Goals:

- Characterize an inductor and learn how it functions
- Learn about Piezoelectric sensors
- Learn how to apply Amplitude Modulation to a signal
- Learn how to use an envelope detector to filter an AM signal
- Create a digital counter to provide a limited visual feedback to the user
- Create arduino code to take in a digital signal and compare that to a stored passcode
- Create a functioning door opener that is activated by a passcode entered through a knocking sequence

Introduction:

A common problem is forgetting keys inside of a room or office and being locked out because of it. To try and solve that problem, we decided to create an inductive door opener for our ES52 final project. Our approach utilized knocks as a code that were transmitted across a door through the magnetic field of a wire coil and received by a hall effect sensor, which after some signal processing was fed into an Arduino microcontroller to be parsed and compared to a code already stored on the Arduino. We also provided the user with visual feedback on when they could enter new data, a knock or a lack of a knock, through a counter and oscillator subsystem that blinks an LED to prompt the user for new input. The core components of our project were: the piezoelectric sensor that fed into a monostable pulse circuit, the sine wave generator created from an LMC555 in astable mode and a Sallen-Key filter, the amplitude modulation circuit that fed into the coil, the receiving circuit that processed a signal from the hall effect sensor, the counter system that provides visual feedback to the user, and the Ardunio system that takes in the digital code and controls the servo.

Characterizing the Coil:

There are three good characteristics to know for every coil that is going to be used, the quality factor, Q, resistance, R, and the self inductance, L. The resistance is an easy measurement, because all that is needed is the multimeter. This can measure the resistance of the entire coil through connecting the two leads to the two ends of the coil. Use the smallest power setting (the coil will be very low resistance) to get the resistance of the coil. For the quality factor and self inductance measurements, a special machine called an LRC meter. This outputs the Q value and the inductance of the coil .

For the coil used in our project, our values were:

Q = 30
R = 21Ω
L = 10.5 mH

Piezoelectric-Monostable Circuit:

The small piezoelectric device, SEN-09198 (Sparkfun), we used was capable of generating a magnitude anywhere from +/-0-90V. However, for the purpose of sensing a knock on a door the typical range was +/-0-5V (with the vast majority of knocks below 1 volt, at about 500 millivolts). A simple knock with the knuckles generated less voltage than hitting the door with a fist. The piezo can also generate negative voltages as it oscillates. To solve these issues, a clamp is needed to protect the other circuit elements from damage. A small signal zener diode, the 1N5232, is a logical choice. We chose one with a zener voltage of about 5.6 volts and a forward voltage of about 0.6 volts.

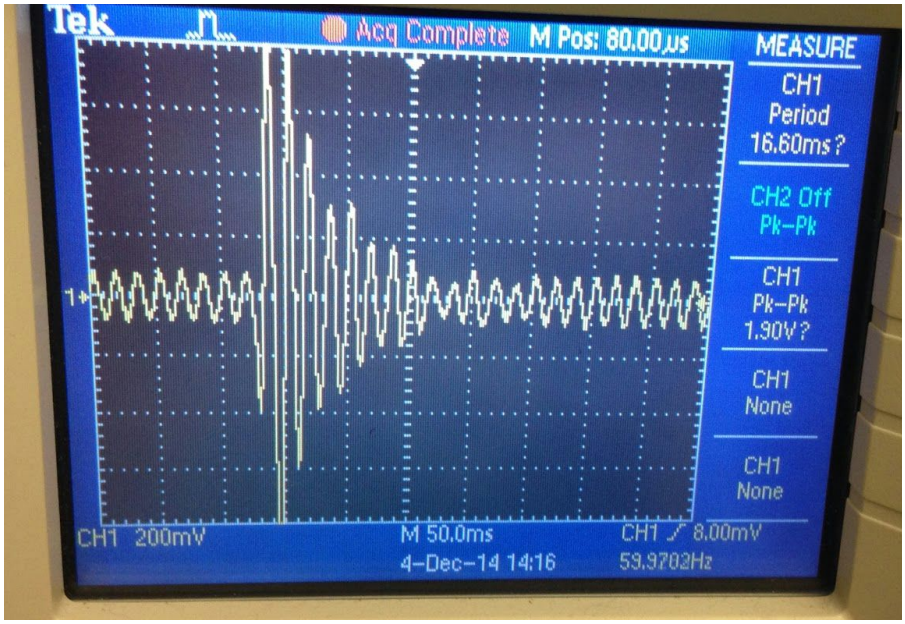
These voltage spikes are brief, not long enough in duration to convey enough information back to the user, and oscillatory in nature, so something was needed to extend this signal. For that, we turned to a monostable circuit, which produced a single pulse of a pre-determined length for every knock. For its ease and simplicity of use, we decided to use an LMC555 chip to produce the monostable pulse we wanted, but we ran into a number of problems: firstly, the 555 chip would only trigger when the voltage input to its threshold pin was less than one third of the source voltage. Considering that the equilibrium voltage for the piezo-electric sensor was at ground, we could not simply connect the sensor's positive lead directly to the 555's threshold pin. One idea was to add a DC offset to the signal, but that would be moderately complex, and would likely take up another op amp, not to mention the variable, fluctuating nature of the sensor.

We decided to design our own inverter circuit using a small signal nMOS transistor, the VN2222. With the inclusion of this component, the threshold pin of the 555 would always be at 5 volts, unless the signal from the piezo sensor, which was connected to the gate of the transistor, was greater than the threshold voltage of the transistor. Most knocks were not greater than the threshold voltage of the transistor, which is about 1.5 to 2 volts, so we decided to add an op amp adjustable gain stage, using an op amp configured in a non-inverting gain setup.

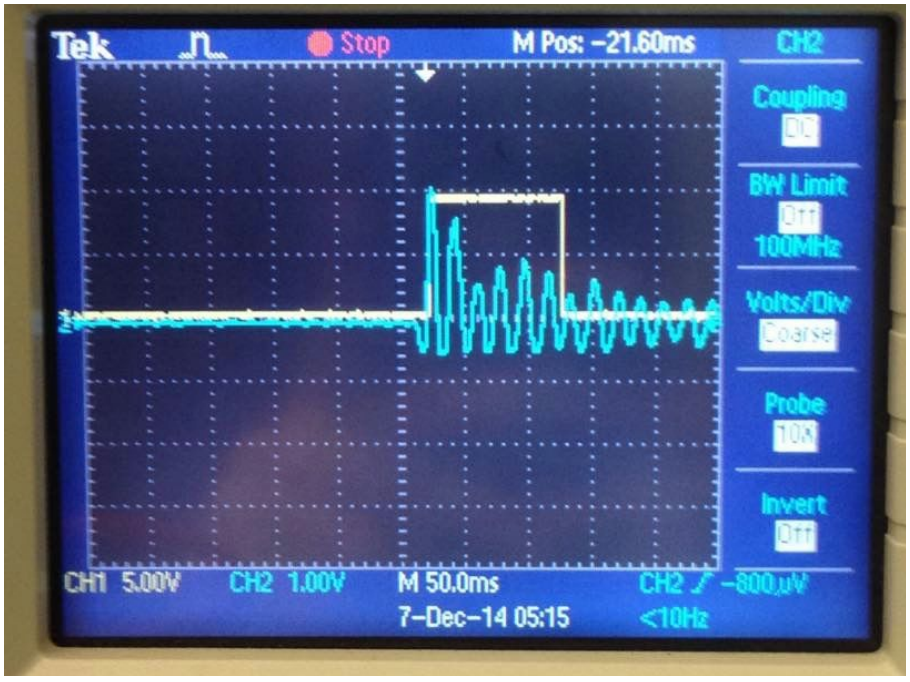
The LMC6482 op amp was used to help amplify the signals detected by the piezo sensor. The inclusion of the 10kΩ potentiometer would allow for setting how sensitive the sensor is to the knocks by how much the potentiometer changes the gain. $Gain = 1 + \frac{Rpot}{1.2k\Omega}$, so gain goes from 1 -> 9.33 depending on the resistance of the potentiometer.

The LMC555's monostable configuration, according to the datasheet, has an output duration of t = 1.1*R*C. A final value of ~2 seconds (R = 2MΩ and C = 1μF) was chosen because this seemed like an ideal amount of time to give the user before a second input was sent. We also connected a red LED through a 510Ω current limiting resistor to show that a knock was detected.

This system worked exactly the way we had hoped, amplifying the knocks' impact on the piezo sensor just enough to affect the MOSFET and trigger the monostable. The largest weakness of this design is that the user must wait for the pulse to end in order to be able to knock again and trigger the monostable. If a user knocks while the monostable is already outputting a pulse, that knock will not be picked up. We added the red LED and later the counter feedback system to help mitigate this potential for error.



This is the output from a test of how the piezo-electric sensor operates. It is clear that the weight on the sensor shakes back and forth after a strong enough motion, creating a damped, oscillating voltage differential.



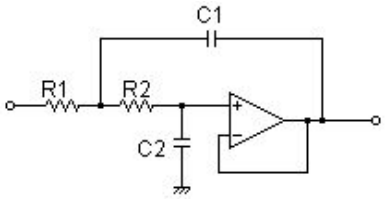
This is the output of the monostable from a signal received by the Piezo sensor (note that this signal is from a 1 second monostable. For our final design, a 2.2 second monostable was chosen). Yellow is the output of the LMC555 and blue is the voltage difference across the piezo sensor.

Amplitude Modulation and Transceiver:

A large, overarching problem central to this project was the question of how to transmit information from one side of a door to another. Transmitting the information visually, while it might work on some doors, will not work on most doors, and so was quickly thrown out. Using XBEE modules to transmit the information as radio signals between each side of the door was another idea, but we felt that that might have been a bit too simple for our project, and would have made more analog aspects of our project difficult to produce, as the XBEE's are inherently digital devices. After hearing about hall effect sensors in lecture and taking inspiration from real world NFC technology, we decided to use a system consisting of a coil of wire, essentially a magnetic antenna, as the transceiver on the outside of the door and a hall effect sensor as a receiver on the other side of the door.

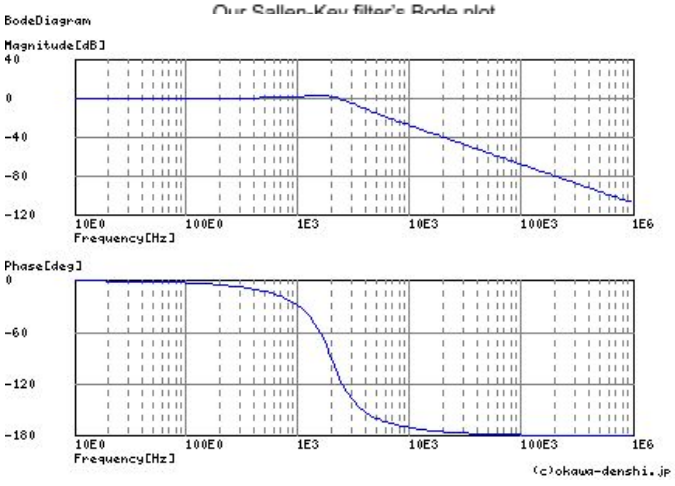
Originally, we had hoped to use a threshold activated hall effect sensor, the Melexis US5881, so that the sensor would produce a full 5 volts if a magnetic field of about 30 gauss or more is detected, and 0 volts otherwise. After experimenting with the coil for a while, we realized that this would likely not be viable, and decided to use the Allegro Microsystems A1301 linear hall effect sensor. This sensor has an output at half the source voltage when no magnetic field is detected, in our case 2.5 volts, and a sensitivity of about 2.5mV/G, which worked much better with the fact that air core coils tend not to produce very large magnetic fields. However, we would no longer just be able to send the signal into the coil, and let the hall effect sensor receiver digitize the signal automatically anymore. We decided to use a digital form of amplitude modulation, called binary amplitude shift keying. We chose amplitude modulation, and particularly this form of it, because signals are easy to modulate and de-modulate this way. Essentially, whenever there is a knock, a carrier signal is produced and runs through the coil, producing the magnetic field detected by the sensor on the other side, otherwise there is no signal produced. This technique modulates the digital signal we want to send to the other side on the amplitude envelope of the carrier signal. The carrier signal is needed, since it is a signal that is more easily transmitted by the coil. We determined a good frequency for this carrier signal by directly connecting the coil to the function generator and the oscilloscope to monitor the output. At about 1kHz or so, the signal was at about it's maximal quality factor; the scope output was somewhat amplified.

To produce a 1kHz carrier signal, we decided to use the LMC555 chip in astable mode, producing a square wave of frequency 1kHz. We decided to use the 50% duty cycle configuration listed in the datasheet, with values of R=5kΩ and C=100nF, to give us a frequency of about 1kHz. Using fourier analysis, we know that a square wave such as the one produced by the 555 is nothing more than the superposition of an infinite number of sine waves at different frequencies. The sine wave we care about has a frequency of 1kHz, so we designed a low-pass sallen key filter with a critical frequency of 2kHz and a quality factor of about 1.25 to help ensure a sine wave of the desired frequency is produced. The Sallen-Key filter was designed with values of R_1=27kΩ, R_2=16kΩ, C_1=10nF, and C_2=1.5nF. Since we wanted to design our project to run off of a single supply, before running the signal from the 555



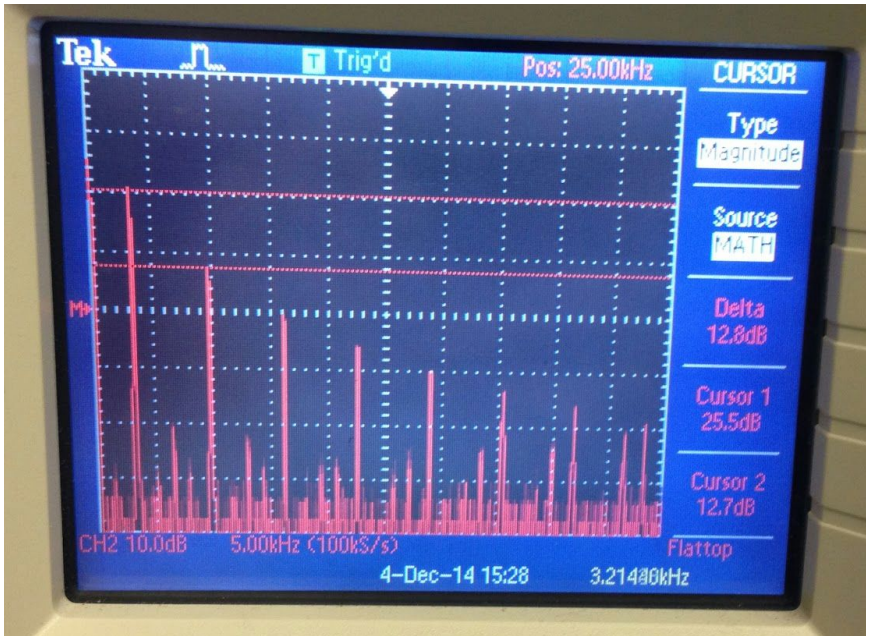
The low pass Sallen-Key filter we used

through the filter, we added enough of a DC bias to ensure that the signal would not be clipped by ground. Since the quality factor of the Sallen-Key filter is greater than 1, it causes some amplification in the signal, so we decided to power the 555 with 5 volts from an LM7805 voltage regulator and the Sallen-Key filter with 9 volts, to limit any potential clipping. The voltage regulator would be used to power all of the discrete digital components, all of the 555 chips, and the piezo amplifier. A yellow LED with a current limiting resistor of 510Ω was later added to show when power was on. Using a voltage divider with resistor values of 10kΩ and 24kΩ, combined with a capacitor of 10nF connected in series to the divider's output, we added a DC bias of: $9V * \frac{24k\Omega}{10k\Omega + 24k\Omega} = 6.353V$, using the scope to determine to ensure calculated values would work.

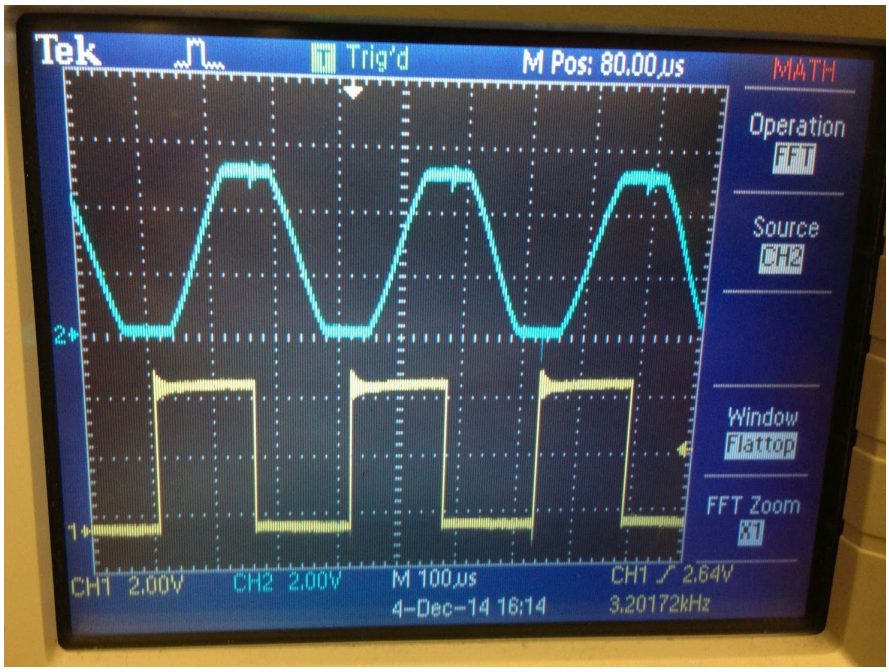


After the gain stage of the Sallen-Key filter, there still was not enough current flowing through the coil to produce a large enough magnetic field. To remedy this, we used the class B power amplifier circuit from Lab 2, using pnp and npn bipolar junction transistors, the MJE2955 and the MJE3055, respectively. To eliminate any crossover distortion, we used the strategy in Lab 2 of running the power amplifier's output as negative feedback into an op amp. Then we eliminate the DC bias we added earlier with a 10μF capacitor in series with the inductor. We also added flyback diodes, 1N400's, between the emitter and collector of each transistor to protect against inductive voltage spikes from the coil. Originally, we used the monostable pulse produced by the piezo signal to control the gate of an nMOS transistor, connecting the end of the coil to ground with a pulse, transmitting the signal. However, we modified this design because of problems with the voltage through the coil going negative, something not good for the MOSFET, and instead connected the monostable pulse to the reset pin of the astable 555 chip. This causes the monostable knock sequence to become modulated on to the carrier signal from the Sallen-Key filter--binary amplitude shift keying.

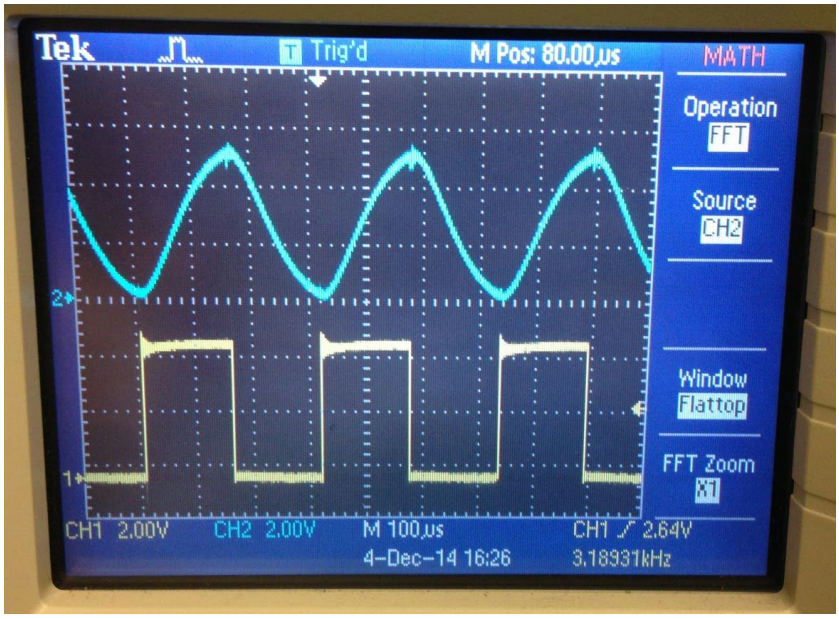
This subsystem ended up working perfectly after an arduous journey to learn about and understand how to utilize amplitude modulation. A knock would trigger the monostable, which would allow the astable LMC555 to activate, producing a carrier signal for the duration of the monostable pulse, which the coil was able to successfully transmit to a hall effect sensor.



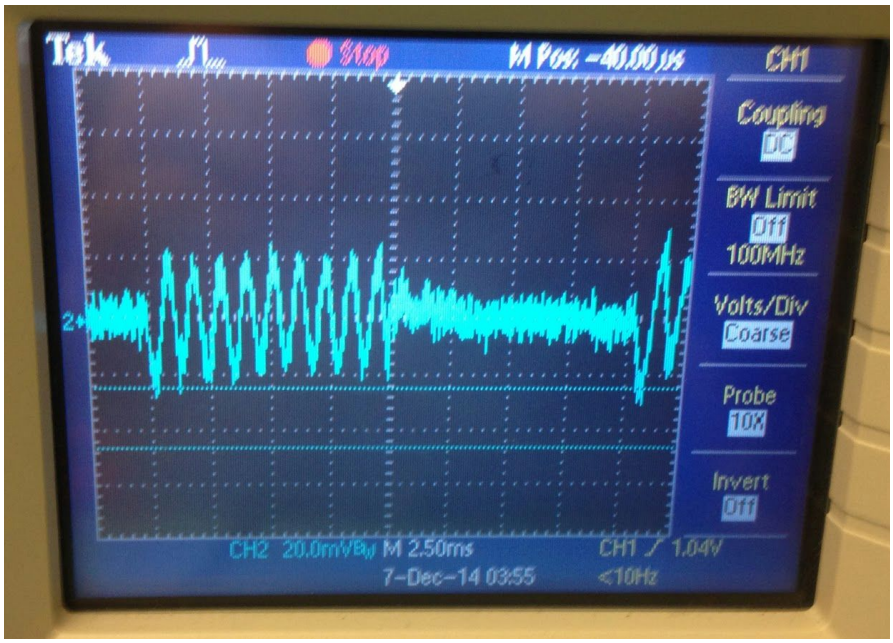
The FFT of the LMC555's square wave output in astable configuration. Note that this signal is just the combination of a number of harmonics at increasing multiples of the square wave's frequency. Filtering out the higher order harmonics leaves us with a harmonic of the desired frequency.



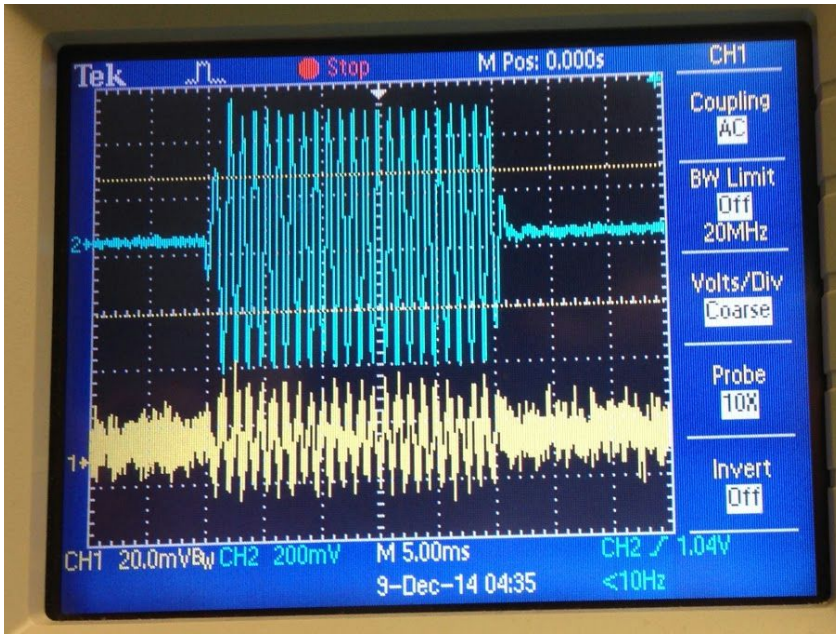
When both the op amp and LMC555 were powered off of 5 volts, clipping was present at the maximum and minimum values of the sine wave. Yellow is the 555 output, and blue is the DC biased and filtered signal.



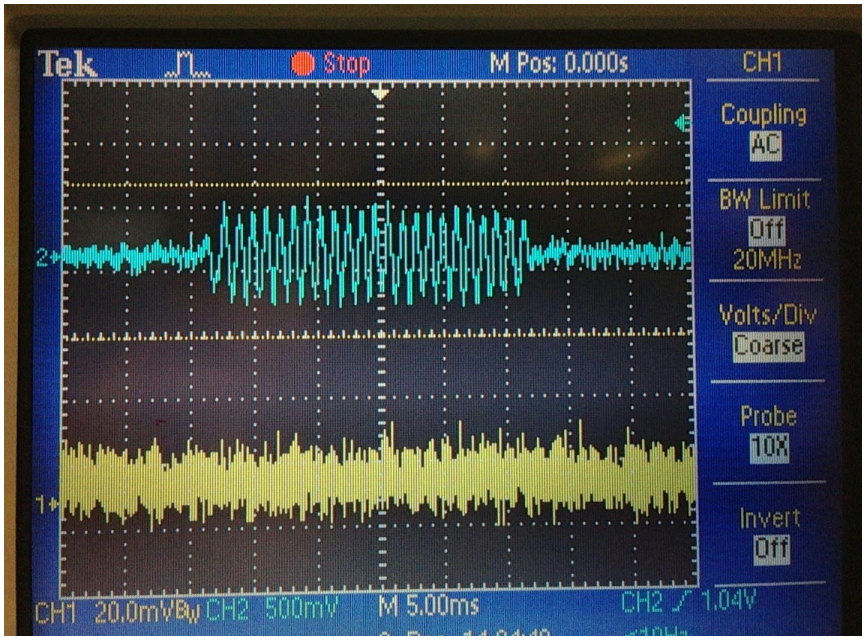
The final result of the filtering was a near perfect sine wave of the same frequency as the 555's output. Yellow is the LMC555's output and blue is the DC biased, filtered signal.



The amplitude modulated signal, as the AC coupled output from the hall effect sensor. The carrier signal was at 1kHz and the modulating signal a 20Hz, 0 to 5 volts, 50% duty cycle square wave signal from the function generator.



Yellow is the output from the hall effect sensor, and blue is the signal after being amplified. Both are AC coupled. The coil was fairly close to the sensor for this test and the modulating signal was provided by the function generator at about 40Hz.

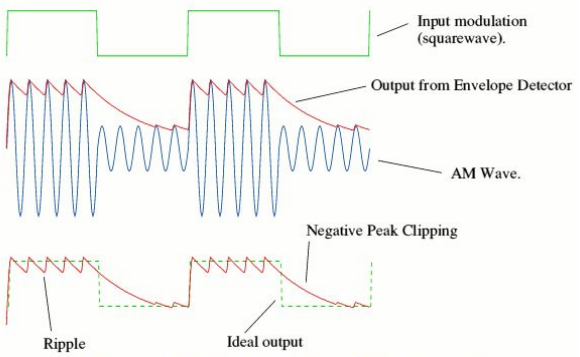


Yellow is the output from the hall effect sensor, and blue is the signal after amplification and filtering. Note that the original signal is very weak, virtually indistinguishable from noise on the scope. This test was performed at a distance to simulate the average thickness of a door, about 3.5 cm.

Hall Effect Sensor and Receiver:

To receive the data being transmitted by the coil, we decided to use a hall effect sensor to detect any magnetic fields produced by the current flowing through the coil. We settled on a linear hall effect sensor with a sensitivity of 2.5mV/G. The magnetic field of the coil was weak, particularly at a distance simulating the thickness of a door. We measured the current through the coil with a multimeter to be about 32mA, which using the Biot-Savart Law, with a coil radius of about 1 cm, and assuming there to be about 1000 turns, leads to a magnetic field at the center of the coil to be about 20 gauss. A distance comparable to door thickness creates a field of about .41 gauss. To solve this problem, we ran the hall effect sensor output through two gain stages with filtering, similar to the design used in Lab 5. In both cases, the low pass filtering had a critical frequency of 500Hz, and the high pass a critical frequency of 2kHz. The first gain stage had a gain of 100, and the second stage had a gain of about 33. In both cases we used the inverting amplifier topology because the signal's cumulative gain would be positive. We separated the gain stages in order to get a better idea of how much gain would be needed, and to avoid the possibility of the op amp acting as a low pass filter, potentially attenuating the desired signal. The non-inverting input of the op amps is connected to a pseudo ground of 2.5 volts created with a voltage divider using two 10kΩ resistors connected between 5 volts and ground. The filtering on the op amps helped to keep noise down, and also ensured that any slight impreciseness in the value of pseudo ground caused by resistor tolerances was not amplified, but rather filtered out. We had quite a bit of trouble identifying the problem when the signal would seem weaker, not realizing that a small impreciseness in the value of the psedo ground would become amplified several times over.

After amplifying the signal, it was sent through an envelope detector constructed from a small signal 1N4148 diode, a 470kΩ resistor, and a 100nF capacitor. This envelope detector demodulates the signal to recover the binary sequence of knocks and no-knocks. The value of the capacitor and resistor create a time constant of 47ms, which is well between the frequencies of the modulating and carrier signals, creating a relatively clean output. The diode causes the output to follow the wave until it peaks, and then the RC circuit causes the signal to fall off with a time constant of 47ms, repeating until the end of one of the signal pulses.

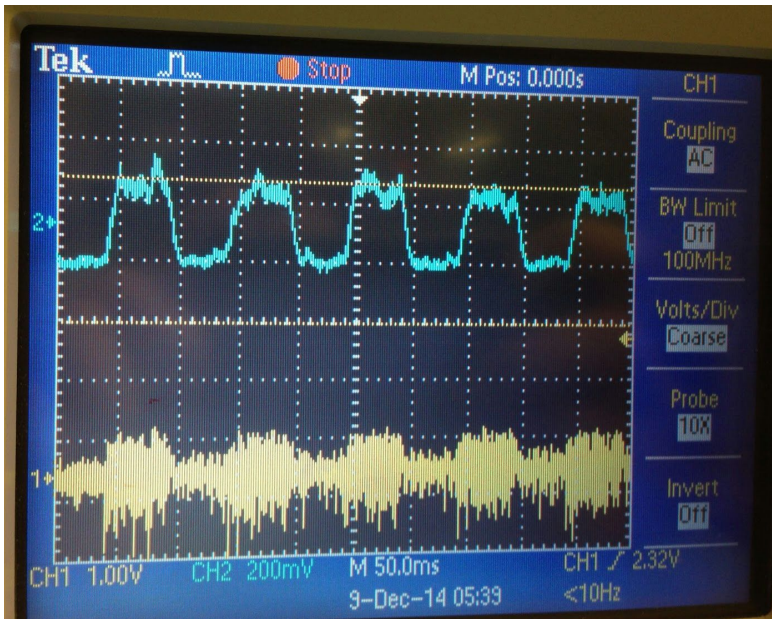


How an envelope detector works

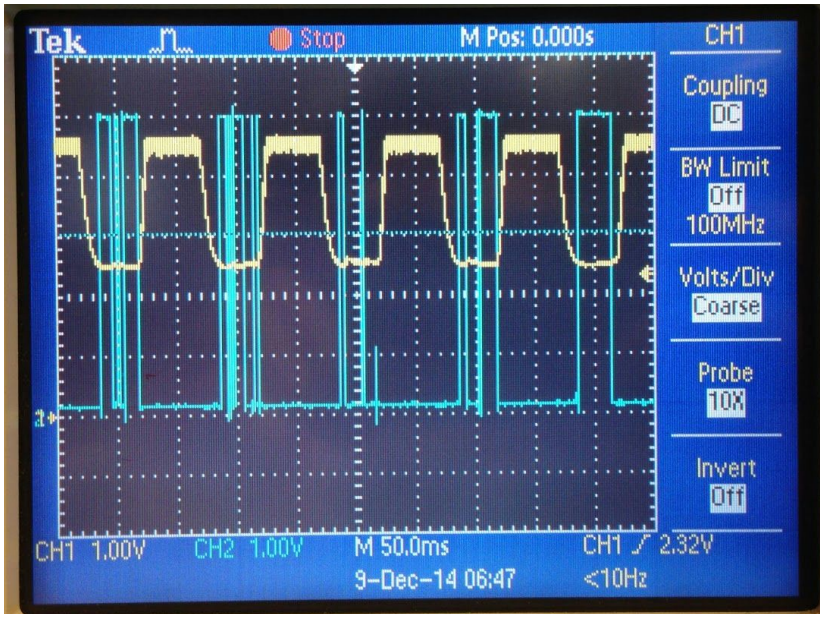
To clean up the signal, amplify it to the full 0V to 5V, and in a sense digitize it, we ran the output of the envelope detector through an LM311 comparator in the inverting topology, with 10% hysteresis from the output of the device, using a 100kΩ and 10kΩ resistor connected as a voltage divider between the output of the LM311 and pseudo ground. Since the output of the comparator was inverted, which was done to provide hysteresis, we ran the output signal through a NOT gate to invert the signal to its correct orientation. We could have done this on the Arduino, but chose not to, in order to integrate some more discrete digital logic into our project.

After the inverter, the signal is connected to one of the digital pins on the Arduino, for our software to parse. We also connected a green LED through a 300Ω current limiting resistor to the inverter output to show when the Arduino received a knock.

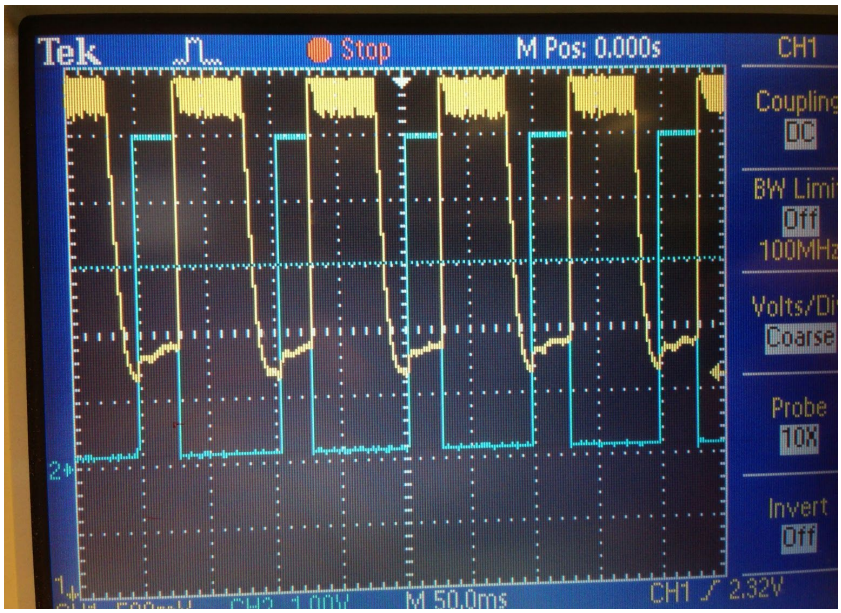
This system ended up working perfectly after a few problems with inaccuracies in the pseudo ground becoming amplified a ton, unintended noise caused by errors with the filtering, and comparator problems due to a lack of hysteresis. The hall effect sensor was able to successfully receive incoming signals from the coil, and the rest of the system was able to successfully process that signal enough to recover the original modulating signal, and feed it into the Arduino.



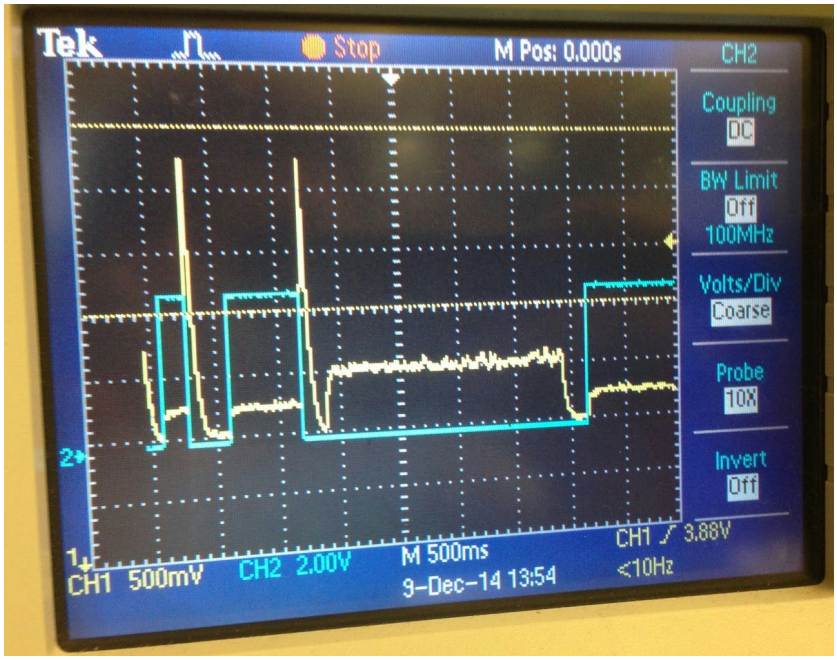
Yellow is the received signal after being amplified, blue is the output of the envelope detector. Both signals are AC coupled, and have a DC bias of about 2.5 volts.



Yellow is the output of the envelope detector, blue is the output of the comparator before adding hysteresis. There is clearly some noise that causes the signal to fluctuate around 2.5 volts.



Yellow is the output of the envelope detector, and blue is the output of the comparator after hysteresis was added. This is the same set of signals as the previous image, but with hysteresis added.



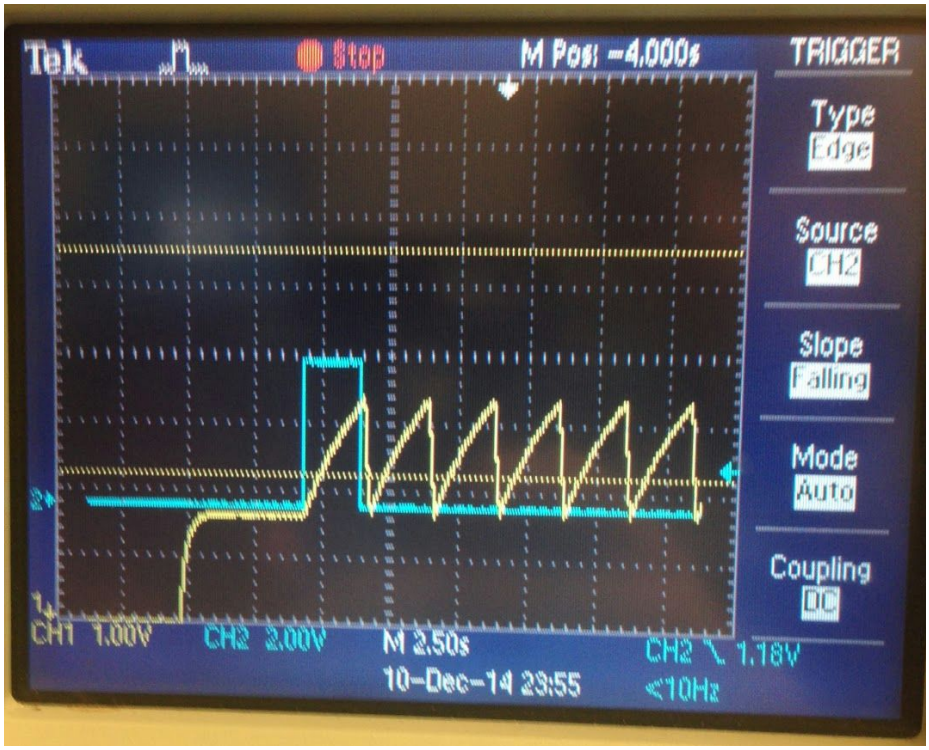
Yellow is the output of the envelope detector and blue is the output of the comparator. The first two spikes were noise that was present before we fixed some mistakes present in our filtering. The plateau in the middle is a knock signal being transmitted completely. The comparator output is inverted, so we ran it through an inverter.

Limited Visual Feedback:

To add some more discrete digital logic components to our project and to add a feedback system for user input, we decided to create a system that would blink an LED every time the user could input new data--knocking or not knocking, since the monostable from the 555 chip will only produce a new pulse after the end of the current pulse. To implement this system, we used an astable LMC555 configuration with a duty cycle of 90% and a frequency of about 0.45Hz, corresponding to the pulse from the piezo-monostable signal. We used values of R_A=240kΩ, the resistor in series between discharge and V+, and R_B=30kΩ, the resistor in series between discharge and threshold, for the resistors and a capacitor value of C=10μF to achieve the desired signal. We used the output of this 555 chip to power the clock of a 74HC161 counter, causing it to increment about every 2.2 seconds. We also connected a blue LED with a 510Ω current limiting resistor to the 555's output, causing it to blink about every 2.2 seconds, prompting the user for input. We did not want the counter and LMC555 constantly running, so we connected one of the counter's enable pins, and the 555's reset pin to the output of a 74HC175 D-flip flop. The flip flop's clock was connected to our piezo-monostable signal, and its input was tied high, so that anytime the user knocks (activates the monostable), the flip flop's output goes high, and enables the 555 chip and counter. To ensure the 555 only outputs 8 pulses, corresponding to the 8-bit knock code, we connected the Q_D and Q_B outputs of the 74HC161 counter to a 74HC10 triple input NAND gate, tying the unused input high. When the counter reaches 1010, the NAND gate output goes low, causing the flip flop's state to go low,

since the NAND gate output is connected to the flip flop's reset pin. The flip flop's transition low resets the counter, since the 74HC161 has asynchronous clear, and stops the 555. Once the entire subsystem was working, we noticed that the first LED pulse would always be much longer than the rest, throwing off the entire synchronization between the monostable subsystem and the counter/feedback subsystem. After some investigation with the scope and assistance from David Abrams, we discovered that the problem occurred because the capacitor needed to charge up from 0 volts for the first pulse and from one third of the source voltage on the rest. To solve this issue, we connected a small signal pMOS, ZVP3306, between 5 volts at its emitter and a 60kΩ resistor between its collector and the non-ground terminal of the capacitor. This MOSFET keeps the capacitor at one third the source voltage once powered on, and by connecting its gate to the reset pin of the 555, the MOSFET is deactivated once the 555 is activated, allowing the capacitor to function normally.

This system worked perfectly, providing exactly the needed visual feedback to make it relatively easy to enter a sequence of knocks. A strange thing was that 90% duty cycle for the LMC555, means that the pulse is high for 90% of the time rather than off, so the blue LED is on, and blips off for a very short time every cycle.



Yellow is the voltage across the counter's LMC555 capacitor, and blue is the monostable pulse from the monostable configured LMC555 chip. Note that the capacitor charges to one third the source voltage after power up, and then oscillates between one third and two thirds the source voltage during operation.

Servo:

The servo was a simple device. The only design considerations that went into the servo involved the type of servo chosen. The servo we finally decided on was a SM-S4304R. This could provide the necessary force needed to open a door handle (servo has a torque up to 71 oz-in). The servo also had continuous rotation. This meant that we would not need to worry about having the servo rotate the door with only 180 degrees of rotation available. All of these decisions were mechanical in nature, but necessary for the ability to open a door.

Arduino Code:

The Sparkfun Pro Micro 5V/16MHz was used for this project. It had the needed interrupts and was available in the lab, which made it an optimal choice. The code needs to take in a monostable output with some leeway of time (for the user to enter another knock or no knock) before it reads in another bit of data. This time delay was estimated to be around half a second, but to be on the careful side, we gave a full second for the user to input the code. If the user knocked before the second was up, then the code would shift in a 1 to the value. If the user took longer than a second, then n number of 0s would be shifted in followed by a 1 (then n 0s depends on how many clock pulses were waited before another knock was entered). The way the code worked was by using an input value, which would be updated whenever the user knocked. This value would be compared to a preset value in the code. The value could be updated by either reading in the analog input every clock cycle, or by using interrupts. The benefit from interrupts is that the time it takes for the user to input the knock is not compounded over the entire passcode sequence. Using this method, the code is updated whenever an interrupt is activated, and if the input value matches the passcode then the servo will be activated.

The code has worked for many of our tests and demonstrations. Using the serial monitor, we were able to look at how the input value changed with each knock. The value would consistently shift out as expected.

Construction Techniques:

Two separate sections are needed for the construction of this project. One is the receiver and requires only one breadboard. The microcontroller is placed at one end of the breadboard, and the hall effect sensor at the other. The middle area is needed for the envelope detector, amplification, comparator, etc. The servo requires three rows near the microcontroller, but that should not be a problem to fit in. Note that this breadboard requires a ground bus and a pseudo ground bus.

The second part is the transmitter. This requires two connected breadboards. The piezo-electric sensor should be on one end of the first breadboard, while the circuitry required for the amplitude modulation spans the other end of the first breadboard. The second breadboard is for the counter and visual feedback subsystem. Note that for these breadboards one bus is needed for 9V and another is needed for 5V.

How to Use:

This mechanism is simple to use. What is required is the ability to knock and to watch LED lights go on and off. To start, if the power switch is off, go ahead and flip that on. Then, whenever you are ready begin to input the knock sequence. The passcode set currently is 11011101. This means that for every 1, you will knock, and for every 0 you will not knock. The signal for when the next input is ready to be entered is when the blue light has flickered once and the red light is off. If you are inputting a 0, then make sure you wait for the 2nd flicker of the blue light, which means you waited for the entire duration it takes to transmit an input signal. If a mistake is made along the way, simply start over whenever you feel comfortable. When the final knock is entered, but servo will activate and rotate the door unlocked. It then remains unlocked for 10 seconds before closing.

Conclusions:

Our project was a success in the sense that we managed to complete all of our main objectives and learn a great deal about many new components (inductors, piezo sensors, hall effect sensors, etc.). Every component worked perfectly, transmitting the needed data across with each knock, providing the user with important visual feedback, and parsing the received data and comparing it with already stored data.

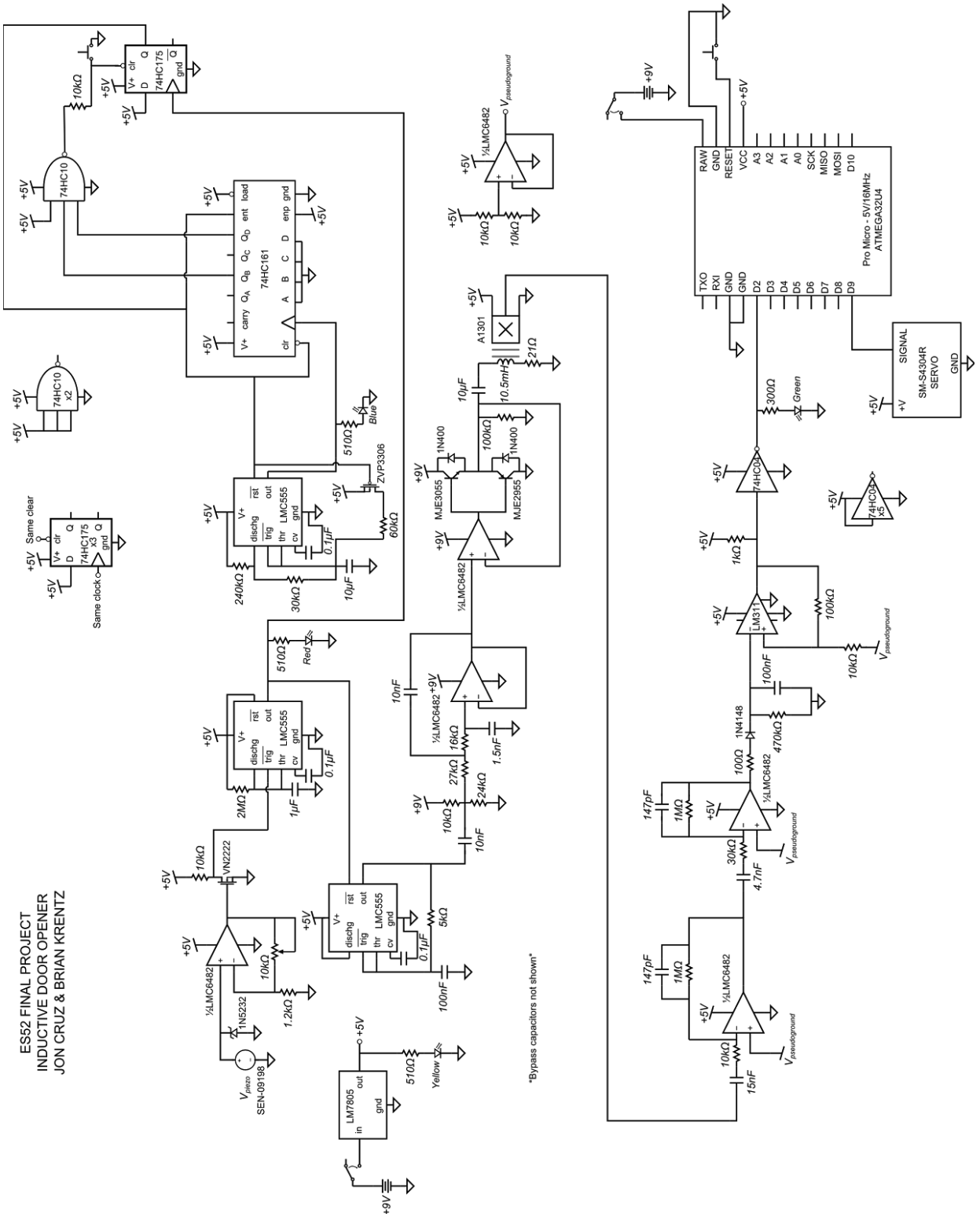
For future improvements, we would like to look into updating our circuit so that the input speed of the knocks would be able to be variable depending on the user. Instead of having a set clock of 2 seconds, the user would only need to keep a consistent tempo as they entered in the information. This could be done by looking at how it was accomplished with morse code inputs. It might also be a good idea to create a second coil to place on the other side of the door, instead of a hall effect sensor. Another coil would allow the circuit on the inside of the door to communicate with the circuit on the outside of the door, potentially allowing the Arduino to convey feedback to the user as they try to enter the code. It would also be nice to implement some sort of power save feature for both the transceiver and receiver, since both draw large amounts of power, the transmitter draws about 150mA of standby current at 9 volts and the receiver draws about 50mA of standby current at 9 volts.

Bibliography

LMC555-Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/Data%20Sheet%20-%20LMC555%20Timer
US5881 Datasheet: http://www.melexis.com/prodfiles/0004824_US5881_rev008.pdf
A1301-Datasheet:http://www.allegromicro.com/en/Products/Magnetic-Linear-And-Angular-Position-Sensor-ICs/Linear-Position-Sensor-ICs/A1301-2.aspx
74HC10-Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/74HC_HCT10_CNV.pdf
74HC161 Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/MM74HC163.pdf
74HC175 Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/MM74HC175.pdf
1N400-Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/Data%20Sheet%20-%201N400X%20Diode.pdf
VN2222-Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/VN2222LL%20B082013.pdf
ZVP3306-Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/data%20Sheet%20-%20P-Channel%20ZVP3306%20Small%20Signal%20MOSFET.pdf
MJE3055 Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/MJE3055T.pdf
MJE2955 Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/MJE2955T.pdf
Zener Diode Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/1N4728-4764.pdf
LMC6482-Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/Data%20Sheet%20-%20LMC6482%20Op%20Amp.pdf
LM311 Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421579.files/LM311.pdf
ATMega-Datasheet:http://isites.harvard.edu/fs/docs/icb.topic1421590.files/Data%20Sheet%20-%20ATMega32U4.pdf

Sallen-Key Filter Design Tool, http://sim.okawa-denshi.jp/en/OPseikiLowkeisan.htm
Envelope Detectors, https://www.st-andrews.ac.uk/~www_pa/Scots_Guide/RadCom/part9/page2.html
Biot-Savart Law, http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/curloo.html
Amplitude Modulation, https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Amplitude_modulation.html
Amplitude Shift Keying, http://engineering.mq.edu.au/~cl/files_pdf/elec321/lect_mask.pdf

Schematic


ES52 FINAL PROJECT
INDUCTIVE DOOR OPENER
JON CRUZ & BRIAN KRENTZ
Bypass capacitors not shown
74HC10
74HC175
74HC161
LMC555
½LMC6482
LM7805
LM311
74HC04
A1301
ZVP3306
VN2222
1N5232
SEN-09198
1N4148
1N400
MJE3055
MJE2955
Pro Micro - 5V/16MHz
ATMEGA32U4
SM-S4304R
SERVO
Red
Blue
Green
Yellow

Arduino Code

```
#include <Servo.h>

// Create servo variable
Servo myservo;
int seconds_open = 10;

// Create variables for code matching
byte value = 0b00000000;                    // current value inputed
int previous_knock = 0;
int current_knock = 0;
byte code = 0b11011101;             // value of passcode
int time_differential = 0;
int clk = 3000;                     // time of pulse + time to hit button

void setup() {
  //servo pin variables
  int servopin = 9;

  // Create interrupt variables
  boolean flag = false;
  int interrupt = 1;

  // Attach interrupts
  attachInterrupt(interrupt, isr, RISING);

  //Attach Servo
  myservo.attach(servopin);
  myservo.writeMicroseconds(1500);

  previous_knock = millis();
}

void isr() {
  flag = true;
}

void update_value() {
  // if td very large then reset value to 1
  if (time_differential > 10000)
    value = value & 0b00000001;
  // shift in appropriate bits to value
```

```
  else {
    value = value << (time_differential/clk + 1);
    value = value | 0b00000001;
  }
}

void open_door () {
  // Code for servo to rotate the door handle, stay open
  // for a few seconds and then rotate the opposite direction
  // to close the door.
  myservo.writeMicroseconds(2000);
  delay(5000);
  myservo.writeMicroseconds(1500);
  delay(1000*seconds_open);
  myservo.writeMicroseconds(1000);
  delay(5000);
  myservo.writeMicroseconds(1500);
}

void loop() {
  if (flag == true) {
    // update time of knocks
    current_knock = millis();
    time_differential = current_knock - previous_knock;
    previous_knock = current_knock;

    // update input value
    update_value();
    if (value == code) {
      open_door();
      value = 0b00000000;
    }
    flag = false;
  }
}
```