ECE 167 - Sensing & Sensor Technologies

Lab 2: Encoder, Ping Sensor, and Capacitive Touch

By: Kyle Jeffrey

Introduction

This lab is the introduction to quadrature encoders, time based sensors, and capacitive sensors. The encoder used in this lab is a rotary switch with RGB LED. The basic task for the students is to implement the quadrature encoder interface (QEI) to determine position and then drive the color as appropriate. The ping sensor is used to determine distance to various objects, and needs to be calibrated to generate accurate distances. Variable capacitance is used in myriad sensors, but in this lab we are going to detect the change in capacitance from touching a pad (so called capacitive touch sensors).

Part 1: QEI

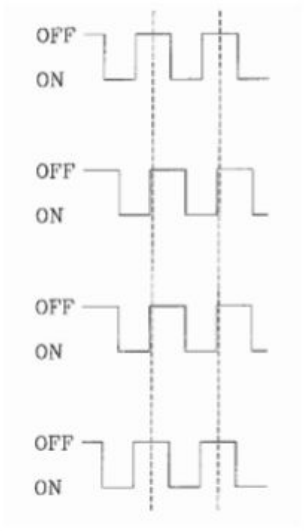
What is a Quadrature Encoder?

Encoders (often called Incremental Encoders, or Quadrature Encoder [QE]) are used to measure angular displacement. Wikipedia (https://en.wikipedia.org/wiki/Incremental_encoder) has a decent reference on incremental encoders. Encoders are used in a wide variety of applications, from joint angles in robotics and keeping track of wheel rotation for navigation (odometry), to keeping track of axes on CNC machines andtracking tank turret angles.

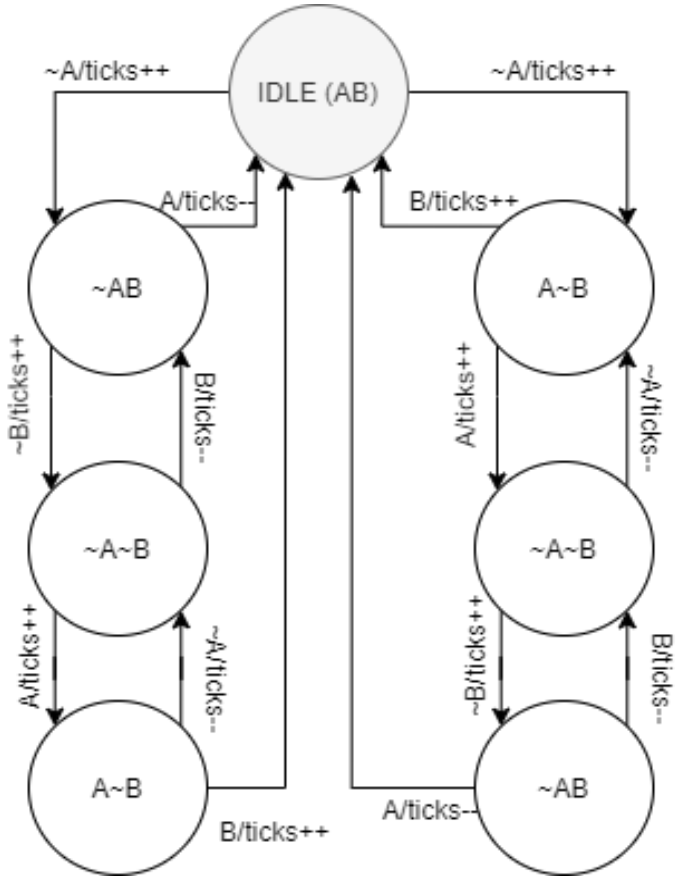
The encoder has terminals A B, and C. A and B will be supplied with 5 volts through a 10K resistor and C will be grounded.

1.1 Reading Quadrature and Creating Angle of Rotation

To read from the encoder, a state machine should be used. The encoder is divided into 24 notches and the encoder emits two pulses 90 degrees out of phase every notch.

<table>
<tr><td rowspan="2">順時針方向
C. W.</td><td>A （A-C 端子間）
A （Terminal A-C）</td><td rowspan="4"> 
</td></tr>
<tr><td>B （B-C 端子間）
B （Terminal B-C）</td></tr>
<tr><td rowspan="2">反時針方向
C. C. W.</td><td>A （A-C 端子間）
A （Terminal A-C）</td></tr>
<tr><td>B （B-C 端子間）
B （Terminal B-C）</td></tr>
</table>

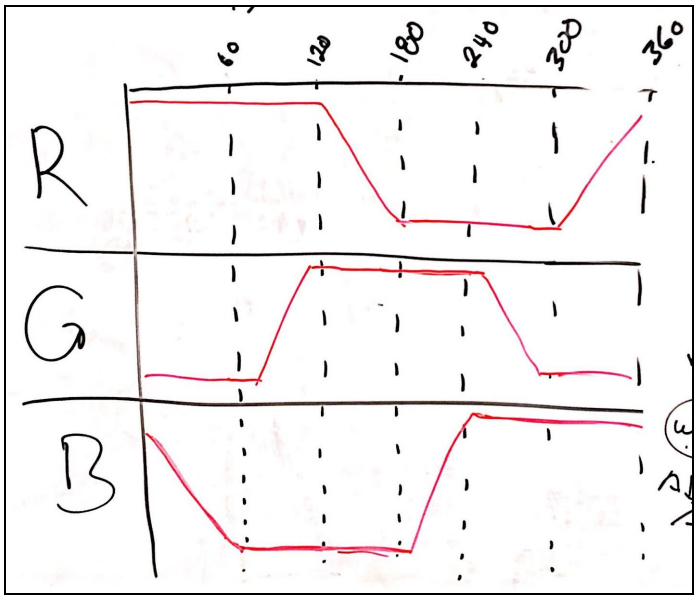
Using the Change Notify Interrupt, there will be 4 flags triggered. This increases the resolution of the sensor by 4, bringing it to 24*4=96 levels of rotation for every revolution. The state machine for that appears below.



This state machine prevents misreading multiple Clockwise or Counterclockwise rotations. It is implemented in code using a switch statement, outlined in the next section of the write up. A running module level variable called “ticks” tracks the rotation from start. CCW rotations decrease the count and CW rotations increment the count. Though, in a later part, the lab specifies to snub the range to in between 0-360 degrees. Ultimately, 1 tick equals 3.75 degrees of rotation, which will be set to 4 degrees to prevent any floating point math.

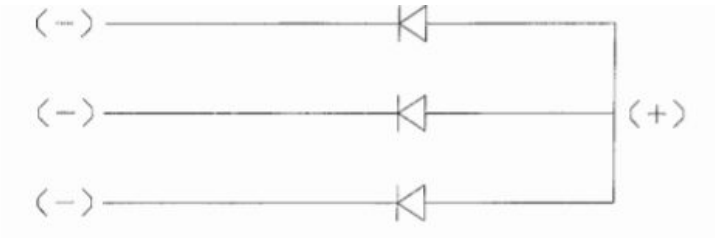
1.2 Mapping Rotation to Color

The PCB adapter that the QE was soldered to has another circuit on the opposite side of the ABC terminals. This circuit controls RGB LEDs within the rotating knob. The lab requires that rotating the knob creates a smooth transition in color from red to green to blue across a full revolution. The mapping for that might look like this graph for color intensity. The y-axis represents intensity and the x-axis represents degrees of rotation.



The circuit for the LED side of the PCB is shown here, taken from the datasheet.

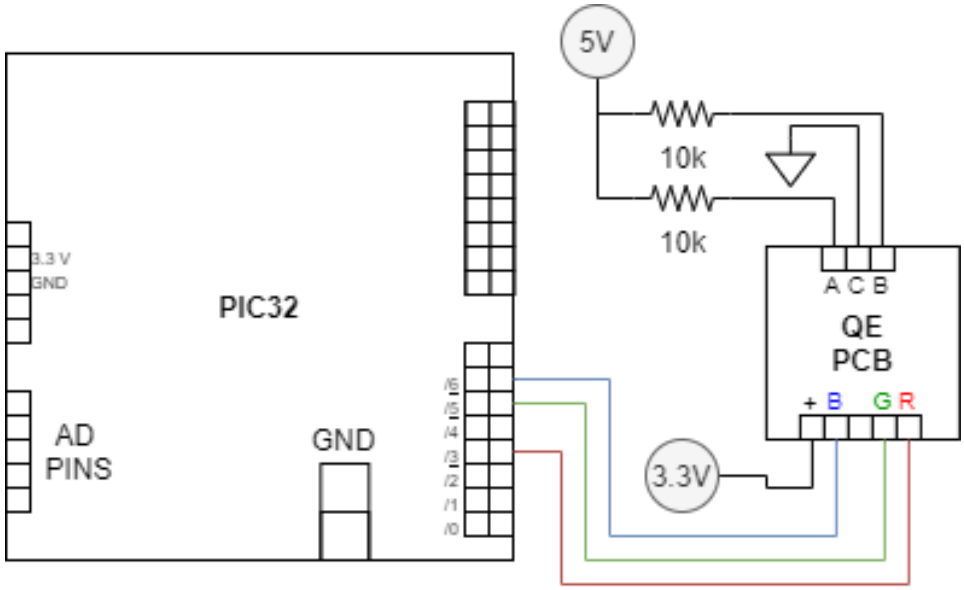
Circuit:



二、L.E.D. 特性 Characteristic:

1 反轉電壓 Reverse Voltage:5V

The circuit works off a 5V supply that is fed through the + terminal on the board, and the LEDs pins are set to ground. To control the intensity of the LEDs, a pulse width modulation(PWM) signal will be used. This simply uses a square wave with a varying duty cycle that rapidly turns the LEDs off or on for a period of time determined by the duty cycle. The PWM library provided by the class will be used to do this, and the software implementation will be shown in section 1.3. Here is a circuit diagram for the total Color Mapping circuit of the QEI.



1.3 Software Implementation

Change Notify Interrupt System:

This lab uses the Change Notify Interrupt system. This interrupt triggers when the configured input pins change values., which were pin 36 and pin 37. The state machine included in the previous section was implemented to keep track of rotation. Pin 36 and Pin 37 are digital pins and would simply give either a 1 or a 0 value, rather than an AD reading which would be 10 bits.

Color Mapping

The LEDs use the PWM library to change the duty cycle of a digital 3.3v signal to the RGB pins on the PCB adapter. Though, the LEDs are driven by a 3.3 DC voltage from the PIC32 board, so when the

PWM signal has a duty of 100 percent, the LED will be fully off, and when the duty cycle is 0, the LED will be full intensity. This code sets the PWM signals for each pin.

A mapping function for each color was run to grab what the duty cycle for each color's PWM signal should be. These functions use the graph provided in the previous section as a guide for color intensity.

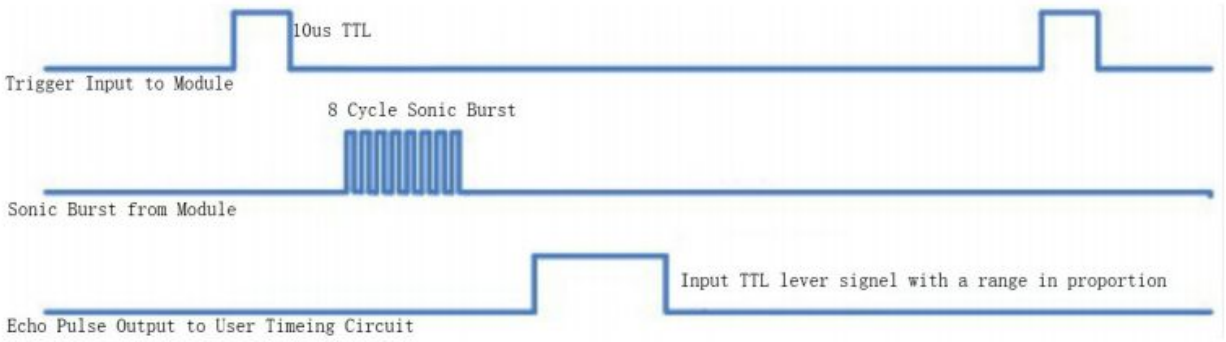
Part 2: Ping Sensor

What is a Ping Sensor?

From the Lab Manual "*The second sensor used in this lab is a time-of-flight ranging sensor called a "ping" sensor. Time of flight sensors work by injecting some form of energy into the environment and measuring the time of flight between the emitter and the reflection (echo) from the object. The kind of energy and the method of measuring the time of flight differentiate the various sensors, ranges, costs, and accuracies. These sensors are used widely in a variety of applications, from mm ranges (VCELS) to 100s of kilometers (RADAR). Typically, light and sound (at various frequencies) are used as the source/*"

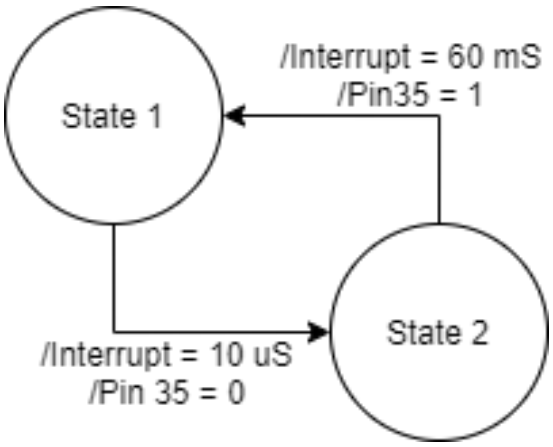
2.1 Ping the Ping Sensor and Capture output

The ping sensor is charged with 5 volts DC and grounded. The pins trigger and echo are an, input and output, respectively. The sensor takes a 10uS pulse every 60 mS through the trigger pin and creates the sonic burst and echo pulse shown here:



Timer Interrupt

This part of the lab uses a timer interrupt that is connected to one of the onboard timers. The interrupt triggers at a constant rate based on settings. So here we use a state machine that runs inside the timer interrupt. The interrupt timing is changed when switching states to give the behavior of a 10 uS pulse every 60 ms.



Sending this signal into the “trigger” pin of the ping sensor creates an echo pulse proportional to the distance the ping sensor is away from whatever object it is pointing at.

BUGS AND ERRORS:

An interesting error occurred using printf() to debug the behavior of the Timer Interrupt. Print created a large enough bottleneck in performance that used too loosely, would actually change output behavior of Pin 34. The print function hung up enough that the Timer Interrupt would skip a trigger.

2.2 Calibrate the Ping Sensor using Least Squares

The echo pulse generated is proportional to the distance. The method of finding the “Least Squares” will be used to create a line of best fit that will serve as the function mapping the period of the echo pulse to a distance. This ping sensor states that the time of the pulse in uS has a 1uS/143in ratio. For every 143 ms, the pulse senses one inch. This will serve as a sanity check for the line of best fit found by the least squares method. The lab manual states “*A quick primer on Least Squares can be found on Wikipedia (https://en.wikipedia.org/wiki/Least_squares) and is worth the read.*“

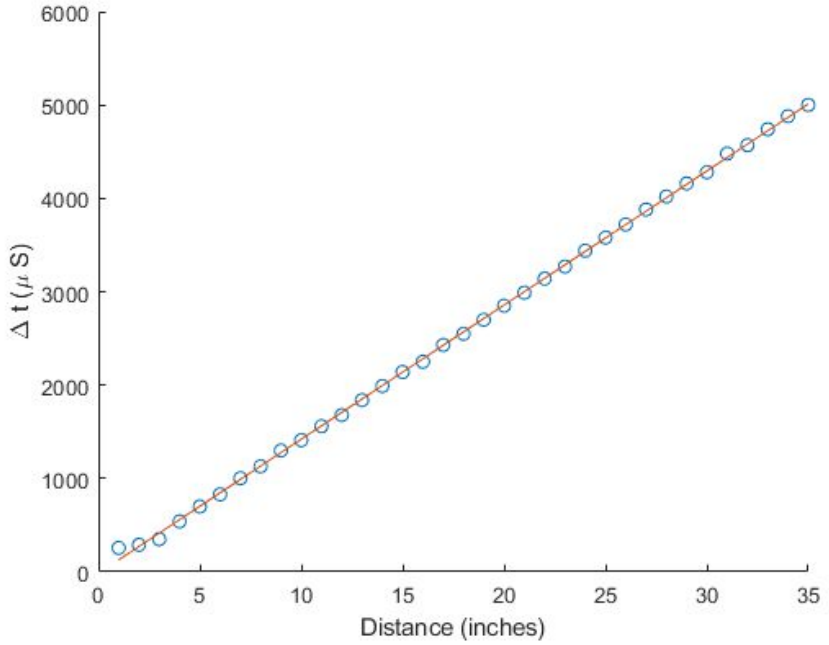
"The equation of the line is: yi = mxi +b where m is the slope and b in the y-intercept. If m and b were known, then for any given output from the sensor, x^ = (ym − b)/m. Thus the best estimate of the distance (x) is using the best estimate of those parameters m^ and ^b. Least squares allows us to estimate those parameters from our set of data pairs. This begins by rewriting the equation of a line in matrix form: "

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{Y}} = \underbrace{\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} m \\ b \end{bmatrix}}_{\mathbf{p}}$$

In this instance, the y-vector will be the period in uS of the echo pulse, and the x-vector will be the inches away from an object that the ping sensor is. When solving the equation for vector p:

$$\hat{p} = \begin{bmatrix} \hat{m} \\ \hat{b} \end{bmatrix} = [X^T X]^{-1} X^T Y$$

After taking data points up to 35 inches, this graph shows the scattered data points that do appear to be quite linear, and it shows the slope/y-intercept of the p-vector when put into matlab.



The vector p = (143,6165, -15.1456). Comparing these values to our sanity check of 143 uS per inch

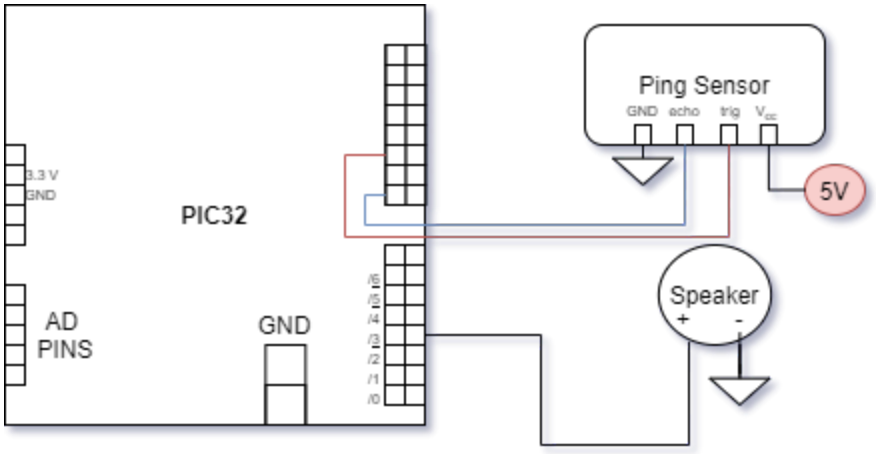
provided by the datasheet, it appears that the estimate is extremely precise using the least squares method. Past 35 inches, the ping sensor begins to get noisy, and maxes out around 40 inches.

Here's the Matlab code used to find the vector **P**:

The data was gathered in inches as the only measuring instrument available at the time was a yardstick. Finding the best fit in inches was also a better heuristic than metric as inches are more familiar. Here's the data used:

2.3 Tone Out Based on Distance

With the line of best fit used to turn the period of the echo pulse into a distance, create a circuit that generates a tone from the speaker. Use the ToneGeneration library provided by the lab to generate frequencies. The library uses pin 3 for a pwm signal. Software filtering could be done for this by way of a running average or step increase, but none was used in this implementation, as the signal wasn't too noisy.



2.4 Software Implementation

This module uses both a timer interrupt and a Change Notify Interrupt system. The Change Notify system grabbed the period of the ping sensor and the Timer Interrupt created the trigger signal ffor the Ping Sensor.

Change Notify Interrupt

The CN Interrupt used the TIMERS library to capture the milliseconds when the echo pulse was one and capture again when the echo pulse was zero, giving the period by subtracting the second reading from the first one and updating a module level variable holding the period.

Timer Interrupt

The TI applied the state machine outlined in section 2.1. The machine used a special reset register,PR4, to change the trigger time of the interrupt. The timer for the interrupt used a 64:1 prescaler for the board's 40 MHz clock bringing the timer to a .625 MHz clock with a period of 1.6uS. Changing the reset register to 7 created ~10uS trigger rate and setting it to 37500 created ~60ms trigger.

Distance Measurement

The least squares solution was used to map uS->inches. Solving for distance using the P vector values, -> distance = (time(uS) - b) / m.

Part 3: Capacitive Touch Sensor

What is a Cap Touch?

From the lab manual, "T*he last section of the lab will be interfacing to a capacitive touch sensor. Many sensors rely on variable ca-pacitance to sense the output, and learning how to measure capacitance is an important part of any sensors discussion. Capacitive touch sensors are used in myriad applications because of their many advantages: the switch is simple to build, has no moving parts to wear out, generates no spark (important for explosive envi-ronments), and can be placed behind a protective barrier such as glass or rubber such that the electronics are sealed from the environment.The basic idea for capacitive touch sensors is that when you touch an object (or place your finger in close proximity) you effectively create an additional capacitance. See the Wikipedia article for a basic explanation of the phenomenon: https://en.wikipedia.org/wiki/Capacitive_sensing. We are using projected capacitance by using a pad that has a gridded ground plane on the other side; a small capacitor is created when touched that induces a small capacitance change (in the pF range and unstable). This change in capacitance is what you will be measuring, and there are various ways to do this measurement."*

3.1 Straight RC Time Constant

The first way of measuring the change in capacitance is a simple RC low pass filter. Put some resistor with a known value in series with the cap touch sensor, drive it with a square wave, and determine either the rise time or cutoff frequency to find the capacitance.

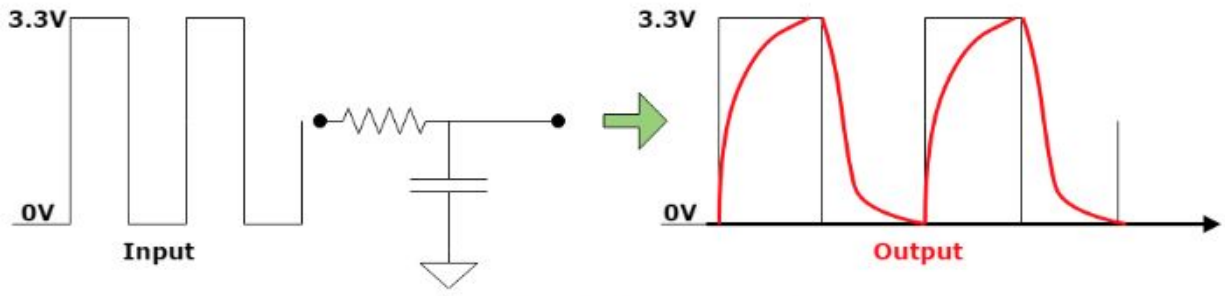
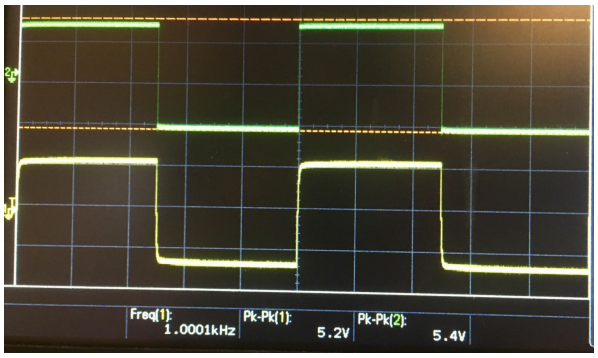


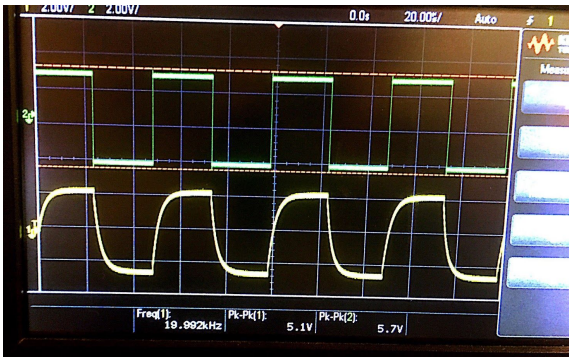
Figure 3: RC Circuit for Capacitance Sensing

In this case, the cutoff frequency was used to determine the capacitance. The cutoff frequency has a gain of -3dB i.e. .707 of the input voltage.

In experimentation, a 100k resistor was used which created the waveforms shown here:

<u>While Not Touching Cap Sensor</u>

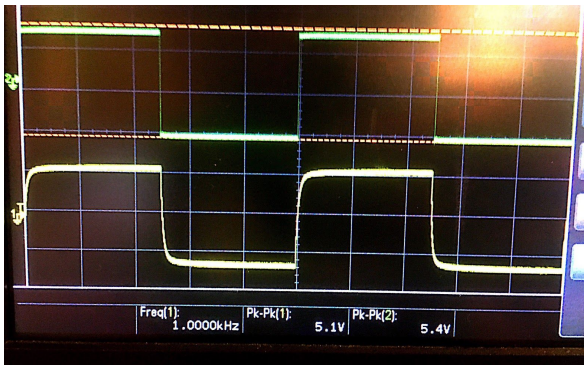


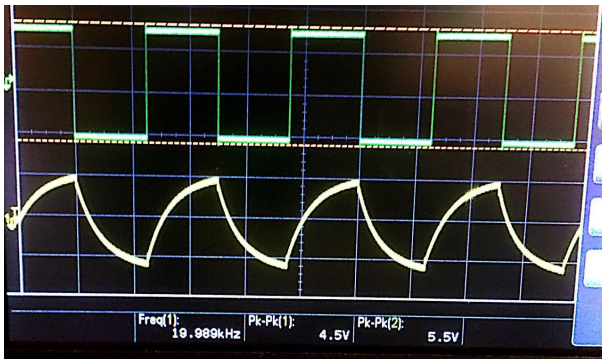


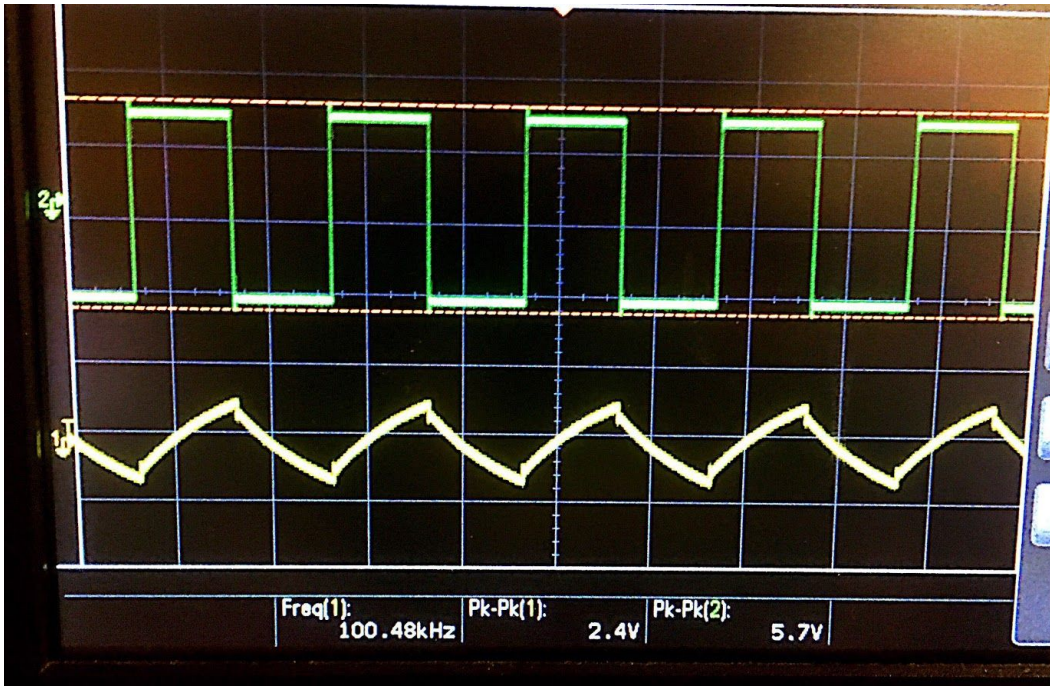
Cutoff Frequency: ~100Khz

Experimental Capacitance: 15.4pF

While Touching Cap Touch







Cutoff Frequency: ~26Khz

Experimental Capacitance: ~60pF

Touching the cap sensor creates ~40pF difference in capacitance, offsetting the cutoff frequency by ~70Khz.

3.2 Capacitive Bridge

From the lab manual, "*By setting two low-pass or high-pass filters with a known resistance/capacitance, and then adding the variable capacitor in a bridge configuration, send the outputs through a difference amp and you can get rid of many of the noise sources, creating a more stable measurement* "

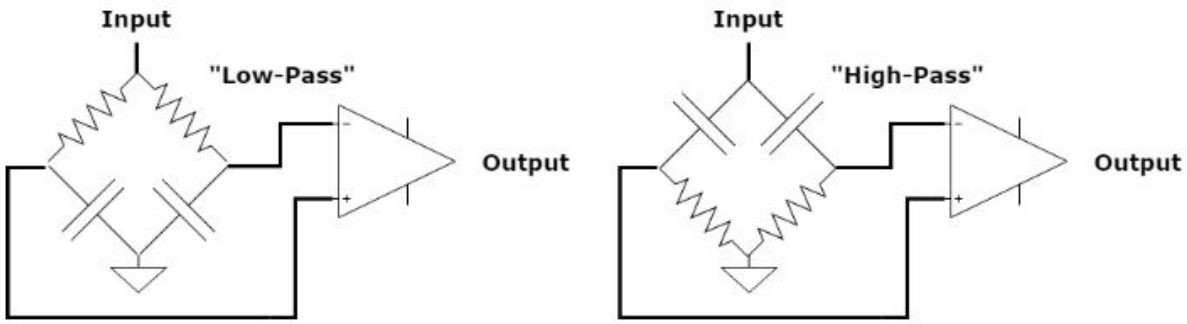
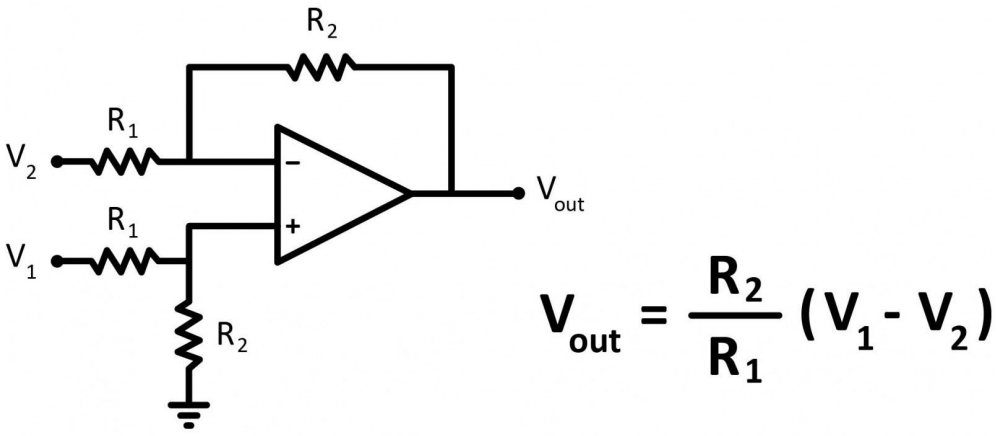


Figure 4: Capacitive Bridges for Capacitance Sensing

This configuration measures a difference in output voltage between the two outputs. This is a technique that can actually be used with the PIC32 Analogue in pins to detect touch. The differential op-amp will create a voltage that is between 0 and 3.3 V with the correct components. The capacitive touch will be put in parallel with one of the capacitors on either side of the bridge. Then, when the sensor is touched, capacitance will be added to the circuit and the difference in voltage can be determined by differential op amp. Make sure that the fixed capacitors have large values compared to the ambient capacitance of the sensor, so the voltage difference is not too large when untouched. The capacitors used in this part were **100pF.** Here is the circuit for a differential Op-Amp:

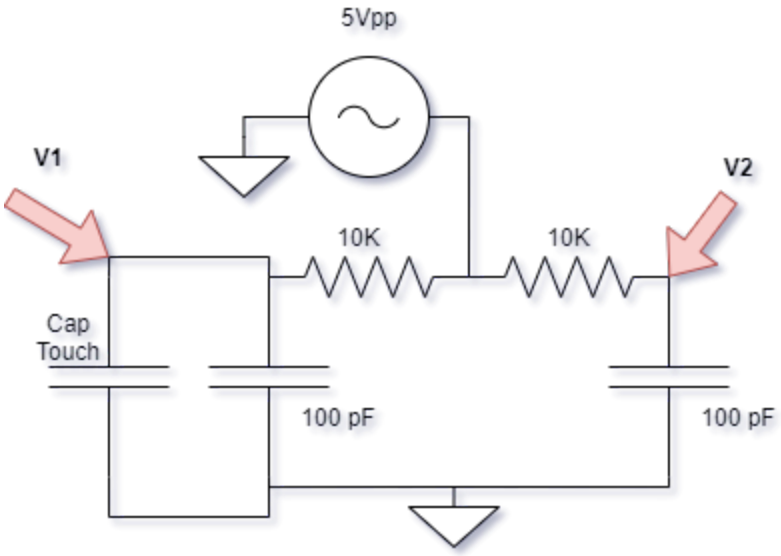


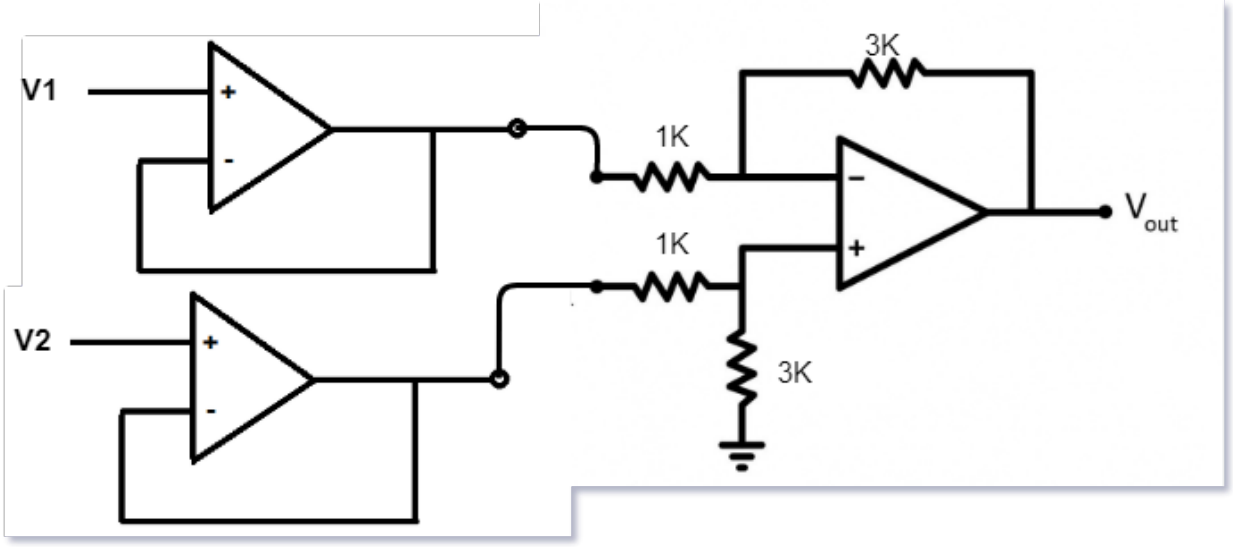
$$V_{out} = \frac{R_2}{R_1}(V_1 - V_2)$$

Differential Op-amp

Notes About Diff Op Amp Assembly:

The inputs V1 and V2 coming from either side of the bridge created large impedance when connected to the Op Amp without buffering. In the end design, both voltages were put through followers to eliminate the impedances.

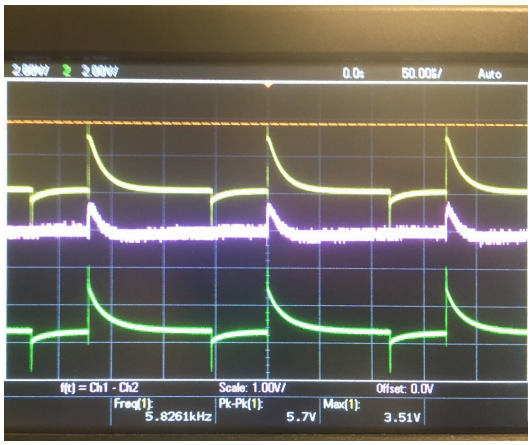




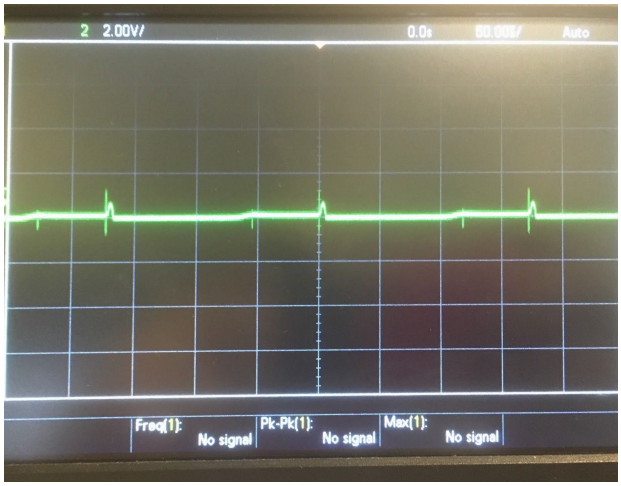
High Pass Bridge Differential Waveform:

V1 - V2 w/o Touch

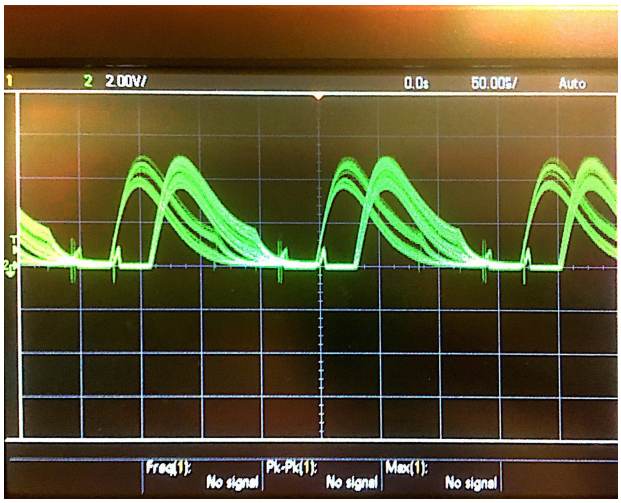
V1 - V2 w/ Touch



Differential Op-Amp w/o Touch



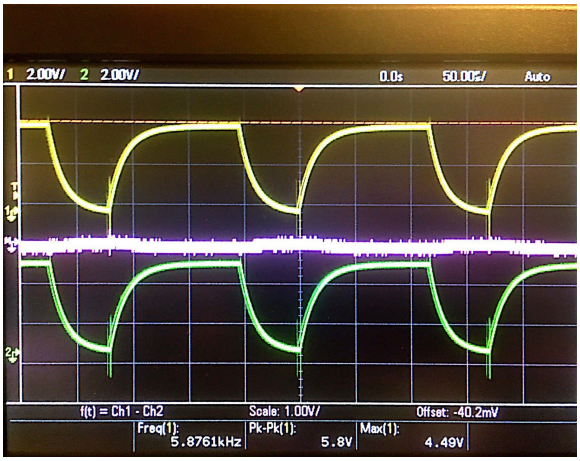
Differential Op-Amp w Touch



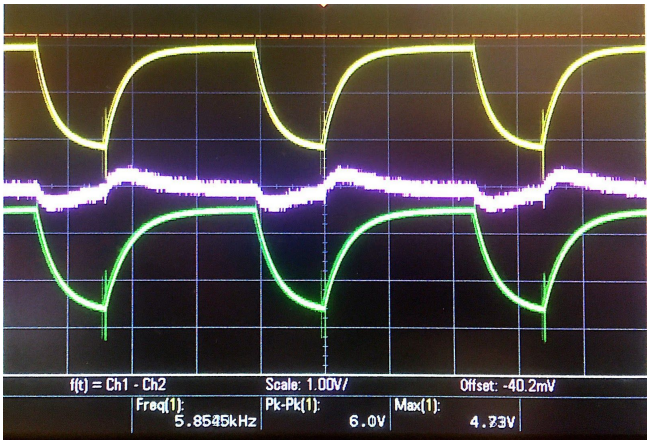
For whatever reason, the Differential output here was very noisy, so the low pass bridge was used in the final design.

Low Pass Bridge Differential Waveform:

V1 - V2 w/o Touch

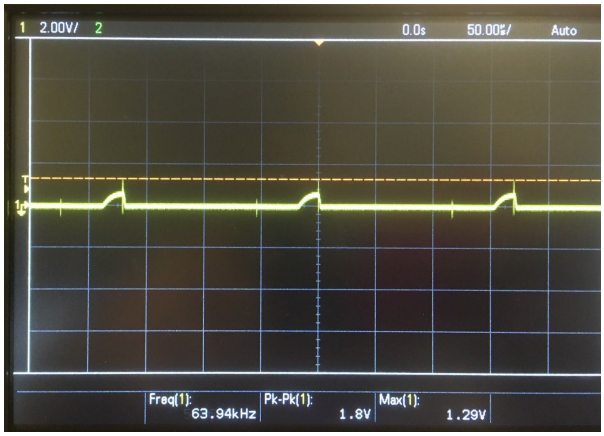


V1 - V2 w/ Touch

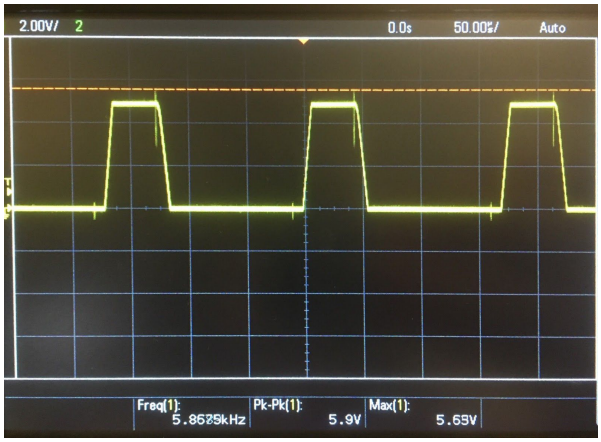


When the differential Op-Amp was used, the negative differences were snubbed, only allowing positive voltages.

Differential Op-Amp w/o Touch



Differential Op-Amp w Touch

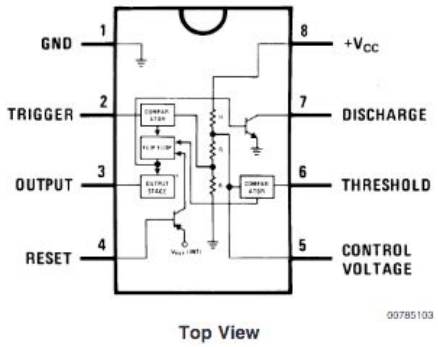


The output of this was put through a voltage divider to cap the output at 3.3V for the I/O pins. Alternatively, Vcc for the Op-Amp could've been put to 3.3v to snub the output.

3.3 Relaxation Oscillator (LM555 Chip)

From the lab manual, "*For the last part of the lab, you are going to implement the relaxation oscillator using an LM555 chip, with the timing capacitor being the parallel of a fixed 22pF capacitor and the touch sensor. The LM555 in astable mode exactly implements the relaxation oscillator, with the thresholds set at 2/3 and 1/3 Vcc. See Appendix A for more details on how to set up the astable mode.The timing capacitor between the threshold pin (6) and ground (1) is going to be a parallel 22pF capacitor and the touch sensor. Make sure you use resistors such that the base capacitance oscillates at a frequency that is reasonable (within the 1-5KHz range). You might need to experiment here. Use the oscilloscope to verify the frequency, duty cycle, and shift in frequency when you touch the sensor. Again, play with the circuit and see what the effects are. Switch the polarity of the touch sensor and see if it makes a difference.*"

The circuit for the LM555 is shown here with the pin setup, Vcc set to 5V:



Top View

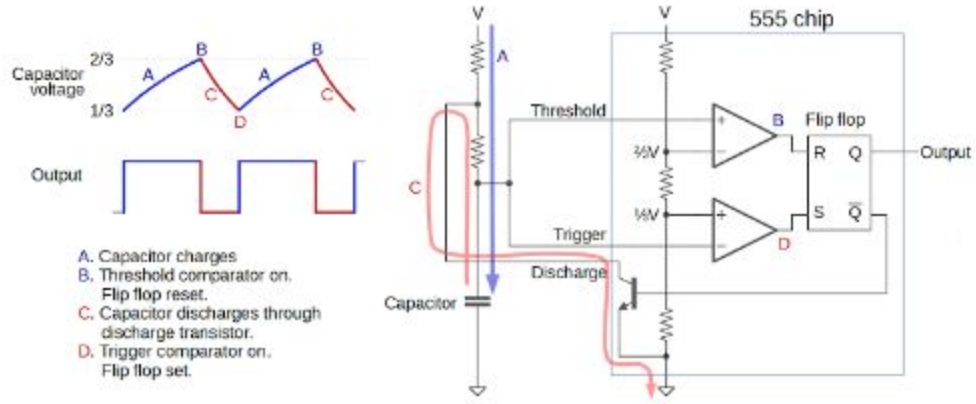
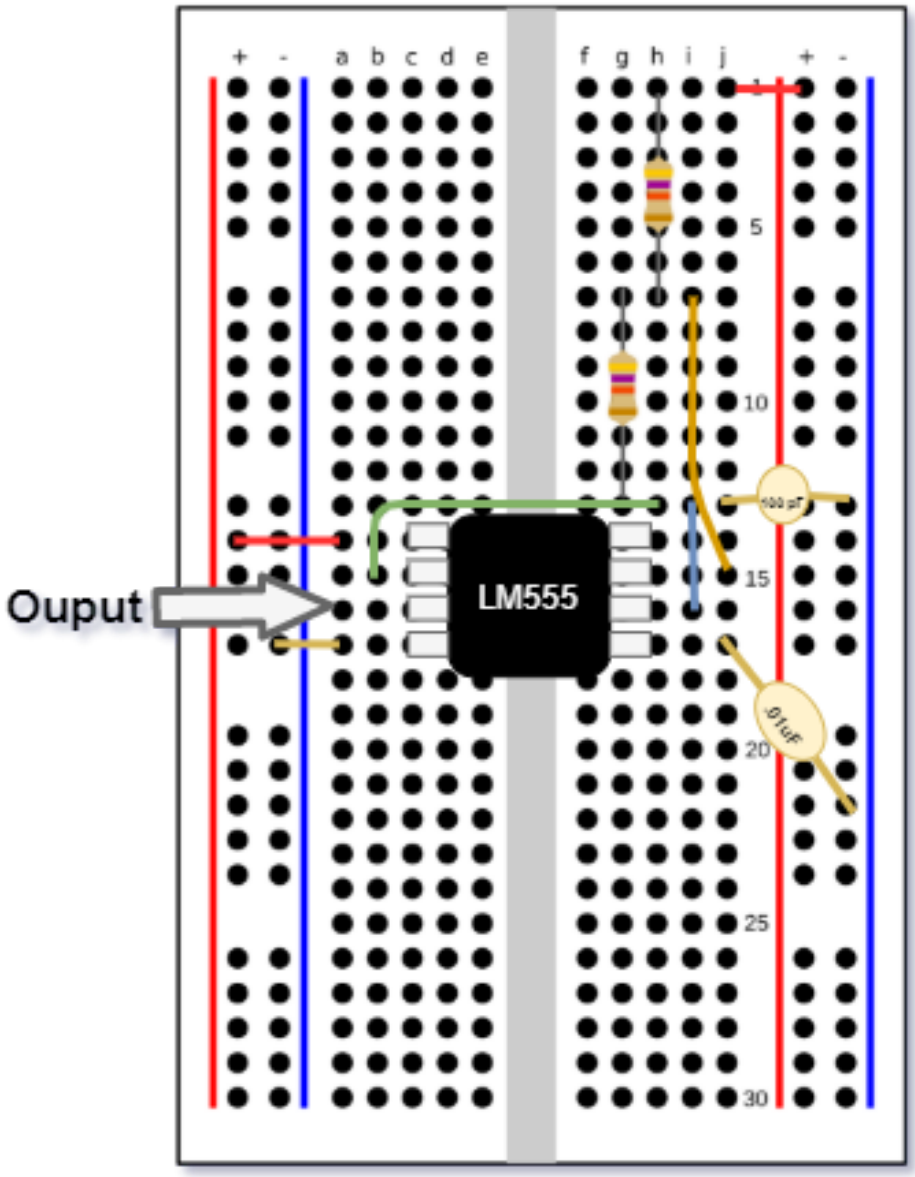


Figure 6: Conceptual Operation of 555 Timer

The resistors in series with the capacitor act as a low pass filter creating an attenuated, triangle wave that the SR Latch used as the duty cycle for the square wave.

Here's a layout of what the breadboard might look like. **I only include this as there was major struggles in getting the LM555 to work.** The resistors were in the MOhm range to achieve a frequency < 5 Khz. The front resistor was 5Mohm and back resistor 1.5Mohm in this experiment, though, ideally the front resistor would be much higher than the back resistor to get closer to a 50 percent duty cycle. A bypass cap was used at the 5V rail to ground to eliminate some AC noise coming from the PIC32.



LM555 Wave Form

This square wave is used to drive the differential Op-Amp circuit and used to create a relaxation oscillator.

3.4 Testing and Issues

LM555 Oscillator

A long time was spent attempting to implement the circuit shown below, included in the lab manual:

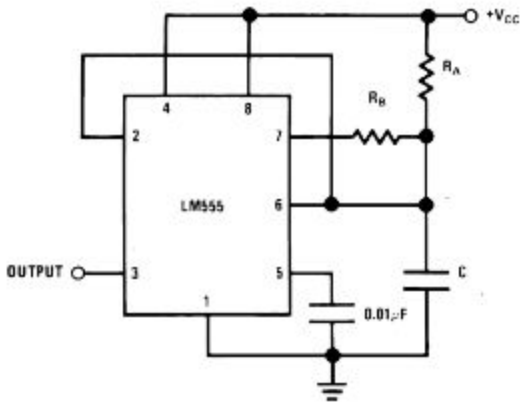


Figure 5: Example 555 Oscillator Circuit

For whatever reason, this circuit did not work as expected. The output was always a DC voltage, and notice that this circuit is different than the one shown in section 3.3 of this lab writeup, that was included within the Lab Manual appendix.

3.4 Software Implementation

Differential Op Amp Analog In

This module didn't use any interrupts to measure the analog voltage. The AD library was used to read the voltage of the differential OpAmp and if the value was above a certain threshold, a touch value was set to true. An off threshold was also used to prevent flickering values.

The values generated from the differential circuit tended to be noisy, so a running average software filtering was used to smooth out the values. The code for that uses an array of any size and the average is calculated:

```
int meanOfArray(int *arr)
{
    int mean = 0;
    int i;
    for (i = 0; i < AVG_SIZE; i++) {
        mean += arr[i];
    }
    mean /= AVG_SIZE;
    return mean;
}
```

Cap Touch Using Relaxation Oscillator

The lab requires writing the CAPTOUCH library using the provided CAPTOUCH header file. This time, an Input Capture interrupt is used, along with a running peripheral timer. The Input Capture interrupt is similar to Change Notify except that it only triggers on rising edges instead of any change in value. This is then used to capture the period of the signal from the timer buffer register, IC4BUF. in this case.

The Cap Sensor when touched increases the period of the signal, so an experimentally determined threshold is used to trigger a touch.

Software filtering is used to average out the signal and reduce noise. A running average was used, as shown in the previous section, with eight values which appeared to help a lot.