Lab 2

ECE 167

Jacob Sickafoose

Part 1 checked off
by **Arturo Gamboa-Gonzalez**
at **3:53PM April 28, 2022**

Part 2 checked off
by **Arturo Gamboa-Gonzalez**
at **5:38PM April 28, 2022**

Part 3 and Some Oscilloscope Circuits checked off
by **Arturo Gamboa-Gonzalez**
at **6:23PM April 29, 2022**

The rest was checked off
by **Cris Vasquez**
at **9:07PM April 29, 2022**

Commit ID:
c65fb3d1e9d86365aea51b013187338bbd8ce829

April 29, 2022

1 Overview

This lab was split into three separate parts. Each had totally different lessons to be learned, and were implemented using different techniques. The first part required both reading a rotary encoder, and outputting three PWM signals to control the color values of an RGB LED.

The second part of the lab required implementation of the timer which changes period based on state, and an input capture pin used to measure how long a pulse remained high. This was then converted into distance values to read from the ping sensor.

The final part of this lab was the most extensive. It began by requiring the implementation of different circuits to explore different methods of detecting a change in capacitance. Each of these circuits was carefully tested and documented. The best and final circuit was then used to implement the code to detect when the capacative touch sensor was pressed. This was done, again using the input capture pin.

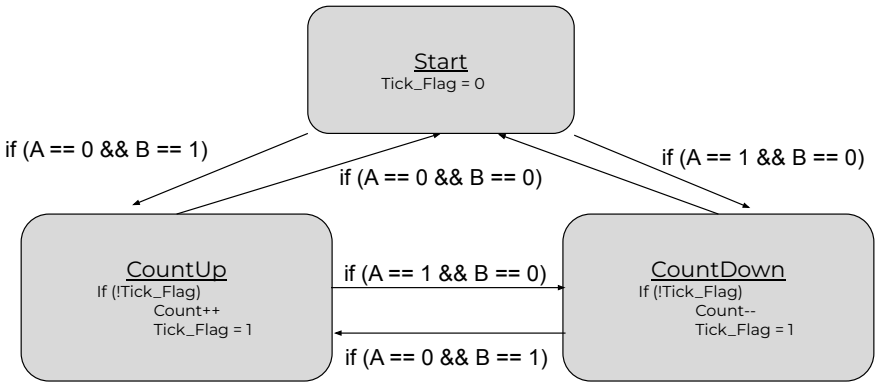
Parts Required:

- Rotary Encoder
- Ping Sensor
- Speaker
- Amplifier Board
- Capactive Touch Sensor
- CA3046 Op-Amp IC
- LM555 Timer IC
- Various Capacitor Values
- Various Resistor Values
- chipKIT32
- Jumper wires

2 Rotary Encoder

This section was broken into two subsections. The first was reading from the rotary encoder and counting the angle properly. The second part was converting that signal to RGB values and outputting that in the form of PWM to control the onboard RGB LED on the rotary encoder board.

For the first section, the lab manual gave the difference between if the rotary encoder was counting up and counting down. I used this difference to implement my state machine which was called inside my interrupt. The interrupt was called whenever the A signal or B signal changed. These two values were passed into my state machine which implemented the following simple logic:



(a) State diagram of the QEI logic

The count was reset back to 0 when it hit 360. The Tick_Flag variable was created to prevent double ticks from happening. Unfortunately it also limits how fast the dial can be spun but I think it is a worthy trade off because of how smooth the value changes and how consistently 1 click = 1 count value. With this implemented, it was time to convert the 0-360 value into an RGB value.

To convert the $0 - 360°$ value into a 0-1000 PWM value for each Red Green and Blue channel I found an equation online to convert HSV into RGB. H, or Hue was the only value changing. The following equation was implemented:

$$C = V \times S$$

$$X = C \times (1 - |(H / 60°) \bmod 2 - 1|)$$

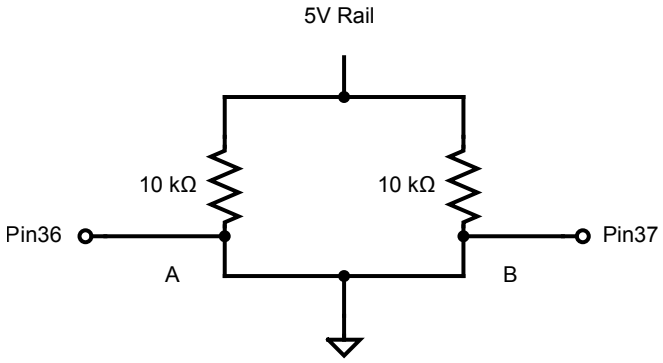
$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0° \leq H < 60° \\ (X, C, 0) & , 60° \leq H < 120° \\ (0, C, X) & , 120° \leq H < 180° \\ (0, X, C) & , 180° \leq H < 240° \\ (X, 0, C) & , 240° \leq H < 300° \\ (C, 0, X) & , 300° \leq H < 360° \end{cases}$$

$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

(b) HSV to RGB conversion

This was implemented with the exception that S and V were locked at 1 and the final scaling by 255 was changed to scale out of 1000. I also implemented it without using float values by keeping every number above 1. The rotary encoder was hooked up in the following configuration as given by the datasheet:



(c) QEI circuit

3 Ping Sensor

As mentioned in the overview, the ping sensor part of the lab introduced two main challenges. The first was getting the timer to properly trigger with the two different periods. The second was using input capture to measure the echo pulse and using that to determine the distance reading from the ping sensor. After this was done, it was trivial to convert that distance to a smoothed out tone because most of that work was completed in the previous lab.

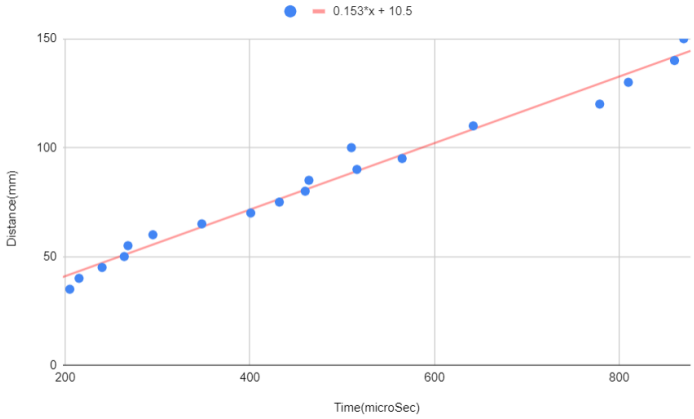
The timer needed to have two different periods. The timer needed to have a high impulse lasting for $10\mu S$ and a low signal lasting for $60ms$. Using the scaling of the timer module and knowing the clock speed of the module, the periods were calculated to be $10\mu S = 0x0007$ and $60mS = 0x927C$. These values were loaded into the period register every time the timer interrupt was triggered, based on what the previous period was. This satisfied the specification for the trigger to be accomplished. The next step was to set up the input capture to give a return value.

The input capture module triggers an interrupt whenever the input pin changes value. It is configurable to trigger the interrupt on the high edge, low edge or both. In this case, the interrupt was triggered on both edges. When the signal went high, a timer was started and it was stopped when the signal went low. The difference between these two timer numbers gave the time of flight in microseconds.

Using this information, I set the microcontroller to display the raw time of flight value onto the OLED so I could compare this value to the distance. The following chart and graph was created with this data.

Time(microSec)	Distance(mm)
205	35
215	40
240	45
264	50
268	55
295	60
348	65
401	70
432	75
460	80
464	85
516	90
565	95
510	100
642	110
779	120
810	130
860	140
870	150

(d) Data Table



(e) Resulting Graph

The least squares regression line was found using the in-built Google Sheets function and applied to the data to implement PING_GetDistance().

4 Capacitive Touch Sensor

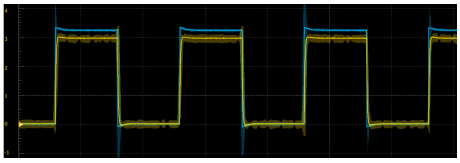
This part of the lab took by far the longest. We were expected to hook up the capacitive touch sensor in a simple RC circuit with multiple resistor values and observing the result of touching the sensor. We then took data setting up a capacitive bridge in two different capacitive bridge configurations and touching the sensor and using that data to trigger a microcontroller TRUE/FALSE value. Finally, we needed to make the relaxation oscillator setup, observe that on the oscilloscope and implement the change on the microcontroller.

4.1 RC Time Constant Based Measurement

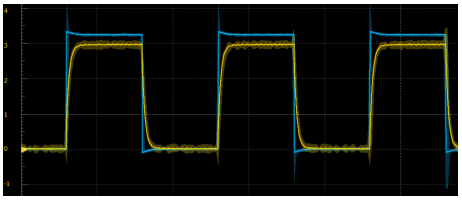
For this section, I measured the rise time and recorded the oscilloscope change between pressing and not pressing the capacitive touch sensor with the different resistance values. The input to the RC circuit was a constant 50% duty cycle PWM signal created by the microcontroller.

$100K\Omega$

With a $100K\Omega$ resistor, the rise time goes from $10\mu S \rightarrow 37\mu S$ when the capacitor was touched. I got the following oscilloscope graph with the PWM input signal represented in blue and the output signal represented in yellow:



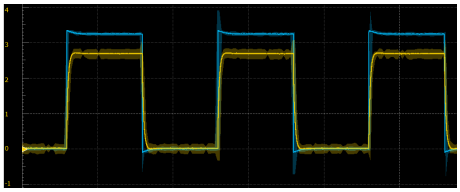
(f) Touch Sensor Released



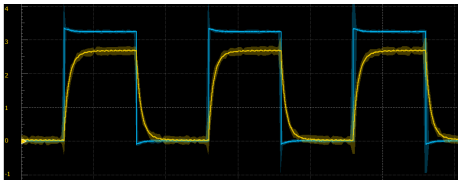
(g) Touch Sensor Pressed

$220K\Omega$

With a $220K\Omega$ resistor, the rise time goes from $20\mu S \rightarrow 78\mu S$. I got the following oscilloscope graph with the PWM input signal represented in blue and the output signal represented in yellow:



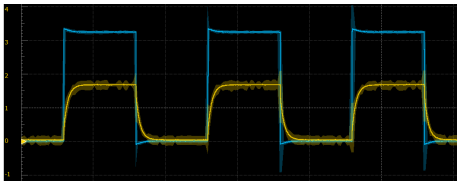
(h) Touch Sensor Released



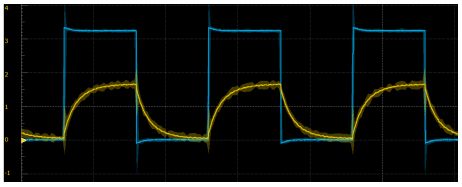
(i) Touch Sensor Pressed

$1M\Omega$

With a $1M\Omega$ resistor, the rise time goes from $58\mu S \rightarrow 210\mu S$. I got the following oscilloscope graph with the PWM input signal represented in blue and the output signal represented in yellow:



(j) Touch Sensor Released



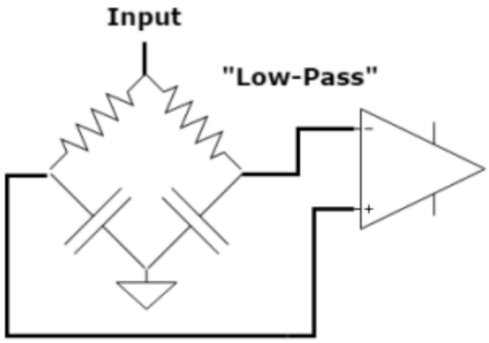
(k) Touch Sensor Pressed

I found that the higher the resistor value, the larger the change when the capacitor was pressed. This made sense because the capacitor value was so low, that the resistor value needed to be large so the frequency was not too high.

4.2 Capacitive Bridge and Differential Amplifier

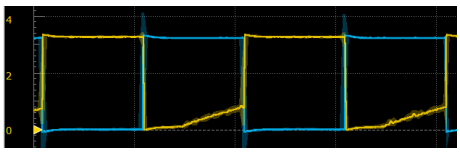
Low-Pass Bridge

For this part, the following circuit was made:

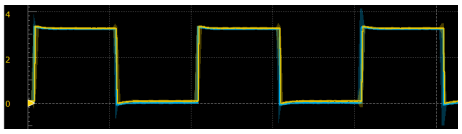


(l) Low-pass Bridge Schematic

Running the same 50% Duty cycle PWM signal through it resulted in the following output with the yellow signal representing the output and the blue signal representing the input PWM wave:



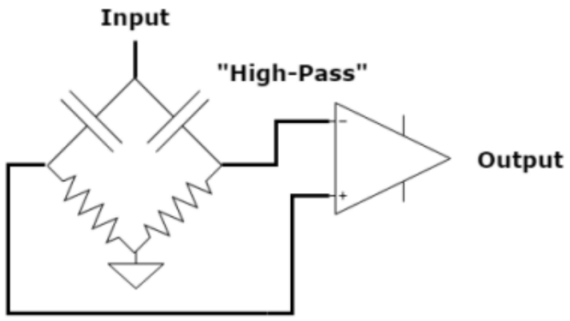
(m) Touch Sensor Released



(n) Touch Sensor Pressed

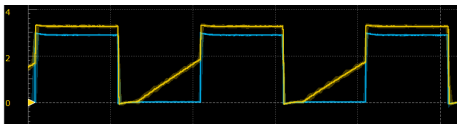
High-Pass Bridge

For this part, the following circuit was made:

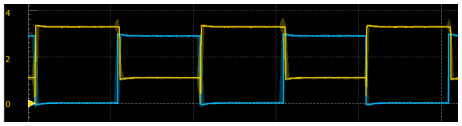


(o) Low-pass Bridge Schematic

Running the same 50% Duty cycle PWM signal through it resulted in the following output with the yellow signal representing the output and the blue signal representing the input PWM wave:



(p) Touch Sensor Released

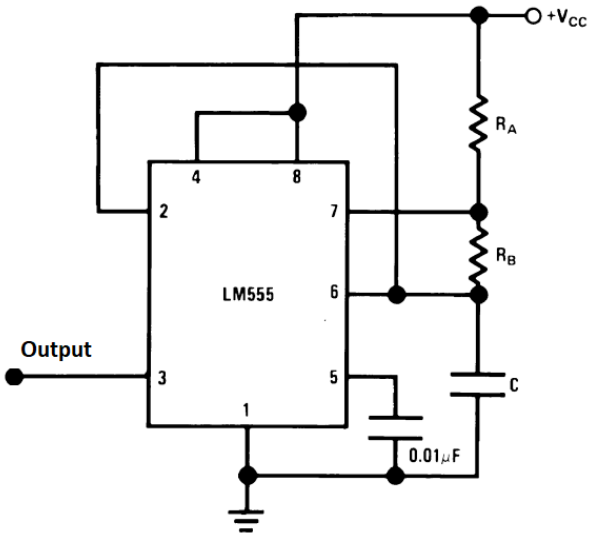


(q) Touch Sensor Pressed

Comparing the Low-Pass and High-pass configurations of the bridge set-up, it seemed as though the only difference between them was a complete swap of when the sensor was pressed and when it was not pressed. In the low-pass configuration, while the touch sensor was pressed, the output signal almost completely matched the input signal. When released however, it was completely out of phase. The opposite was true with the high-pass configuration. In order to for this circuit to be detectable by the microcontroller, the differential amplifier was required to output a signal depending on this change in the relation between the input and output.

4.3 Relaxation Oscillator

In this part, we were required to utilize the LM555 timer in order to implement the following relaxation oscillator circuit:



(r) Relaxation Oscillator Schematic

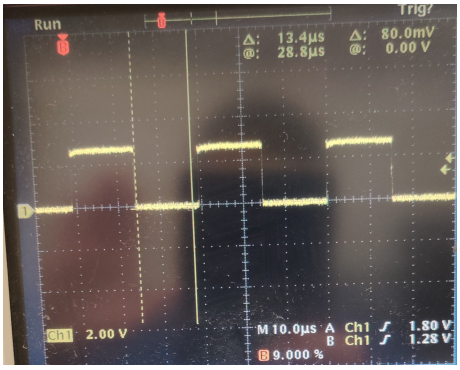
In this schematic, the C value was replaced with our touch sensitive capacitor in parallel with a $22pF$ capacitor to add to the capacitance. In order to achieve a roughly 50% duty cycle, the following formulas taken from the datasheet were implemented:

$$t_{on} = 0.69 * C * (R_A + R_B)$$
$$t_{off} = 0.69 * C * R_B$$

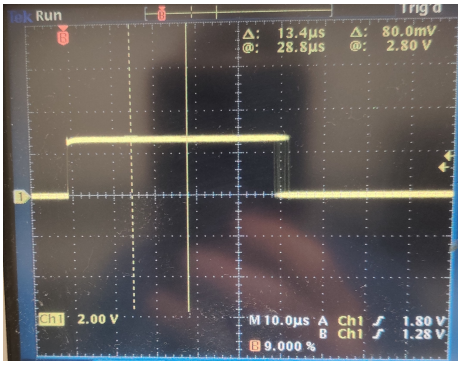
Where

t_{on}: Length of high output pulse in seconds
t_{off}: Length of low output pulse in seconds

All that mattered between the R_A and R_B values was their ratio so I chose $R_A = 4.7K\Omega$ and $R_B = 470K\Omega$. When hooking it up to the oscilloscope with and observing the output, I got the following:



(s) Touch Sensor Released



(t) Touch Sensor Pressed

It can be observed that once the capacitor was pressed, the length of the high signal was lengthened according to the formula when the C value changed. The input capture module was used on the microcontroller to record the length of the high pulse just like with the ping sensor. This time the elapsed time was recorded and depending on how long the wavelength became, the TRUE/FALSE was triggered representing whether it was pressed or not.

In order to find the threshold value, the elapsedTime raw value was output to the OLED screen and observed between pressing and releasing the touch sensor. The difference in the value between these two states was vast and made it easy to detect with certainty if the sensor was being pressed.

5 Conclusion

This lab felt like the culmination of three separate labs. The first two were not hard but making each of the circuits required for the final part was an extreme pain. I also learned after failure that the circuit diagram in the lab document for the relaxation oscillator was wrong and did not work. The one that worked was in the data sheet which I initially just thought must have been outdated. I am happy with how well the touch sensor can be detected with the final relaxation circuit. Overall, this lab had a lot of different components to it to learn and at least it feels like I learned a lot from it.
Est. Hours: 17