Senior Capstone Design Project 2007

Final Report:
Implementation of RFID Payment for Parking Meters

April 26, 2007

Advisor: Prof. Charles DiMarzio

Group Members:

Matthew Dickman
Leo Rakita
Gleb Zverinskiy
Michael Rolfes
Dan Quigley
# Table of Contents

I. MOTIVATION ................................................................................................................................. 3 

II. PROBLEM ANALYSIS .................................................................................................................. 4 
   A. BOSTON TRANSPORTATION ................................................................................................. 4 
   B. RFID TECHNOLOGY ............................................................................................................... 5 
   C. ANALYSIS .............................................................................................................................. 5 
   D. USER SCENARIOS .................................................................................................................. 7 
   E. IMPLEMENTATION .................................................................................................................. 8 

III. DESIGN ......................................................................................................................................... 10 
   A. OVERVIEW ............................................................................................................................. 10 
   B. HARDWARE INTEGRATION ..................................................................................................... 10 
      i. Timer Components ............................................................................................................... 10 
      ii. Phidgets RFID Kit ............................................................................................................... 110 
      iii. Emitter Follower Circuit ................................................................................................. 12 
      iv. Final Product ................................................................................................................... 13 
   C. CONTROL SOFTWARE ............................................................................................................. 14 
      i. Overview ........................................................................................................................... 14 
      ii. The Graphical User Interface ........................................................................................... 14 
      iii. Control Logic .................................................................................................................. 16 
   D. BILLING SOFTWARE GUI ......................................................................................................... 18 
      i. Overview ........................................................................................................................... 18 
      ii. The Graphical User Interface ........................................................................................... 14 
      iii. Backend Operations ........................................................................................................ 21 

IV. PROJECT SCHEDULE ..................................................................................................................... 22 

V. COST ANALYSIS ............................................................................................................................ 23 

VI. TASK ASSIGNMENTS .................................................................................................................. 26 

VII. IDEAS FOR ENHANCEMENT ..................................................................................................... 27 

VIII. CONCLUSION ............................................................................................................................ 29 

IX. APPENDICES ............................................................................................................................... 30
I. Motivation

On the first day that our capstone group was placed together, we discussed many possible projects to undertake. After brainstorming and deliberating we concluded that our project will be one that is simplistic in nature as well as logical to its environment. To be successful in the completion of our project we would all have to be highly motivated and have a basic understanding of the underlying problem. This motivation came from our past experiences and the knowledge that comes with those experiences.

During our team meetings, our group realized that we had similar issues with commuting into and around the city of Boston. When driving around, more often than not, we’d have to find a spot to park, which almost always takes longer than desired. In addition to the time it takes to find a meter, another issue is that the meter only accepts change as payment. Then the decision of how long to stay at the meter has to be determined upon arrival. So, as a commuter you’re asked to first drive around blindly looking for an open space hoping to get lucky, then estimate how long you’re going to be there, then pay the meter in change, with no other option of payment. Our group looked at these issues and thought that there could be a more efficient system.
II. Problem Analysis

a. Boston Transportation

We began research into whether Boston had looked at this as a current problem, and if so, what they have done to work towards a solution. Similar transportation issues that the city of Boston has recently handled are “The Big Dig”, environmentally safe busses, new trains, and the introduction of the “Charlie Card”. The Big Dig is the ongoing improvement of the city’s roads and tunnel system that seems to consistently add projects to this overall city-wide improvement. This led us to believe that Boston is a city looking to improve its current transportation systems and is therefore a good candidate for our capstone project. The environmentally safe compressed natural gas (CNG) busses were released in 2004 in a hope to increase the city’s overall environmental cleanliness as well as to save money versus older diesel engine busses. Around the same time period new trains we released on the B-line to increase attractiveness to the green line as well as to save money with more efficient machinery. Most recently in December of 2006, the MBTA released its RFID card and replaced its old coin operated turnstiles with sliding glass doors. All these recent improvements and changes to the city’s methods of transportation led our group to believe that the next logical step would be to improve or change the current parking meter system. We realized that we could capitalize on this area and build a working prototype. All these changes Boston has made over the recent years have been, in one way or another, an attempt to make the city’s transportation seem more attractive and hopefully gain more consistent revenue. Our group realized that we would have to take this as one of our motivation factors in this project to make our project marketable to the city of Boston.
b. RFID Technology

Before the development of the Charlie Card we decided on the use of RFID (Radio Frequency Identification) technology as a main aspect of our project. Some of our group members took courses in wireless technology and electromagnetics where it was mentioned how RFID has been booming in the recent years due to a few important factors; lower cost, increased security, and increased ease of use. The technology’s costs have diminished exponentially because of the mass production of passive tags for commercialized products. One of the most publicly familiar passive tags was released in 1997 by Texas Instruments and Digital Signature Transponder for the company Exxon-Mobil under the name “Speedpass”. RFID technology has been around since the mid-1920s however this advancement allowed for the public to be confident in its security as well as to enjoy on a user level RFID’s ease of use.

c. Analysis

As a group, our dedication to solving a logical real-world problem stems from our experiences at Northeastern University and its real-world emphasis on education. At our cooperative education (co-op) jobs we discovered that solving the problem is not a complete solution, we must solve the problem efficiently and in this case that means being cost effective. During our original research we did come across a RFID parking meter that was on the market of the summer of 2006 however the costs exceeded $10,000. Our group decided that we would be able to prototype a system and keep the overall unit cost below $500.
With strong motivating factors encouraging us, we began to analyze the problem in more depth. The first step in the analysis of a problem is to define the problem thoroughly and clearly. Our problem was that the current parking meter was inefficient due to four main factors; the meters accept only change, multiple city laborers are needed, the length of stay must be predetermined, and they provide often inconsistent revenue for the city. The current parking meter system runs very simply; find a spot, guess the possible time of occupancy, and then put in the respective amount of change. As Boston becomes more digital based, with only a small percentage of city business taking only cash, change becomes less available on any given person. This affects the meter system because if a person does not have any available change then that person would not be able to pay for the meter and therefore is put in a position with two logical outcomes. The person runs to the nearest store to get change for the meter, putting his vehicle in unnecessary risk of a ticket, or the person does not pay for the meter at all leaving the city with no income unless ticketed. The current change payment parking meter system also has the issue of needing multiple laborers to maintain it. The system also needs a meter maid to check for expired meters as well as employees to collect the change multiple times a week. An armored vehicle is needed to transport the collected change to a sorting and counting facility where many laborers are employed as well. The change parking meters work on a time increment method, where 25 cents could equal 15 minutes of time. This creates the issue of judging the length of stay which can be difficult to determine as unforeseen circumstances may arise that would differ from the original assessment. This situation can end in either underpaying, where the risk of a ticket arises, or overpaying, where more money is put into the meter than needed. All
these issues lead to inconsistent revenue for the city. With these issues in mind we could declare our capstone objective to be the following: To implement an RFID payment and logging system that can be integrated into a standard parking meter, which allows for increased ease of use as well as more efficient transactions.

d. User Scenarios

Now that we have a solid goal that allows us to define what we want to accomplish we determined if there is a specific need for a solution. In this area we looked at the two main parties that would be affected by these issues, the civilian and the city community. The civilian would benefit by not having to use change. This would allow the civilian to not get stuck in a situation where he or she would be left with an unpaid meter when going to get change. Also the civilian would not have to determine the time of stay on arrival. With a logging system, the civilian could “swipe in” and “swipe out” allowing for no overpaying or underpaying, only paying for what the civilian used, the exact amount of time. This minimizes the risk of getting an unnecessary ticket, which would put the civilian at ease and subsequently will allow for more consistent use of the meter. With the ability to pay for the exact amount of time, the civilians would be more enthusiastic to pay for the meter versus not paying for a few minutes and risking a ticket. The civilian will also be at ease to know that the whole transaction can be encrypted and their account information is secure. With protocols developed by Texas Instruments for the Exxon-Mobile Speedpass, and more recently RSA securities, the civilian can be assured of the secure transactions. Misplaced or lost RFID cards allow for less concern than cash or credit cards as the only purchase that can be made with these cards would be to pay
for parking meters that are not a high priced item, at most $1 an hour. The civilian will also benefit from the smooth, simple, and fast transitions. The civilian only needs to hold the RFID tag next to the meter for a second when swiping in and a second when swiping out, no fumbling through change or navigating through long menu systems.

The city also benefits from upgrading to the RFID parking meter system. With RFID the transactions become digital, cutting down on labor that was necessary for the coined system. Since the RFID parking meter system allows for coins to be used as well the labor wouldn’t be completely eliminated, however the amount of employees could be increased saving money for the city. Since the transactions are so easy and efficient for the civilian they are more willing to pay for the meters and that will create more consistent revenue for the city, making money where the city would usually get zero income. One of the most positive outcomes of having a system that is digital and creates data is that the system allows for a wide range of possibilities for future applications.

e. Implementation

Seeing that both sides can benefit greatly from a new parking meter system our group decided it was important in analyzing a problem to discuss the alternative solutions. We looked at three possible solutions, staying with a coined system, using a credit card kiosk, and using our RFID parking meter system. Comparing these solutions we looked at how the systems judge time. Both the coin system and the credit card kiosk use a time incremented process which brings up the issues over overpaying and underpaying, where the RFID uses a logging system which logs and charges the motorist for the exact time of use. Another comparative issue was how long it took the civilian to
use the system. Our group averaged the time it took to put 4 quarters into a meter at approximately 10 seconds. We averaged the time it took to use a credit card kiosk similar to the ones in place in San Diego at approximately 30 seconds. The credit card kiosks in San Diego work by first swiping the credit card then, choosing the spot where the car is parked, then entering the amount of time that will be paid for, then waiting for approval and finally the ticked printing out. The RFID parking meter system would only take approximately 2 seconds, 1 second to swipe in and 1 to swipe out. The next issue that was raised to compare the possible solutions was security. The coin system is not a security issue since cash doesn’t trace back to any account information. The credit card kiosk system is a security risk since the credit card information, if misplaced, could be used to purchase multiple different items and opens consumers to identity theft. The RFID parking meter system uses an advanced encryption algorithm and even if lost wouldn’t be a security issue since the only use for the tags would be to pay for time at parking meters. The feature that was compared last is the ability to open up to future applications. With a coin operated system the city remains stagnant and is not open to future applications. With either a credit card kiosk or the RFID parking meter system the data is there to move forward with new applications in information management. After analyzing the problem and having a strong motivation to move forward with our project we were able to move into the design phase of our project. Boston is clearly progressing away from old technology, seen in the recent improvements in city roads, busses, trains, those respective payment systems and so should Boston’s parking meters.
III. Design

a. Overview

Our final design has gone through a number of changes over the process of the course because of encountered difficulties. Initially, we received a fully operational parking meter from the city of Brookline – the same type that is in use today. Using an existing meter would have made less work to integrate our design and also would have been a clean solution. Unfortunately, after spending weeks trying to figure out a way to integrate our board with the parking meter our success was absent. In the end we could not interface with the parking meter due to the complexity of the waveforms that are sent when a coin is inserted. In order to interface our RFID board with a parking meter we wanted to find a single point on the meter that goes from a low voltage to a logical high when a coin is inserted. If were able to find that point on the meter then we could just tap in when the RFID was activated via a tag and on the outside – the equivalent of inserting a coin. In the end, after measuring signals anywhere and everywhere we couldn’t come up with a place to tap into. We then decided that our best plan of attack would be to move forward with the creation of our own meter, engineered from various parts and cost effective components.

b. Hardware Integration

i. Timer Components

It was time to rethink the design and figure out how we can make our own parking meter. For that purpose it was agreed upon to use a kitchen timer because the essential parts were all there and manufactured affordably. The counting
mechanism, the 7-segment display and the controls were all together and we didn’t have to worry about making our own. A simple kitchen timer was decided upon for its simplicity, and price. The best way to interface with the timer was to tap into the buttons that that were on it. We discovered that two electrical contacts get shorted together when any button on the timer is pressed. Please refer to the Figures 1-3 below for more detail on the timer and relay assembly.

Figures 1-3: (from left to right) The kitchen timer from Target; The solder points on the timer circuitry to short the function; The relays soldered to control the short from 5V output.

ii. Phidgets RFID Kit

We did a great deal of research online and chose to use a fairly affordable RFID kit from a company called Phidgets based out of Canada. The RFID kit produces a 5V output so we decided upon using relays to activate the timer via the RFID kit which would short the contacts on the timer buttons. The RFID board only has two outputs so that means that we can only control two buttons and therefore the button press on the meter for all the purposes needs to be a combination of just those 2 buttons. The relays needed to be controlled by a 5 volt pulse and the relays we chose were from Radio Shack because of abundance and price. Later we discovered that the second output was only strong enough to power an LED, so we needed to construct and emitter follower circuit to amplify the power outputted to
trigger the reset relay. Please refer to Figures 4-5 below for more details on the Phidgets RFID kit.

![Phidgets RFID Kit specifications and final application.](image)

**Figures 4-5**: Phidgets RFID Kit specifications and final application.

### iii. Emitter Follower Circuit

At the testing of the final design stage our group has encountered a major problem. One of the outputs on the RFID board could drive a relay but the second LED output did not provide enough current. When the relay was connected, instead of giving out the 5 volts that we needed, the voltage would drop to approximately 1 volt. In order to counteract this problem we assembled an emitter follower circuit using a BJT-PNP transistor, two resistors and a 9-volt battery for power. The output from the RFID board went into the base of the transistor and acts the biasing switch. The connection is made through a 160Ω resistor because even when switched off there is a tiny voltage give off that would bias the junction. On the collector side the 9-volt battery is connected that powers the circuit. The emitter side is grounded through a 3.9MΩ resistor and the relay is connected at the same point as the emitter and the resistor. When no pulse is sent the voltage at the relay junction is zero but when a pulse is sent the transistor gets biased into letting current through and bumping the voltage at the relay junction to 5 volts.
We were able to test this circuit using various resister values, as well as try various designs before we can to a final circuit that worked correctly. This circuit turned out to be the key component to our system and we needed to be able to trigger both the reset feature and start/stop feature of the timer. In Figure 6-7 below you can see a diagram of our circuit, as well as the fabricated product soldered together.

![Circuit Diagram](image)

**Figures 6-7**: The circuit diagram for the altered emitter follower circuit and the final design.

### iv. Final Product

After resolving the issues and troubles we had, we were able to integrate the compact parts from inside the timer with an existing meter housing, as well as store the emitter follower circuit in a compact and concealed location with the RFID receiver. We tested out all of the hardware features thoroughly and integrated it with the control and billing software. **Figure 8** shows the final product assembled and working efficiently. We also made a video of the meter function and operations. If you would like a copy of this video please contact the authors.
c. Control Software

   i. Overview

   Integrating all of the circuitry was no easy task. The Phidgets RFID kit came bundled with a variety of evaluation software. One of the programs, written in Visual Basic, consisted of a simple graphical user interface with an arsenal of interactive tools to control the radio frequency receiver. We modified this particular program in order to successfully integrate the various hardware components of the meter. It was chosen due to the programming language’s event-based architecture and its simple, yet powerful, commands that enabled us to accomplish our design goals. The overall system consists of a GUI control application built around a feature rich API. This was incredibly useful when it came to debugging issues we had throughout the course of the project, especially when our RFID kit burned out the day before the initial Capstone Competition.
ii. The Graphical User Interface

Figure 9: A standard view of the software that integrates all of the hardware components, as well as the logic surrounding the equipment.

There are a series of checkboxes at the top of the dialog that control the various outputs on the RFID receiver. The overall look and feel can be seen in Figure 9 above. The top two checkboxes, “DigitalOutput0” and “DigitalOutput1”, were used in controlling and triggering our electronic timer. Each box, when checked, applies a voltage to its respective output terminal. This enabled us to connect wires from each of the output terminals to a set of relays. The relays were subsequently connected to the electronic timer, one relay to the controller for starting and
stopping the clock and the other to the controller for resetting the timer. Of course, during regular meter operation all of the triggering and resetting would be automated, but these checkboxes enabled simple diagnostic and troubleshooting operations whenever necessary.

The swipe log, located below the checkboxes, is a list of each swipe “in” and “out” of the parking meter. The table also displays only accepted and logged swipes. Once a tag is swiped “in”, no other individual can swipe that current user out. Subsequently, no other tag will be displayed in the table until the current user swipes “out”. The log contains each tag number, and a text label specifying whether the tag is an “in” or “out” swipe (and whether the person actually swiped their tag or let the meter timeout). If the person is at the meter longer than two hours, the tag is automatically logged out of the meter and the transaction is printed to the swipe log and a separate log tile containing all of the daily transactions.

iii. Control Logic

All of the operations handled behind the scenes were integrated into the same Visual Basic graphical user interface. The primary back-end functions were handling when a tag is swiped “in” or “out” of the parking meter. When a new tag is swiped “in”, the RFID receiver will send a 5 volt pulse to the relay connected to the controller for starting and stopping the electronic timer. This operation starts the timer. When that same tag is swiped “out”, the RFID receiver will also send a 5 volt pulse to the relay connected to the controller for starting and stopping the electronic timer. This operation stops the timer. However, there is also another pair of 1 volt pulses sent to an emitter-follower circuit. This raises the output voltage of
the pulse to 5 volts, which is subsequently forwarded to the relay connected to the controller for resetting the timer. This operation resets the timer and causes it to flash. The flashing signifies that the meter is unused and/or unpaid.

Another important feature of the code is disallowing tags from swiping a different tag “out” of the meter. This operation prevents people from swiping innocent customers “out” of the meter while they are out shopping or running errands. After a tag is swiped “in”, the tag number is locally stored and compared after each subsequent swipe. If the swipe is not made by the original tag, the program does nothing. If the swipe is made by the original tag, the program logs the swipe and resets the meter to accept any new “in” swipe.

One of the last, but equally important features of the code is its ability to automatically swipe “out” customers that forget to swipe “out” of their spot. Once a tag is swiped “in”, an internal clock is started in the program. If the counter reaches two hours before the user swipes “out”, the tag is automatically swiped “out”. The transaction is logged normally and the meter is reset to accept any new “in” swipe. If a user swipes “out” of the meter before the timer expires, the timer is stopped and reset. It will begin counting from 0 seconds the new time a tag is swiped “in”.

The code was a highly effective prototype in translating our conceptual designs into reality. Some possible areas for future improvement would include, but are not limited to, verifying each tag is valid before it is logged and accepted, allowing other users to override the meter or log another user out of the spot if they have left without swiping “out”, and having the meter automatically swipe users out once the
time reaches 6:00 PM (or a similar time when the meter is no longer checked for usage). The program code can be seen in Appendix A at the end of this report.

d. Billing Software GUI

i. Overview

The software components were integrated using both scripts written in C++ and a graphical user interface created using Visual Basic. The original software, which integrated the hardware, created a standard log file. The log file kept a record of each of the swipes with a date and time stamp, as well as the tag number. This enabled a secondary piece of software to scan the log file based upon various search criteria and construct a billing statement based upon that tag’s usage at the meter.

ii. The Graphical User Interface

The graphical user interface has a relatively simple design. A user can select any of the currently used tag numbers from a drop-down menu and then click the “Search” button to generate and preview a sample billing statement. If the user does not select one of the tag numbers, clicking the “Search” button will do nothing. Clicking the “Print” button will produce a blank page to be printed. Once one of the tag numbers is selected, clicking the “Search” button will then successfully generate a billing statement. Details of this interface can be seen below in Figures 10-12. Also, you may contact the authors if you would like a soft copy of the executable.
Figure 10: A screenshot from our secondary software that will generate billing statements based off of the RFID swipe log file created by the control application.

Figure 11: The drop-down menu contains a list of the tag numbers found in the log file.
Figure 12: After clicking the “Search” button, a preview of the billing statement is provided in this textbox.

Once the “Search” button is clicked, the Visual Basic code actually calls a C++ script to traverse the log files and generate the billing statement. Once the script successfully completes, the Visual Basic code will load the resulting file that contains the finished billing statement. This file is imported into the large textbox located in the bottom land-hand corner of the dialog. If the resulting preview is acceptable, a user can print the statement using the “Print” button at the bottom right-hand corner of the dialog. It will produce a standard windows page setup dialog where a printer can be selected, and the margins and layout can be customized. From there, the file will be printed to paper.
iii. Backend Operations

The C++ script that traverses the log files and generates the billing statement is relatively simple in design. The code begins its operations by importing the previously selected tag number into memory, and searching through a log file containing all users’ billing information. This allows the tag number to be matched with a specific name and address. The user’s information is specially formatted and outputted to a text file. This serves as the header for the billing statement.

The main operation of the script is next. The log file containing all of the meter’s transactions is searched for swipes by the selected tag number. The total time and cost of each transaction is then calculated and outputted onto the billing statement. A running total of the charges is maintained locally, where it is later outputted at the bottom of the billing statement after the transaction file has been completely searched.

The final formatting and design additions are made to the billing statement file, and it is ready to be imported back into Visual Basic for preview and editing purposes.

The code was another highly effective prototype in translating our conceptual designs into reality. Some possible areas for future improvement would include, but are not limited to, enabling search criteria other than just tag numbers, enabling statements to be constructed for multiple meter (and log file) usage, and allowing more exact charges to be made for meter usage (as it currently rounds transactions up, in 15 minute intervals). The C++ program code can be seen in Appendix B at the end of this report.
IV. Project Schedule

This capstone experience proved to us, the importance of having a strong project schedule and sticking to it. We also had to learn how to deal with quick fires that come up, as well as stick to our main project goals. One week we saw that we risked “rat-holing” on the specific topic of integrating with the existing meter and reverse engineering the design. We soon learned that this was difficult and developed three alternate plans to stay on track. We ultimately decided to build out own meter which we would integrate into and existing meter’s housing. We knew our strengths as well as our weaknesses, technically as a team, and focused our efforts to make the greatest impact in the project. Below you can find a brief overview of our process as well as some of the issues we ran into along the way that changed our design goals.

<table>
<thead>
<tr>
<th>Project Development Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>July – August 2006</strong></td>
</tr>
<tr>
<td><strong>August 2006</strong></td>
</tr>
<tr>
<td><strong>September – October 2006</strong></td>
</tr>
<tr>
<td><strong>November – December 2006</strong></td>
</tr>
<tr>
<td><strong>January – February 2007</strong></td>
</tr>
<tr>
<td><strong>February – March 2007</strong></td>
</tr>
<tr>
<td><strong>March 2007</strong></td>
</tr>
<tr>
<td>Date</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>April 2007 – Week 1</td>
</tr>
<tr>
<td>April 2007 – Week 2</td>
</tr>
<tr>
<td>April 17, 2007</td>
</tr>
</tbody>
</table>

Instituting weekly reports and constantly reviewing the goals of our project helped us stay on course and handle issues when they arose. Thanks to the hard work of our team, and some creative and resourceful ideas on everyone’s part, we were able to stay on track and make our project a success.

V. Cost Analysis

As it was stated in previous sections of this report, maintaining a low cost was essential in the overall development and design of our project. This was one of the main reasons we steered away from more expensive implementations such as a proximity sensor, wireless communication of tag information, or “Gumstix” computer integration into the meter. We wanted to prototype a device that was small enough to integrate into existing meters so as to minimize cost and make it more marketable to municipalities who invest a large amount of money into these systems.

Over the course of our research, we came across a large number and variety of meters which ranged in cost anywhere from $400 - $10,000, on average $1000 per meter. Our
initial estimate was to maintain a product under $500, and we were able to beat that goal with an overall cost of approximately $115. Below you can see a breakdown of the overall costs associated to the development of our product, as well as the unit cost for us to repeat the process and create these units on a mass scale.

<table>
<thead>
<tr>
<th>Description</th>
<th>Vendor</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phidget RFID Kit (EM4201)</td>
<td>Trossen Robotics</td>
<td>$86.53</td>
</tr>
<tr>
<td>Infrared Emitting LEDs</td>
<td>&quot;You-Do-It&quot; Electronics</td>
<td>$7.13</td>
</tr>
<tr>
<td>Digital 2 Hour Parking Meter</td>
<td>eBay / PayPal</td>
<td>$34.99</td>
</tr>
<tr>
<td>Soldering Kit, solder, IR/LED Detector</td>
<td>RadioShack</td>
<td>$17.26</td>
</tr>
<tr>
<td>Timer Clock (x2)</td>
<td>Target</td>
<td>$23.98</td>
</tr>
<tr>
<td>PhidgetTextLCD 20x2 : White</td>
<td>Phidgets Inc.</td>
<td>$134.62</td>
</tr>
<tr>
<td>SPDT 5V 1AMP Relay</td>
<td>RadioShack</td>
<td>$4.49</td>
</tr>
<tr>
<td>SPDT 5V 1AMP Relay &amp; 50FT White Kynar Wire</td>
<td>RadioShack</td>
<td>$8.48</td>
</tr>
<tr>
<td>Plexiglas Sheet</td>
<td>The Home Depot</td>
<td>$2.29</td>
</tr>
<tr>
<td>Three - SPDT 5V 1AMP Relay</td>
<td>RadioShack</td>
<td>$13.47</td>
</tr>
<tr>
<td>Loose Fasteners</td>
<td>Economy Hardware</td>
<td>$3.08</td>
</tr>
<tr>
<td>4 PK 1x1/2 Corner Iron &amp; Misc. Hardware</td>
<td>True Value Hardware</td>
<td>$4.17</td>
</tr>
<tr>
<td>Misc. Hardware</td>
<td>True Value Hardware</td>
<td>$0.72</td>
</tr>
<tr>
<td>SPDT 5V 1AMP Relay</td>
<td>RadioShack</td>
<td>$4.49</td>
</tr>
<tr>
<td>Phidget RFID Readers</td>
<td>Trossen Robotics</td>
<td>$158.87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$504.57</td>
</tr>
</tbody>
</table>
We had a number of extra costs over the course of our development, which can be seen in the first table. The bulk of these expenses were replacement RFID Receivers and LCD displays reserved as a backup if we needed them. We also bought some extra parts and supplies in anticipation that something may fail, such as sensitive relays. We are very proud at the fact we were able to accomplish this with such limited overall cost.
VI. Task Assignments

During the past two semesters various group members took upon themselves a number of tasks. As the project evolved, so did the various assignments that needed to be fulfilled. Initially, Gleb and Leo were responsible for acquiring the parking meter. This was done with the assistance of the city of Brookline. Meanwhile, the rest of the group was focused on obtaining the necessary documentation to help reverse engineer the meter. This proved to be more difficult than anticipated since POM (the company that made our meter), was unwilling to give up their schematics or provide any kind of assistance. At this point Matt was researching the various RFID systems we were considering implementing but not having the knowledge of the inner workings of the parking meter prevented us from ordering one early on. Since our initial intention was to be able to interface the RFID with an existing, working parking meter our primary concern was to figure out how the meter works. Our main concern was to figure out how time could be added through an external input. After months of unsuccessful trials by virtually every group member, it was agreed upon that another approach would be to use a timer in place of the meter and program a RFID kit to trigger/reset the timer. By this time, Matt picked out a RFID kit with dual 5V outputs which was also not to difficult to program. Mike took it upon himself to program the kit accordingly. Meanwhile, Dan was responsible for fabricating the body of the meter. Matt, Leo and Gleb were working on the hardware, such as figuring out how to connect the timer to the RFID board, which relays needed to be used, creating the emitter follower circuit, and soldering all the components. Most of the soldering was done by Leo, while Gleb and Matt constructed the emitter follower circuit needed for driving one of the relays. The table below gives an overall summary of team roles and areas of interest.
<table>
<thead>
<tr>
<th>Team Member</th>
<th>Roles and Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matt Dickman</td>
<td>Team leader, weekly reports, project planning, parts research and acquisition. More emphasis on hardware design.</td>
</tr>
<tr>
<td>Gleb Zverinskiy</td>
<td>Parking meter acquisition, hardware development, meter reverse engineering, software requirements input, hardware focused.</td>
</tr>
<tr>
<td>Michael Rolfes</td>
<td>Primary software developer, researched API capabilities, designed both the control and billing graphical user interfaces.</td>
</tr>
<tr>
<td>Leo Rakita</td>
<td>Meter acquisition, hardware oriented, primary solder and construction of timer and relays, software requirements input.</td>
</tr>
<tr>
<td>Dan Quigley</td>
<td>Hardware focused, overall planning and design for problem solution, original case development and meter reverse engineering.</td>
</tr>
</tbody>
</table>

**VII. Ideas for Enhancement**

Upon completion of our project it became more and more evident what could be done to further improve our product. Perhaps the most glaring issue that needs to be addressed was our inability to connect the RFID kit to the parking meter. Having the ability to do so would go a long way towards convincing city officials to use our product since they wouldn’t have to replace the original housings, and would save a lot of time and effort by not having to qualify and test an entire parking meter. All the testing would have to be done to only the part we made, since the rest of the meter already withstood the test of time. It is often said; “If it’s not broken, why fix it?” our motto was “If it’s not broken, why not improve it?”

Another enhancement we were considering was using a proximity sensor. Unfortunately, we simply didn’t have time to integrate that into our design even though it was something we considered in the beginning. There were two upsides to this feature. First, the person who just parked would not have to manually punch out before leaving. Should he forget to waive his tag, the meter would sense that the vehicle is no longer in the parking spot and reset automatically without the person having to pay for the time he didn’t use. As it stands now, they would get charged for two hours, only then the meter would reset. The other benefit would assist the
meter maids. The proximity sensor would make the meter only flash if there is a car in the parking space but the meter was not paid for. When the BTD truck drives through a street and counts how many meters are unpaid, it would only register the spots that are occupied reducing the number of “false alarms”.

Yet another possible improvement was to integrate the Fast Lane technology. In doing so, the user would no longer need to have an actual tag in his pocket or key chain. The meter would automatically activate once the vehicle is parked and deactivate once it leaves. Since the EZ pass is directly tied to a credit card number, it would be easy to automate this system making it easy to charge the users. Another advantage of such system would be to eliminate the necessity for a proximity sensor as previously mentioned. This approach adds the possibility of false alarms, i.e. car triggering a meter while simply driving by it. However, it would be an easily solvable problem. One possible solution would be to allow the meter to trigger after the car has not moved for two minutes while in range of the tag reader.
VIII. Conclusion

We were able to learn a great deal through our experience in this capstone design process. It was another facet of experience different from traditional co-op where you work on a team of those somewhat older than you. In this process we were able to work directly with our peers on our schedules which was difficult, but at the same time allowed for a different level of communication than we would traditionally see in the workplace. There were a number of individuals who stepped up and took initiative whether it was spending days getting code to work, or taking considerable time in the lab to identify problems in our reverse engineering attempts.

The best part about our project is that it was truly an interdisciplinary experience. It was not strictly a hardware project. We were able to poll back on our experience in previous coursework and labs to design the hardware, develop the software systems, and test everything accordingly. A number of our team members also pursued Business minors which brought another facet to our Capstone experience. As you could tell from this report, we did a fair amount of research and analysis to determine the feasibility of our product once it’s designed not just the actual design and creation. This is an important facet to understand as we progress into the real-world, which is often dominated by profits and marketable products.

It has been a great experience and we have all been left with a strong sense of accomplishment and completion as we finish our undergraduate educations here at Northeastern University. We would also like to thank the faculty, staff, and advisors who helped make our project a reality and guided us along the way.
IX. Appendices

APPENDIX A – RFID Control GUI Code

Option Compare Database

Private WithEvents PM As PHIDGET.PhidgetManager
Dim WithEvents RF As PHIDGET.PhidgetRFID
Dim strLastTag As String
Dim IsStarted As Boolean
Dim LastTag As String
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

Private Sub Form_Timer()
'Output the tag data to the text file
Open "C:\tagdata.txt" For Append As #1
Write #1, Date$; Time$; LastTag
Close #1
'Switch the flag to false
IsStarted = False
'Stop the clock
RF.OutputState(0) = True
Sleep 250
RF.OutputState(0) = False
'Reset the clock
RF.OutputState(1) = True
Sleep 250
RF.OutputState(1) = False
Sleep 250
RF.OutputState(1) = True
Sleep 250
RF.OutputState(1) = False
'Set the timer
TimerInterval = 0
'Provide Information to user
Me.[TblTagIDs subform].Form.Recordset.AddNew
Me.[TblTagIDs subform].Form.Recordset!tagid = LastTag + "- closed by timeout"
Me.[TblTagIDs subform].Form.Recordset.Update
'Reset tag number
LastTag = 0
End Sub

Private Sub Form_Load()
'MsgBox "here"

[30]
TimerInterval = 0

'Set PM = New PHIDGET.PhidgetManager
Set RF = New PHIDGET.PhidgetRFID
Call RF.Open

chkDigitalOutput0.Enabled = False
chkDigitalOutput1.Enabled = False
chkDigitalOutput2.Enabled = False
chkDigitalOutput3.Enabled = False

'Use the following code to attach reader directly without using the PhidgetManager
'Set RF = New PHIDGET.PhidgetRFID
'Call RF.Open(False)
End Sub

'When the reader appears, assign the events to RF
'Private Sub PM_OnAttach(ByVal PHIDGET As IPhidget)
'    If PHIDGET.DeviceType = "PhidgetRFID" Then
'        'End If
'    End Sub

Private Sub RF_OnAttach()
    lblAttached.Caption = "PhidgetRFID" & " Version " & RF.DeviceVersion & ", Serial# " & RF.SerialNumber
    Call But_ClearTable_Click
    'Check the form to determine if the Reader has been enabled and set it on the device
    chkDigitalOutput0.Enabled = True
    chkDigitalOutput1.Enabled = True
    chkDigitalOutput2.Enabled = True
    chkDigitalOutput3.Enabled = True
    If chkDigitalOutput3 Then RF.AntennaOn = True Else RF.AntennaOn = False

End Sub

Private Sub chkDigitalOutput0_Click()
    If chkDigitalOutput0 Then RF.OutputState(0) = True Else RF.OutputState(0) = False
End Sub

Private Sub chkDigitalOutput1_Click()
    If chkDigitalOutput1 Then RF.OutputState(1) = True Else RF.OutputState(1) = False
End Sub

Private Sub chkDigitalOutput2_Click()
    If chkDigitalOutput2 Then RF.LEDOn = True Else RF.LEDOn = False
End Sub
Private Sub chkDigitalOutput3_Click()
    If chkDigitalOutput3 Then RF.AntennaOn = True Else RF.AntennaOn = False
End Sub

'When the event OnTag is thrown by the PhidgetRFID Reader
Private Sub RF_OnTag(ByVal TagNumber As String)

    'If the TagNumber is the same as the last tag then ignore it
    'If chkOnlySingleEntries Then
    '    If TagNumber = strLastTag Then Exit Sub
    'End If

    If IsStarted = False Then
        Me.[TblTagIDs subform].Form.Recordset.AddNew
        Me.[TblTagIDs subform].Form.Recordset!tagid = TagNumber + " - opened by swipe"
        Me.[TblTagIDs subform].Form.Recordset.Update
        strLastTag = TagNumber
        'Output the tag data to the text file
        Open "C:\tagdata.txt" For Append As #1
        Write #1, Date$; Time$; TagNumber
        Close #1
        'Switch flag to true
        IsStarted = True
        'Store tag number
        LastTag = TagNumber
        'Start the clock
        RF.OutputState(0) = True
        Sleep 250
        RF.OutputState(0) = False
        'Start the Two Hour Timer
        TimerInterval = 7200000

    ElseIf LastTag = TagNumber Then
        'Output the tag data to the text file
        Open "C:\tagdata.txt" For Append As #1
        Write #1, Date$; Time$; TagNumber
        Close #1
        'Switch the flag to false
        IsStarted = False
        'Reset tag number
        LastTag = 0
        'Stop the clock
        RF.OutputState(0) = True
        Sleep 250
        RF.OutputState(0) = False
        'Reset the clock
        RF.OutputState(1) = True
        Sleep 250

    End If

[32]
RF.OutputState(1) = False
Sleep 250
RF.OutputState(1) = True
Sleep 250
RF.OutputState(1) = False
'Stop the Two Hour Timer
TimerInterval = 0
'Provide Information to user
Me.[TblTagIDs subform].Form.Recordset.AddNew
Me.[TblTagIDs subform].Form.Recordset!tagid = TagNumber + "," - closed by swipe"
Me.[TblTagIDs subform].Form.Recordset.Update
strLastTag = TagNumber
End If
End Sub

'Delete all lines from the table TblTagIDs
Private Sub But_ClearTable_Click()
    CurrentDb.Execute "DELETE FROM TblTagIDs;"
    Me.TblTagIDs_subform.Requery
End Sub

'When the reader disappears, say so
Private Sub RF_OnDetach()
    lblAttached.Caption = "Attach an RFID tag reader"
    But_ClearTable.SetFocus
'Set RF = Nothing
    chkDigitalOutput0.Enabled = False
    chkDigitalOutput1.Enabled = False
    chkDigitalOutput2.Enabled = False
    chkDigitalOutput3.Enabled = False
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Call But_ClearTable_Click
End Sub
APPENDIX B – Billing GUI Code

```c
#include "stdafx.h"
#include "iostream"
#include "fstream"

using namespace std;

int _tmain(int argc, _TCHAR* argv[]) {
    int j;
    char tagnumber[10];

    //input the number being searched for
    ifstream inputtagnum ("tagnumber.txt", ifstream::in );
    inputtagnum.seekg(0, ios_base::beg);
    for (int x = 0; x < 10; x++)
        tagnumber[x] = inputtagnum.get();
    inputtagnum.close();
    char namenumber[10];

    //locate the personal information assoc. with the tag
    ifstream inputname ("namedoc.txt", ifstream::in);
    inputname.seekg(0, ios_base::beg);
    bool isfound = false;
    while (!isfound) {
        for (int z = 0; z < 10; z++)
            namenumber[z] = inputname.get();
        if (namenumber[z] == tagnumber[z] && z == 9)
            isfound = true;
        else if (namenumber[z] == tagnumber[z])
            isfound = true;
        else {
            inputname.ignore(256, '@');
            inputname.ignore(1, ' ');
            z = 10;
        }
    }

    ofstream bill ("bill.txt", ofstream::trunc);

    //print billing header
    char line[100];
    char line2[100];
```
bill << "-----------------------------------------------------" << endl;
bill << "----------- Billing Statement -----------" << endl;
bill << "-----------------------------------------------------" << endl;
bill << endl << endl << "Mail to:" << endl;
inputname.getline(line,100);
bill << line << endl;
inputname.getline(line,100);
bill << line << endl;
inputname.getline(line,100);
bill << line << endl;
inputname.getline(line,100);
bill << line << endl;
inputname.getline(line,100);
bill << line << endl;
inputname.getline(line,100);
bill << line << endl << endl;
bill << "---------------------------------------" << endl;
bill << "---------- Billing Statement ----------" << endl;
bill << "---------------------------------------" << endl;
bill << endl << endl << endl;
inputname.close();

ifstream inputtrans ("tagdata.txt", ifstream::in);
inputtrans.seekg(0, ios_base::beg);

int insec, inmin, inhour, outmin, outsec, outhour;
double total = 0, trans = 0;

while(!inputtrans.eof())
{
    //analyze transactions
    inputtrans.getline(line,100);
    inputtrans.getline(line2,100);
    for (int j = 25; j < 35; j++)
    {
        {
            //figure out the time difference
            insec = (int)line[21] + ((int)line[20] * 10);
            inmin = (int)line[18] + ((int)line[17] * 10);
            inhour = (int)line[15] + ((int)line[14] * 10);
            outsec = (int)line2[21] + ((int)line2[20] * 10);
            outmin = (int)line2[18] + ((int)line2[17] * 10);
            outhour = (int)line2[15] + ((int)line2[14] * 10);
            // insec = outsec - insec;
            // if (insec > 60)
            // {
            //     outmin += 1;
            //     insec -= 60;
            // }
            // if (insec != 0)
            //     outmin +=1;
            if (outsec > 0)
                outmin += 1;
        }
    }
}
inhour = outhour - inhour;
if (inhour != 0)
{
    if (inhour == 1)
        outmin += 60;
    else if (inhour == 2)
        outmin += 120;
}

inmin = outmin - inmin;

trans = ((inmin/15) * 0.25);
if (inmin%15 != 0)
    trans += 0.25;

//print out transaction
for (int g = 1; g < 11; g++)
    bill << line[g];

    bill << inmin << trans << endl;

    total += trans;
}
else if (line[j] != line2[j])
    j = 35;
}

//print out total cost to customer
    bill << Total: $ << total;

inputtrans.close();
bill.close();
return 0;