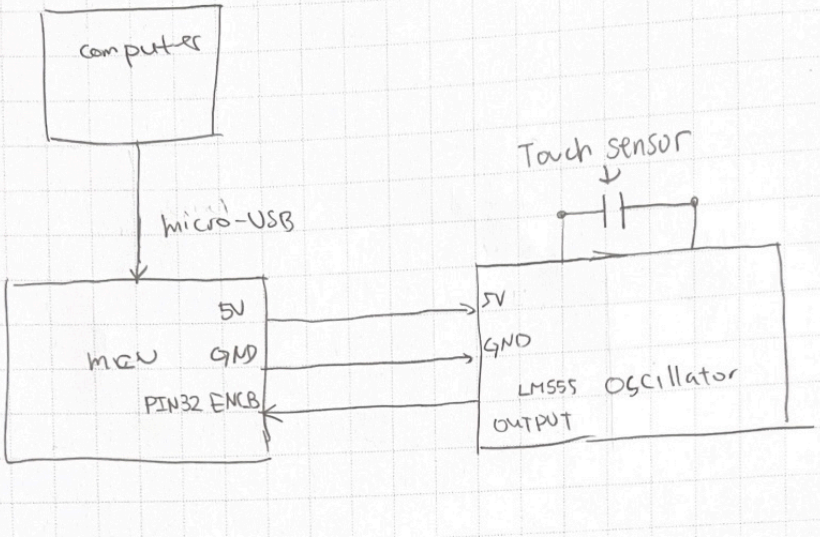
BC

TITLE

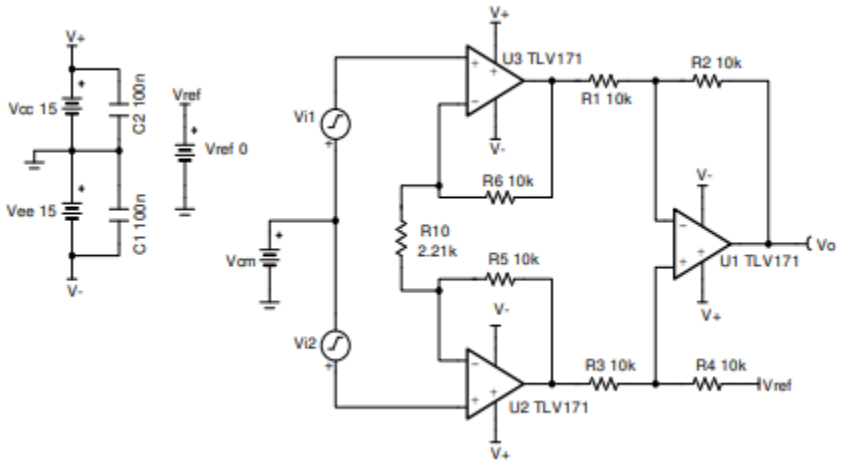
PROJECT

Continued from Page

CAPTOUCH Block Diagram



The simple RC circuit diagram can be found in the lab2 documentation. The instrumental amplifier circuit diagram can be found in a Texas Instruments document called Analog Engineer's Circuit, Three Op Amp Instrumentation Amplifier Circuit, as seen below.



The bridge circuit is shown in the Lab2 documentation, as seen below.

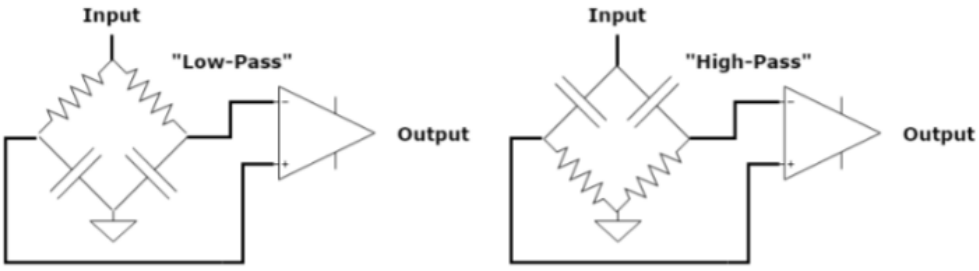


Figure 4: Capacitive Bridges for Capacitance Sensing

The LM555 relaxation oscillator circuit diagram can be found in the lab2 documentation, as seen below.

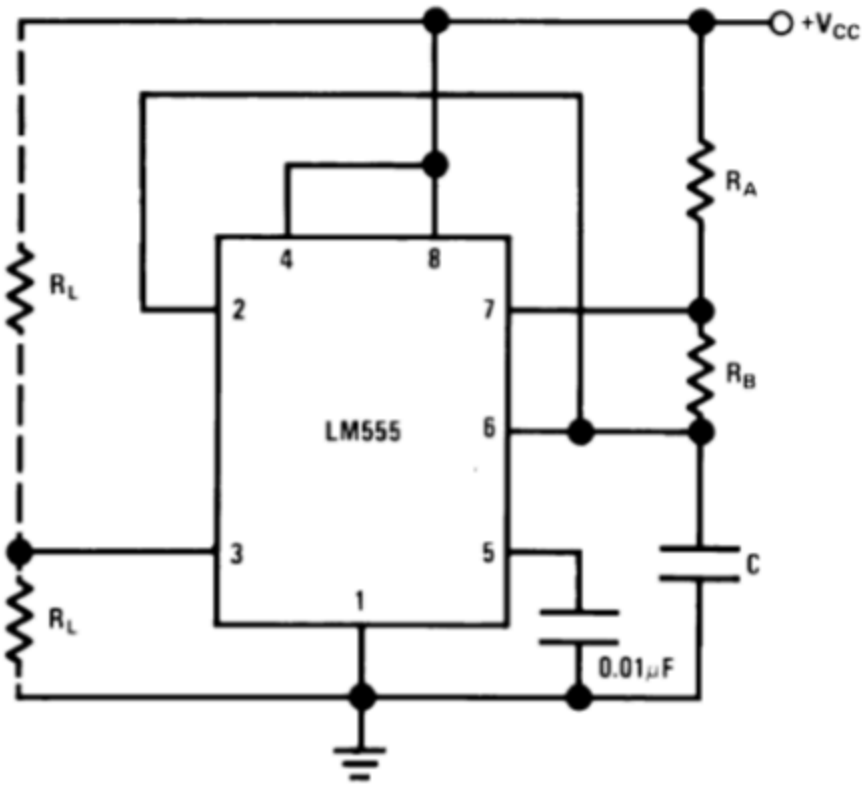


Figure 5: Example 555 Oscillator Circuit

3. Hardware Components

The SparkFun Rotational Encoder is an incremental encoder that generates two output signals, A and B, in a quadrature configuration to track rotational movement. These signals are phase-shifted by 90°, enabling both the magnitude and direction of motion to be determined. For example, if Signal A transitions high while Signal B remains low, the encoder is rotating clockwise (CW). Conversely, if Signal B transitions high while Signal A remains low, the rotation is counterclockwise (CCW). The encoder also features an onboard RGB LED that can be controlled via PWM signals to produce various colors, providing visual feedback for system states or user-defined indications. The LED requires a 5V power supply for operation and is integrated into the encoder for added functionality. This setup ensures reliable position tracking while offering versatile feedback options.

The HC-SR04 Ultrasonic Sensor is a non-contact distance measurement module capable of detecting objects between 2 cm and 4 m with an accuracy of ±3 mm. Operating at 40 kHz, it works by emitting an 8-cycle ultrasonic burst and measuring the time taken for the echo to return after reflecting off an object. The distance is then calculated using the formula found in the datasheet attached in Lab2 documentation. The module has four pins: VCC (5V) for power, GND (0V) for grounding, Trigger (TRIG) to initiate a measurement with a 10 µs pulse, and Echo (ECHO) which outputs a high signal for the duration of the returning echo. The width of this high signal corresponds to the time taken for the ultrasonic pulse to travel to the object and back. A minimum 60 ms delay between consecutive measurements is recommended to prevent signal interference. For optimal accuracy, the detected object should have a minimum surface area of 0.5 m² and be as smooth as possible.

A capacitive touch sensor detects touch by measuring changes in capacitance. When a finger approaches or touches the sensor pad, it introduces an additional capacitance, altering the existing electric field. We use projected capacitance, where a gridded ground plane beneath the pad forms a small capacitor that changes in the picofarad range upon contact.

The lowpass and highpass bridge circuit is often used in conjunction with an instrumentation amplifier (represented by an op-amp symbol) to amplify the differential voltage between the two midpoints of the RC network. This allows for precise filtering and amplification of specific frequencies, making the setup ideal for signal conditioning applications where only certain frequency ranges need to be measured or analyzed.

An instrumentation amplifier (in-amp) is typically constructed using three operational amplifiers (op-amps) to achieve high gain, high input impedance, and excellent common-mode rejection. The first stage consists of two buffer amplifiers, each connected to one of the input terminals, ensuring minimal loading on the source signals. These buffers feed into a differential amplifier in the second stage, which subtracts the two input signals and amplifies the difference while rejecting common-mode noise. A resistor placed between the buffers sets the overall gain, allowing easy adjustment. This design enables the in-amp to accurately amplify small differential signals in the presence of large common-mode noise, making it ideal for precise low-level signal measurements in applications such as sensor interfacing, medical devices, and audio systems.

The LM555 timer is a versatile integrated circuit commonly used in a variety of timing and pulse generation applications. In astable mode, it functions as a relaxation oscillator, producing a continuous square wave output without the need for external triggering. This is achieved by configuring the 555 timer with four resistors and two capacitors connected to its

pins. The resistors control the charge and discharge rates of the capacitor, which determines the frequency of the oscillation. As the capacitor charges and discharges, the output alternates between high and low states, creating a square wave. The frequency of this wave can be adjusted by changing the resistor values, allowing for precise control over the timing.

4. Software Components

The software for the quadrature encoder interface (QEI) is implemented in C, with initialization handled by the `QEI_Init()` function, which configures GPIO pins PB4 and PB5 for rising and falling edge interrupts. Signals A and B are concatenated into a 2-bit value, formed by combining the values of A and B (`{A, B}`), representing one of four states in a finite state machine. The `QEI_IRQ()` function compares the current state to the previous state to determine rotation direction, incrementing or decrementing the signed int `encoder_position` accordingly, tracking positive and negative rotations. External interrupt handlers (`EXTI9_5_IRQHandler()` and `EXTI4_IRQHandler()`) call `QEI_IRQ()` to ensure efficient state processing. The `QEI_GetPosition()` function retrieves the current position, while `QEI_ResetPosition()` resets it to zero for re-initialization or relative tracking. This design provides accurate, low-overhead position tracking, ideal for applications like motor control or user input.

The main program initializes the board, encoder, and PWM modules to read the encoder's position and convert it to a degree value. The degree is normalized to fall within ±0–360° and 15° per tick for easy interpretation. The `QEI_SetColor()` function sets the PWM duty cycle for RGB LEDs based on the normalized degree, cycling through colors as the encoder rotates. Note, setting PWM duty cycle to 0 is the same as turning on the color and duty cycle of 100 turns off the color, and frequency is not required to set the colors.

The second part of the PING code implements the functionality for an ultrasonic sensor using a combination of GPIO pins, timers, and interrupts on the STM32 microcontroller. The `PING_Init` function initializes the necessary peripherals, including setting up the GPIO pin for the trigger signal (PB8) and configuring Timer 3 to generate interrupts at a frequency of 1 MHz using Auto-Reload Register (ARR) timers. The ARR timers are used to define the duration of the pulse signal and echo signal intervals. In the case of generating the 10 µs pulse, the timer's auto-reload value is set to 10, ensuring the pulse lasts for precisely 10 microseconds. For the 60 ms interval, which is the time the system waits after sending the pulse, the ARR is set to 60,000, corresponding to a 60 ms period.

The GPIO pin PB5 is set to trigger interrupts on both rising and falling edges, which correspond to the echo signal from the ultrasonic sensor. The `EXTI9_5_IRQHandler` interrupt service routine (ISR) handles the changes in the echo pin's state, capturing the start and end times of the echo signal based on the rising and falling edges.

The `TIM3_IRQHandler` ISR controls the timing of the ultrasonic pulse. In the `current_state` machine, when the state is 0, it generates a 10 µs high pulse on the trigger pin (PB8) to initiate the ultrasonic measurement. After 10 µs, the state changes to 1, where the pulse is turned off, and the timer is set to 60 ms, which is the maximum range for the sensor. The sensor's echo is measured by capturing the duration between the rising and falling edges of the echo signal.

The `PING_GetTimeofFlight` function calculates the raw duration of the echo signal in microseconds by subtracting the start time from the end time. Finally, the

`PING_GetDistance` function uses the time of flight to compute the distance to the object, applying the speed of sound (340 m/s) and converting the result to millimeters. This distance is then returned as the measurement.

The MATLAB code that implements a least squares regression to calibrate an ultrasonic sensor by relating the actual distance measurements (x_i) to the sensor readings (y_i). The process begins by forming the X and Y matrices, where X contains the actual distances and a column of ones to account for the intercept term, and Y holds the sensor readings. Using MATLAB's backslash operator, the code solves for the slope (m) and y-intercept (b) of the best-fit line, providing the relationship between the sensor readings and actual distances. A conversion function is then created to convert sensor readings to the corresponding distance values. The code also plots the raw data points and the fitted line for visual validation, and calculates the prediction error by comparing the actual sensor readings to the predicted values from the regression model. The mean and standard deviation of the error are displayed to assess the model's accuracy. The least squares regression is implemented in C at the end by using the formula to adjust the raw distance measurement based on the slope (m) and y-intercept (b) of the regression line, effectively normalizing the raw data to a corrected value.

The CAPTOUCH.c code implements a capacitive touch sensor using an external interrupt on GPIO pin PB5, with the goal of detecting touch events and filtering frequency measurements. The `CAPTOUCH_Init` function initializes the GPIO pin and sets up an interrupt for the rising edge of PB5, while also initializing the system timer to capture the time of touch events. The interrupt service routine (`EXTI9_5_IRQHandler`) calculates the time between two consecutive rising edges to determine the frequency of the touch sensor, and applies a moving

average filter to smooth the frequency values. The `CAPTOUCH_IsTouched` function returns a boolean indicating whether a touch is detected, based on the filtered frequency. If the frequency is below a threshold, it returns `FALSE` (no touch), otherwise, it returns `TRUE` (touch detected).

5. Testing and Result

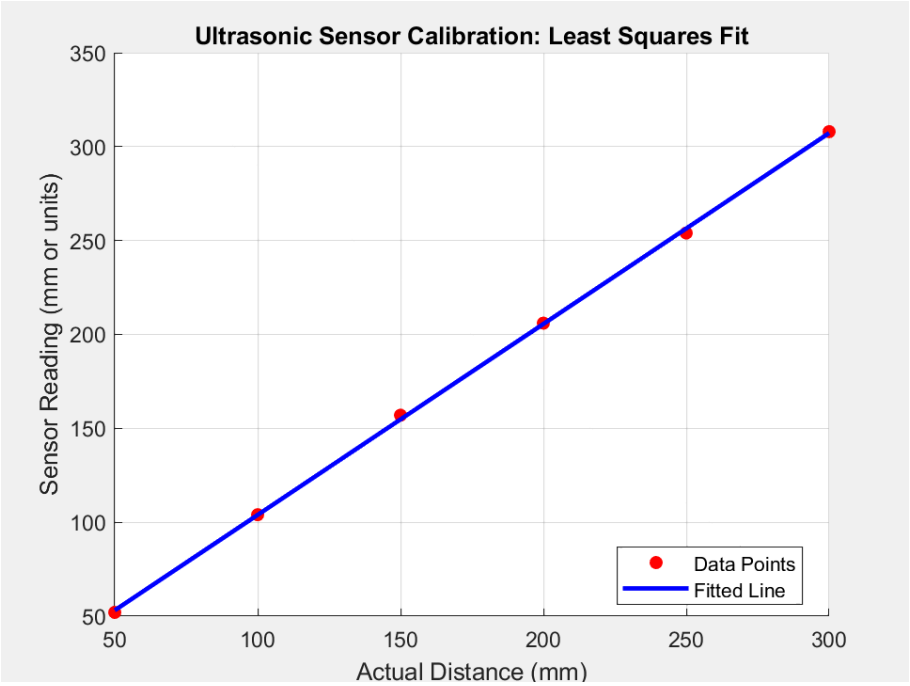
To validate the functionality of the encoder, I tested its behavior through software. The primary test involved monitoring the state transitions of the encoder signals A and B. Specifically, I verified that a state change from `00` to `01` corresponds to a counterclockwise (CCW) rotation, while a change from `00` to `10` indicates a clockwise (CW) rotation. This ensured that the encoder's response matched the expected rotational direction and was not inverted, confirming the correct operation of the quadrature signal processing. The color wheel encoder was done by setting each 15 degree increment to its corresponding color code.

To calibrate the ultrasonic sensor, I collected raw distance readings from the sensor and measured the corresponding actual distances.

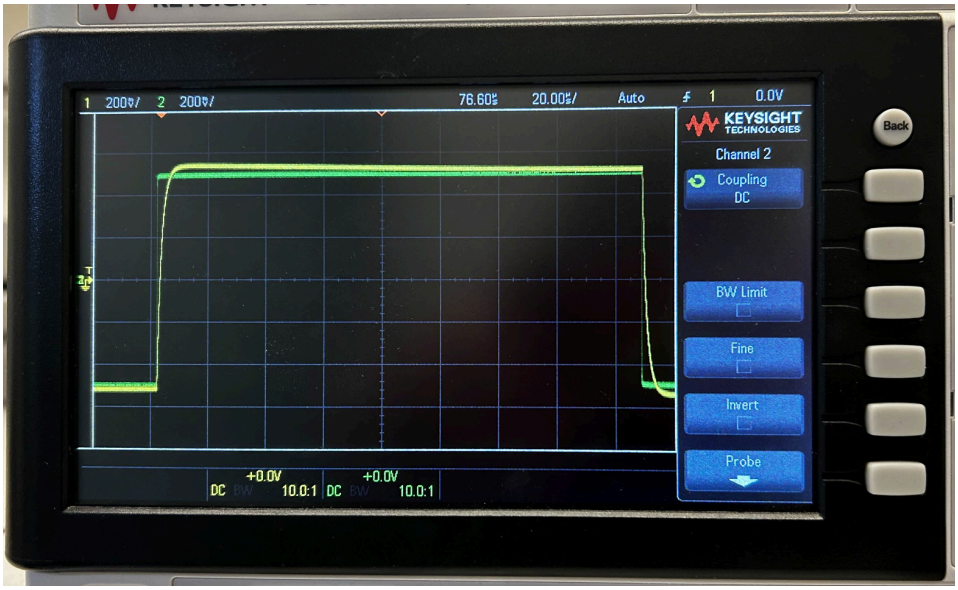
Sensor Raw Reading (mm)	Actual Distance (mm)
51	50
104	100
156	150
201	200
262	250
319	300
356	350
397	400
452	450
506	500

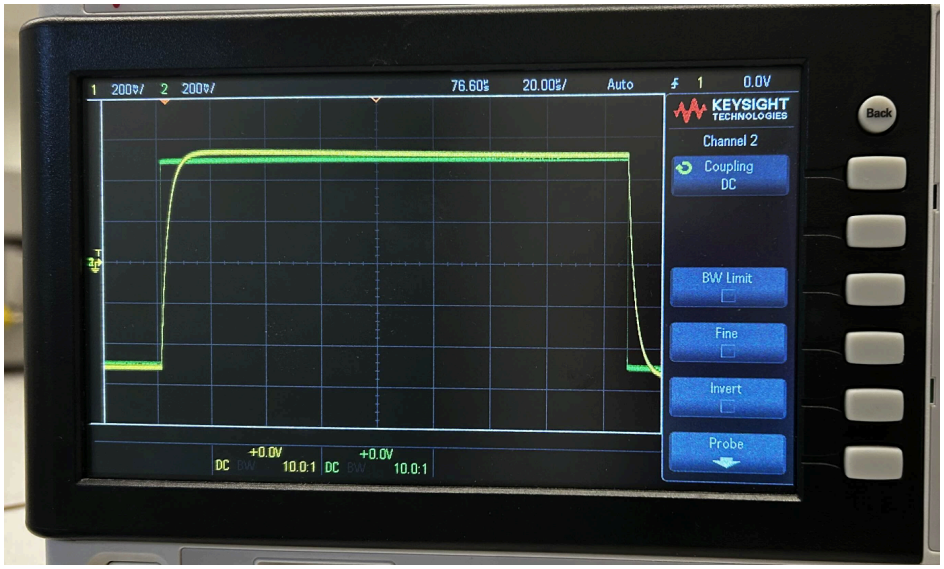
Using MATLAB, I performed a least squares regression on the data to generate a model, which resulted in a slope (m) of 1.0166 and an intercept (b) of 2.2667. This model was then

implemented in C code to convert the raw sensor readings into accurate distance measurements during operation. The graph can be seen below.

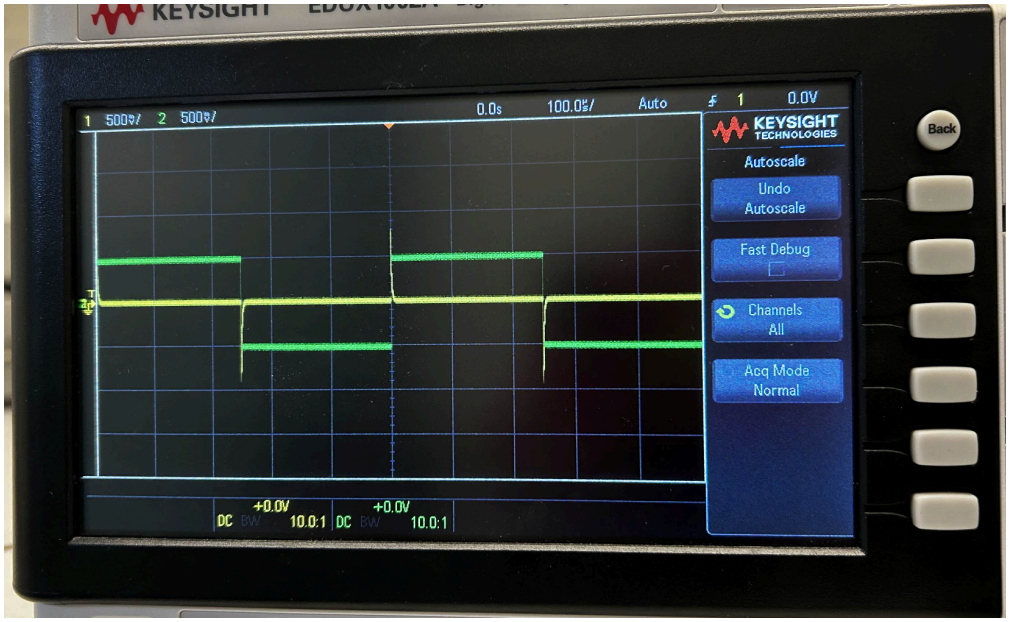


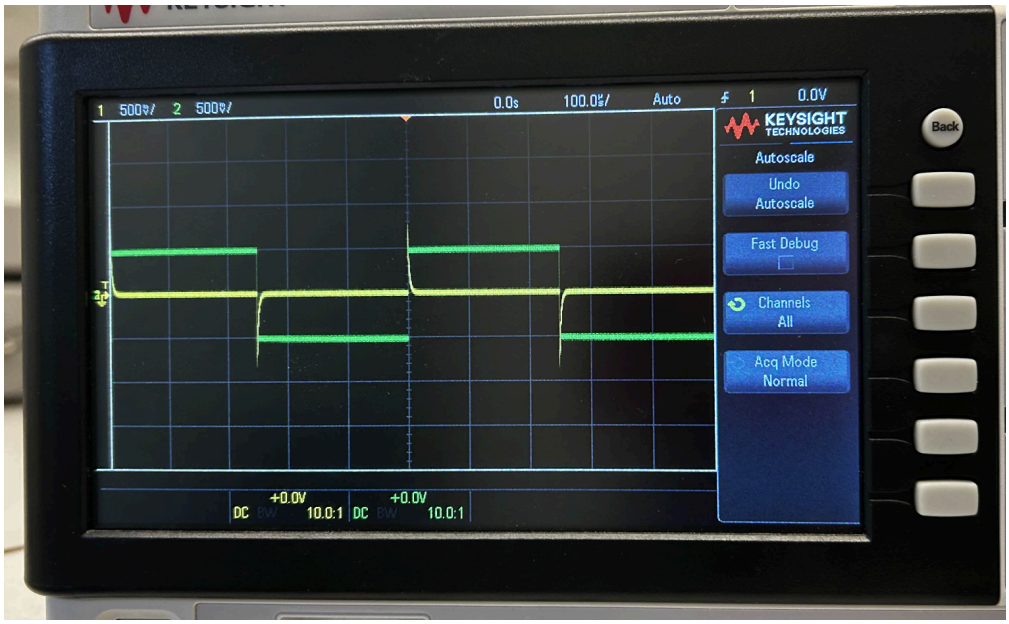
I used a simple RC circuit connected to the capacitive touch sensor, with a 300kΩ resistor, and applied a square wave signal with a 1V peak-to-peak amplitude and a frequency of 2kHz as the input. I observed the change in capacitance when I touched the sensor, and monitored the resulting signal on an oscilloscope. Touching the two exposed pins on the sensor will cause a short, so do not touch the pins. The most significant change in capacitance was observed when both sides of the capacitive touch sensor were touched simultaneously. Additionally, the sensor exhibited the most pronounced change in capacitance when the polarity was switched from the P pin to the G pin, which caused a noticeable shift in the signal, clearly visible on the oscilloscope, as seen below, before and after touching the sensor.



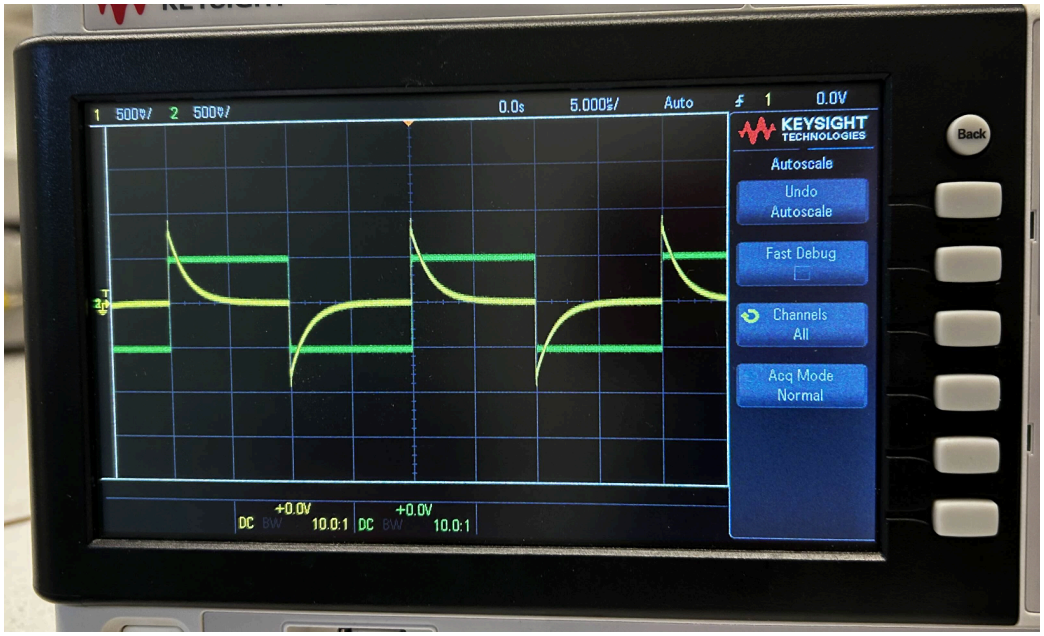


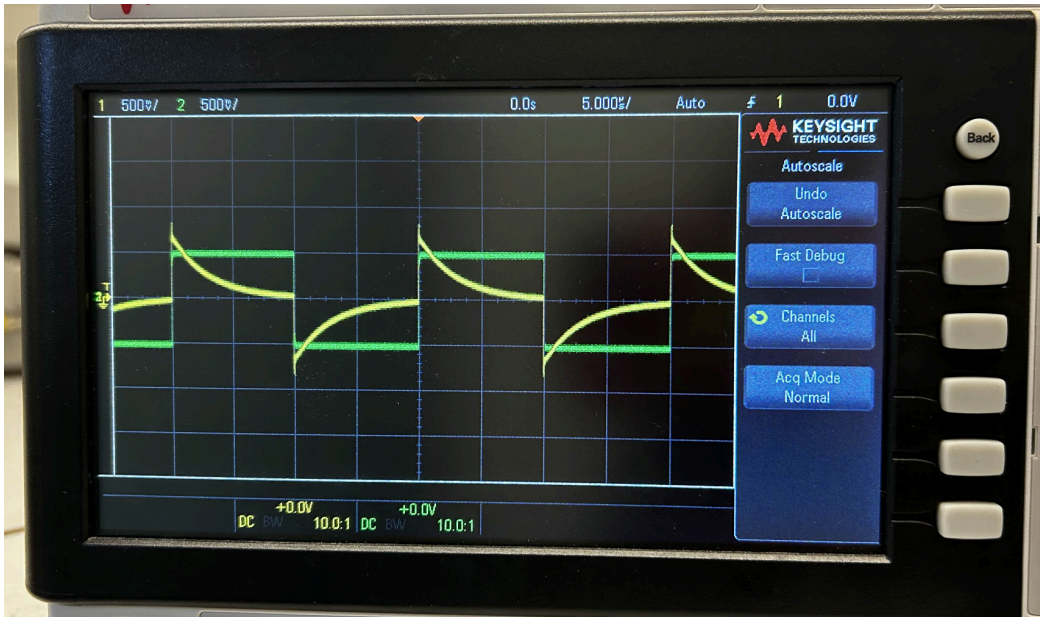
In the RC bridge circuit (resistors 10kΩ), I placed the capacitive touch sensor in parallel with one of the capacitors, which resulted in a noticeable change in capacitance when the sensor was touched. The change in capacitance was clearly visible on the oscilloscope. In the low-pass configuration, the effect of touching the sensor was noticeable with the 2kHz input signal, as the circuit's response shifted in response to the capacitance change, as seen below.





However, in the high-pass configuration, the 2kHz input frequency was too low to generate a significant response. To achieve a more pronounced effect, I increased the input signal frequency to 50kHz, which produced noticeable data on the oscilloscope, allowing me to clearly observe the changes in capacitance as I interacted with the touch sensor, as seen below.





Lastly, for the in-amp, it was built exactly following the circuit diagram provided by the datasheet shown earlier.

Using the LM555 oscillator circuit, I initially set the two load resistors (R_A and R_B) to 3.9kΩ and 3kΩ, respectively, with an input voltage of 5V, and both R_L resistors were set to 10kΩ. These values obtained from the datasheet. However, this configuration resulted in an unexpectedly high frequency of around 200kHz, which was outside the desired 1–5kHz range. To adjust the frequency, I swapped both Ra and Rb to 1MΩ, which successfully lowered the frequency to approximately 5kHz when the touch sensor was not engaged. When the sensor was touched, the frequency dropped further to around 2kHz, providing a more suitable response for the experiment.

I connected the output of the LM555 oscillator to the MCU to read the frequency change using timers. Initially, I made the mistake of placing a `printf` function within the interrupt handler, which introduced delays and caused the timers to become unreadable. After removing the `printf` from the interrupt, the frequency readings became stable, with values of around 200 microseconds when the sensor was untouched and around 300 microseconds when it was touched. By setting the threshold to 250 microseconds, the system worked as expected, providing reliable readings of the sensor's state.

6. conclusion

In conclusion, the best results were achieved when the polarity of the touch sensor was oriented from the P pin to the G pin, as this produced the most significant change in capacitance. Additionally, the high-pass configuration, combined with a higher frequency signal input of 50kHz, provided the clearest and most noticeable response on the oscilloscope. This lab was both enjoyable and slightly more challenging than the previous ones. It took me about 4-5 days in the lab to complete, spending 5-6 hours each day working through the steps. Building analog circuits like the LM555 oscillator and integrating them with the digital MCU to interact with the sensor was a particularly cool experience, as it involved combining both analog and digital techniques to achieve the desired results.