111a Final Project

Mike Zhong

Lab Partner: Melissa Hazlewood

14 December, 2015

Abstract

We have studied and replicated the behavior of Cal1ID cards as Radio Frequency Identification (RFID for short) tags. RFID tags communicate over 125kHz radio waves and use both Amplitude Modulation and Frequency Modulation to encode their IDs. We built a simple card reader circuit, and eventually adapted it and installed it to unlock the front door at Cloyne Co-op; we have also prototyped a circuit that replicates the behavior of Cal1ID cards.

1 Introduction

Cal1ID cards and RFID readers are ubiquitous here at the campus of UC Berkeley (Go Bears!) – there are numerous situations in which one scans their Cal1ID card: to unlock certain doors, to transfer meal points upon entering the dining commons, to authenticate access at the RSF, and so much more. For our Physics 111a Final Project, we tried to figure out exactly how Cal1IDs work as RFID tags, and replicate their behavior.

Initially, we prototyped a simple Cal1ID reader using the function generator, a simple circuit consisting of 5 circuit components, and LabVIEW to analyze the signal. The heart of the RFID reader is an LC circuit tuned to 125kHz – the inductor emits an oscillating magnetic field which which inductively powers a Cal1ID card placed next to it. Upon being placed next to our circuit, a Cal1ID card causes interference to the LC circuit, so that the output voltage is an AM-modulated. The frequencies at which the the wave is AM-modulated encode the bits that the card transmitted. It is with this setup we cracked the code the Cal1ID cards use to encode its ID. Later, decoding a card's transmitted ID number was automated with a LabVIEW script.

After news of a working prototype of the Cal1ID reader spread, I was commissioned to build a reader that could upon authentication open the front door at Cloyne Co-op, where I reside. Because it was impractical to install a signal generator, DAQ, and Windows-computer-running-LabVIEW for such a specific task, we decided to use the Arduino Nano, an open source microcontroller board, to drive the 125kHz carrier wave and process incoming card scans. Because the Atmega328 chip is not nearly as powerful as LabVIEW to collect analog samples and take Fourier transforms, we had to filter the incoming signal so that the Arduino would count the number of pulses for each bit's time window to

compute the transmitted frequencies. To transform the Amplitude Modulated output signal into pulses, we used a peak detector to extract the amplitude envelope, a band-pass filter to smooth and center it, and an Op-Amp comparator to convert to peaks and troughs to discrete high and low voltages. C++ code was written for the Arduino to analyze input signals and check authenticity of a scanned Cal1ID card. Upon verification, the Arduino outputs a success signal which enables a transistor that is installed in parallel to an already existing handicap switch that automatically opens the door.

Finally, we tried to see whether we can replicate the behavior of a Cal1ID card. The circuit that allows Cal1ID cards to communicate with RFID card readers through interference is simply a LC circuit tuned to 125kHz and a transistor placed in parallel. When the base and emitter is closed, the LC circuit behaves normally and absorbs much of the RFID reader's circuit, causing a higher load, and ultimately a trough in the AM modulation. When the transistor is turned on, the inductor is shorted, so the card's LC circuit is no longer tuned to 125kHz and thus no longer absorbs much of reader's signal – the AM modulation is at a peak. Much to our surprise, our circuit was fairly successful, and was used to enter the SPS room for snacks when no officers were around to open the door.

2 Card Reader Prototype

Our initial Card Reader Prototype consisted simply of three blocks: the signal generator at 125kHz to drive the circuit, an inverting amplifier that acts as a voltage follower to allow more current to be drawn, and an LC circuit tuned to 125kHz whose inductance is big enough to drive the RFID circuitry in the Cal1ID circuit. See Figure 1.

Figure 2 is the circuit diagram of this circuit. We initially tried to build the circuit without the voltage follower, but it did not work because not enough current could be drawn from the signal generator for a notable output signal. We ended using an inverting amplifier with gain G = 1, resistors R = 100k, for our inverting amplifier. Our reasons for choosing 100k resistors is very similar to many of our later circuit component choices – it was the first value that we arbitrarily tried that worked. With that being said, 100kΩ is a high input impedance, which is desirable for our circuit.

To build our LC circuit, we hand-crafted an inductor by wrapping wire around a 5cm x 4cm x 2cm wooden block around 25 times. We measured the inductance to be 0.090mH, so we picked a capacitor to have a capacitance of 18nF, as

$$f = \frac{\omega}{2\pi} = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi\sqrt{(0.090mH)(18nF)}} \approx 125.044kHz$$

With the LC circuit was in place, it was found that driving the circuit with a 100mV p-p sine wave allowed the nicest signal being produced. As to why higher amplitudes do not work as nicely befuzzles me to this very day.

Without a card present, Vout is simply the result of passing a 125kHz wave through a LC circuit tuned to that frequency. See Figure 3. Once a card is attached, however, Vout is no longer a simple 125kHz sine wave – the card causes interference to the RC circuit, and AM-modulates the wave at roughly

13kHz. See Figures 4 and 5. (A cause for this interference pattern is explained in Section 4, where we discuss building a Cal1ID replicator.)

It is this Vout that we fed into an analog input port of LabVIEW's DAQ. LabVIEW collected 100,000 samples at 1.25MHz, and passed it through a low-pass filter with cutoff frequency 50kHz to remove the 125kHz carrier wave; see Figure 6. It is this signal we tried making sense out of.

2.1 Decoding the AM Modulated Signal

As one can see in Figure 6, there there are two frequencies being transmitted – a sort of binary code. It was by the classical method of using a printout + pencil + ruler we deduced that the transmission rate was 1 bit per 400 microseconds. Each bit was transmitted over a window of 400microseconds, either at 12.5kHz (5 cycles per 400 microseconds) corresponding to a 1, or 15kHz (6 cycles per 400 microseconds) corresponding to a 0. This type of hardware encoding is called binary frequency-shift keying (FSK) [1].

With that knowledge, we wrote a Python script to easily sift through and display 400 microseconds of data at a time so that we could hand-decode the 0's and the 1's. It was after a tedious amount of jotting around 300 bits down were we able to figure out that the signal repeated itself with a period of 96 bits – the transmitted word is 96 bits long in length. Every card we subsequently tested had the same periodicity, so we deduced that this 96 bit wordlength to be a universal property.

Decoding the numbers actually transmitted with these 96 bits was somewhat difficult – we tried searching through the bits to find certain numbers on the card. It was only after realizing that the longest sequence of repeated bits was three 1's, and three 0's, all back-to-back, were we able to deduce that Manchester Encoding [2] was used to add redundancy to ensure accurate transmission. In Manchester Encoding, the original word is doubled in length, by following each "0" with a "1" parity bit, and vice versa. After removing the initial three 0's and three 1's, we were left with 90 remaining bits (starting immediately after "111"). We were able to reduce the remaining 90 bits into nice 45 chunks of "01"s and "10"s. It was in these 45 bits, we were able to find actual values that were printed on the physical Cal1IDs – most importantly, from card to card, only the last 21 bits varied: the first 20 corresponding to the 6-digit code on the back of the Cal1ID, and the 21st simply a parity bit ensuring that the total number of "1"s in the last 21 bits was an even number.

2.2 Automated LabVIEW code

Later, we adapted our LabVIEW VI so that it would do the decoding automatically. After feeding the sampled signal through the Low Pass Filter, LabVIEW would break it into 400 microsecond chunks. For each of the 400 microsecond chunks, it would take a FFT, and compare the amplitudes at the 5-cycles/period and 6-cycles/period bins – if the 5-cycles/period bin had a higher amplitude, the 400 microsecond chunk corresponded to a 1, and vice versa.

Deciding the best phase to start the 400 microsecond chunk was actually a bit of a problem. If by chance LabVIEW's 400 microsecond splits occurred in the middle of bits, some 400 microsecond chunks would contain both 1 and 0 bit frequencies – it would be nearly impossible to decode. Subsequently, our original

LabVIEW script that did nothing about this issue had a success rate of decoding of about 10 percent. To find a better phase, we tried four starting values: 0ms, 0.1ms, 0.2ms, or 0.3ms. We would compare decode around 4 milliseconds (10 bits) of signal to see which of the four phases had most coherent signal (i.e. which had the largest amplitudes in the 5-cycles/period and 6-cycles/period bins) and used that phase to finish the decoding process.

To account for noise, we averaged 10 cycles (for a total of 960 bits, or 0.384 seconds) for the final 96 bit word. After that, we searched for and removed the string "000111", and undid the Manchester encoding redundancy by deleting every other remaining bit, so that 45 bits remained. We converted the second-to-last 20 bits corresponding to the 6-digit Cal1ID code to human-readable decimal, and displayed it. Of the 10 student Cal1ID cards we tested with our complete circuit, LabVIEW was successful in decoding all 10 cards' 6-digit code.

3 Cloyne Card Reader

Once news spread that we have successfully built a Cal1ID reader, my good housemates of Cloyne Court Co-op asked me whether we were willing to implement a card reader to open the front door. We decided to take on the challenge.

To build a practical card reader, we would have to find a way to substitute two of components of the prototype Card Reader: the function generator to drive the LC circuit, and LabVIEW to decode the received signal. We decided to use Arduino Nanos for their cheap cost (on Amazon.com they are approximately $10 per board), capabilities for voltage input and output, and ease of programming. The Atmega328 chip that is in the Arduino Nano is clocked at 16MHz. The Arduino itself runs on 7-12V DC power, and has 12 Digital I/O pins, and 6 Analog Input pins [4]

Initially, we tried using a Square Wave Oscillator circuit found in [3] in place of the function generator. Unfortunately, we were unable to get a consistent 125kHz signal, so in the last minute we decided to use a 2nd Arduino Nano. In our Cloyne Card Reader circuit, instead of driving Vin with a function generator, we had an Arduino Nano output a 125kHz square wave that oscillates between 0V and 5V, and passed it through an inverting amplifier of gain 1/50 so that the inductor would be driven by a square wave between 0V and 0.1V. This was done by modifying R2 of Fig. 2 so that R2 = 2k.

Unfortunately, collecting an analog sample takes around 100 microseconds, corresponding to a sample rate of 10kHz. Our signal has frequencies more than an order of magnitude higher, so we could not simply translate the code on LabVIEW to C++. Instead, we had to filter the signal that the Cal1ID transduced into a digital signal that could be fed into the digital pins of the Arduino. The hardware filter we ended up building extracted the AM envelope from the transduced signal and converted the peak of the wave to 5V, and the troughs to 0V. The Arduino could then count the number of pulses every 400 microseconds – 5 pulses per period corresponds to a "1", 6 pulses a "0".

3.1 Analog Filtering

Our block diagram continues where Figure 1 ends; the Vout of Figure 1 is the Vin of 8. For this circuit, we have also included two Voltage Followers (not

indicated in the block diagram) to ensure no funny business and interference would occur by not having high input impedance and low output impedance.

Initially, we have a peak follower to extract the envelope of the AM modulated signal. We chose values C1 = 10nF and R3 = 2.2k for a decay time, $\tau = (10nF)(2.2k\Omega) = 2.2us$. This is a fraction of the period of the 125kHz wave, $\frac{1}{125kHz} = 8us$ and lead to a nice extracted envelope shown in Figure 9.

We pass this through a band-pass filter, that is a high pass filter with cutoff of 35.37kHz ($f = \frac{\omega}{2\pi} = \frac{1}{2\pi(1.5k\Omega)(3nF)} \approx 35.37kHz$), and a low pass filter with cutoff of 0.013 Hz ($f = \frac{\omega}{2\pi} = \frac{1}{2\pi(14M\Omega)(900nF)} \approx 0.013Hz$). A voltage follower with R = 10kΩ is between the two to prevent strange interference from not having high input impedances and low output impedances. The high pass filter is used to remove the blippy-noise, and the low pass filter used to remove any DC component to center the remaining signal to zero. The resulting signal is rather smooth and sine-like, and is depicted in Figure 10.

We fed this signal into the positive lead of an Op Amp that behaves as a comparator, as its negative lead is grounded and is without any feedback. It outputs -12V for the troughs and 12V for the peaks of the signal. Because an Arduino's digital pins take in 0V and 5V as input, we converted the -12V to 0V, and 12V to 5V using a linear DC voltage transformer [5]. If the final Op-Amp outputs -12V, by symmetry Vout would be 0V. However, if the final Op-Amp outputs 12V, the circuit reduces to a voltage divider, in that the Op-Amp output is equal to the 12V source – the two 27k resistors can be combined in parallel for one 13.5k resistor. Thus $Vout = 12V * \frac{10k\Omega}{10k\Omega+13.5k\Omega} \approx 5.1V$. As depicted in Figure 11, the ultimate Vout is close to a digital signal, oscillating between 0V and 5V.

3.2 Arduino Signal Processing

The Arduino is programmed to constantly listen for signals by counting how many pulses have occurred in the 200 microsecond windows. Once there have been consecutively two or more pulses per window for 20 windows, the Arduino is activated and actively starts recording the signal. It sums 5 cycles of 96 bits into an array of length 192 = 96 bits*(400 us/bit)*(1 sample/200us). We have the sample rate to be twice the frequency of the bitrate to combat the problem of choosing the write phase, discussed in section 2.1.

To decide the best phase, the Arduino reduces the array of length 192 into length 96 by summing every other element. There are two ways of doing this (summing elements 0+1, 2+3, 4+5, ... ; or elements 1+2, 3+4, 5+6, ..., 191+0) corresponding the two possible phases. The Arduino then checks which of the two phases led to the highest variance, calculated as the expected squared deviation from the mean across each one's array of length 96.
$Var = E(X - E(X))^2) = \frac{(X[0]-\bar{X})^2+...+(X[95]-\bar{X})^2}{96}$
The array with a higher variance corresponds to a more matched phase, and is the chosen array,.

Once having chosen a phase, the Arduino computes the average number of bits per window over all 96 bit-windows. Then, each bit-window is converted to a 1 if has a lower number than the average, and a 0 otherwise. This is the extracted 96 bit code.

The Arduino does exactly the same processing to extract the 20-bit, 6-digit

card code from the 96 transmitted bits, as was described in Section 2.2. Once having the 6-digit card code, it checks whether the code is on the white-list, which is stored on an external SD card. If so, access is granted – otherwise, access is denied.

For security reasons, an encryption scheme was implemented so that the one could not easily obtain people's card numbers just by viewing the contents of the SD card. We used salted SHA-256 encryption to encrypt each member's card number. When the Arduino checks whether a received card-code is on the white-list, it performs the same encryption, and checks whether the encrypted hash is stored on the SD card. Upon authentication, the Arduino outputs a signal that is connected to the base of the PNP transistor that acts as a switch. The base and emitter of the transistor is attached in parallel with the already installed handicap mechanical switch, and behaves the same way.

3.3 Installation

One of the biggest challenges of installing this circuit to Cloyne was that the Op-Amps requires +12V and -12V power supplies, and the Arduino 7V. To solve this problem, we used three phone chargers that plug into the wall that converted the wall's AC voltage into a constant DC voltage, two at 12V, one at 6V. We made a common ground by connecting appropriate leads, and used 0.1 microFarad capacitors to attenuate 60Hz noise. We used Ethernet cable for long-distance wiring.

It has been approximately 4 days since installation, and 47 of Cloyne's 120 members have registered their Cal1ID cards. Many of the registered users have shared moments when they thought they were locked out but – I have yet to hear of an unsuccessful report.

4 Card Substitute

We then tried to recreate the behavior of a Cal1ID card. What was most perplexing was how it was possible for a Cal1ID card to cause the AM modulation when being held close to a RFID reader. It turns out, inside each Cal1ID card, is also an inductor. Through Mutual Inductance, it is able to influence RFID readers' inductors. Our Cal1ID card replicator is illustrated in Figure 12. It consists of an LC circuit tuned to 125kHz, a transistor acting as a switch, and an Arduino. The Arduino controls whether the collector and emitter of the transistor is shorted.

If shorted, the circuit looks like Fig. 13. The output of the Card Reader's circuit can be computed by solving this system of differential equations:

$$Vin(t) = L2 * Q'' + M12 * q'' + \frac{Q}{C}$$
$$0 = M21 * Q'' + L1 * q''$$

where Vin(t) is the driving 125kHz signal, M12 and M21 the mutual inductance between the two inductors and are equal, Q is the charge in the reader's capacitor, and q the charge of the replicator circuit. Because q only appears as q" in both equations, we can remove it, to get the final equation:

$Vin(t) = (L2 - L1) * Q'' + \frac{Q}{C}$

Given that L1 = L2, we have that $Vout(t) := \frac{Q}{C} = Vin(t)$.

If not shorted, the circuit is just a LC circuit that has a resonant frequency of 125 kHz – see Fig. 14. The output of the Card Reader's circuit can be computed by solving this system of differential equations:

$Vin(t) = +L2 * Q'' + M12 * q'' + \frac{Q}{C1}$
$\frac{q}{C2} = M21 * Q'' + L1 * q''$

If we assume L = L1 = L2 = M12 + M21, and C = C1 = C2, this is solvable by using the substitutions, $x = Q + q$ and $y = Q - q$. We easily solve for x by subtracting the two equations: $Vin(t) = \frac{x}{C} => x = C * Vin(t)$ To solve for y, we add the two equations: $Vin(t) = 2L * x'' + \frac{y}{C} = 2LC * Vin''(t) + \frac{y}{C}$ Given that Vin(t) is a 125kHz sine wave, and LC is tuned to 125kHz, we have: $Vin(t) = 2Vin(t) + \frac{y}{C} => y = -C * Vin(t)$

Because $Q = \frac{x+y}{2} = 0$ and $q = \frac{x-y}{2} = C * Vin(t)$, we see that $Vout(t) := \frac{Q}{C} = 0$, as the Cal1ID absorbs all of the oscillations.

As was thus illustrated, shorting the LC circuit corresponds to a peak of the AM modulated signal as Vout = Vin, while the not shorting the LC attenuates the reader's signal and corresponds to a trough of the AM modulated signal as Vout = 0.

With a single digital output pin, we output a 5V square wave, whose frequency is either 15kHz or 12.5kHz, corresponding respectively to a "0" or a "1", every 400 microseconds. The resistors act as a voltage divider so the transistor would see at most 2.5V and 5V / 1k = 5mA. The Arduino would cycle through a boolean array of length 96 whose bits were the 96 bits of a Cal1ID card.

We were able to use this circuit a couple of times around Leconte hall. Sam, the GSI, has a nice story about an unexpected visit to the lab room. Also, we were able to obtain lab snacks from the SPS room when no officers were around to open the door.

5 Conclusion

The process of completing this project has taught me a lot in various walks of life. Learning how Cal1IDs work when one scans them to a reader has increased my appreciation for the laws of electromagnetism and security schemes. More importantly, I was able to apply what I have learned this semester with regards to LC circuits, linear filters, Op Amps, and transistors, to build circuits that are actively in use and making the world a brighter place. Cal1ID cards function as RFID tags by interfering with a card reader's LC circuit by AM-modulating its amplitude. The encoded bits are in the frequency modulation using FSK encoding. Using a peak detector and band-pass filter, one may extract exactly the frequency modulation that the Cal1ID communicates. The mechanism by which a Cal1ID card creates this interference is through a variable LC circuit – it absorbs a variable amount of the card reader's signal through mutual inductance, causing AM modulation.

When reviewing this final lab report for quality of the final project, I hope that the Reader understands that none of this material will be used for the

Worser Bad, i.e. illegal activites.

6 Acknowledgment

Must thanks must be given. First and foremost, a big thanks to Joel "Moses" Fajans, whose entertaining lectures and cheerfulness made Physics 111a so much more bearable (See Fig. 15). Big shoutouts to the GSIs, in particular, William, Sam, and Dylan Reese – Ambassador for Peace – for all the long hours put into helping us students. A personal thanks goes to James Martins, for helping drill holes in the wall and setting up the Cloyne Card Reader. Thanks to Roy Tu for teaching me how to adapt C++ code for Arduinos, and to Allan Zhao for inspiration for the Arduino decoding algorithm. Thanks to Mitar Milutinovic and Derek Leung for the helpful conversations about card key encryption. And last but not least, a big thanks goes to Jingyi Li for happening to have 0.1 microFarad capacitors at 1AM the night of the installation.

7 References

[1] https://en.wikipedia.org/wiki/Frequency-shift_keying
[2] https://en.wikipedia.org/wiki/Manchester_code
[3] http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/square.html
[4] https://www.arduino.cc/en/Main/ArduinoBoardNano
[5] http://electronics.stackexchange.com/a/115509

8 Figures

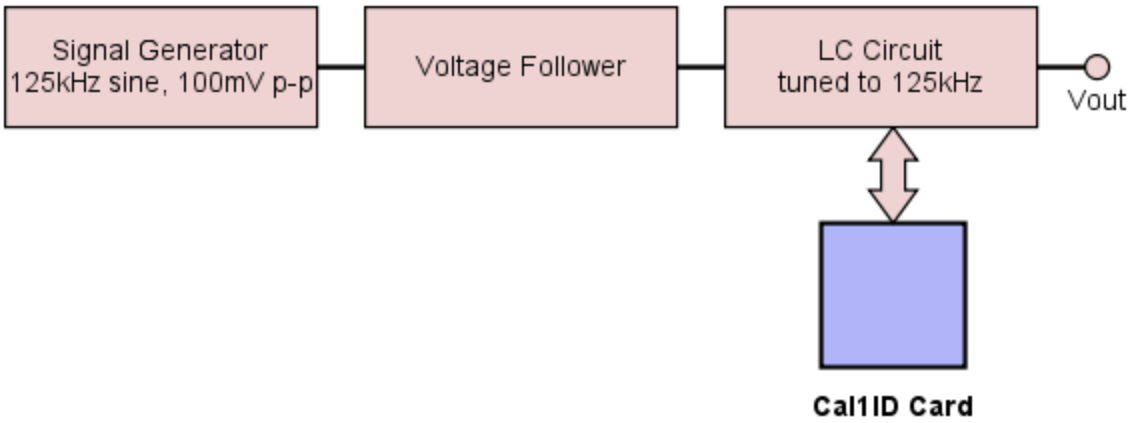

Fig. 1 Block diagram for Card Reader Prototype

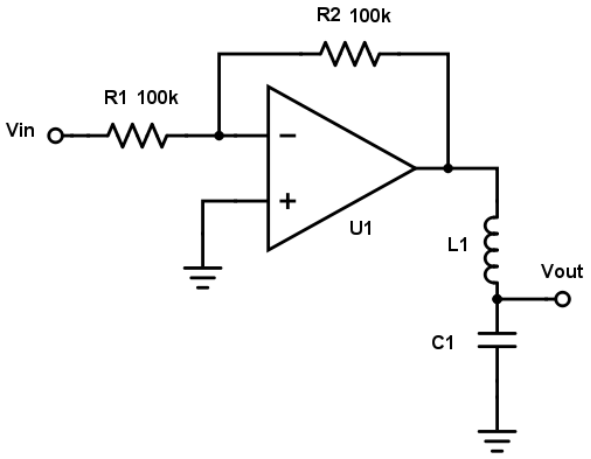

Fig. 2 Circuit diagram for Card Reader Prototype. Attach signal generator to Vin, drive with 125kHz sine wave at 100mV p2p. Inverting Amplifier acts as voltage follower; L1 and C1 chosen to be tuned to 125kHz

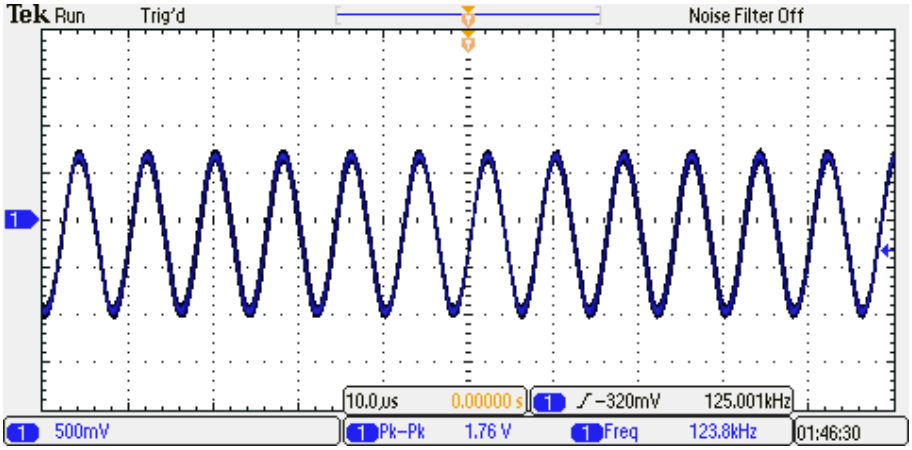

Fig. 3 Vout of Fig. 2 when no card is present

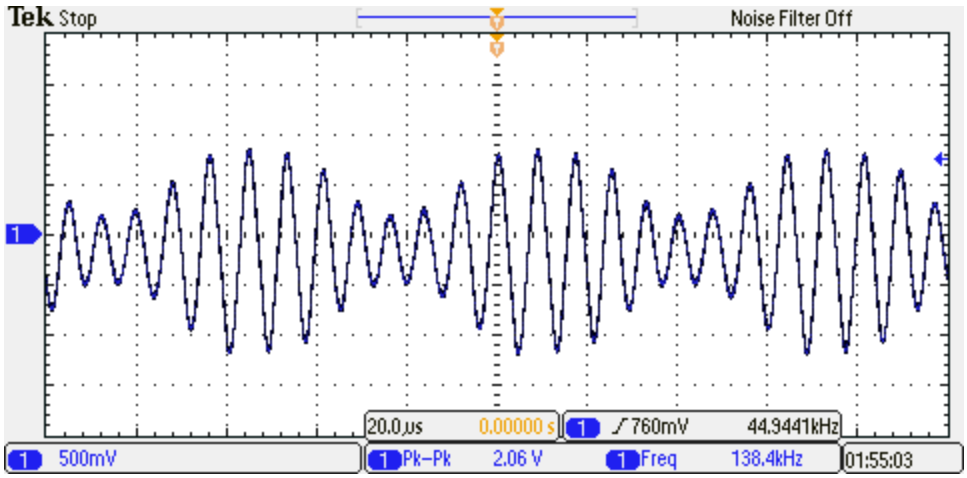

Fig. 4 Vout of Fig. 2 when card is placed on the inductor

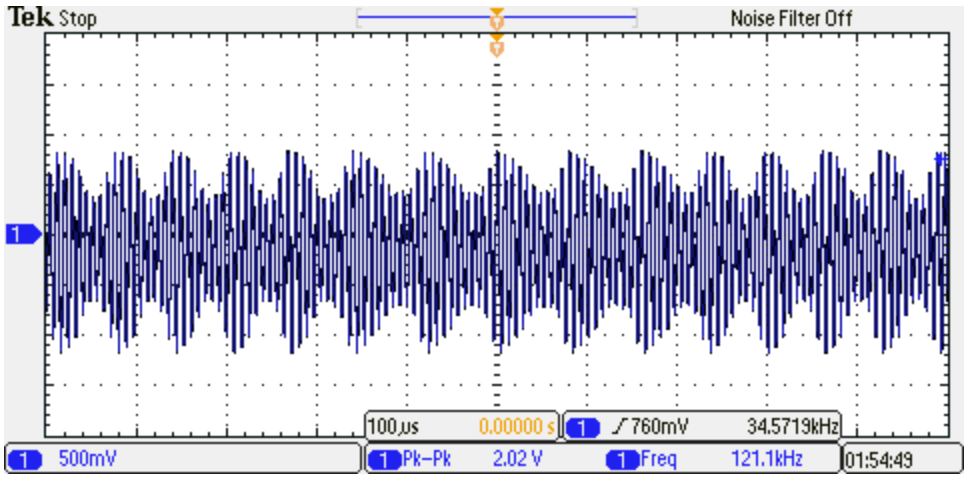

Fig. 5 Zoom out of Fig. 4. Clear low-frequency sinusoidal AM modulation.

[to be attached]

Fig 6. The raw trace after passing Vout of Fig. 2 into Labview and low passing to remove the 125kHz carrier wave. There are two discrete chunks of signal.

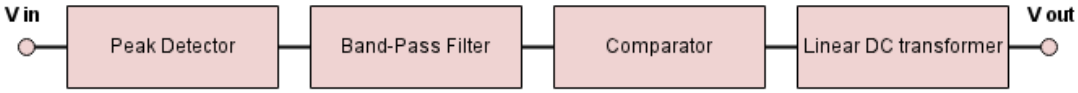

Fig. 7 Block diagram for Cloyne's Card Reader

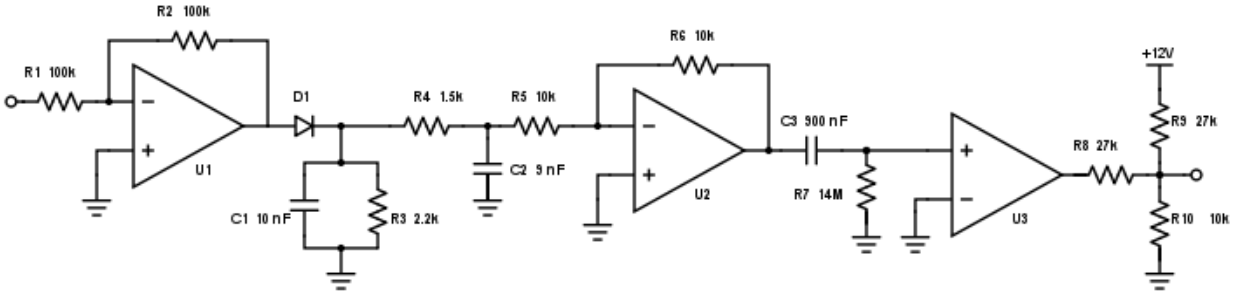

Fig. 8 Circuit diagram for Cloyne's Card Reader. Vin is Vout of Figure 2.
There are voltage followers between Vin and the peak detector, and the LPF and HPF.

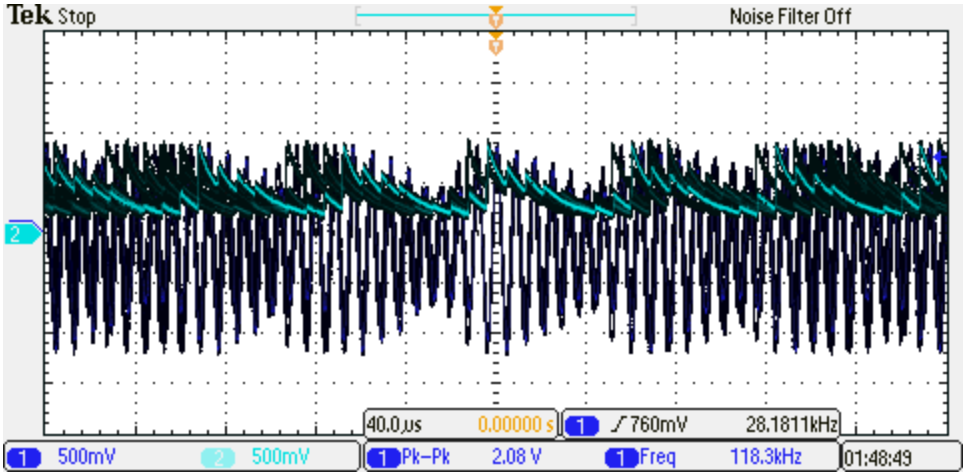

Fig. 9 Signal right after peak detector, compared with Vin of Fig. 8.

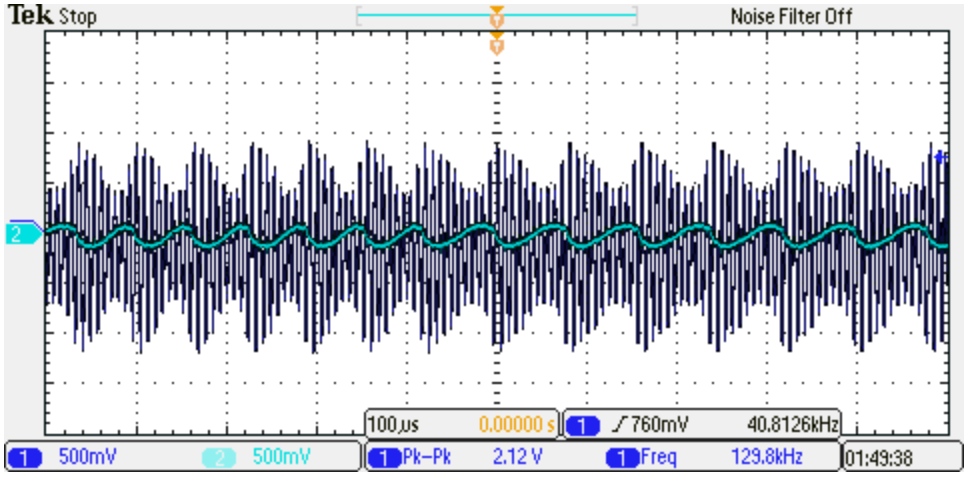

Fig. 10 Signal right after peak band-pass filter, cp Vin

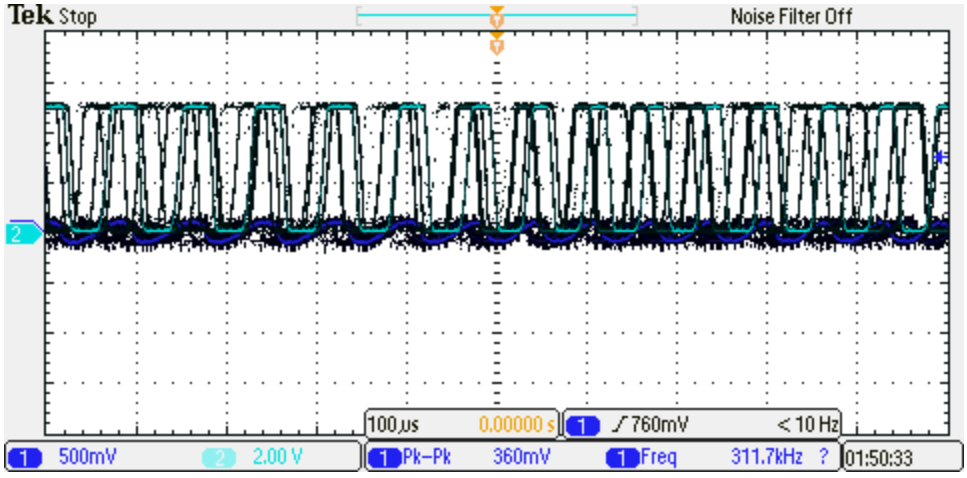

Fig. 11. Final Vout fed into Arduino, cp Signal right after band pass filter

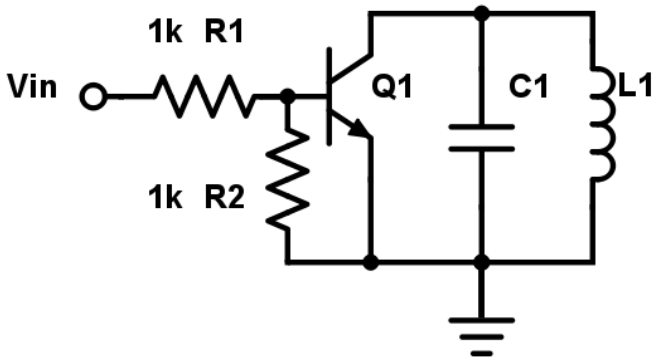

Fig 12. Circuit diagram for the Cal1ID Card Replicator

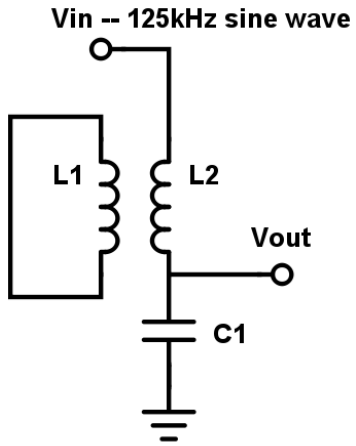

Fig 13. What Cal1ID replication circuit reduces to when transistor is closed, placed next to a card reader.

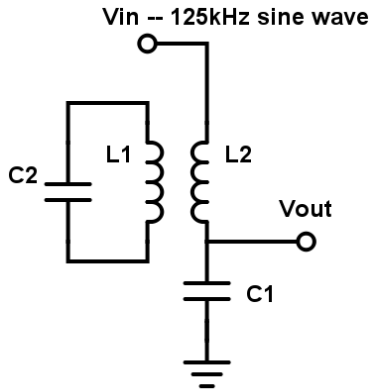

Fig 14. What Cal1ID replication circuit reduces to when transistor is open, placed next to a card reader.

Fig. 15 Holiday Joel and the author