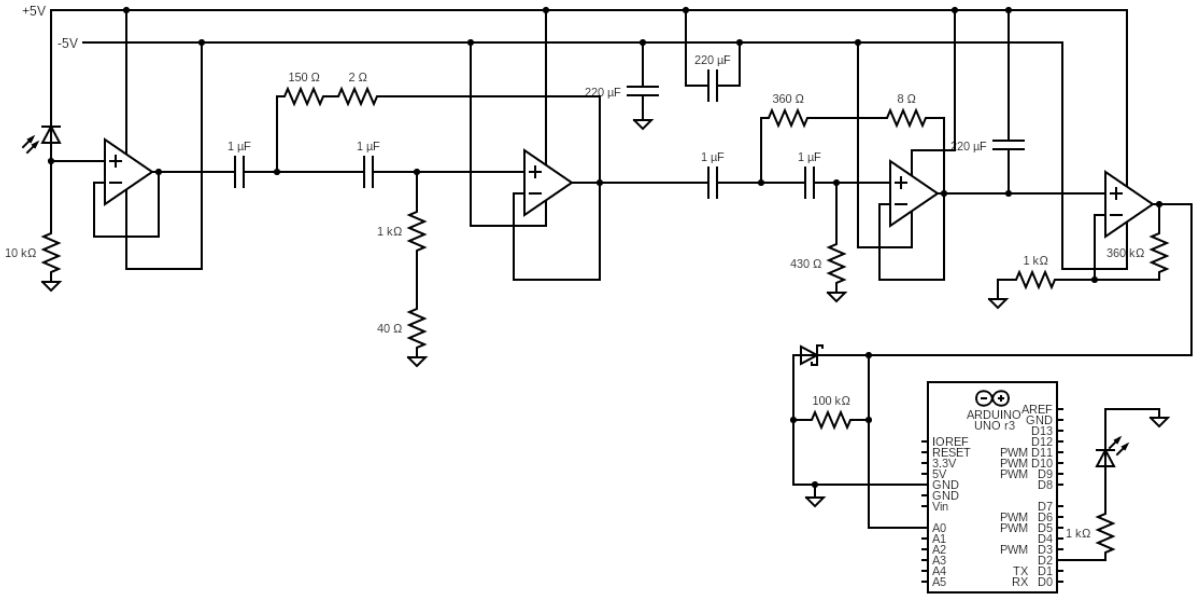
# Lab 2 Report

By: Chad Wynter, Charlie Wolfe, Kyle Lastine

## Design Documentation

**Hardware Design**: Our system consists of a photodiode, unity gain buffer, two-stage filter, and noninverting amplifier. The output of the circuit is read by an analog pin on an Arduino Uno, which then controls an LED.




The first op amp functions as a unity gain buffer that repeats the output of the photodiode, to prevent the filter loading the photodiode and flattening out the signal. All op amps are powered with +5V on the positive terminal and -5V on the negative terminal. There is a resistor going to ground to create a voltage from the current output of the photodiode.

The following two op amps are filters in the Sallen-Key configuration, with capacitors set to 1uF because that was the material we had available. We were then able to solve for the appropriate resistor values to filter out <120 Hz and allow for the 500 Hz transmitter light to pass untouched. Some resistors had to be substituted with combinations of resistors, which is reflected in the schematic.

The final op amp stage is a noninverting amplifier. We had some issues with the Arduino reading the output of the filter, so we amplified the signal with a gain of 361. The output of the amplifier is connected to a Schottky diode used to protect the Arduino

pins from a high negative voltage. When the output sinusoid reaches <-0.4, the diode activates and the signal is connected to common ground instead of the input pin.

**Software Design:**

Python Software

The python script is used to listen to the serial instance coming from the Arduino and to send indication texts when the Arduino tells the python script the signal has been interrupted. To read the serial stream, we use a python library called pyserial. Each time a line is read, the script will check to see if a hardcoded indicator appears in the output. If the indicator is present, the python script will take the current time, append the time to a set message, and send the text message using gmail. The SMS functionality is largely unchanged from what we used in Lab 1.

**Pseudocode/code structure:**

```
# serial = serial instance
# gmail = gmail instance to send texts
# While True:
#       serial_message = read serial output
#       if indicator is in serial_message:
#               current_time = get current time
#               message = “Critical safety event at ” + current_time
#               gmail.send(phone number + cell carrier, message)
```

Arduino Software

The software on the Arduino is used to process the signal read in from the analog pin connected to our circuit. To do this, we store the last ten values read from the analog pin. Each time we read a value, we take the difference between the minimum and maximum values in the stored values. If the difference is greater than a hardcoded threshold we set, we determine that the signal has been broken. The intuition behind this is that if our hardware based filter is working, the input square wave should only have a sufficiently large difference between the min/max values if the desired signal is actually present, and not being filtered out.

Once our software detects that the signal has been broken, it will send a message over the serial port to tell the python script running on a separate computer to send a text message. It will also write to a digital output pin to turn on the LED. Each time the signal

is broken, the Arduino will only send one indication message across the serial port. The Arduino will not send another indication message until the signal has been restored and then broken again. This is to avoid texts constantly being sent while the signal is broken.

**Pseudocode/code structure:**

```
# last_readings[10]
# text_sent = False
# While True:
#       value = analog_read(CIRCUIT_OUTPUT_PIN)
#       shift_and_store(last_readings, value)
#       max_difference = get_max_difference(last_readings)
#       if max_difference > 300:
#               digital_write(LED_PIN, OFF)
#               text_sent = False
#       else:
#               if !text_sent:
#                       Serial.write(indicator)
#               digital_write(LED_PIN, ON)
#               text_sent = True
```

## Design Process and Experimentation

**Introduction to final design**

The Final design of our system was mostly hardware based, with the only software portions used to send a text when the signal was broken, and also to determine a threshold of the analog pin on the arduino in order to turn on and off the LED. The hardware began with a single photodiode that was used to take in light readings. The voltage generated by the diode was then passed through an op amp that was configured to be a unity gain circuit, which ensured that we would be passing in an adequate signal into our filter even if the transmitter circuit was further than a meter. Next, that signal went into a two stage high pass filter that consisted of two op amps in an integrator circuit with capacitors, and resistors that had the job of filtering out signals that were not close to 500 Hz. At the output of the filter circuit, we used a non inverting amplifier configuration to amplify the signal leaving the filter, so that we could make accurate decisions based on the readings taken. The Led was hooked up to the arduino separately from the photodiode circuit, and only consisted of a red led and a resistor. The software for the arduino examines the average of the past 10 readings taken, and then makes a decision on whether to turn on the LED (when the signal was broken), or

turn off the LED (when the signal was restored). The steps that it took to lead to this final design are described below.

**Hardware Design 1 (no hardware filtering)-** When just starting the lab, we decided to take a software based approach to the filtering since we have more software experience than hardware experience. We did not get very far with this approach as we soon realized that a hardware based filter would be a better solution for us, so no real software code towards the software filter was created. What we did though, was construct the photodiode with various resistor values, and then tested what types of values we would get into the analog pin in different lighting conditions. At first we thought that the photodiode was broken, as we kept reading in the maximum allowable value into the arduino analog pin. This factored into our decision in trying a hardware based approach since we figured we needed to filter the signal at least a little before reading in any value into the analog pin.

Experiments

1) We tested a simple photodiode schematic, and tried getting readings into the analog pin. We were looking for the analog pin reading to change in different lighting conditions. The schematic for the photodiode and resistor combination is located in Appendix A.
    - We first tested this with a 1k resistor. When changing lighting conditions, the analog pin was continually outputting 1023, which is the max value of the analog pin.
    - We also tested resistor values of 100, 500, 2.5k, 5k, 10k, 100k, and 1M. Each test ended in a failure as the analog pin continually outputted 1023.

**Hardware Design 2 (filtering fully with hardware plus non inverting amplifier)-** After deciding to go with a hardware based approach, we used some help from the filter design tool to construct a two staged high pass filter. We decided on high pass because we were looking to allow high frequencies to pass through the filter, and then impede other low frequency signals that may be trying to pass through. At first, we decided on using the AD8657 op amp since it was recommended, but we ended up using another op amp which would change our design slightly. This is because when the AD8657 op amp arrived, we noticed we made a mistake in its ordering and we got the microchip version of the op amp. After changing to the OP275 op amp, we constructed the two stage filter and then began testing the outputs of each individual stage, or op amp. We were expecting to see a more and more filtered signal after each stage, but that was not

what we observed at first. The signal we observed was weak, and did not seem to filter the signal correctly. We did not change any of the resistors or capacitors because we knew that was the correct configuration as stated by the filter design tool. What we did notice though, is that we had two rails that went to ground, and two rails that went to 5 volts. This is significant because for the filter circuit, in was configured in such a way that we needed one of the rails to be connected to -5V, as the ground portion of each op amp would be connected to that instead of ground. After we made that change, we saw a significant performance increase after each stage of the filter, leading us to believe that it was filtering correctly. We tested this extensively as outlined below. After we were satisfied the filter was configured correctly, we began working on a non inverting amplifier at the output of the second stage of the filter, that had the purpose of amplifying the signal before we took it into the arduino and made decisions based off of the value. The values of the resistor used in this non inverting amplifier decided the overall gain, and we first decided to go for a gain of 100. This led us to choose resistor values of 10k and 100, as that would assure a gain of around 100, but that did not work.The output of the non inverting amp was not amplifying the signal that was given by a signal generator, and was in fact distorting the signal a bit. After some consulting, we decided to go for a higher overall gain of around 350, and to choose higher resistor values, the resistors we ended up using have values of 360k and 1k, and that seemed to provide the signal amplifying that we were looking for.

Experiments

1. Each stage of the filter was filtering correctly, resulting in a high pass output that filtered out low frequencies. Signal was provided by a signal generator hooked up to the input of the filter.
    a. Using an oscilloscope, we tested the outputs of the first stage of the filter and the second stage of the filter. The waves on the oscilloscope did not seem to move as expected, and the output of the second stage was not correctly determining which frequencies should pass through. We checked if our resistor / capacitors were correct, but did not change any of them.
    b. After changing the '-' section on the op amp from ground to -5V, we had to use a different power generator that could supply a -5V to one of the rails. After doing so, we checked again the outputs of each stage of the filter. The first stage looked much better as it seemed to filter out some noise, and the second stage seemed to behave as expected since when changing the frequency provided from 500 to 400 to 300 to 200 to 100, the output of the filter would begin to show less and less of a sine wave before showing nothing,

which told us that the filter was filtering out unwanted frequencies correctly.

2. The non inverting amplifier at the end correctly amplified the signal to where we would be able to make correct assessments
    a. Using resistor values of 10k and 100, we first observed the output of the second stage of the filter, and then the output of the amplifier using the oscilloscope. With an input of 500 Hz generated from the signal generator, the output of the amplifier did not seem to accurately amplify the signal from the output of the filter, and instead seemed to add some weird noise to the signal shown in the oscilloscope.
    b. We next tried resistor values of 360k and 1k, as we were informed that it was bad to use small resistor values, and to go for a higher gain. After changing the resistor values, we again observed the output of the second stage of the filter, and the output of the amplifier using the oscilloscope. The waves shown performed more as expected this time, as the output of the filter was a wave, and the output of the amplifier was a much larger wave.

**Hardware Design 3 (adding schottky diode after output)-** After getting the non inverting amp configured correctly, we decided to hook up the output of our circuit to the analog pin of the arduino, with the intent of testing the values we received from the filter/amplifier configuration provided from a signal generator. The values we were getting were not as expected, and the arduino was taking sporadic measurements. After some research, we noticed that the minimum voltage accepted by the arduino was -0.5V, which was a problem for us because the output of the amplifier went between 5 V and -5 V instead of ground. In order to correct this issue, we decided to set up a schottky diode as shown in Appendix A, in order to cut off any signal being received that was less than the turn on voltage of the diode, which is very low. At first we thought that the arduino would be broken since we were trying to read in voltages that were less than 0, but it seems to have some sort of protection against that. After trying that configuration with a schottky diode and a 100k resistor we tested the outputs of the circuit with both the oscilloscope and then with the observed values from the analog pin.

<u>Experiments</u>

1. The oscilloscope voltage readings were cut off at 0 V, with a wave that alternated between 5 V and 0 V

    a. This test was a success, as we simply measured the output voltage at the end of the circuit, and ensured that the voltage did not dip below 0 V as the arduino would not be able to handle that signal.
2. The arduino reads in helpful values that allows us to collect values to take An average.
    b. This test was a success, as we were able to take in an average of the last 10 values read in from the signal generator through the filter in order to set a minimum value of when to turn on the red led.

**Hardware Design 4 (adding unity gain after photodiode)-** When connecting the final piece of the circuit, we first decided to hook up the photodiode similar to what was done on the demo circuit, with just a resistor going in the voltage signal and the photodiode going into ground. This seemed to work, as the system behaved as expected, but it did not continue to behave as expected for distances past a foot. This means that if the transmitter was only a foot away from the receiver, the red led was turning on as the receiver was not taking in any signal at all. This was due to the fact that electrons were being lost instead of everything going into the filter circuit. To fix this, we added in a unity gain buffer which was supposed to perform the duty of capturing the full signal taken from the photodiode before sending it to the filter circuit. We first set up the unity gain using a 1k resistor going to the input from the photodiode because we thought that we needed a large resistance after some testing as detailed below, the output of the signal was getting a very weird wave on the oscilloscope, and the maximum distance from the transmitter did not increase by much. We then tried different resistor values at the beginning of the circuit that was connected to the photoresistor, with values of 10k, 100k, and 500k. These all provided similar results as the first resistor test. We ended up not using a resistor at all, as the resistance that was provided when passing through the unity gain buffer proved to be just enough to provide a good signal into the filtering circuit.This ended up working, as we were able to put amplifier over a meter away from the receiver, and we were still getting a good signal. The tests we performed are outlined below.

<u>Experiments</u>

1. With the photodiode connected to just the resistor going into the filtering circuit, the red led would turn on when the signal was broken.
    a. This test was a success, as when we put our hand between the transmitter and the photodiode, the red led would turn on as expected

2. With the lamp turned on, the red led would turn on when the signal was broken and stay off when the path between the transmitter and receiver was clever
    a. This test was also a success, as the receiver was filtering the lamp as it should have been.
3. The transmitter could be positioned at a distance of over a meter with no noticeable drop in performance.
    a. This first test was a failure, as when the transmitter was over a foot past the receiver, we were getting no signal into the filter circuit.
    b. This was tested again with a unity gain buffer and different resistor values of 1k, 10k, and 100k. We got a bit better distances, but the output of the whole circuit behaved weirdly, and gave a weird wave which did not really seem to accurately show a filtered output.
    c. After removing the resistor altogether, we tested the distance again. Now, we were able to pick up a strong signal for distances well over a meter, and there was not a large drop in performance when the lamp was turned on.

**Software Design (Python texting script)-** We had initially experimented with using a python script to read serial output in Lab 1 when we had planned on using an Arduino. We changed our plans, so we never had to use it. The SMS functionality we also used for Lab 1 and was left largely unchanged.

Experiments

1) Using a python script to read serial messages.
    - At the first implementation, we had issues with the messages across the serial port being extremely delayed. We were able to fix this by increasing the baud rate.

**Software Design (Arduino Programming)-** The main experimentation done with the software on the Arduino had to do with the previously discussed threshold of the maximum difference of the past 10 read values. This threshold value changed as we updated the circuit.

Experiments

1) Processing the signal output from the circuit.

- When initially using a signal generator to test our circuit, the threshold was able to be set very low due there not being much noise.
- When testing the filter on the provided transmitter, we realized we had to change the threshold to account for the added noise. The filter still worked well enough so there was a significant enough difference between an active and broken circuit so the threshold was still viable. Surprisingly, even when using older iterations of our filter that did not perform well, the threshold method we used was still able to tell the difference between an active and broken/noisy signal after tweaking the threshold to account for it.

## Test Report

We did much of the testing of the circuit with a signal generator, since we saved the photodiode setup for last. There were five main parts of this system that needed testing, being the filter circuit, the non inverting amplifier circuit, the schottky diode that cut of voltages below 0, the unity gain circuit that was connected to the photodiode at the beginning, and then the systems software that read in analog values and then took averages in order to determine if the red led should be on or off. We tested the system in parts after they were completed in that order, so we were only able to test the requirements of the system after the photodiode was connected correctly. We first tested the function of the two stage filter by observing the output of each stage on the oscilloscope. We also compared the output of the filter at different frequencies on the signal generator, to see if it performed well at filtering out unwanted frequencies. Changes we made to correct any issue in testing this system are outlined in the Design Process and Experimentation section. Next, we tested the function of the non-inverting amplifier that was placed at the output of the two stage filter. This was testing using the oscilloscope again, and observing if the output was amplified from the output of the second stage of the filter to the output of the amplifier. The next thing tested was the output of the system when a schottky diode and resistor were connected in parallel at the end of the output. This again was tested using the oscilloscope, as we just observed the voltage ensuring that the wave never went below 0 V. After ensuring that was the case, we could then connect the output to the arduino, connect the input to the photodiode, and then test the readings from the analog pin. When testing, what we were looking for was a noticeable drop in the average of readings taken when the signal was cut, and then for that average to increase again when the signal was restored. After

ensuring that this was the case and the system operated well at a good distance, we then repeated those tests using the lamp, to ensure that the filter circuit was correctly filtering out the frequencies the lamp was giving off. Once we noted that this was the case, we were ready for checkoff. The chart below outlines the final tests that we performed in order to ensure that the system performed up to user specifications as outlined in the lab requirements.

| Test | User requirement addressed | Expected result | Actual result | Pass/ Fail | Corrective Action |
|---|---|---|---|---|---|
| LED indicator | Device indicates if there is a signal | LED is off when there is a signal, on otherwise | When transmitter pointed at receiver, LED turns off; on when blocked | Pass | None |
| Low frequency performance | Device does not respond to low frequency light | LED is on when in the presence of ONLY 120Hz light | LED never turned off when 120Hz lamp pointed at it | Pass | None |
| Text message | User contacted when signal is broken | User receives a text when signal broken | Text message received at appropriate time | Pass | None |
| Distance performance | Device must perform at range | Device performs as in the first test at a range of 3ft | LED turned off if transmitter pointed at it from 3ft away | Pass | None |
| Light performance | Device must perform in the presence of noise | Device should filter out noise from 100W bulb | LED turned off if 100W light near transmitted | Pass | None |

## Project Retrospective

**Outcome-** The project was an overall success, with the finished product being a working device that met the given specifications. However, there are some aspects of the project that could have been improved to be more reliable or efficient.

SMS messaging was done through email and sent to cell service provider servers to be routed to mobile numbers. This worked, however messages were often delayed and sometimes not received at all due to unreliable email servers for cell carriers.

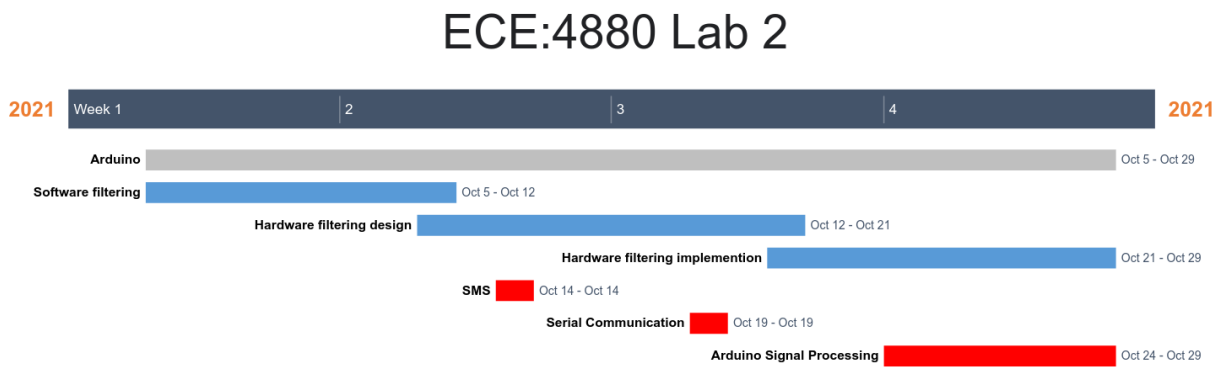
Furthermore, we had to specify which cell service the phone number we were texting was. While the issues of this method were out of our control, we could have switched to a different, more reliable service such as Twilio to send our SMS messages.

**Incomplete Sections -** All fields in the Text Report Table were passed.

**Member Contribution & Workload Distribution -**
**Kyle -** Hardware Design / Implementation, Lab report
**Chad -** Hardware Design / Implementation, Lab report
**Charlie -** Software Design / Implementation, Lab report

**Project Management-**
We did not use a specific/named project management style but in essence, we broke the project down into smaller sections and took one on at a time, testing and integrating as we went. Once one part of the system worked, we moved on to the next and tested subsequent sections as they were completed. This allowed us to quickly find the source of issues and fix them, as only one part of the overall system had changed. This is reminiscent of test-driven development. We also regularly communicated about which sections were being worked on and finished, and had to collaborate on certain parts that involved multiple sections of the project.

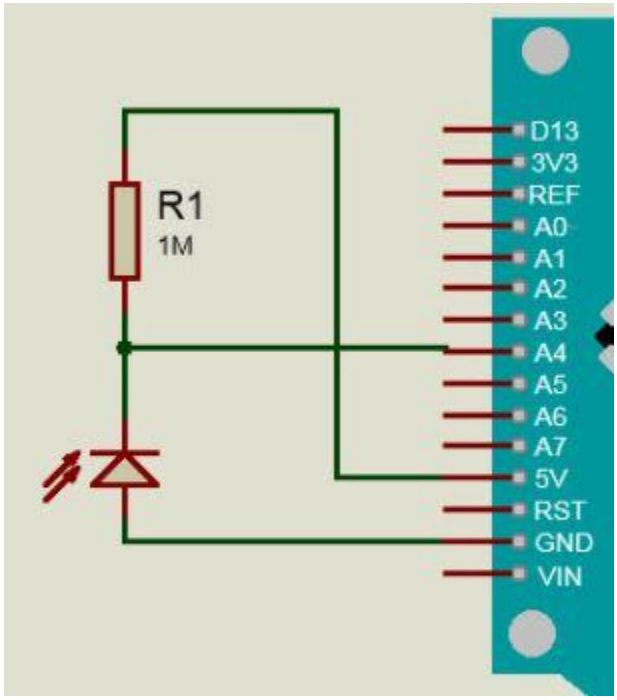
**Gantt Chart-**



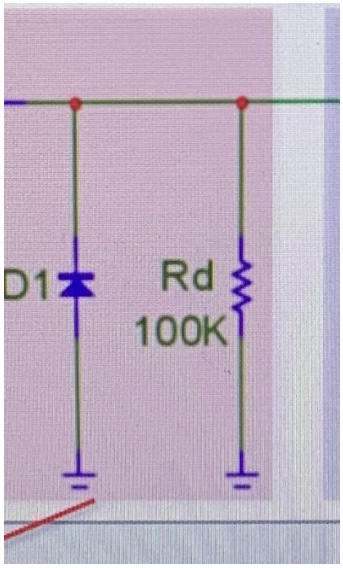

Above is the Gantt chart indicating when each task was started and completed. The gray colored boxes show the device we worked on, the blue boxes show the hardware tasks and the red boxes show the software tasks.

## Appendix & References

Appendix A:




Appendix B:



Libraries used:

- pyserial - To read serial output from the arduino using a python script.
- smtplib - To send emails (for texts) using gmail.

Website used to aid in filter design:

- Filter Design Tool | Filter Wizard | Analog Devices