Team Members: Cole Arduser, Luke Farmer, Sam Nicklaus, Sam Loecke
ECE:4880 Senior Design
Lab 2 Report

Senior Design Lab 2

Introduction

The goal of lab 2 was to implement an “electric eye” safety sensor system. The system consists of a transmitter and receiver to detect obstacles in between our two circuits. Here is an example schematic from the lab 2 requirements document:

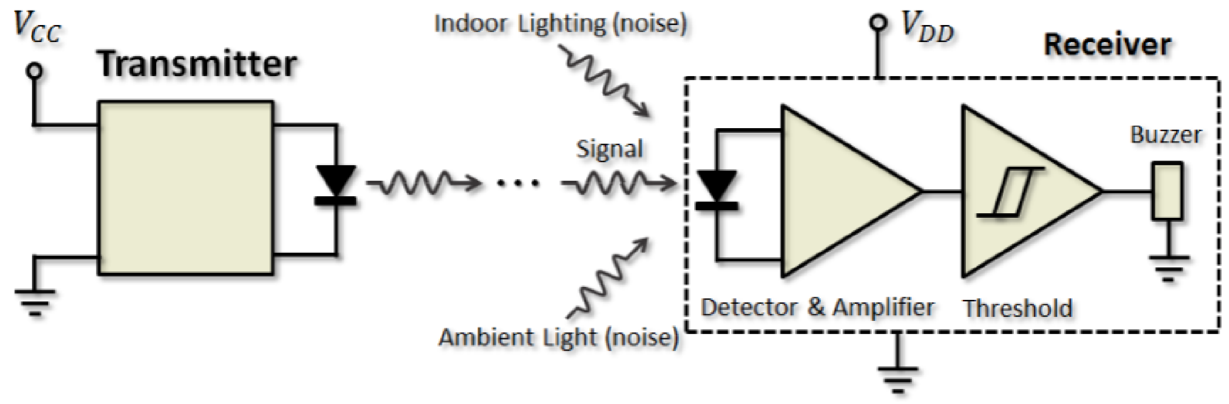


Figure 1: Overview of the system from the lab requirements

Our task was to design and implement the receiver side of the system as the transmitter was provided. We also need to send a text alert when the light beam is interrupted. In order to create this system, we used both hardware and software components. Our system is mostly a hardware system, but we do use a Raspberry Pi with a software program to send a text when the beam detects an obstacle. The main tasks that we needed to complete were receiving, filtering, amplifying, and detecting the transmitted signal.

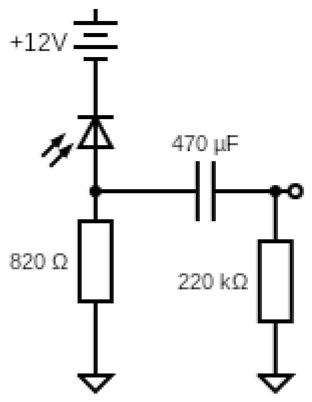
Design Documentation

Our design that we used for our solution to the lab prompt, uses both hardware and software aspects, but is a mostly hardware solution. We chose to use a hardware filtering and conversion technique to detect the transmit signal. The hardware receives, filters, amplifies, detects and converts the transmit signal. The output of our hardware is then fed into a Raspberry Pi GPIO pin to interface with our software component. In this section, we will give an in-depth specification of both our hardware and software design.

Hardware Overview:

Our hardware has six main sections: receive, buffer, filter, amplifier, detector, conversion for the raspberry pi. The receive section includes an IR photodetector to receive the signal from the transmitter and a resistor to convert the current produced by the photodetector into a voltage that we can use to sense the transmit signal. The next section is the buffer which uses an op-amp configured to have a voltage gain of 1 with a high input resistance and low output resistance to buffer the rest of the circuitry from the front end receive section. Then the signal is fed to the filter. The filter is specifically designed for the ~550 Hz signal that the transmitter uses. This filter is a 6th order butterworth band pass filter that is used to filter out the other frequencies on our input voltage. After the filter, the signal is passed through a non-inverting op-amp configuration with a gain of around 300 V/V to amplify our signal. After the amplifier, we have a peak detector to take the input sine wave and create a DC like voltage to determine the state of our circuit. This DC like signal is then fed into the gate of a mosfet which is used to isolate the rest of the circuit from the raspberry pi GPIO pin. The mosfet then uses the power rail of the raspberry pi to set a GPIO pin high and low based on the signal received from the transmitter.

Receive Block:



The receive block uses an IR photodetector (OP505A) diode to create a current with the received signal. We then use a resistor to ground to convert the current produced by the photodetector into a voltage that we can amplify, filter, and use to detect the presence of the transmitter signal. This resistor value was chosen to protect the photodetector from high current. The diode turn on voltage is .4 V, this means the resistor will have Vcc - .4 V = (12 -

Figure 2: Receive Block Circuit Diagram .4) = 11.6 V. The diode max current rating from the datasheet is 15 mA. To have a maximum current of 15 mA, Resistor in Ohm = (11.6 V / 15 mA) = 773.33 Ohms. We used an 820 Ohm resistor in the circuit.

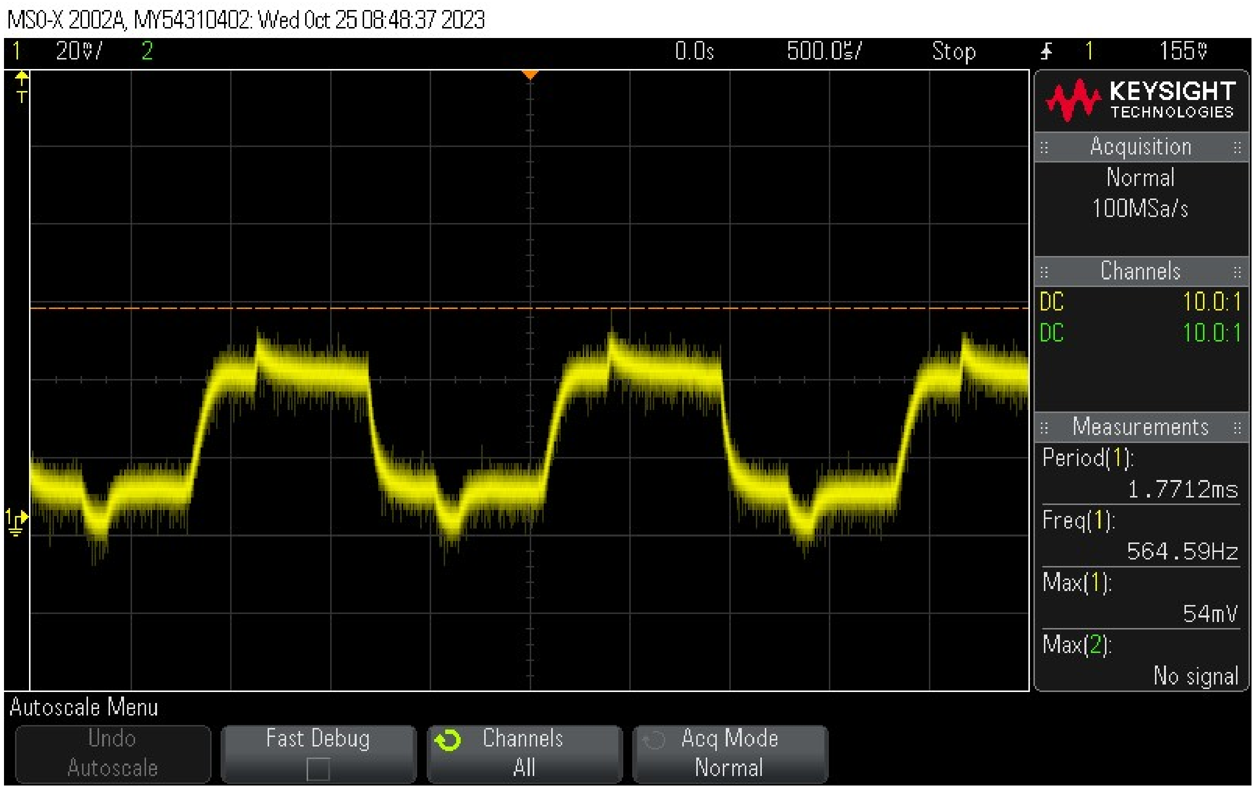
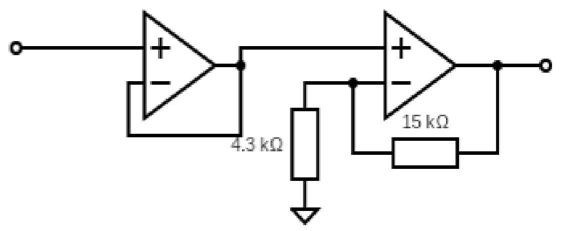


Figure 3: Signal observed on the output of the receive block

As you can see from Figure 3, the signal is very small with a max of 54mV and not very clean, but we do see our transmitter frequency of ~564 Hz.

Buffer:



After the receive block, we wanted a way to isolate and protect our receive front-end from the circuitry that comes next. To buffer the circuit and eliminate the loading effects from the next blocks, we used an op-amp in a buffer

Figure 4: Buffer Circuit Diagram configuration to get a very high input resistance and a very low output resistance. To do this, we used a TL082CP Op-amp in the buffer configuration shown. Then after the buffer, since the TL082CP package has two op-amps in the one package, we used the second op-amp to create an inverting op-amp with a slight gain to amplify our signal slightly before being fed into the filter. The gain for this inverting configuration was set at 15 K-Ohms / 4.3 K-Ohms = 3.49 V/V by the ratio of the two resistors.

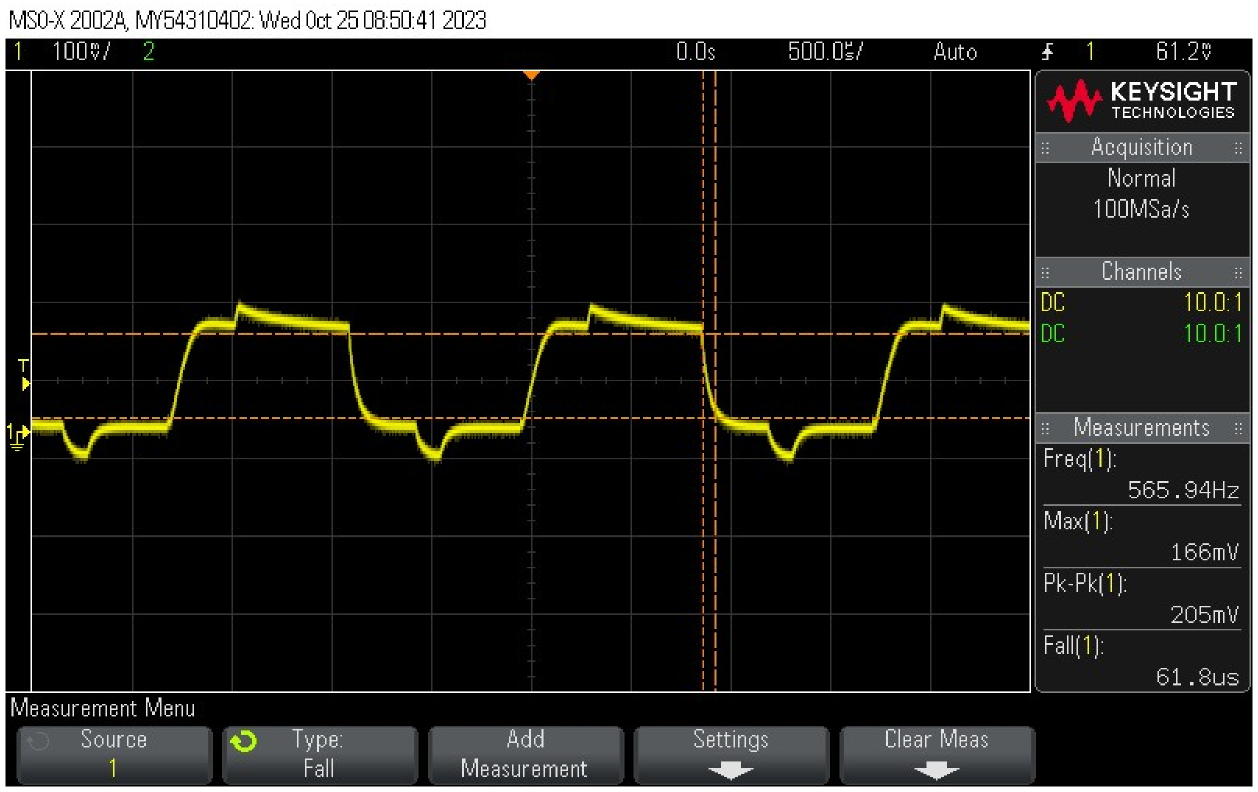


Figure 5: Signal captured at the output of the buffer block

In Figure 5, we see a mirror of the receive output, but larger with a max of 166 mV for an observed gain:

$$A_v = V_o / V_I = 166 \text{ mV} / 54 \text{ mV} = 3.07 \text{ V/V}$$

Filter:

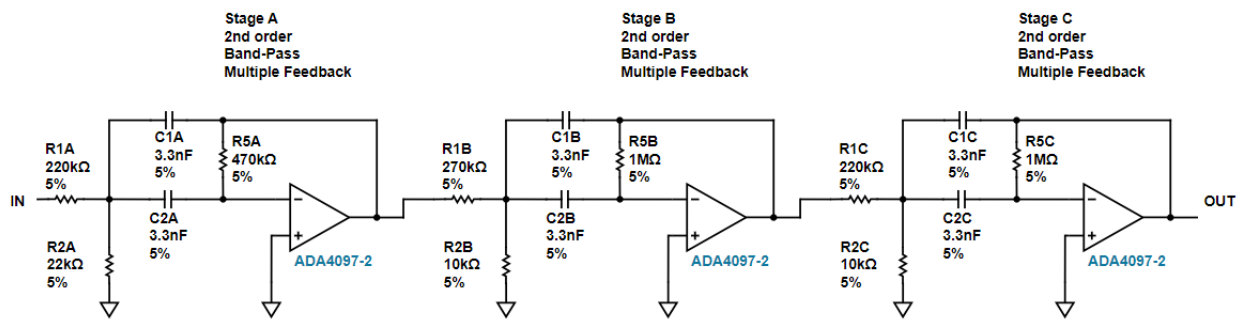


Figure 6: Filter Circuit Diagram

To design our filter, we used Analog Devices, Filter Wizard to simplify the process. This software allows us to specify a bandpass filter with specified center frequency, passband, and stopband. The application then produces a circuit schematic to use to build the filter. The filter we used is a 6th order butterworth, bandpass filter. We used a center frequency of 500 Hz, a passband of 200 Hz, and a stopband of 2 KHz.

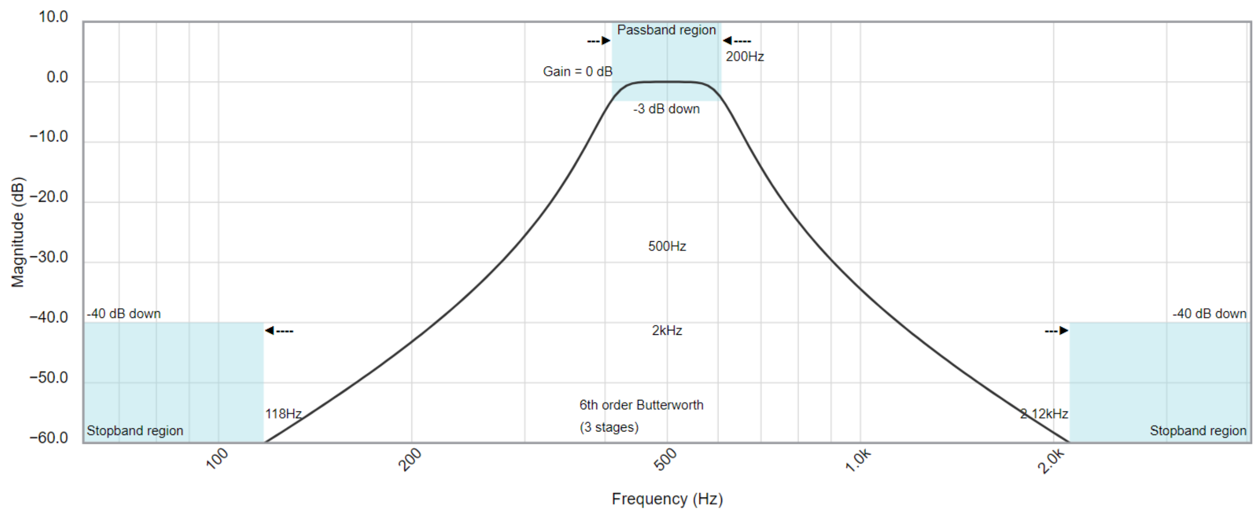


Figure 7: The graph here shows the characteristics of the active filter

Later on in our testing section we will show how the filter actually performed in our circuit over a wide range of frequencies.

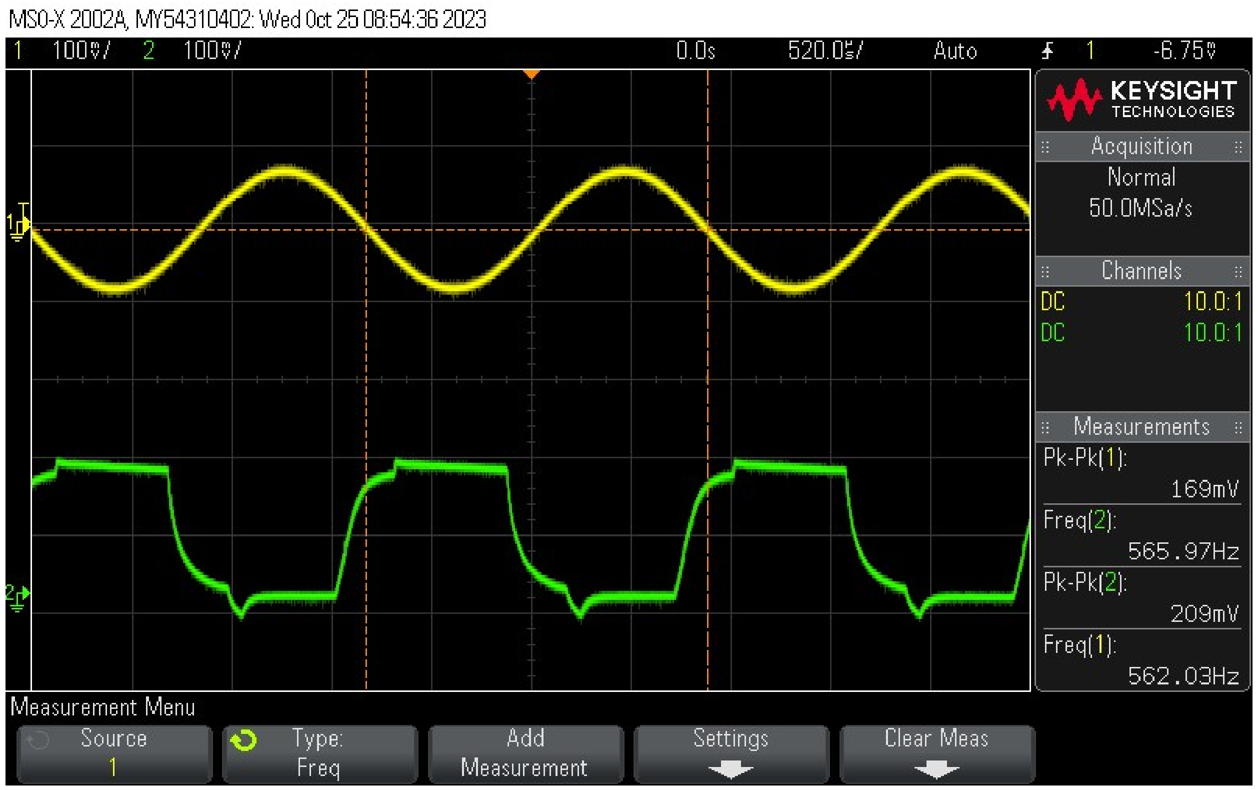


Figure 8: The Input of the filter (Green channel 2) Vs the Output of the filter (Yellow channel 1)

In Figure 8, we see that the signal is smoothed out by the filter into a nice sine wave as well as maintaining roughly the same magnitude and frequency.

Amplifier:

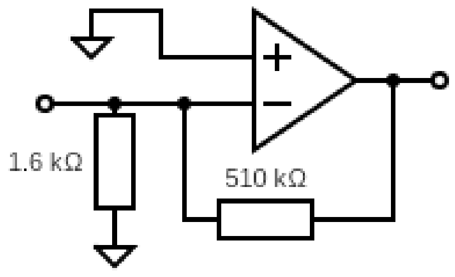


Figure 9: Amplifier Circuit Diagram

The amplifier block is used to amplify the output of our filter so that our receiver can be used over longer distances away from the transmitter. To amplify the signal, as shown in the circuit schematic, we used an inverting op-amp configuration where the gain of the amplifier is set by the ratio of the resistors. In our circuit we used resistor values of 510 K-Ohm and 1.6 K-Ohm, which gives a gain of

$$Av = R2/R1 = 318.75 \text{ V/V}$$

The inverting configuration means it will invert our signal, but since it is a sine wave and we are not worried about polarity, this will not affect our circuit. The gain was chosen to amplify the signal when the transmitter is far away so that we can still detect the signal. The gain was limited because when our transmitter is close to our receiver, a high gain will push our signal all the way to the power rails and would make it hard to detect the presence of our desired signal.

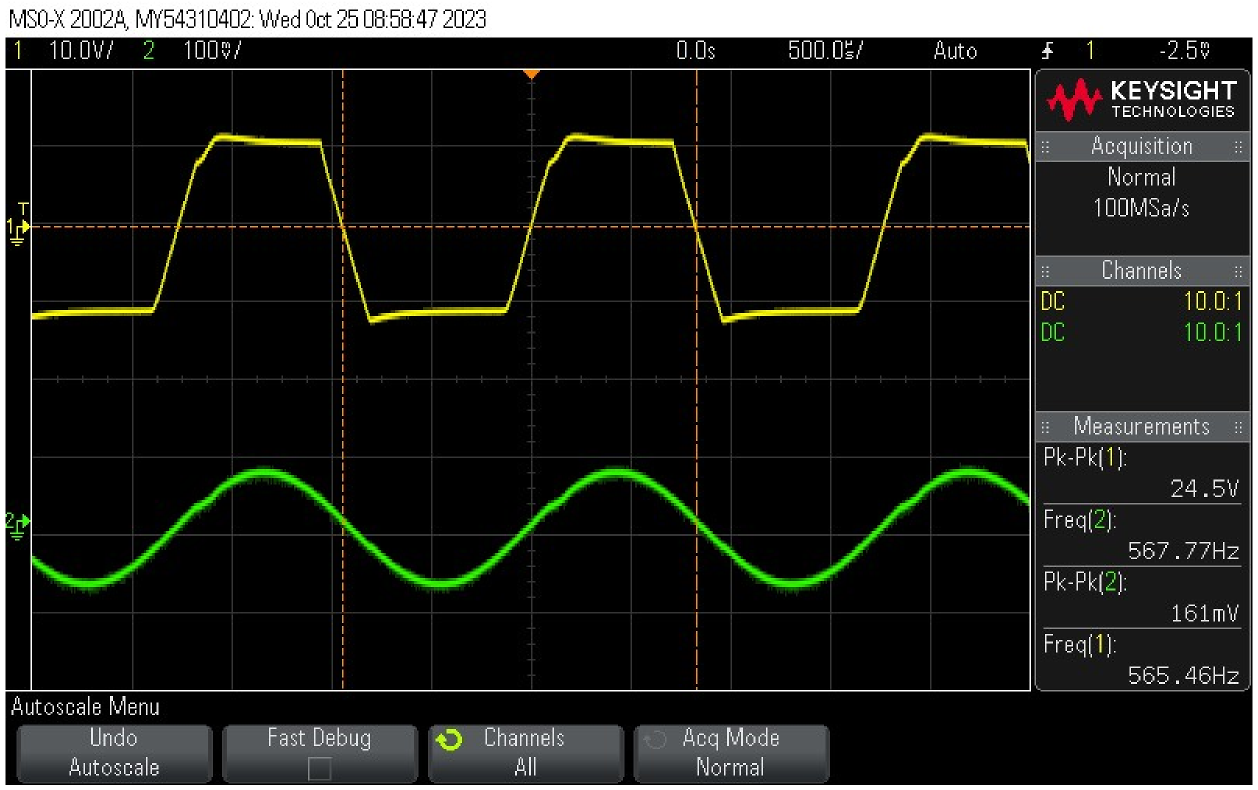


Figure 10: Gain block input (Green channel 2) Vs Gain Block Output (Yellow channel 1)

From Figure 10, we can measure our observed gain:

$$A_v = V_o / V_I = 24.5 \text{ V} / 161 \text{ mV} = 152.174 \text{ V/V}$$

This gain is noticeably smaller than the theoretical gain of 318 V/V and that is because as we can see in Figure 10, the output signal is clipped at the top and bottom of the sine waves because we are reaching our positive and negative power supply rails.

Peak Detector:

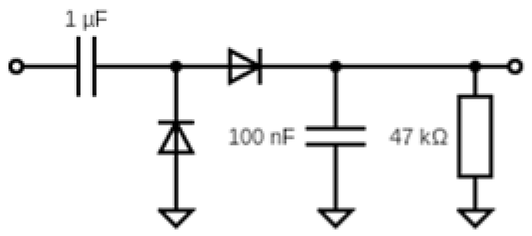


Figure 11: Peak Detector Circuit Diagram

After the filter, our signal is a sine wave with our desired frequency. Now we want to convert this wave into a DC like signal that can be used to set on and off a mosfet gate. To do this, we use a peak detector circuit. The purpose of the peak detector is to create a signal that follows the peak of our sine wave. This gives us a DC (almost) signal that can be used to feed into the gate of a mosfet. Here is an example plot of the output of the filter and the output of the peak detector:

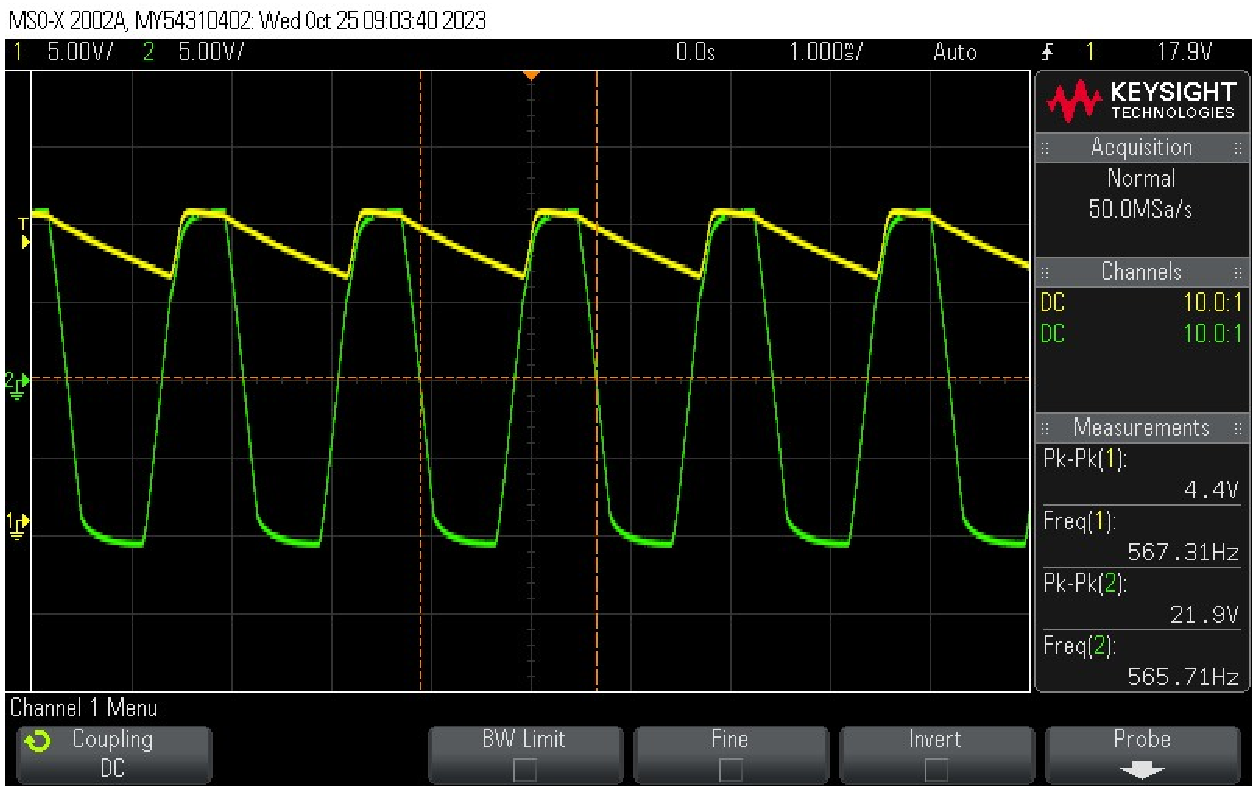
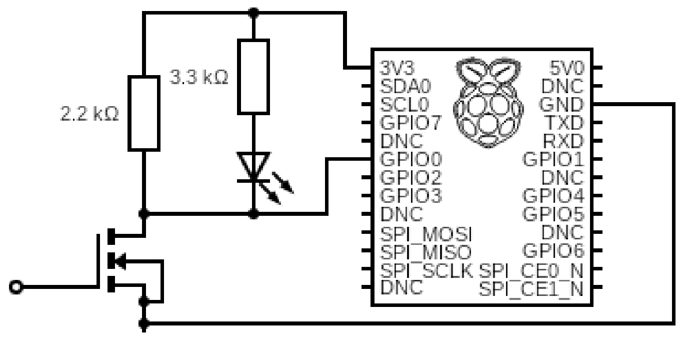


Figure 12: Filter Output (Green Channel 2) Vs Peak Detector Output (Yellow Channel 1)

From Figure 12 we see that the output of the peak detector follows the peaks of the filter output signal. The ripple voltage is quite large at 4.4 V making this more of an AC signal, but it will work for turning on our mosfet as it stays positive rather than our filter output that swings negative.

Conversion for Raspberry Pi:



In order to take our peak detector output signal and convert it into a pure DC signal

Figure 13: Raspberry Pi Conversion Circuit Diagram

with acceptable voltage and current we used a mosfet to isolate the preceding circuit blocks from our Raspberry Pi GPIO pin. The mosfet is used as a voltage controlled switch that pulls the GPIO pin low when the mosfet is on and conducting (signal detected). When the mosfet is not conducting (no signal detected), the GPIO is set as high as you can see from the circuit schematic. The power rail of the Raspberry pi is used here to provide the correct 3.3 V voltage for the GPIO pin. The resistor was calculated to limit the current into the GPIO pin. The desired max current rating was retrieved from the Raspberry Pi datasheet. This allowed us to set our resistor value to limit the current to less than the maximum rating. The combination of the two resistors is how the max current is set. The equivalent resistance is R1 * R2 / (R1 + R2). With our set up, this sets the current into the pin as

3.3 V/(3.3 K-Ohms * 2.2 K-Ohms/(3.3 K-Ohms + 2.2 K-Ohms)) = 3.3 V / 1.32 K-Ohms = 2.5 mA

When the signal from the transmitter is seen by the receiver, the output voltage is pulled to 0 V and sets the raspberry pi GPIO pin low. In contrast when the signal is not seen, the output voltage is pulled back up to 3.3 V and sets the GPIO pin high. This is how we can tell in our raspberry pi code if the transmitted signal is seen or not.

Software Overview:

For our software, we modified our texting code from the previous lab to fit our needs. We used the smtplib library to send text messages via email to one of our devices. After following the documentation we configured the text message to send when an interrupt was triggered (when the GPIO pin was high). Python's RPi.GPIO library was used to accurately detect when the signal from the circuit was high or low and called the function accordingly.

Design Process and Experimentation

Hardware Design:

The hardware design considerations we needed to figure out were:

Front end receiver

Buffer

Filter

Gain Block

DC Conversion

Raspberry Pi interface

Front end receiver impedance type:

Solution	Implemented?	Reason
Resistor in series with photo detector	Yes	Easy way to convert current from photodetector into a voltage we can use. Easy to set the impedance of the resistor. Does not function as a DC short to ground.
Inductor in series with photo detector	No	Converts current from photodetector to an AC voltage and shorts the DC voltage to ground. The impedance depends on frequency and so at the frequency we want (530 Hz) to get a good impedance, the inductor would have to be pretty big.

Front end receiver resistor value:

Solution	Implemented?	Reason
Large resistor > 10K	No	Create a large voltage from the output of the receive diode. Allows for longer distance transmission. Fails when in the presence of the lamp since the voltage goes to high and can not see our transmitted signal
Medium resistor 1K < R < 10K	Yes	Create a medium voltage that allows us the see our signal from medium distance and succeeds in the presence of the light most of the time
Small resistor < 1K	No	Create a small voltage so the lamp will not affect the signal at all. Only allows for limited distance

Buffer:

Solution	Implemented?	Reason
Op-Amp using just one channel in buffer configuration	No	Works as a buffer, keeps unfiltered signal small so unwanted frequencies are also small
Op-Amp using both channels of the IC to buffer and then slight gain	Yes	Works as a buffer, utilizes the hardware we have, provides the filter with a larger signal which is easier to filter, also amplifies unwanted signals

Filter:

Solution	Implemented?	Reason
Software Filtering	No	Keep the hardware simple, group isn't sure how to software filter, CPU intensive
Hardware Filtering	Yes	Group knows how to hardware filter, keeps software simple, Filter Wizard makes design of the filter easy

Gain Block:

Solution	Implemented?	Reason
Small Gain	No	Small gain makes sure we do not hit the power rails. Does not provide enough amplification when the receiver is farther away from the transmitter
Medium Gain	Yes	Provides detecting from farther distances, still minimizes the clipping at the power supply rails
Large Gain	No	Provides detecting from even farther distances, clips very badly at close distances

DC Conversion:

Solution	Implemented?	Reason
Peak Detector	Yes	Outputs a DC like signal with the DC

		value of the peak of input, less diodes needed. Signal is not a really good DC value as it still has a large ripple voltage
Bridge Rectifier and Capacitor	No	Outputs a DC like signal with DC value of the peak of input, requires four diodes instead of just two, creates a better stable DC signal

Raspberry Pi Interface

Solution	Implemented?	Reason
Use Raspberry Pi on board ADC to convert voltage to a digital value	No	Easier on the hardware, no need to level change the voltage, requires more software to read from the ADC
Convert voltage to Raspberry Pi voltage and use digital GPIO pins	Yes	More hardware, provides isolation between Pi and rest of the circuit, easy software for reading from GPIO pins on Pi

<u>Software Design:</u>

The software design considerations were:

Device to run program

Library for text alerts

Pin polling technique

Device to run program on

Solution	Implemented?	Reason
Raspberry Pi	Yes	The Pi has WiFi capabilities which allowed for us to use the text library we chose. In addition, we had familiarity with this type of interfacing already because of our Internet of Things class and the implementation of Lab 1 with the Pi.
Arduino	No	The Arduino doesn’t have WiFi capabilities, so we would have needed a different text library. With that, we would have need to make a completely new code environment which is something we were able to avoid by using the Raspberry Pi.

Library for text alerts

Solution	Implemented?	Reason
Smtplib and Email Text Message	Yes	We used the smtplib python library to send text messages from our email. We did this because it was a free way to send a text message. We used the email because we knew our Raspberry Pi would be WiFi capable, so it was easier to sign into an email than it was to connect to a phone number.
Twilio	No	We chose not to use this because after your free trial, you have to pay. We decided it wouldn't be cost effective to implement

Polling Technique

Solution	Implemented?	Reason
Interrupts	Yes	This was great for our application because we didn't want our Pi doing anything until the pin changed. When the pin changed, we do one action (send a text), which works will because an interrupt triggers on the change from low to high and not the pin being high.
Continually polling of GPIO pin	No	Too inconsistent and not nearly as efficient. We did a short test run with this implementation, but we quickly found that it wouldn't be suitable for this application.

Test Report

Hardware Tests:

Front End Receiver Tests:

Current through diode within spec (less than 15 mA):

Current = 12 V (max voltage) / R = 12 / 820 = 14.6 mA

Results: PASSED

Square wave from transmitter seen across the resistor (f = 560 Hz):

Figure 2 shows the square wave the diode produces

Results: PASSED

Buffer Tests:

First Stage Output Mirrors Input:

Input:

Figure 2 shows the output of the receive diode which is the input for the first stage of the buffer

Results: PASSED

Second Stage Output has Correct Waveform and Voltage Gain:

Theoretical gain of channel 2 op-amp = 3.49

Figure 3 shows the output waveform of the buffer block

Measured gain = 3.07

Results: PASSED

Filter Tests:

Filter Suppresses Frequencies Outside of Passband:

Input Voltage Amplitude = 3 V

Frequency	Output Peak-Peak Voltage
60 Hz	96 mV
100 Hz	96 mV
500 Hz	2.85 V
750 Hz	800 mV
1 KHz	193 mV
10 KHz	113 mV
50 KHz	113 mV
100 KHz	109 mV
500 KHz	105 mV

750 KHz	109 mV
1 MHz	109 mV

Results: PASSED, 500 Hz is passed, everything else is suppressed

<u>Filter Allows Passband Frequencies Through:</u>

Input Voltage Amplitude = 3 V

Frequency	Output Peak-Peak Voltage
400 Hz	630 mV
450 Hz	1.69 V
500 Hz	2.85 V
550 Hz	2.05 V
600 Hz	1.33 V

Results: PASSED, 400 Hz suppressed, everything else is passed

<u>Filter Output Waveform Mirrors Input when In Passband Frequency:</u>

Figure 5 shows the input and output of the filter when in the presence of a correct transmit signal

Results: Waveform distortion but acceptable, PASSED

Gain Block Tests:

<u>Gain Matches Expected Gain:</u>

Theoretical gain = 318

Measured gain = 152

As shown in Figure 6

Results: PASSED, measured much less than theoretical but that is because of clipping at the power supply rails

Peak Detector Tests:

Output Voltage Follows Peak of Input:

As shown in Figure 7

Results: PASSED

Output Voltage Is A DC Like Value:

As shown in Figure 7

Ripple Voltage = 4.4 V

Results: PASSED

Conversion for Raspberry Pi Tests:

Node for Raspberry Pi GPIO is 3.3 V when Beam is Blocked:

Output Voltage: 3.3 V

Results: PASSED

Node for Raspberry Pi GPIO is 0 V when Beam is Unobstructed:

Output Voltage: 0 V

Results: PASSED

Max Current Through Raspberry Pi is GPIO is within Spec:

Raspberry Pi max current into GPIO: 5 mA

Max current into Raspberry Pi GPIO = 3.3 V / (Rlim) = 3.3 / 1.32 K-Ohms = 2.5 mA

Results: PASSED

Overall Hardware System Tests:

Output Node Voltage ~ 0 V when Beam Unobstructed up to At Least 3 Feet Distance:

Distance	Voltage
6 Inch	0 V
1 ft	0 V
2 ft	0 V
3 ft	0 V

4 ft	0 V
5 ft	0 V
6 ft	0 V
7 ft	200 mV

Results: PASSED

<u>Same Test as Above, with the Lamp Pointed at the Receiver:</u>

Distance	Voltage
6 Inch	0 V
1 ft	0 V
2 ft	0 V
3 ft	0 V
4 ft	0 V
5 ft	0 V
6 ft	1.6 V
7 ft	3.3 V

Results: PASSED

<u>Output Node Set to 3.3 V when Beam Obstructed:</u>

Output voltage = 3.3 V

Results: PASSED

<u>Output Node Set to 3.3 V when Beam Obstructed and Lamp on Receiver:</u>

Output voltage = 3.3 V

Results: PASSED

<u>Software Tests:</u>

Software Test	Result

1) Send “Hello World” text	Pass
2) GPIO 17 to trigger on interrupt	Pass
3) Send text on interrupt	Pass
4) Text message sends time and date	Pass
5) Text message sent when GPIO High	Pass
6) Text message is sent in the correct format	Pass

Project Retrospective

In lab 2 we successfully completed and fulfilled all requirements given to us while staying on time and on budget. We played to our strengths, as Sam Loecke is EE and took team leader of the project. We all learned from Sam and tried to split the workload when we could but Sam contributed the most by far. We split the project up as follows:

Samuel Nicklaus - Sourced circuit components, assisted in creation of texting software on the raspberry pi, assisted in testing the circuit

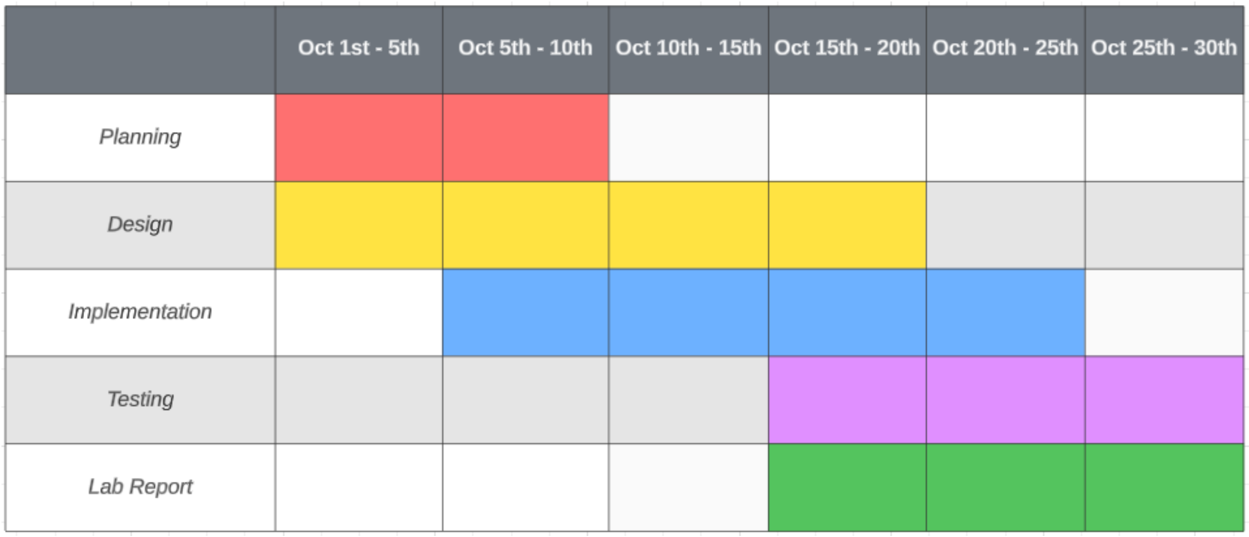
Cole Arduser - Transmitter and Filter design and creation. Helped test filter.

Luke Farmer - Raspberry pi code and testing

Sam Loecke - Circuit design, Implementation, and testing

The work distribution for this lab was more lopsided than the previous since the project was circuit heavy. EEc individuals worked heavily on the initial circuit design while CSE individuals created the software, and assisted in testing. We implemented a Waterfall methodology as we were already given strict requirements and guidelines. This helped a lot as we started our project by having a long meeting going over how we wanted to design the circuit. What we were most concerned with was the knowledge gap between individuals for circuit design.

Our biggest challenge for this lab was dealing with the 100w light bulb as after our initial circuit creation, the distance at which the transmitter worked was not meeting the requirements. Originally we thought the biggest challenge when dealing with the light bulb would be filtering out the different frequencies so we focused on the filter. The front end of the receiver ended up being the issue with the light as in the presence of the light the voltage was pulled to power and did not capture the transmit signal. We adjusted the front end resistor value to create smaller voltages and that helped us to deal with the light. If we were to have an area for improvement the front end design is where we could have gone back to the drawing board after seeing the effect of the light.



	Oct 1st - 5th	Oct 5th - 10th	Oct 10th - 15th	Oct 15th - 20th	Oct 20th - 25th	Oct 25th - 30th
Planning	🟥	🟥				
Design	🟨	🟨	🟨	🟨		
Implementation		🟦	🟦	🟦	🟦	
Testing				🟪	🟪	🟪
Lab Report				🟩	🟩	🟩

For the most part, we were able to stick to our plan and timeline. Actually we were ahead of schedule for the majority of the lab which allowed us to test further and try to increase the overall distance of the device.

Conclusion

Overall the goal of this lab was to create an “electric eye” safety system that can detect obstacles. We accomplished this with the system we built and over achieved on the range as we were required to reach 3 feet but we reached all the way to 8 feet. We made good design decisions in our circuit but also had to do some redesigning of the front end and buffer blocks to make our system reliable in the presence of the lamp. By using mostly hardware elements in the circuit we were able to keep our software relatively simple and limited to sending texts.

Appendix & References

Raspberry Pi 4 documentation
https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/, 2023

Python Raspberry Pi GPIO library documentation https://pypi.org/project/RPi.GPIO/, 2022

SMTPLIB Python Library, https://docs.python.org/3/library/smtplib.html, 2023

"Filter Wizard." Analog Devices. https://tools.analog.com/en/filterwizard/, 2023

Anton Kruger - ECE:3410:Electronic Circuits - Lecture Slides

NPN Silicon Phototransistor OP505A,
https://www.mouser.com/datasheet/2/414/OP505A-42094.pdf, TT Electronics, NA