RFRD
Radio Frequency Readout Device

FINAL REPORT

Team: 18

Advisor: Dr Qiao
Client: Dr Qiao

Team Leader: Brandon Baxter (IC Team)
Communications Leader: Luke Myers (IC Team)
Key Concept Holder: Kurt Turner (IC Team)
Webmaster: Vaughn Dorsey (Reader Team)

Robert Buckley, Kellen Yoder, Mehdy Faik,
Aaron Haywood, Michael Miller

Email: May1718@iastate.edu
Website: may1718.sd.ece.iastate.edu

Revised: 4-23-17

Contents

1 Introduction

1.1 Project statement

The goal of this project is to develop an RF sensor system that can send a capacitance value from a passive tag. In an acronym, the goal of our project is to develop a Radio Frequency Readout Device (RFRD).

1.2 Purpose

This project, in just the application for which we designed it, allows us to take a capacitance measurement from a distance. For two metal plates with a uniform dielectric between them, a capacitance measurement is essentially a distance measurement - so anywhere this tag is implanted would let us gauge the distance between two metal plates. The specific application will be connecting an RF tag to each of the bolts on a street-side light post that will include eight capacitance sensors per bolt. It will be used to determine whether the bolts are tight or loose.

In theory the system should be adaptable to take any physical measurement that is expressible as a voltage. This means that our system should still work properly for measuring, say, heat or light level if all we do is replace the capacitance transducer with one for heat or light.

1.3 Goals

Our main goal is to measure capacitance using an IC that will harvest RF energy from a reader to power the tag and sensor, and transmit the data back to the reader. The reader will subsequently transmit this data to either a smartphone or a computer, which will upload the data to the cloud when available. More specific goals for our project are included in the list below:

1.3.1 Goals Accomplished

- Successfully transmit a 13.56 MHz signal across two (square coil) antennas
- Conduct performance tests for our capacitive sensor design on Cadence
- Build a breadboarded IC prototype of our logic circuit and corresponding power rectifier, 13.56 MHz clock counter
- Create data modulator for our breadboarded IC prototype that can operate with a 13.56 MHz signal
- Build a breadboarded prototype of our capacitive sensor
- Transmit modulated data from our logic circuit prototype across two antennas
- Develop a user interface for accessing the data from the Microcontroller and displaying it
- Finalize the capacitive sensor design
- Design rectifier: Source pull and harmonic balance simulation for optimal power transfer
- Implement optimal rectifier through to hardware implementation, both initial test and to account for antenna coupling network.

- Design an IC in Cadence that uses the same principles as the discrete component prototype to test the power consumption of a more realistic finished tag
- Design antennas for 13.56 MHz given practical constraints.

2 Deliverables

We have delivered the following for our project:

- A prototype, custom PCB capable of collecting and transmitting dynamic data through discrete component simulation of an IC
- A prototype, custom PCB capable of harvesting DC power from a 13.56 MHz wave
- A parameterized, straightforward environment for antenna simulation at current and future operating frequencies
- A website containing information and documents regarding our project
- Cadence simulation of a system capable of transmitting a 13.56 MHz wave with the start bits, stop bits, tag ID, and capacitance reader data to test power consumption
- Prototype antenna capable of being used to test signals sent and received by our tag prototype.
- Prototype user interface software capable of storing captured data in the cloud

3 Design Standards

3.1 System specifications

3.1.1 Non-functional

- IC
  - The RFRD tag cost should be minimized (initially <50¢)
  - The tag size should be limited to the scale of millimeters, but is less crucial than cost considerations.
- Reader
  - The size of the reader is inconsequential.
  - The reader does not need to be battery-powered.
  - The cost of the reader is negligible compared to the cost of the tag.
- Antenna
  - Reader antenna should be sturdy and a manageable size (should not need to be carted around, for example)
  - IC antenna should be appropriately durable and small enough to fit in its application (about 4” x 4” max footprint).

3.1.2 Functional

- IC
    - The tag needs to be passive. It must harvest all of its power from the incoming signal from the reader.
    - Tags must include an antenna for signal reception/transmission.
    - Tags must include identification data in the form of nonvolatile memory that can be utilized to uniquely differentiate the tags for each of the bolts on a light post (or a similar method of identification, such as programmed clock delay).
    - Capacitance sensors need to be able to determine whether the bolt is ‘tight’ or ‘loose’ (will readout a ‘0’ for tight or a ‘1’ for loose).
- Reader
    - The RFRD reader must be able to successfully retrieve capacitance data, from the RFRD tag at a minimum distance of 5 meters.
    - The reader should transmit and receive a signal with carrier frequency of 13.56 MHz for the purpose of testing and possibly 900 MHz for the final product in which the capacitance and identification data for each tag will be embedded.
    - The reader must be able to receive the incoming signal from each tag, capture the corresponding capacitance data, and identify which tags are linked with loose/tight bolts.
- Antenna + Rectifier
    - Receiving antenna must capture enough energy to be able to drive the tag.
    - Transmitting antenna must be able to handle enough input power to drive the receiving antenna.
    - Transmitting antenna must have low enough reflection at transmitting power level to avoid damaging reader system.
    - Rectifier must be efficient enough to convert incoming RF power to DC power.

3.2 PROPOSED DESIGN/METHOD

Data Collection:

The main purpose of the project is to collect dynamic data from a capacitance sensor. This data collection would normally be done with a microprocessor and sensor devices in a normal, powered system. Due to power, size, and cost restrictions, we formulated a different strategy. Our sensor tag uses a small amount of energy to charge the capacitor for a predetermined interval to detect whether a capacitor is above or below a set level. This voltage level is read into a simple shift register to be read back to the antenna. Our approach allows for an extremely small circuit footprint, and low cost at scale, but is limited in sensing resolution and has a high initial cost.

Signal Generation:

The 13.56 MHz signal for the prototype will be created with a commercial function generator. We have been working on creating an oscillator and amplifier to generate the signal from our reader as well, but the power from the reader is too low for the discrete component prototype. Due to range requirements, we believe the final design will need an amplified microwave oscillator working at 900 MHz.

Signal Modulation:

Prior to transmitting the data across the antennas, the signal must be modulated. Due to its simplicity, we used amplitude shift key (ASK) modulation to prepare the data signal from transmission for testing purposes. To achieve ASK modulation, a simple RF mixer chip can be used to multiply the data signal with the carrier signal being received from the reader. The output would be a modulated version of the carrier signal where the amplitude is higher for a data bit ‘1’ and lower for a data bit ‘0’.

Signal Amplification:

A power amplifier was designed to boost the signal from the IC detected by the reader’s antenna for the demodulator. As part of the reader, the design is not restricted by power constraints. The amplifier is designed for the prototype, and must be optimized to the final design frequency and power requirements. The current component values should not be expected to function properly on the production model reader. As a class A power amplifier, it can be made to accommodate the higher frequency without changing the basic design.

Signal Demodulation:

A demodulator was utilized to demodulate the modulated signal received from the tag (restoring it to a form where a high voltage represents a data bit ‘1’ and a low voltage represents a data bit ‘0’). The output of the demodulator could then be sent to an Arduino.

Reader to PC Bridge:

An Arduino Mega 2560, powered by an Atmel ATMega2560 microcontroller, will interpret the data bits for the User Interface. The input and output pins of the Mega will be used to interact with custom hardware to send powering signals to, and to receive and decode data from our custom tag. It will use a tethered USB connection and a serial-over-USB connection to receive commands from and transmit decoded data to the user interface.

User Interface:

To allow a user to interface with this system, a simple user interface will be developed that will be able to command the reader to read tags and to receive data from the reader when it has finished retrieving data from said tags. The interface will also be able to store gathered data in a database on a server in the cloud for centralized storage and access to the data. It will be built using Visual C# on a Windows-based Laptop and the datastore running on Microsoft SQL Server, hosted on a remote server.

Rectifier implementation:

We considered and implemented half-wave and full-wave diode configurations with various filter schemes under impedance-matched conditions in simulation. We found that a half-wave rectifier performs consistently better than a full-wave rectifier under optimal impedance-matched and filtering conditions for each. We also found that while an ideal filter between the antenna coupling and the diode served to increase output voltage by over 10%, there were no practical methods of implementing such a filter using real components. Lumped components were too lossy and microstrip lines were too big at our frequency.

Antenna implementation:

At our operating frequency, designing antennas of compact size and favorable S-parameters is difficult. After coming to a conclusion with optimizing antenna simulation performance, near-field operation and low IC power requirements make the task workable. So the problem of finding, an S11 below -20 dB while at 13.56 MHz and enforcing a max footprint sizes on the order of tens of square inches was resolved by low-enough power demands.

Cadence design:

In addition to creating a breadboard prototype out of discreet components we have been working on a single VLSI based IC chip design. Due to being required to use passive RFID technology and the cost non-functional requirements, we decided that a custom IC would be the best solution for this problem. We have created many of the parts of the breadboard prototype on the IC including: charging and discharging lines to test capacitance, reading anticipated capacitance range, a parallel in serial out data shifter, and data modulation.

We are still working on creating a schmitt trigger, memory for the ID and start/stop signal, and bonding pads for the pin out. We have created a rectifier (mentioned above) in a separate program which will need to have as many parts as possible merged with this.

3.3 DESIGN ANALYSIS

Data Collection:

The data collection would be assessed based on speed, accuracy, and precision. The tag should send a signal and receive data within one second to the reader, preferably quicker, to allow for quick, easy operation. Accuracy of measurements would be important to be able to determine if the bolt is loose. With the sub-millisecond repeating measurements, obtaining an average reading should be possible with a one second pulse to the tag. Precision would be important so that the proper tag readings are assigned to the correct tag. This can be done through the ID bits, or imposing a delay on each tag to prevent overlap. Manufacturing and shipping specifications would dictate which method is used.

Modulation:

Proper Amplitude Shift Keying modulation can be assessed by analyzing the signal output of the mixer chip with an oscilloscope. Provided with the output signal of the multiplexor (containing the data) and the RF carrier signal as inputs to the mixer, we should receive a modulated version of the RF carrier signal where the carrier's amplitude is higher when the data signal is high and lower when the data signal is low. By viewing both the modulated and data signals on an oscilloscope, we should be able to see whether or not this is the case. If it is, we can progress to the next step of data transmission across the antennas.

Demodulator:

The demodulator must accurately and reliably output distinguishable "high" and "low" data bits that can be read by the AtMega2560. The signal received by the reader antenna is amplified and filtered. The reader uses a low pass filter as an envelope detector to block the 13.56 MHz carrier wave, passing a waveform that can be read as a high/low data string.

User Interface:

For the reader's user interface, to validate the software's design and test that it is working as designed, there aren't many methods available to ensure the the methods we use are proper. Since the user interface could be developed in several different ways, using different languages and frameworks. The best way would be to simply test the software and see if it functions as intended with the hardware and can send commands, receive data, and connect to the database and store gathered data. Alternatively, though our project is rather unique, the design of our software could be compared to the working software of other RFID-based projects to see if they are

similar in nature or if we might have missed something that needs to be implemented.

Antenna + Rectifier:

For the antenna and rectifier, success is a very simple measurement. After transmission, antenna coupling, and rectification, there needs to be enough energy on a capacitor to power the IC for its own work cycle of data collection and transmission. In this respect the antenna and rectifier are one: If a sacrifice in one allows for a net system improvement by improving the other, then that design can be seen as more successful. From the IC team, we were quoted as needing 80 mW x 20 ps = 1.6 pJ, and the current antenna-rectifier stores 13.22 uJ on a 1 uF cap - more than sufficient.

4 Implementation (Development)

4.1 Interface Specifications

For the RFRD Project, there were a number of interfaces that connected the various parts of the project together. The first interface that the user would encounter is the user interface software, which sends commands to and receives data from the RFRD reader. The software needs to be simple to use and contain enough functionality for workers in the field to be able to do their job efficiently. Additionally, the software needs to be able to store all gathered data on a server in the cloud such that it can be retrieved at another time.

The reader will be an Arduino-based unit and should be connected via a serial-over-USB connection to the PC running the software. A Bluetooth connection will be preferred at some point later in development. This connection will be used to receive commands from the software and to deliver decoded data that is received. This reader is also connected to an antenna, which is used to send power to and receive transmitted data from the IC tag, which is decoded by onboard hardware.

Our custom IC tag will then have two interfaces. The first will be an antenna, which is used to receive power from the reader to power itself and to send gathered data back to the reader to be displayed on the reader interface. The data will be gathered from a capacitance sensor, which is the other interface for the IC tag, the data being dependent on the use of the IC tag.

4.2 Hardware/Software Specifications

IC Tag

The actual product would involve the manufacture of a custom Integrated Circuit due to the size and power requirements. Due to IC manufacture costs and time constraints, we built a PCB prototype to test the theory and application of our design for the tag. The prototype requires external power as a result of the parasitics involved with discrete components and PCB traces. The two revisions of PCB, without

and with the modulator, allow for testing at our 13.56 MHz frequency. The second revision also allows us to input a sinusoidal wave and change the data read rate.

Modulator:

We used an SA602A double-balanced mixer and oscillator chip for the ASK modulator. The SA602A is intended for high performance, low-power communication systems and operates at up to 200 MHz. Its small scale (3.9 mm width) was also ideal for purposes of space efficiency on our printed circuit board designs.

Antennas/Rectifier:

The antennas are square coil strip lines oriented facing each other. They are to be 8 to 16 mils thick with traces 40 to 60 mils wide. The mechanical safety of the antennas is a legitimate concern, both during manufacturing and during operation. In operation there likely will have to be some mechanical installation on the reader maintaining the position of the transmitting antenna. This could just be a dielectric that clamps to the antenna.

The rectifier is a board about 1" x ½" in footprint at most, and can easily be compacted further in case of space concerns. As with the transmitting antenna to the reader, maintaining the position of the receiving antenna and the safety of its attachment to the rectifier circuit is a nontrivial mechanical problem. This is also a task which will likely depend on having some dielectric and clamp system on which to mount the receiving antenna.

Reader:

The reader prototype is designed to accommodate the 13.56 MHz signal using large discrete components. A production reader operating at 900 MHz will require smaller components on a PCB to function properly. The prototype will be powered by a microcontroller to make development easier and more hardware focused, with the goal of transitioning to a Raspberry Pi as the control computer to take advantage of the built in network connectivity for sending the data to a phone or directly to the remote server.

Initially, the plan for the reader software was a mobile-based solution connected to the reader via Bluetooth to display data and store it in the cloud, as this would make the system very portable. However, as the project moved along and more focus and resources were put toward the tag side of development, this created a need to simplify the development of the reader. With this in mind, it was decided to switch to a tethered solution, foregoing the bluetooth connection as it would require more hardware to be added, using a Windows-based application running on a laptop with a USB connection to the reader. This application will have the ability to use the reader to scan for IC tags, interpret and store data in the cloud, and have the ability to view previous results.

4.3 Process

IC Tag:

The initial mock up was done with a breadboard, but the parasitic capacitance of the breadboard caused entirely too much noise to test at 13.56 MHz. Although, we were able to test functionality at lower frequencies. We ordered version A of the PCB prototype in late February, but due to problems during the ordering process, it wasn't initially ordered, and didn't arrive until the first week of April. This board did significantly reduce noise in the prototype compared to the breadboard, and we immediately reworked the design and ordered version B with our additions. This board will allow us to test the entire tag on one board, with options for changing the data read rate to help with modulation and antenna transmission testing.

Modulation:

In determining a proper form of modulation for our project, we spoke with Dr. Sang Kim who said that the simplest form of modulation would be amplitude shift key (ASK) modulation. This would be an easy way to modify the message signal (containing high and low amplitude values corresponding to the capacitance sensor data) for proper signal transmission. This form of modulation can be performed by something as simple as multiplying the message signal with a carrier wave via a mixer chip.We ordered a couple SA602A mixers to test this modulation process with our breadboard IC prototype. Details for testing are included below.

In speaking with Dr. Amariucai, we determined that ASK modulation may prove insufficient for the longer range signal transmission required for our application. While ASK modulation worked for our proof of concept, signal and channel noise will make accurate data transmission difficult with ASK modulation in future developments. A better form of modulation in the future would be binary frequency shift key modulation (BFSK), which will be discussed in Appendix II.

Antennas/Rectifier:

The development process started by seeking to optimize antenna and rectifier performance alone.

Antenna optimization starts with the selection of a general geometry. This geometry should be a) functional, (eg: Not a dipole) b) easy to simulate (Easy to generate a parameterized model for and quickly simulate), and c) easy to implement in hardware (eg: A three-dimensional helix of specific trace width and thickness will create problems).

Antenna optimization then is accomplished by optimizing the two-port S-parameters between the transmitting antenna (port 1) and the receiving antenna (port 2) over a specified channel of space, and tweaking geometric parameters of either antenna.

Rectifier performance is optimized alone by seeking out places where impedance matches and filters or resonators can improve circuit performance after accounting for non-idealities.

After both rectifier and antenna performance have been optimized the systems can be integrated. In the near-field, the antenna coupling can be represented by a two-port impedance network. This impedance network can be implanted into the rectifier's chain of power flow (which leads from the transmitting antenna to the rectifier output). The rectifier can then be re-optimized to account for the presence of the antenna coupling using the same process that was used to optimize it without the antenna coupling. The antennas and rectifier are now integrated, and are ready for integration into the reader (Ensure reader has correct output impedance) and IC (ensure enough energy is on capacitor to run IC).

Amplifier/Demodulator:

The amplifier was designed with discrete components for an output of 20 dB gain. This will generate enough signal strength to provide a readable input for the the demodulator. The demodulator is an LPF envelope detector that can pass the filtered signal as a square waveform to the Arduino for processing.

Reader Hardware:

For development purposes, we decided to use an Arduino Mega 2560 one of our teammates owned as our bridge between the user interface and the analog hardware needed to communicate with the tag. Arduinos are rather easy to program and the Mega provides a lot of I/O pins and built-in analog to digital conversion hardware on the device, allowing for changes to code to be made easily and giving us plenty of room to expand as necessary. It is also easy to communicate with thanks to a Serial-over-USB interface that Arduinos typically provide by default, making it easy to push changes to the device and to send and receive data to and from the user interface host.

User Interface:

The design process for the user interface was largely an ad-hoc, change-as-needed process, due to the user interface software not being a large focus of the project. After gathering the requirements of the software, the first thing that was done was to determine what language it should be developed in. Since the target platform was now for a PC, there were many options available, all with their own quirks and features. After some consideration about what kind of computer would be available in a work truck, figuring it'd likely be a Windows-based machine, it was decided to use C# and Visual Studio to develop the GUI, as Visual Studio provides an easy-to-use GUI editor and C# is a fairly easy language to work with, making the GUI decently easy to change if the need arises. For a data storage backend, since we were already using Microsoft technologies, it was decided to run SQL Server 2016 on a server provided by ECpE's Electronics Technology Group.

Work then began on designing both how the software would look and what the data in the database would need to be. The database schema, or structure, did take a bit of time to flesh out, as there were a couple of trains of thought to consider regarding how to store the data effectively. It ended up boiling down to a table to hold information about locations and then a table to hold bolt information, though this will likely need to change in later versions of the project to accommodate for mass

production and large scale quantities of the bolts. The actual design and layout of the UI was largely based around testing different layouts and seeing if they are easy to work with. The initial thought was to provide a window per function with a starting window having links to open other windows, but this proved to be cumbersome. So, a tabbed window was created to house all the functionality, the tabs making it far easier to extend the software's functionality. Code soon linked the user interface to the database, allowing data to be stored and retrieved.

Finally, the last added portion was connecting to the Arduino to send commands and receive data. This was an easy process, as a serial driver is provided through the .NET Framework that C# runs on, and after some experimentation, yielded a small driver that can send and receive from the Arduino with ease. This driver is also able to receive any amount of data without issues to make changes to data formats easier. What data is received from the Arduino is then parsed out by the program and displayed before being stored in the database, the program doing a lot of the interpreting of the data.

5 Testing Process & Results

IC Tag Prototype:

The IC prototype was initially built on a breadboard, with discrete components, to work out the functionality of the capacitance reading. The extreme parasitics of the breadboard in relation to the capacitor size restricted testing to below about 500 kHz. The circuit would sometimes work at 13.56 MHz but was extremely noisy. We ordered a PCB with the core functionality of the tag to help alleviate these problems.

Before receiving our PCB, we conducted preliminary data transmission testing using a set of simple antennas consisting of three turns of wire that were mounted on two pieces of cardboard that were 10" x 10". We first sent the output data from our IC directly to the transmission antenna to make sure we could transmit data through our antennas. After observing that our simple antennas could transmit a signal successfully, we then integrated the modulator into our system.

We first observed the output of the modulator directly on an oscilloscope and saw that, despite the presence of significant noise, we were able to observe successful modulation of the data signal with the RF carrier wave. Connecting the modulator with our transmitting antenna, we were able to successfully transmit the modulated signal and receive it on the other antenna. We could still distinguish between the high and low portions of the modulated signal on the receiving end of the antennas, though the amount of noise present in the signal made it difficult to distinguish at a range of more than a few centimeters between the antennas.

Although we had ordered version A of our board through the University in early March, there was confusion on their end (as a result of our ordering both a PCB for our IC and a PCB for our rectifier) and the order wasn't placed until nearly a month later. When we finally received our board, it greatly improved the noise to

signal ratio, allowing for testing at 13.56 MHz. Although, it did not include a clock generator or modulator. Version B added these parts, as well as the ability to slow down the data rate to ¼, 1/8, or 1/16 the rate of the incoming antenna signal. These slower data rates are needed to make the modulated signal more readable after transmitting back to the reader.

The revision B PCB was able to read out changes in capacitance to the modulator, which was only marginally able to perform. The modulated data was almost indistinguishable, and debugging this part is still in progress. The limitations of available equipment are also a limiting factor in testing.

Cadence Testing:

The design in cadence was built on the 0.5 micron process. This process is slightly outdated, but is still used in some analog devices. Newer processes would make the circuit more efficient and able to handle the higher frequency better, so we figured this slightly outdated process would be good for our testing.

In order to test our design we created the essential components, the capacitive sensor and the shifter logic in cadence at the transistor level. Once those two main components were complete we expanded to include the logic to control the system, the modulation, and the initial clock. There are still many parts that need to be converted from test components to printable components, but the core of our design is implemented and working according to tests.

In order to test the design we ran transient analysis on the design. We were not able to run the analysis on the full clock division, because it would take thousands of computations, and many hours to complete the analysis. As such, we do not have access to the resources to store such a simulation. So we tested at 1/32nd the division and were able to get good results. The data was read in from the capacitor sensor well and the frequency modulation worked.

User Interface Testing:

Testing for the user interface was largely usability driven, with a focus on making sure that it worked as intended or the current graphical design was intuitive to use. The criteria for determining this was mainly based around if the UI elements made sense to work with, the code backing them did the functions that they said they would do, and any code gathering or submitting data was properly receiving data from the source or was saving the data in the correct fashion.

Because of the focus on the IC portion of this project, the user interface testing was largely done without hardware and using, focusing primarily on testing the design of the UI and the database interactions to make sure they made sense from a higher level perspective and produced desired results. This did end up helping considerably, as it pointed out problems with both how interactions with the UI didn't make sense or needed improvement and pointed out issues with how the database was structured for storing the data. Testing with hardware did eventually occur, which allowed for expanded development of the software to understand the output that was being sent from the Arduino.

Antenna Testing:

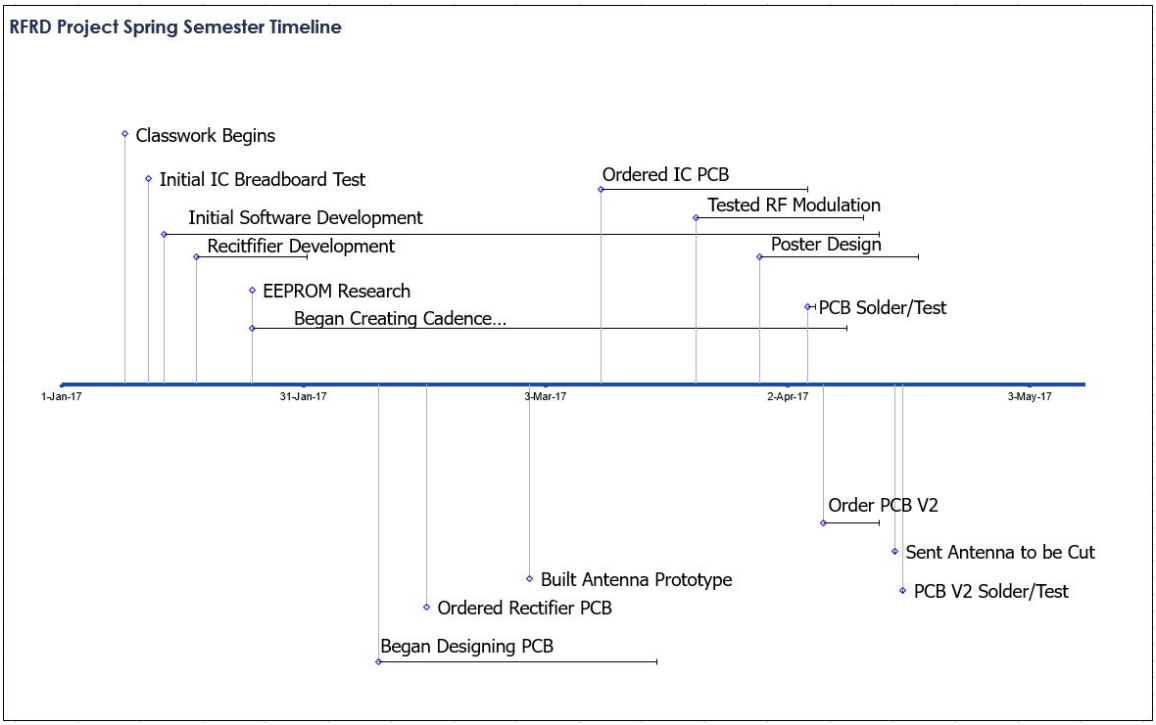
No antenna has been implemented in hardware; only in simulation (besides the simple antennas built for modulation testing as described above). The first design iteration had S11 = -3 dB and S21 = -20 dB. However, it soon became apparent that no practical manufacturing process was available at ISU to cut copper safely to traces 40 to 60 mils in width. The only practical method of implementing the antennas was to mount them on dielectric, which de-tuned the antennas and required a re-design. By the time they were re-designed, it was too late to order a PCB implementation. This re-designed antenna coupling pair has S11 = -2.5 dB and S21 = -25 dB at about 14 MHz.

Rectifier Testing:

The first iteration of the rectifier performed with 10% RF to DC power conversion efficiency. It was designed to match to a 50Ω source input, and demonstrated the use of input impedance match and pre-diode filtering. It culminated research in various areas regarding the function of the rectifier, such as: Reflection at harmonics of operating frequency, modeling the load, and simulation strategies.

The second rectifier, then, was also designed to match to a 50 Ω source, as well as account for the two-port Z-network that represented the antenna coupling. As of this writing it only has testing in simulation. It offers about 1E-5 J of energy storage for 5 mW from the reader - an order of magnitude or two more energy than is necessary to run the IC for one complete cycle.

6 Semester Timeline



7 Conclusions

The final results of this project are not what we would have hoped, but they are around, slightly above, what we anticipated. The project is very complicated and has a few major problems to get past by future groups if it continues. The largest long term problem is that, at the end of the project, this will have to be created on a single integrated circuit. This means a large investment in creating a fabrication mask and then building the IC.

Although the final goal has a long way to go, we have made good headway on developing a working design as a proof of concept. We created a basic design prototype on a PCB that shows how to build the device, and created a cadence design with most of the core components completed and working. The reader will also need further development, but this should be helped by having a prototype tag to work with. Progress has been made on the reader as well in the design of the user interface, oscillator, and demodulator.

Any group that continues this project will need to look into two main issues we currently see with the Cadence IC. First, how to create the rectifier diodes that will be needed for the design, as they need to have a much smaller voltage drop than diodes we normally make in Cadence as undergraduates, and may need to be external in the end. The second problem will be how to generate a square wave from a sine wave for use in the IC. We planned to use a schmitt trigger as a comparator to generate the waveform, but we have been unsuccessful at creating an internally compensated op-amp that will work for this component.

8 Appendices

Appendix I: Operation Manual

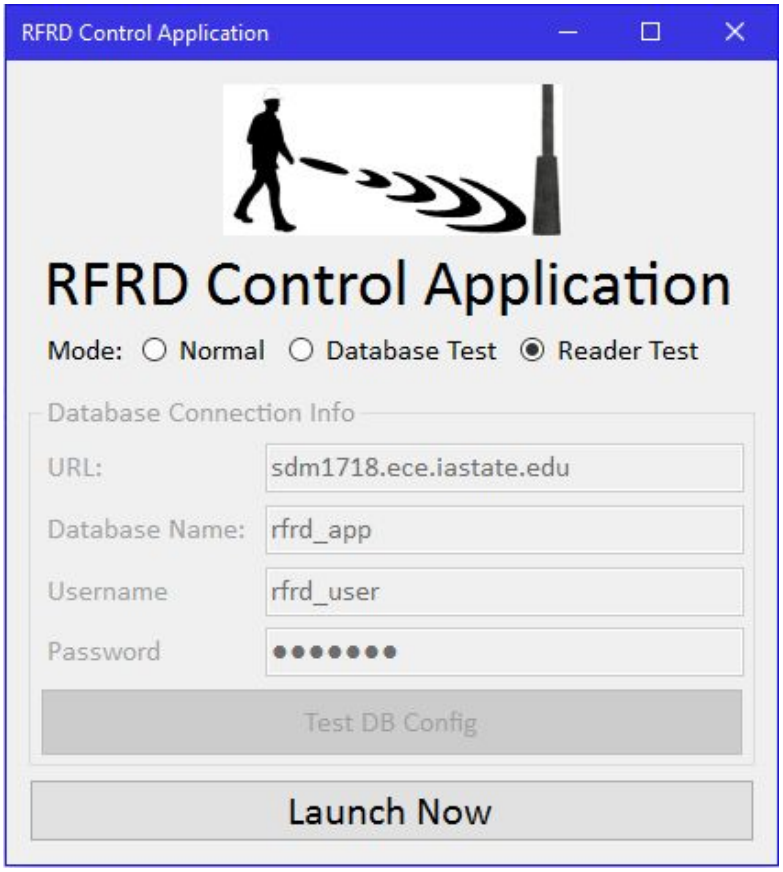
User Interface Notes

The prototype User Interface has three modes that it can operate in. The first is normal operating mode, which will connect to the database and the reader hardware and use data output from the reader hardware as the software is intended to do. The second is a database test mode to ensure the software is able to work with the server it's being pointed to, using random values to act as 'tight or not' values being read from the IC tag. Finally, the third mode is used as a standalone reader testing mode, requiring no internet connection to operate and simply reading out what data the reader is sending back. We will focus on the Reader Test mode for this documentation, as the Normal and Database Test modes require further setup.

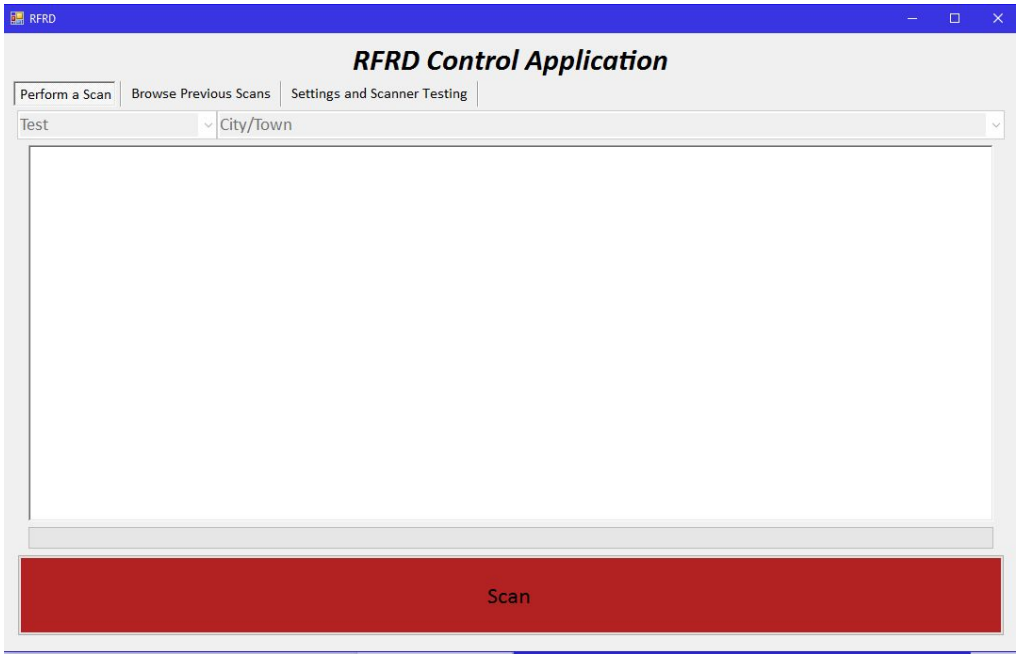
To run the program in mode 3, you will need a computer that is capable of running at least Windows 7 with at least .NET Framework 4.5.2 installed. The .NET Framework is available from Microsoft for free, though it may be installed already. Running the applications installer should automatically install it if it isn't. Additionally, you must have the Arduino hooked up to the computer via one of the available USB ports on the system. Windows should be able to provide a driver for this automatically.

To begin, download the last published build using the link in Appendix IV. It will be located under the Documents tab and should be an archive containing all of the files needed to install the application. Unzip this folder to a convenient location and then run Setup.exe. It will then verify that your computer can run the application and install it. Once the installation is done, the application will open and, after being closed, is available from your Start menu under the name rfrd_control_application. It can be uninstalled through the Control Panel.

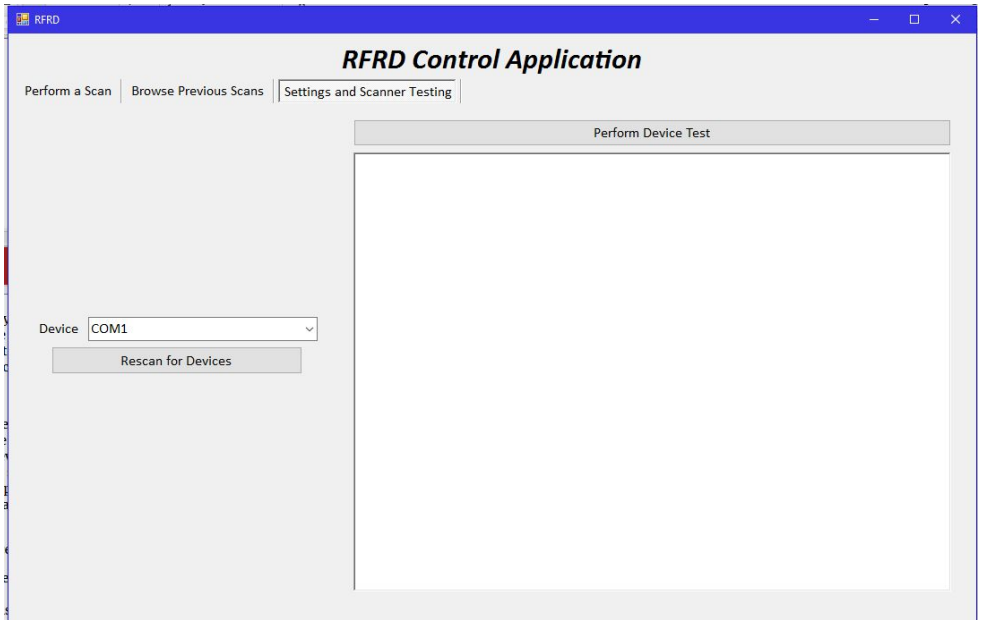
Once the program does start, you will see a prompt like the following. Select the third tab and press the large 'Launch' button to start the application.



The main window of the application will then load. From here, you can access the three main tabs of the application. The first tab, which handles scanning, is the first page that will be shown.

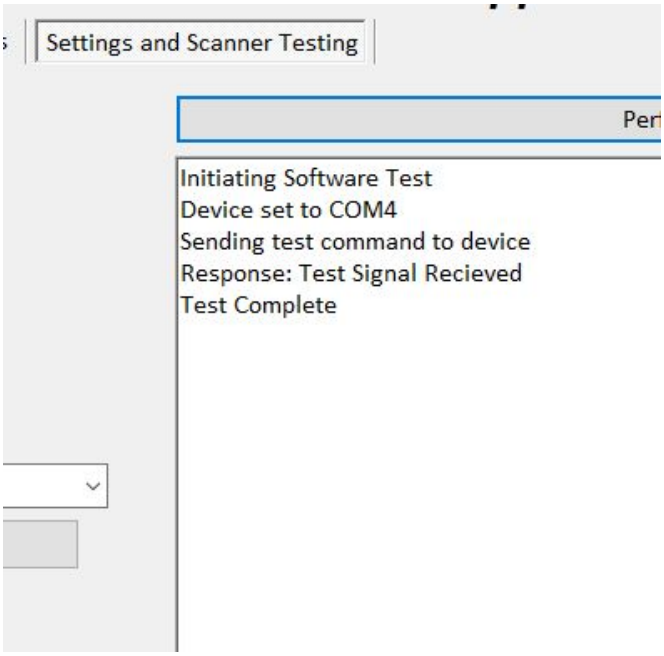


Before scanning, you should verify that the settings are correct so that the program is talking to the right device. To do this, click on the Settings tab. This tab will display settings related to the scanner port being used and a testing interface to check the scanner connection easily.

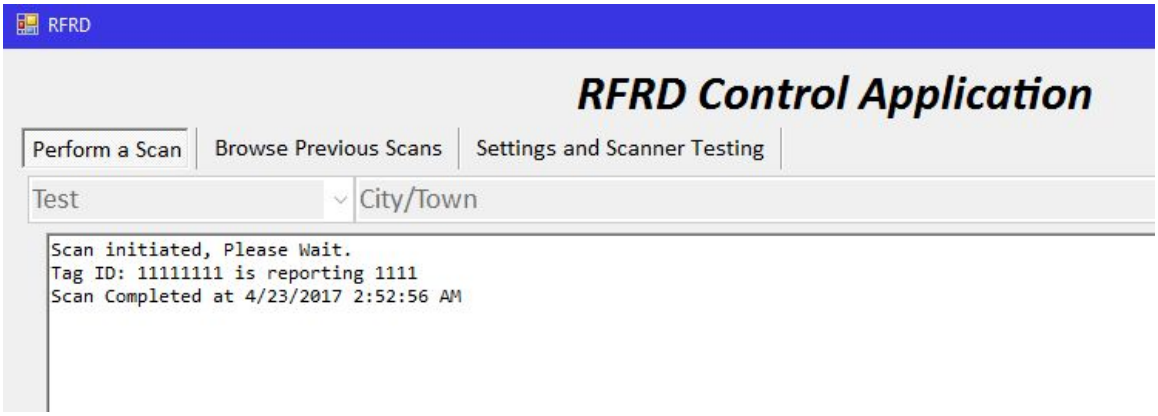


Adjust the COM settings using the dropdown and make sure the testing interface reports that the testing signal has arrived before continuing. Usually, the Arduino shows up as the second COM device, if you only have one device connected to the machine. If none of the COM devices seem to be the Arduino, then check the

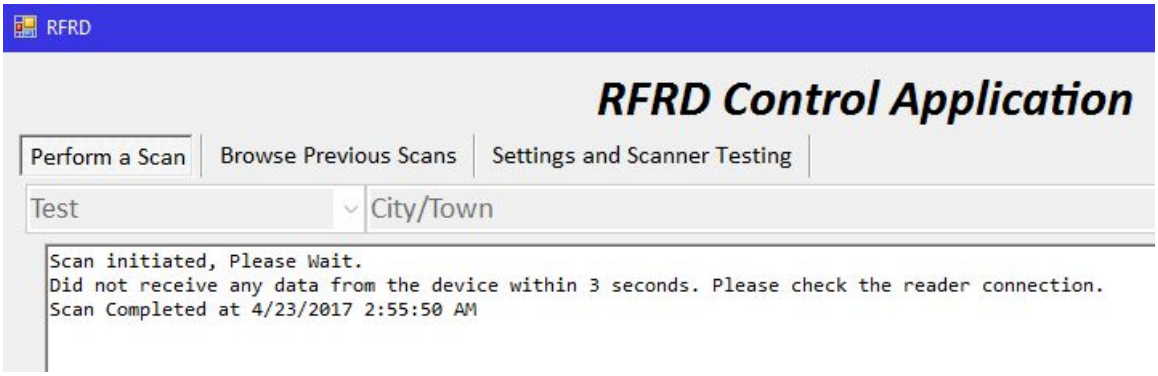
connection with the device and restart the application. If that doesn't work, the Arduino may need programmed. See the Reader section of this manual for information on how to do that.



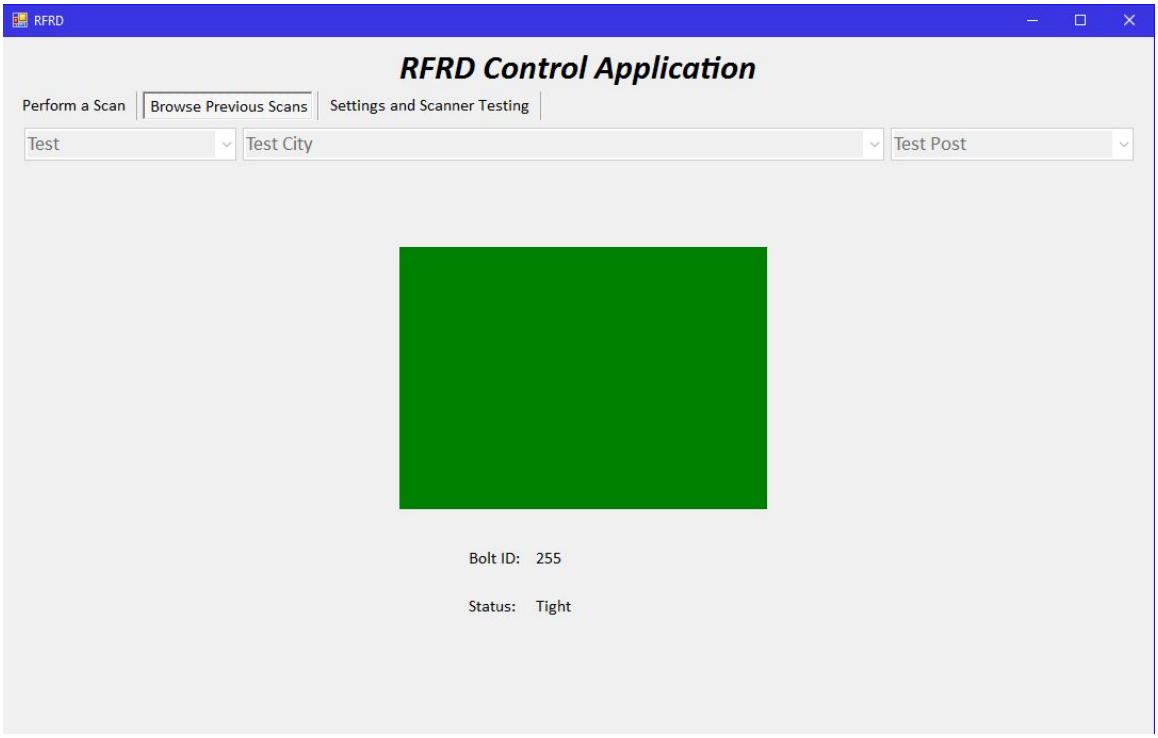
After ensuring the software is properly connected to the device, navigate back to the Perform a Scan tab that you were previously at. Clicking the large, red 'Scan' button below will initiate a scan using the reader hardware. If everything is connected properly, you should receive data back from the device, as shown below.



Otherwise, if something isn't connected right, or the wrong COM port has been set, you will see the following message. Check connections and settings and try again.



After the data has been read successfully using the reader hardware, the recovered data can then be viewed in the second tab, 'View Previous Results'. In the Normal and Database Test modes, you can use this screen to select between cities and states that are available and choose one of the lamps the city has. It will then show a diagram of the bolts, or bolt in the case of the reader test, with each bolt showing it's ID and whether it is tight or loose, using both a color (green for tight, red for loose) and text. In the Reader Test mode, these choices are disabled.

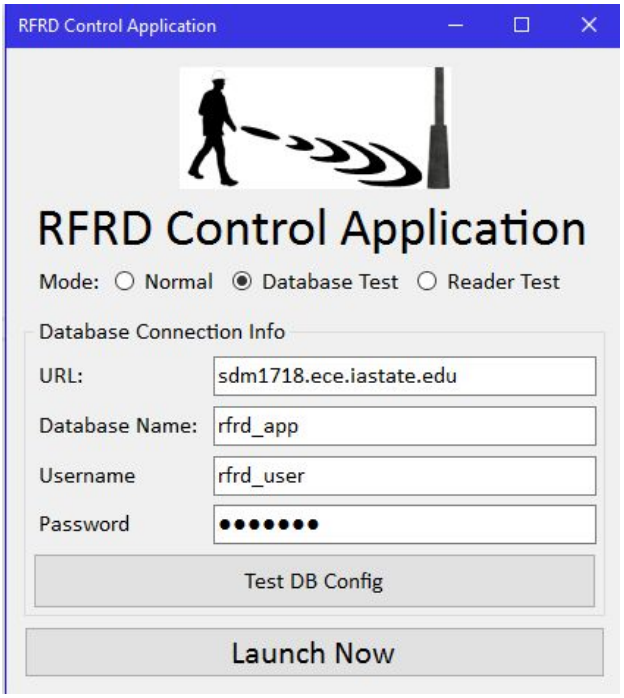


This is all of the functionality available in the software at the time of this writing. The next portion explains how to setup a database to run modes 1 and 2 of the software. In order to have the software actually storing data, you will need a Microsoft SQL Server instance installed somewhere for the application to access, as the program only supports it at the current revision. For development and testing purposes, an instance of SQL Server Express can be installed on the development

machine or a remote server or virtual machine, whichever is more convenient. Information about how to install and administrate it is available online.

Once it is installed, you will need to set up the necessary tables for this version of the software. To avoid problems with not having the tables right, a SQL code file to set up the entire database is available on our website. See Appendix IV for a link to the SQL file. Once you've obtained the code, simply run it as a query on your SQL Server instance and, barring any issues with running it, you'll have the tables needed for this build of the software. You can also create a db user, though it'll need some extra config, for the application or just use the admin user for the SQL Server instance. Documentation on using SQL Server is not provided in this document, but there are tutorials available online that should cover anything you need to know.

With the database setup, the Normal and Database Test modes should now work as intended. Re-open the application and select the desired mode at the start prompt. In the middle of the window, fill out the connection information for your database, including the url, which will have no protocol attached, as shown below, the name of the database being accessed (if you're using the provided SQL code, then leave it the same), and the username and password for the application to use to access the database. If all goes well, the program will be ready to work with the database.



Reader Notes

For the Arduino, in order for it to act as the the bridge from the user interface to the analog reader hardware, it must be programmed with some code to take in input from an analog pin and output to a digital pin on the Mega. The code is

available from the link in Appendix IV. You will also need the Arduino IDE in order to program the Mega, which is available for download at arduino.cc and should include everything that you need to program the device. Once that is setup and the Mega has been programmed successfully, plug the device in using the USB port of your computer. Then plug in the wires leading to the analog reader hardware into the board, the output going to the sending circuitry using digital pin 34 and the input using analog pin 8. Then follow the instructions for the user interface and you should be good to go. If you need help with performing these tasks, guides are available from Arduino that provide step by step instructions to setup the Arduino IDE, program the device, and even on how to extend the code.

Antenna / Rectifier Notes and Setup

The antennas have one input. They take an RF signal from some RF source. From a systems point of view, the only thing between two antennas should be the channel over which communication is occurring. One antenna is connected to the reader, and one antenna is connected to the tag.

The rectifier has one input and one output. It is on the tag side of our overall system (as opposed to the reader side). Its input connects to the tag-side antenna. Its output is a DC voltage on a capacitor which stores enough energy to run one cycle of a low-power IC.

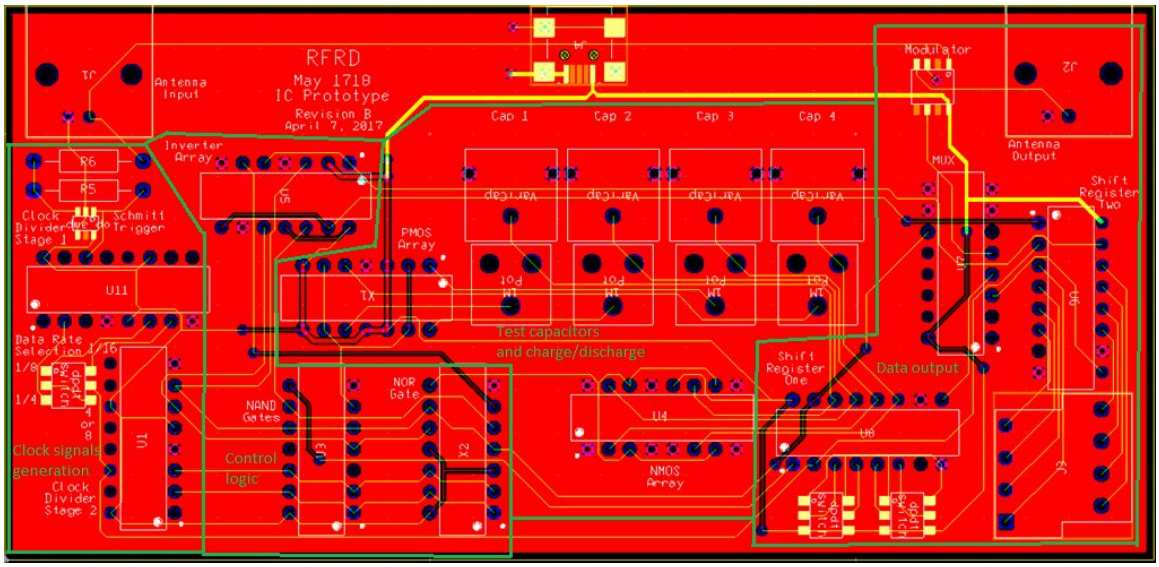
IC Prototype Notes and Setup

There are two versions of IC prototype PCB, version A and version B. Version A has the core functionality of the IC. The inputs are a clock signal and a 5 volt power supply. There is also a single output for the raw serial data.

Version B will accept a sine wave antenna input, with 5 volt power input, and will output modulated data to an antenna. This board will also allow three data rates at ¼, ⅛, and 1/16 of the incoming antenna signal frequency.

The data generation on each board consists of two 8-bit shift registers, which output to a mux in sequence. Shift register one starts with a set of switches that allow the operator to select high or low for the first four bits in the sequence. The remaining four bits on this register are the capacitor readings. The second register reads its values from a set of jumper connections.

These chips are controlled through the nand, nor, and inverter logic chips, with the signals from the 4-bit counter chip. The pmos and nmos array chips charge and discharge the capacitors.



Cadence Notes

In cadence we separated the different parts of the project and created them as Verilog designs using behavioral Verilog, synthesized it using RC into structural Verilog, and then imported it into Cadence. With this process we created the counters and shift registers to be used in the design.

With these two main components imported into cadence we created the analog design for the capacitive sensor. The first analog component designed was the inverters for the capacitance charger. They had to be specially sized to control the rise and fall times so the capacitors would charge to the correct values. The second analog component we attempted to create was an internally compensated op-amp to create a schmitt trigger, but were unable to finish the design.

At the moment we are using a selection of NAND and NOR gates to directly control the charge/discharge times. When the design is closer to finishing, these should be changed to verilog and combined into a single, larger Verilog circuit to automatically generate the layout using Encounter. We are currently modulating the output data through the use of a simple MOSFET modulator.

Appendix II: Alternative and/or Initial Designs

Discussion of option to design for 900 MHz:

At 900 MHz, we can get antenna power transfer orders of magnitude better (tens of dB decrease in insertion loss and reflection, respectively) than we do at 13.56 MHz with orders of magnitude less size (wavelength goes from 22 m to 33 cm). We can start using microstrip lines in place of passive components, saving loss, tightening design tolerances, and putting more of our electronics in our control. For example, the filter at the rectifier to eliminate diode harmonics is realizable at 900 MHz because the microstrip lines used for it can be of physically practical length.

The problem with designing at 900 MHz is the logistics of hardware implementation. Because we had no experience with RFID systems in the past, we used 13.56 MHz because it allowed us to quickly turn over hardware prototypes and assess their functionality at very low administrative, logistical, and financial cost, while giving us a frequency that was good enough for decent power transmission on reasonably sized antennas. If we tried to prototype and develop with 900 MHz as fast as we did with 13.56 MHz, our testing would be destroyed by parasitics and distortion inherent to the tools we have at our regular disposal.

Now that we have a proof of concept system, shifting the frequency back up to 900 MHz is just an improvement on an existing system - one that will allow a more efficient and longer range device.

MSP 430 & Capacitance Sensor Chip Design:

There were some other options considered when designing this system, which included us researching off the shelf components that could be used to make our RFRD tag. We found that we could use an MSP 430 processor and a capacitance sensor chip to design the RFRD project, but this came with some constraint breaking problems. First of all, using both of these together would bring our cost way over our the fifty cent requirement per tag. The other constraint broken would be that the size would drastically exceed the millimeter mark and push us into the centimeter area which is an order of magnitude over the constraint. The power requirement for these was also an issue discussed as it would push our reader to not supply adequate power to the tag which would render the tag useless.

Reflection coefficient method of determining capacitance:

Before the client made it known that he wanted a fully adaptable, abstract system that only needed replacement of a transducer, it was proposed to measure reflected RF to determine the capacitance of the light post. Essentially, the tag would be the receiving antenna linked to a series RLC (transmission line accounted for as well), with two capacitors. The R, L, and one of the Cs would be known, or fixed by us, while the second C would be the capacitance we want to measure.

The principle, then, is that a frequency other than the resonant frequency of this passive tag system will lead to some reflection that bounces back out of the tag's antenna.

The reader can then use two antennas: One to transmit a variable frequency, and another to pick up reflection from the tag's antenna. This second antenna on the reader will pick up the minimum possible power when the first antenna is transmitting at the resonant frequency of the tag's circuit. When this resonant frequency has been found, the capacitance of the unknown capacitor can be calculated.

Changing the known values would alter the range of the unknown capacitances we can measure and how much spectrum would be required to traverse the possible range of measurable capacitances. Also, antennas used can be frequency-independent, designed near the operating frequency.

Binary frequency shift key modulation:

Binary frequency shift key (BFSK) modulation would better serve the purposes of our project than amplitude shift key (ASK) modulation and should be utilized in future developments of our RFRD device. ASK modulation was originally utilized because of its simplicity. However, while it was sufficient for our testing purposes this semester, the fact that our message signal contained a considerable amount of noise in comparison to signal amplitude meant that the two distinct amplitudes of the corresponding modulated signal were indistinguishable at a range of more than a few centimeters. Since BFSK modulation would transmit the data in the actual frequency of the signal itself, the presence of noise would not have as detrimental an impact on the ability of our system to transmit data successfully.

Appendix III: Other Considerations

Continuation of the project should first focus on finishing a Rev C of the tag prototype fixing any problems that have been found and creating a physical antenna using the design we have created. The next step would be to work on the reader, implement the designs for the oscillator, amplifier, and demodulator and test with the tag prototype.

Once the prototype system is working the design should be converted to Cadence. At the moment the main option available is using the 0.5µ process, but the team should look into smaller processes to work at higher speeds.

Due to the amount of notes compiled for the next Antenna/Rectifier team, we decided to upload this document to our project's website. It is available at the following link: http://may1718.sd.ece.iastate.edu/docs/1718_nextAntennaNotes.docx

For the next software developers working on the user interface, the software could definitely use work. The first thing I would recommend doing is restructuring the SQL tables and see if a better layout for the data can be created, as there's probably a more optimal way to store the data that wasn't considered. Next, the

capability to add new bolts to the system should be added. This was intended to be in the first release, but ended up being scrapped in favor of improving the other parts of the code. Finally, the program is not visually appealing at all, so if someone wants to get creative and knows more about UI design using Visual Studio, they could really make the application look a lot better. Alternatively, the original mobile app version of this could be developed instead, but this is up to the future developers.

As for the Arduino code, there's room for improvement there as well. At the time of development, there was no hardware to actually test the code with, so while the code does work decently well and the Arduino will return data, I am unsure if the code will work with the analog hardware that will be needed to communicate with the IC tag. Additionally, a smaller Arduino could be used to power the reader, as the Mega had far more than enough pins to make things happen. Just make sure that the smaller one has all of the hardware features the Mega does (ADC, analog and digital out, Serial-over-USB, etc) and, with a few tweaks, it should work fine. Finally, if a sense of adventure is desired, a bluetooth module could be added to the Arduino to facilitate Bluetooth communication. Should work decently well and it was something that was desired to be in this iteration, but we decided to play it safe and just use USB communications.

Appendix IV: Code

Since our project has a number of code files to make the project work and these files can be quite long, we have decided to store the code on our website and provide links for you to access it.

User Interface Visual Studio Project:

http://may1718.sd.ece.iastate.edu/docs/1718_uicode.zip

Latest User Interface Build:

http://may1718.sd.ece.iastate.edu/docs/1718_uibuild.zip

Arduino Reader Code File:

http://may1718.sd.ece.iastate.edu/docs/1718_reader.ino

SQL Server Setup File:

http://may1718.sd.ece.iastate.edu/docs/1718_rfrdsql.sql