University of Puerto Rico

School of Engineering

Department of Electrical and Computer Engineering


# Progress Report


An Electronic Voting System

ICOM 5047 Section 031

Javier Torres Santiago ID# 802-03-8529

Sylvia Rodríguez Rodríguez ID# 802-03-7033

Angel Vega Cortés ID# 844-02-9732

Laura Cruz Rodríguez ID# 802-03-1797

Wednesday, October 29, 2008

# Table of Contents

# 1. Abstract

JSAL has completed the first development phase for the Sparrow prototype, and is currently in progress with the second phase. All deliverables were met for phase one, and with the progress achieved in phase two all deliverables should be met by November 5th, 2008. The cost of both phases has not surpassed our proposed budget.

Deliverables for Phase One:

- Software
  - Create administrator application
  - Create database
- Hardware
  - Assembly and testing of electronic components

Deliverables for Phase Two:

- Software
  - Add additional features to administrator application
  - Database data encryption
- Hardware
  - eVote Screens
  - Implement write-in vote feature
  - Implement keypad navigation

## 2. Executive Summary

We have successfully completed the first development phase for the Sparrow prototype and are currently in the midpoint of the second development phase. Regarding phase one, the software aspect consisted of the administrator application and the database. While the hardware aspect consisted of the assembly and testing of electronic components.

Beginning with the software, the administrator application will permit the designated employee to log in into the system and have the functionality of: registering voters, verifying the status of the kiosks, and revising the usage of the kiosk by the voters. It will also, allow the printing of a Voter Verified Paper Audit Trail (VVPAT). The database is fully implemented for the Sparrow, and has been tested for correctness.

As for the hardware, all electronic components where received in good condition (some parts where delayed in the shipping, but no major drawbacks were made to our schedule.) The entire circuit has been mounted and tested for functionality. We have managed to successfully establish the communication between the three main components: the LCD screen, microprocessor, and the navigation keypad. There were several drawbacks in the hardware that did not affect our schedule and where resolved in due time (i.e. faulty cabling and adding an inverter design to manage the contrast of the LCD screen).

Regarding our progress up till now in phase two, the software aspect consists of encrypting the data in the database and adding new features to the firmware (i.e. permit voter to cast a write-in vote). Furthermore, the hardware aspect has partially implemented keypad navigation. At this pace, we will achieve all the deliverables of phase two by November 5$^{th}$, 2008.

Finally, the budget for both phases one and two has not surpassed our proposed cost.

# 3. Introduction

The eVote project seeks to help solve the problem surrounding the inefficiency in Puerto Rico's traditional voting system. With eVote, the votes can be managed and tallied electronically, and the probability of human error is reduced.

During the past months the team has been able to deliver all scheduled functionality and meet all planned deadlines. On September 10, 2008 the team presented the eVote proposal to professors and fellow students and on October 17, 2008 all team members participated in individual oral exams and discussed their contributions to the project.

In the past few months several changes have been made to the original proposal: from small changes in the database design, to a completely new write-in feature for the LCD. These changes and their impact have been documented both in a change control document and in this report.

This report presents the progress made throughout Phase 1 and part of Phase 2 of the eVote project and also details remaining future work. This progress report also contains a brief work breakdown structure. The main portion of this document contains all the development made by each team member and explained with technical detail. The report also contains an appendix with all code related to the project.

# 4. Progress Report (Gantt chart)

## 4.1. Phase I

For the first phase of eVote, it was planned in the Gantt chart that the team was to complete various tasks.

1. The software interface and format of the voting receipt
2. The Login of the voting officials registration software
3. The Database
4. The assembly of hardware required for an LCD screen test
5. Testing of phase one components

The team has successfully completed these tasks and the Gantt chart remains unaltered. Due to the team's dedication we were able to complete an additional task set for phase two which is the implementation of the database with the interface allowing the eVote registration software to register voters and mark their voting status in the database. This prevents any voter who has voted from voting again since the software will verify the status of a voter's number when they arrive to a voting center. If they are allowed to vote, then they are assigned a kiosk which is also saved in the case that the kiosk malfunctions. In the case of a kiosk malfunction, a list can be compiled to inform voters whose votes were found invalid.

The Login software was also completed which is necessary to make sure only voting officials can log into the software and register voters. This feature compares account information with the database and if the data given is valid, then the official can proceed to the main registration program.

The hardware of this phase consisted of the microprocessor, liquid crystal display (LCD) and an inverter that was found necessary to provide negative voltage to the liquid crystal display. The datasheet indicated that a voltage of -15 volts to -10 volts is necessary to adjust the contrast of the LCD from light to dark. These components were connected together and now function with the preliminary test screen firmware implemented for the first phase of eVote which ended October 17, 2008.

The testing was also completed to make sure each team member completed what was asked of them and if their work meets the Gantt requirements as well as if everything is completely functional.

## 4.2. Phase II

For the second phase of the eVote development the following tasks were set to be completed by November 5, 2008.

1. Create queries for voter check-in
2. Check-in history in administrator application
3. Implement voter check-in with identification number
4. Complete Screens for eVote device
5. Keypad navigation for eVote screens
6. Database encryption and decryption
7. Write-in screen and navigation
8. Testing of all phase two components

The tasks from one to four have been completed. Items five through eight are currently under development and should be completed before the 5th of November.

## 4.3. Future Work (Phase II and III)

As mentioned in 4.2. as of right now only four items remain to be completed for the second development phase of eVote, the keypad navigation, database encryption, write-in and the testing of the components that have been completed.

For the third phase it has been proposed that the team complete the remaining hardware integration of the cables used to provide communication (UART – universal asynchronical receiver/transmitter) between the computer software and the microprocessor, and the implementation of the hardware status (eVote and printer) within the administrator program. The UART will be used to lock the eVote device once the voter has finished their voting experience. Also when an elections official registers a new voter, the eVote device will unlock allowing the voter to enter and begin completing their vote. After this is implemented, the voting summaries will be sent from the eVote device to the computer software where methods will be implemented to store the votes into the database. A simulation to prove that eVote is scalable using threads that will act as individual voters voting in concurrency with other active eVote devices. The hardware will also be encased after being removed from the breadboard and soldered onto a different board permanently.

After all the development of eVote has been completed, extensive testing of all the hardware and software components will begin. To wrap up the project, a user manual will be created to ease new arrivals into the voting procedure with eVote.

## 5. Budget Analysis

There have been some additional hardware expenses. A navigational keypad and casing were bought. The navigational keypad, cables and inverter hardware components had a cost of 37 dollars while the casing cost was $13 dollars and 46 cents. The sum of the additional hardware is 50 dollars with 46 cents. Adding this additional cost to the materials total sums the amount of 229 dollars and 3 cents. Comparing the new total for hardware materials we can see that the cost is still below the total that includes the overhead cost.

| Materials | Total Cost |
|---|---|
| Microcontroller | $10.08 |
| Development Kit | $149.99 |
| LCD | $18.5 |
| | |
| Materials Total | $178.57 |
| Overhead Cost | 95% |
| Total | $348.21 |

Table 1 - Original Hardware Budget for Sparrow

As far as the project goes there has been additional hours worked. Javier has 15 over time hours working with the hardware. Laura is expected to work about 8 overtime hours during the following weeks in order to implement the write-in feature that was requested last week. The 15 hours of overtime of Javier's work have a total cost of $360.6, adding these overtime hours to the personnel cost gives a total of $33,546.07. Comparing the new personnel total with the total expected (the total after the overhead) we can see that the personnel expenses are still below the proposed cost.

| Personnel | Annual Income | Hourly Salary | Weeks on schedule | Hours per week | Total Salary | Salary plus benefits (Social Security and Pension Benefits) |
|---|---|---|---|---|---|---|
| 3 Computer Engineers | | | | | | |
| Laura Cruz | $40,000 | $19.23 | 12 | 30 | $ 6,922.80 | $7,490.47 |
| Angel Vega | $40,000 | $19.23 | 12 | 30 | $ 6,922.80 | $7,490.47 |
| Sylvia Rodriguez | $40,000 | $19.23 | 12 | 30 | $ 6,922.80 | $7,490.47 |
| 1 Project Manager | | | | | | |
| Javier Torres | $50,000 | $24.04 | 12 | 30 | $8,654.40 | $9,364.06 |
| | | | | | Personnel Total | $33,185.47 |
| | | | | | Overhead Cost | 95% |
| | | | | | Total | $64,711.67 |

Table 2 - Original Personnel Budget for Sparrow

# 6. Work Distribution

## 6.1. Work Breakdown structure

The following is the work breakdown structure for the prototype Sparrow and shows the different modules that the project contains with their corresponding subtasks. The work break structure has been modified to reflect the recent changes in costumer desired features, the write-in and a scalability simulation to verify the database is still effective with thousands or more of items within it.

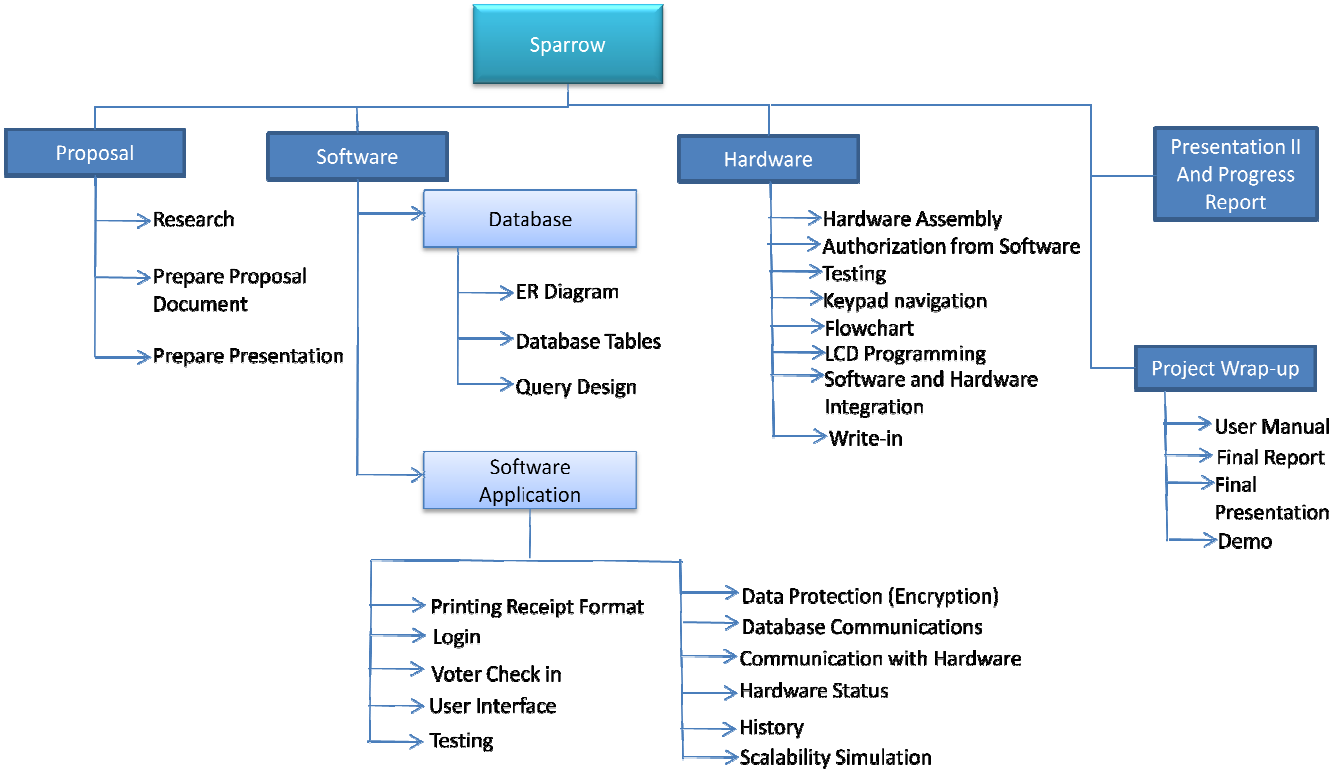



Figure 1 – Work breakdown structure

## 6.2. Task division

The following are the tasks divided upon the JSAL team according to the Gantt chart. Please refer to the Appendix of this report to view the Gantt chart.

### 6.2.1. Javier Torres Santiago

Phase 1:

- Integration of hardware components
  - LCD
  - Inverter
  - Msp430FG4619 microprocessor
- Firmware Initialization of LCD screen
- Provide initial test screen to prove LCD functionality
- Test the administrator application

Phase 2:

- Optimize code to allow space for additional write-in feature
- Implement all the eVote screens
- Teach Laura and Sylvia the code syntax for implementation of phase 2 tasks
- Test Angel's encryption and decryption code

Phase 3:

- Integrate the UART
- Add firmware code to interact with computer administrator software
- Solder all hardware components together to store into casing
- Test Angel's tasks of the scalability of eVote simulation

### 6.2.2. Laura Cruz Rodriguez

Phase 1:

- LogIn Form
- Design & Create Voting System Tables
- Test User/Kiosk Tables

Phase 2:

- Write-In Design & Development
- Test Screen Navigation

Phase 3:

- Lock Sparrow Device
- Activate Sparrow Device
- Test Screen Corroboration & Additional Navigation

### 6.2.3. Angel Vega Cortes

Phase 1:

- Create Voter and Kiosk tables in the database.
- Implement a database connection test application in C#.
- Design the Sparrow ER Diagram in cooperation with Laura Cruz.
- Coordinate the installation necessary software.
- Test the Voting System tables done by Laura Cruz.

Phase 2:

- Database data encryption.
- Test the software application.

Phase 3:

- Scalability Simulation
- Store voting summary into database.
- Implement Sparrow computer driver.
- Test the software application.

### 6.2.4. Sylvia Rodriguez Rodriguez

Phase 1:

- User Interface Design (Admin Form Design)
- Printing Format of receipt
- LCD Test Screen Corroboration

Phase 2:

- Administrator voter check
- Voter check in with id implementation
- Check in history
- Keypad Navigation
- Budget adjustments
- Sparrow Device Screen Corroboration

Phase 3:

- Device Status
- Printing completed with barcode text conversion
- Database voting summary corroboration

# 7. Javier Torres

## 7.1. Phase I

### 7.1.1. Microprocessor

The microprocessor used for the eVote project is the MSP430FG4619 as seen in Figure 2. This microprocessor was chosen due to the fact that it has 100 pins allowing us more room to allocate additional pieces of hardware without the worry of not having ports available. Another reason this microprocessor was chosen was due to the UART (Universal Asynchronous Receiver/Transmitter) pins it includes which is needed for communication with the eVote computer software.

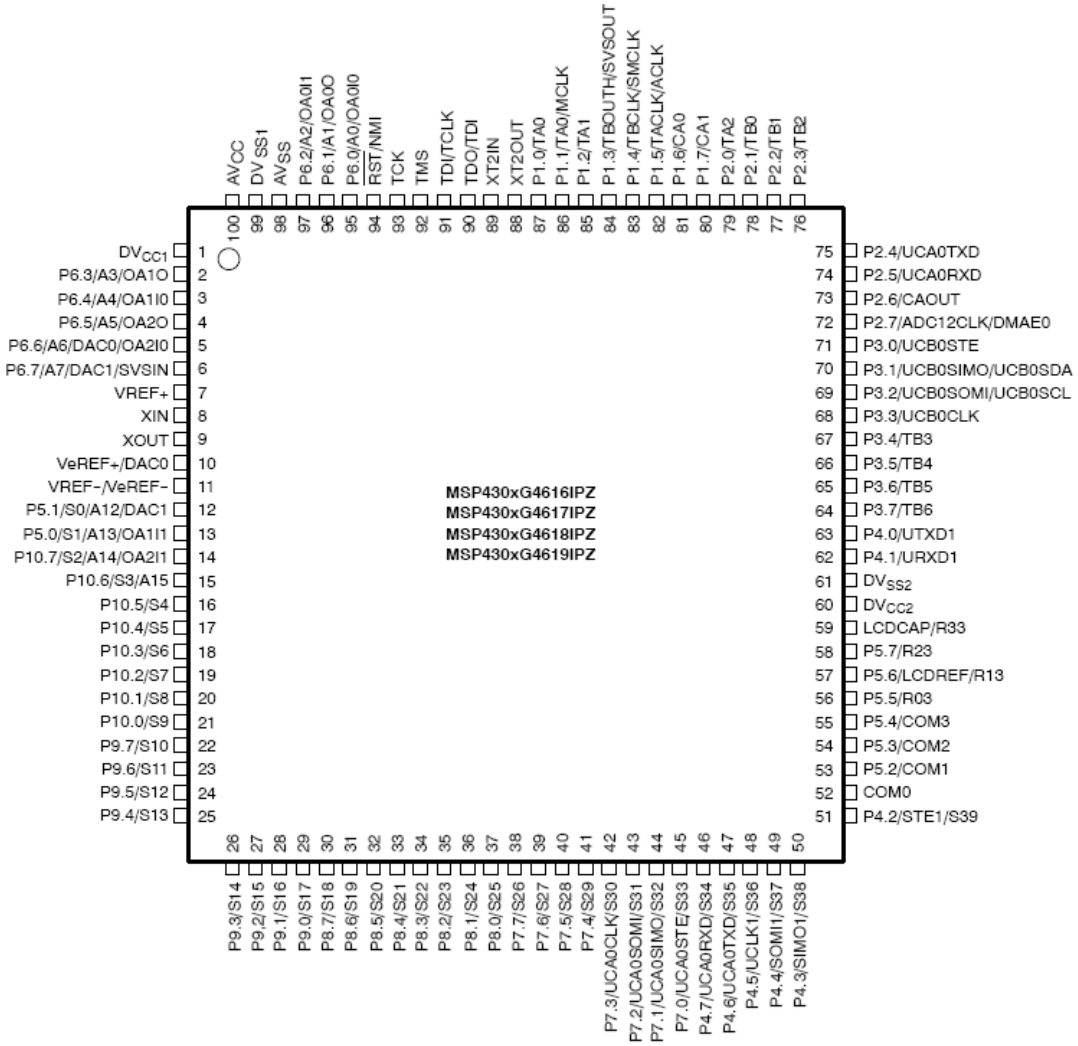


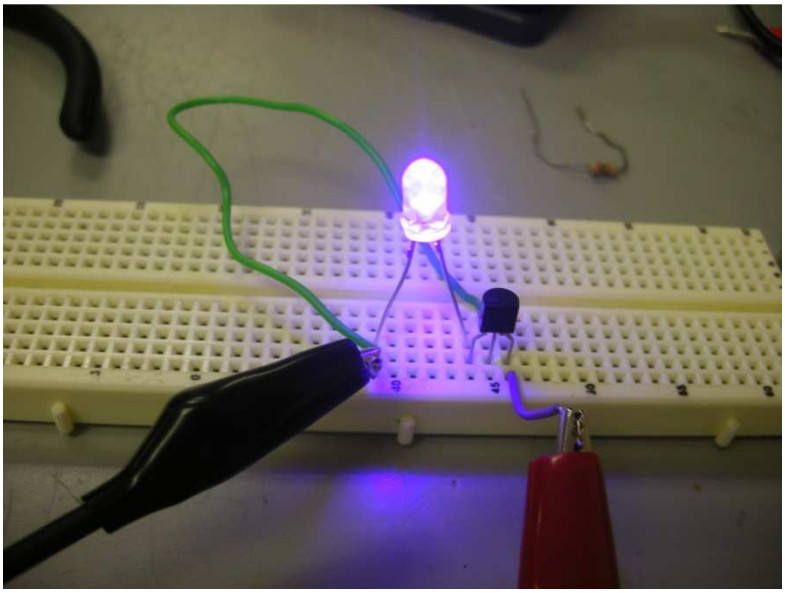

Figure 2 - MSP430FG4619 Pin Layout

Additional soldering was required since the microprocessor development board came with the pin slots separately. The program manager soldered the development board as seen in figure 3.


Soldering - Figure 3

### 7.1.2. Regulators

Tests were done to learn the behavior of regulators and if they truly output the voltage we wanted for the hardware components safely. The tests were done with LED (Light emitting Diodes). JSAL acquired 3.3 and 5 volt regulators. The 5 volt regulators were intended to be used for the LCD screen which requires an input voltage of 5 volts. The test can be seen in figure 3.



3.3 volt Regulator Test - Figure 4

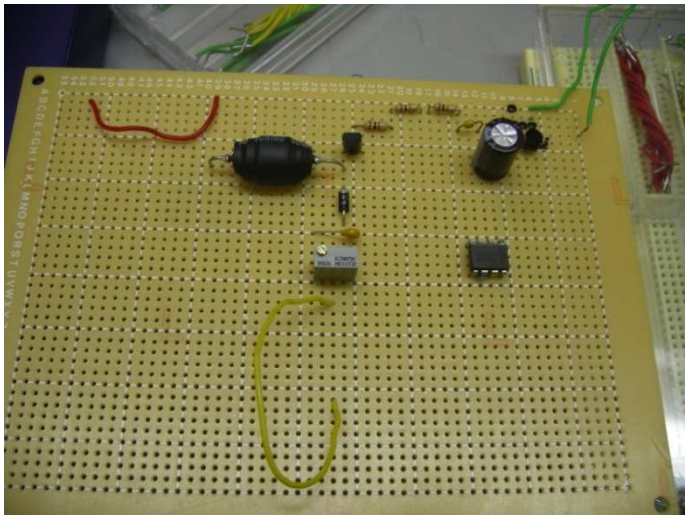
### 7.1.3. Inverter

The inverter circuit was found online and copyright permission was acquired so that we could implement the circuit for the LCD screen which requires negative voltage to control the intensity of the contrast. Instead of purchasing the device for fifty dollars, it was decided to use the designs of the inverter and buy the components separately. The total cost to make the inverter was less than ten dollars. The inverter is of great importance since with the potentiometer, it is possible to adjust the darkness to an acceptable level of viewing to suit each individual's preferences for prototype testing purposes. The contrast for the final phase will be kept at a specific level and will not be adjustable on the user end.

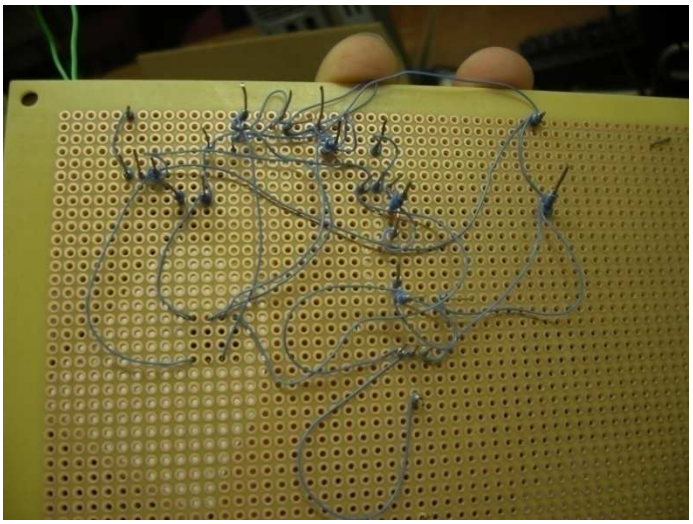
A small modification to the design was made to make sure that the LCD was protected and receiving a healthy voltage level at all times. This modification was the implementation of a voltage regulator. The regulator would only allow 5 volts to pass on into the circuit making sure if by any chance the voltage was to be higher, the regulator would prevent the circuit and LCD from being damaged.

The two kilo-ohms resistor was changed to a 220 ohms resistor to allow a higher level of negative voltage to enter the LCD. With the 2kΩ resistor and potentiometer connected, the voltage would only output -9 volts and the desired output was -10 to -15 volts. Without the potentiometer, the inverter at first gave out -33 volts and once the potentiometer was inserted into the load of the inverter, the voltage lowered significantly to -5.5 volts which was unacceptable. For this reason, the resistor was changed and the voltage fell to the acceptable range of -10 to -15 volts. It was noticed that after the change of the resistor, the voltage levels would return to -5.5 volts even with the resistor change. After careful analysis nothing was found wrong with the circuit, but with the cables used to connect to the voltage supply source. If the cable was moved, the voltage output was lowered inexplicably, but after adjusting the cables and testing with a multimeter an acceptable position for the cables was found and the inverter finally gave out the negative voltage required. Although it was necessary to change the resistor to that of less resistance, the cables used to provide voltage to the hardware was found to be faulty. By turning the potentiometer of the inverter, the output would raise the negative voltage ,if turned clockwise and lower it if turned counterclockwise.

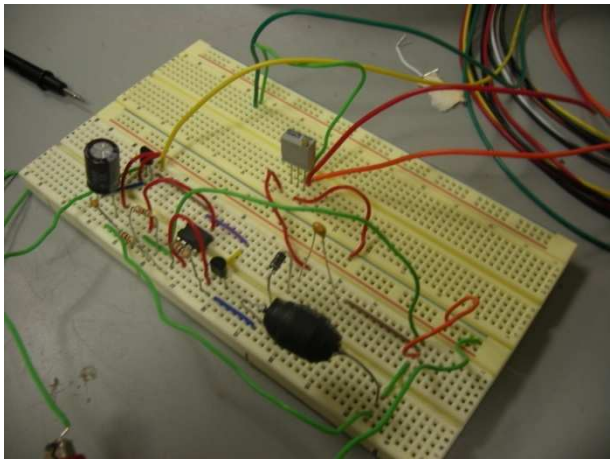
The inverter was then wire-wrapped, but then it was noticed that this would prove to be an inconvenience if any additional circuits need to be tested along with the inverter or if anything needed to be moved around, so it was reverted back to the breadboard.



Wire-wrapped inverter circuit - Figure 5

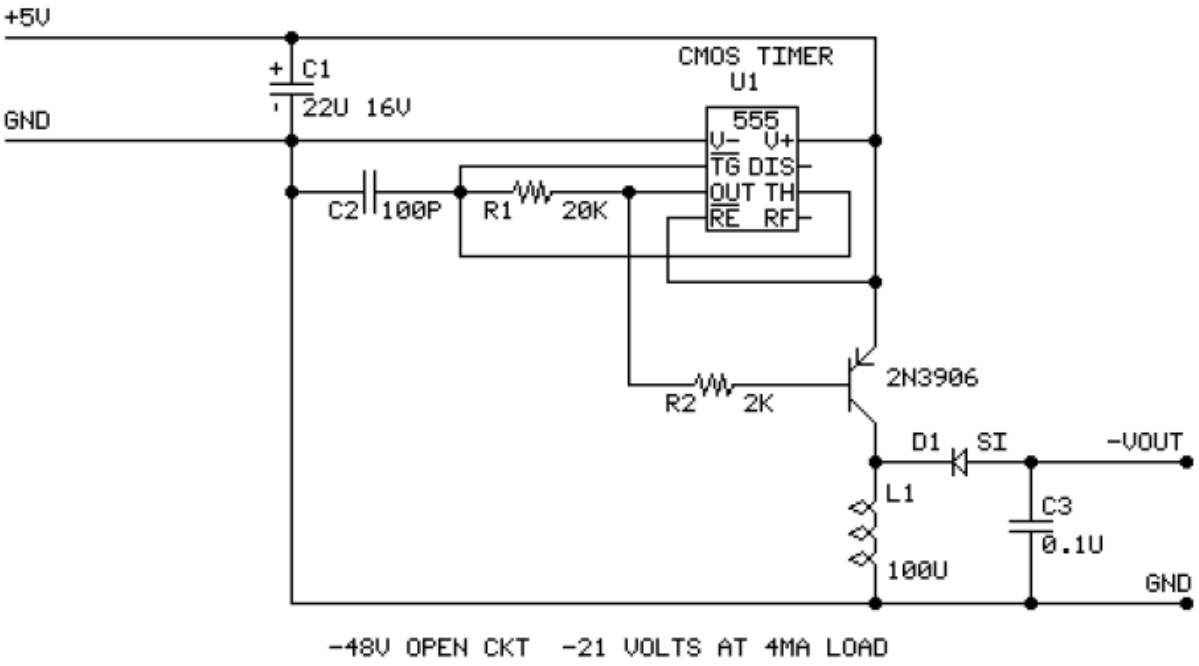


Wire-wrapped inverter circuit - Figure 6



Inverter on a breadboard - Figure 7

Shown in Figure 7 is the design used for the inverter.



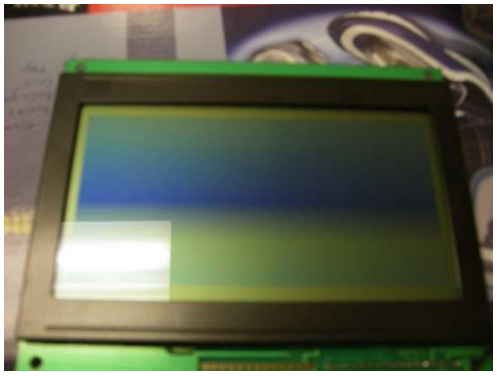

Inverter Design - Figure 8

### 7.1.4. Liquid Crystal Display

The LCD used is called a Hyundai # HG25504 which is 5.8" x 4.58" in size. The size was the main reason for the selection of the LCD since it would easily permit us to implement the voting selection without worrying about size limitations for the text to a certain extent. With this size we can easily implement the 4 voting choices to be implemented in eVote for the governor and the resident commissioner. If we were to choose a smaller display, the amount of text would be severely limited. In the market there are other displays available, but at the cost of hundreds of dollars. With this LCD the prototype expectations can be met. The LCD has its own microcontroller so with the assistance of the datasheet it is possible to access preprogrammed characters which are used to type information onto the screen. The LCD screen was the most difficult aspect of this phase due to the complex initialization of firmware needed. Although the Datasheet provides the items that need to be initialized, the firmware needed to be perfect or nothing would be displayed. After carefully verifying the c language code, the initialization was successfully implemented making the LCD programmable to receive text parameters.

At first the LCD was not responsive to the code and would display horizontal lines and random wave pulses, these results can be seen in Figures 8 and 9 respectively

LCD Horizontal Lines - Figure 9



LCD Wave Pulses - Figure 10

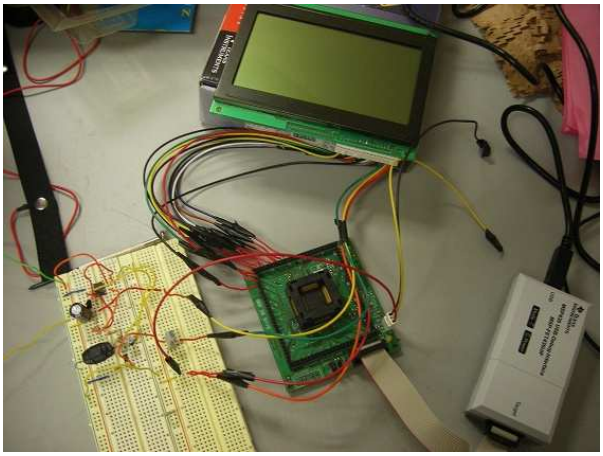
### 7.1.5. Overall Hardware

The hardware connections for the LCD to the microprocessor development board were used on two different ports, ten and nine. Referring to figure 2, the Connections between the LCD and microprocessor are as follows:

- Port 10.5 = Reset
- Port 10.4 = Read
- Port 10.3 = Write
- Port 10.2 = Chip Select
- Port 10.1 = Data Select

- Port 9.7 = Data bit 7
- Port 9.6 = Data bit 6
- Port 9.5 = Data bit 5
- Port 9.4 = Data bit 4
- Port 9.3 = Data bit 3
- Port 9.2 = Data bit 2
- Port 9.1 = Data bit 1
- Port 9.0 = Data bit 0

Figure 10 shows the hardware connections made with each component. The ground of the microprocessor was connected to the common ground of the inverter since the 5 volts the LCD receives comes from the circuit found in the breadboard. The 5 volts being supplied to the LCD are fed first to a regulator and then to the LCD screen. Also seen in Figure 10 is the debugger that is connected with the development kit. This debugger permits computer code to be downloaded into the microprocessor. After the code is downloaded, it can then initialize the LCD and give an output as seen in figure 11.



Complete Phase 1 Hardware connected - Figure 11


Test Screen on LCD - Figure 12

Port ten was used to manage the control signals needed to write information into the LCD microcontroller. Port nine is used for data which is sent to the LCD's memory, while adjusting the control signals to write them into the LCD as data or a command. A logic one for the microprocessor is 3 volts, and a logic zero is approximately 0 volts.

### 7.1.6. Firmware

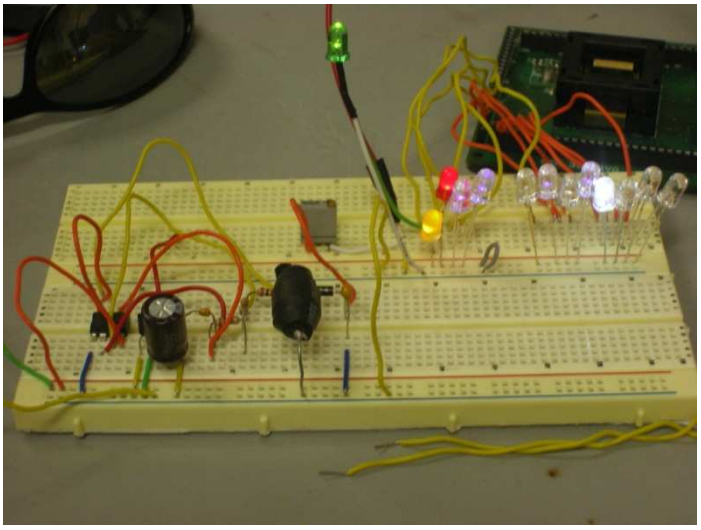
The code was done in C language to allow for a less complex structure of code. A variety of different functions were developed to make the phase one hardware functional. The LCD required a specific setup procedure in order to initialize the system parameters. The setup was completed to permit thirty-two characters per row with a total of 16 rows. This means that a total of 512 characters are allowed to be displayed at one time on the LCD screen. The initialization of the LCD basically adjusts the address on screen where to begin typing text as well as graphic pixels. The LCD has two layers, one

is for text and the other is for graphics. The graphics layer allows a pixel by pixel approach to create images onscreen. During initialization, all memory is erased address by address, which is the most time consuming part of the LCD initialization firmware.

For the testing phase, due to some programming errors, a LED test was implemented to verify if the correct system initialization command was being sent as well as if the correct control signals were active when they were supposed to be. Figure 12 shows the LEDs used to verify the control signals and data bits being sent into the LCD. The code was run line by line to verify each the LEDs with the datasheet's hexadecimal value. For example if we want to move the send the address 010C to the LCD, then the following code can be seen in binary with the LEDs:

```
P9OUT = 0x0C;    //00001100    Least significant bytes go first
writeData();      //verify control signals of write, chip select and data select which are altered in
                  //this function
P9OUT = 0x01;    //00000001most significant half of address
 writeData();     //verify control signals of write, chip select and data select which are altered in
                  //this function
```

When sending an address, 16 bits of information is required, this is why the least significant part of the address is sent first and then the most significant afterwards. Port nine contained the data and commands sent to the LCD so it was possible to use the LEDs to verify the binary information being sent. This testing was very useful since many errors were caught such as functions that would send incorrect binary data to the LCD.



LED Firmware Test - Figure 13

## 7.2. Phase II

### 7.2.1. Firmware

Due to the additional need for a write-in, it was necessary to find ways to optimize the code for the eVote firmware. As of phase I, the code was using individual functions to type individual characters and required that each letter be typed individually. For example, to type “hello” the following would be typed:

typeH();

typeE();

typeL();

typeL();

typeO();

This code has been optimized by creating a function called “typeText” which accepts as parameters, the text to be typed and it’s length. The length is very important since the text is saved into an array. In order to complete the typing of the text, the iterations of the switch that is implemented is repeated by the amount of times found in the length of the text to be typed. For example to write “hello” the following would be done:

typeText(“hello”, 5);

Now instead of calling five different functions to type this word, a function now types it for you. Inside the function is a switch which uses character literals as cases. By comparing each item in the character array of "hello" with a switch, it types the corresponding character.

```
Switch (a[i])
{
Case 'h' :
//code for typing the letter h
break;
…etc
}
```

The screens were implemented in both English and Spanish due to recent events; it is now a law to have bilingual ballots. Please Refer to the appendix for the screen shots of the currently implemented screens.

### 7.2.2. Overall Hardware Progress

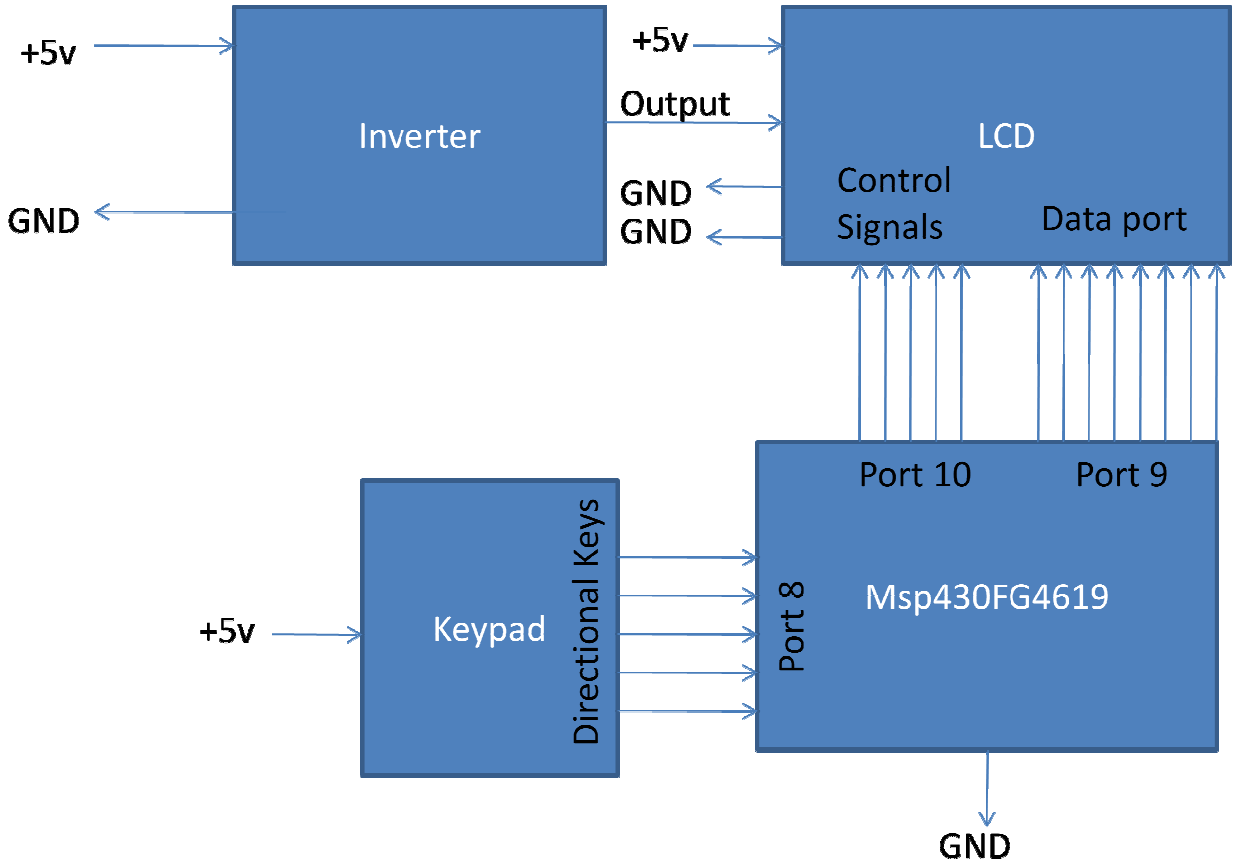



Figure 14 - Hardware connection Progress

The above figure represents the overall hardware progress of eVote as of phase II. The only missing hardware element is the UART (Universal Asynchronical Receiver/Transmitter) cable which is to be implemented in phase III. The ground of each item has been set to a common ground that all the hardware components share although they are various noted ground elements in the above figure. Sylvia was responsible for the keypad implementation so more details on the functionality of this hardware component can be found section 8.2.2. of this report.

# 8. Sylvia Rodriguez Rodriguez

## 8.1. Phase I

### 8.1.1. Administrator Form

The administrator form was created using Visual C#. The administrator form contains three tabs and two menus items. The tabs that can be found on the administrator form are the Registration Tab, the Status tab and the History tab. There is another tab that can be seen on the administration form; the Print Test Tab. The Print Test tab is used for testing purposes and will be deleted for the third phase.

The Administrator form was partially implemented. The registration tab was completed and tested. The user (official worker of the elections) can now verify if a voter has voted or not at the current elections and if the voter is assigned to that particular unit and precinct.

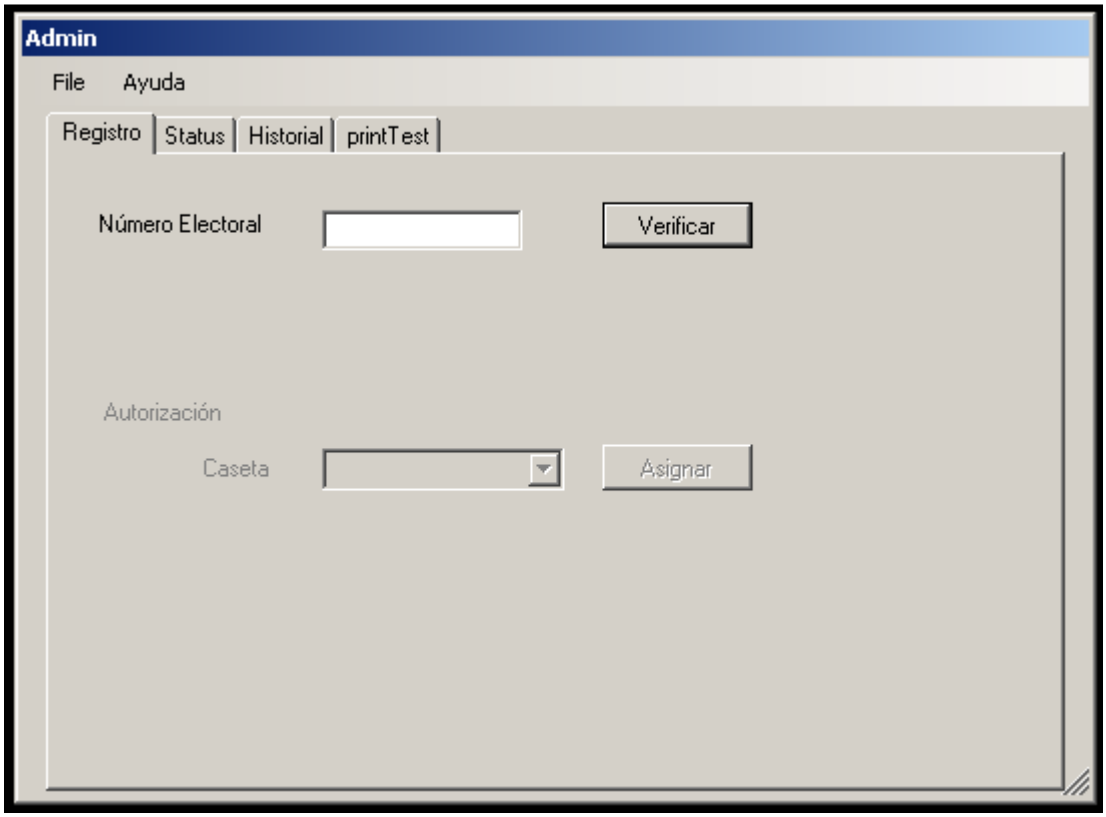



Figure 15 - Registration Tab view

If the voter is authorized to vote, the application allows the user to assign a kiosk where the voter can find the e-Vote device and vote.

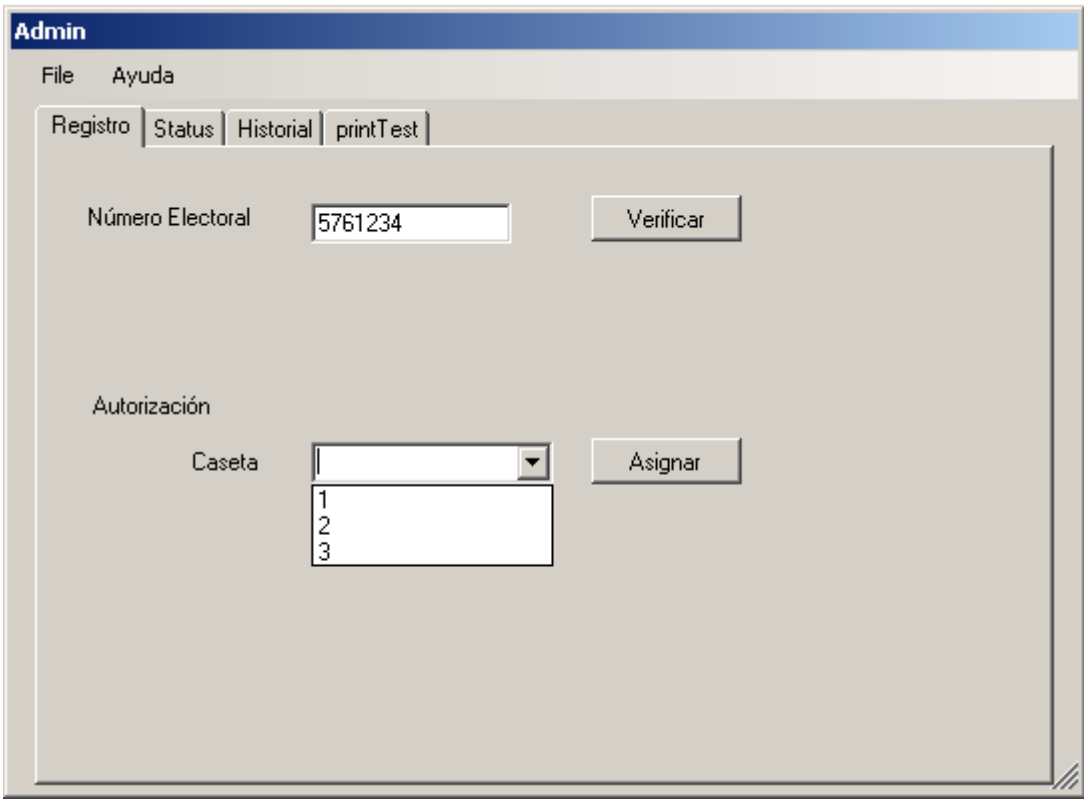



Figure 16 - Assignment of kiosk to a voter

A message will appear indicating that the assignment is successfully completed.

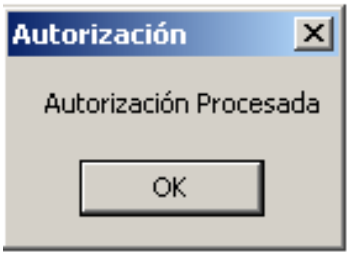



Figure 17 - Successful assignment message

If there are no available kiosks a message will appear indicating there was an error assigning a kiosk to a voter and will allow the user to wait and reassign a kiosk later.

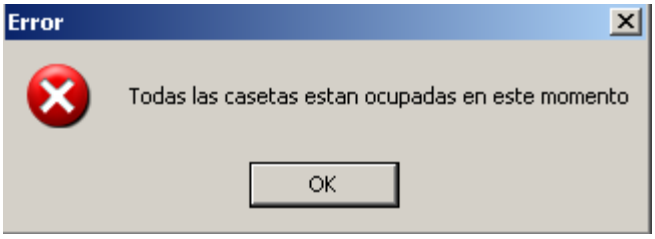



Figure 18 - Message that indicates an error on the kiosk assignment

If the voter is not authorized to vote, a message will appear indicating the reason.

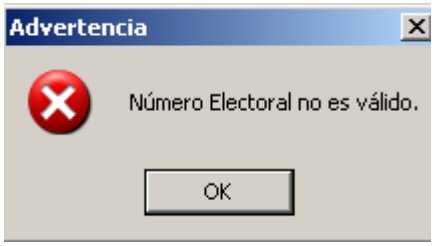



Figure 19 - Not a valid electoral ID

A File Menu was implemented and tested. The file menu allows the user to end the session and log it off while returning to the Login Application.

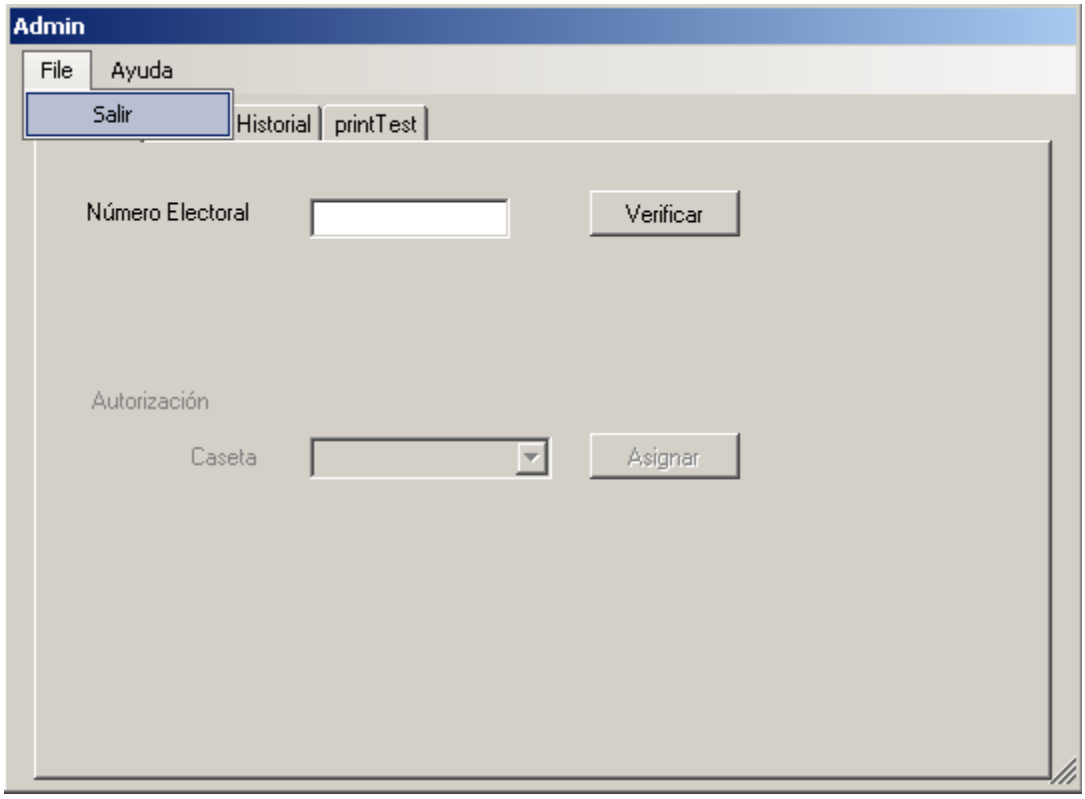



Figure 20 - File Menu

A help menu was also implemented. This menu contains information about e-vote and its copyrights.

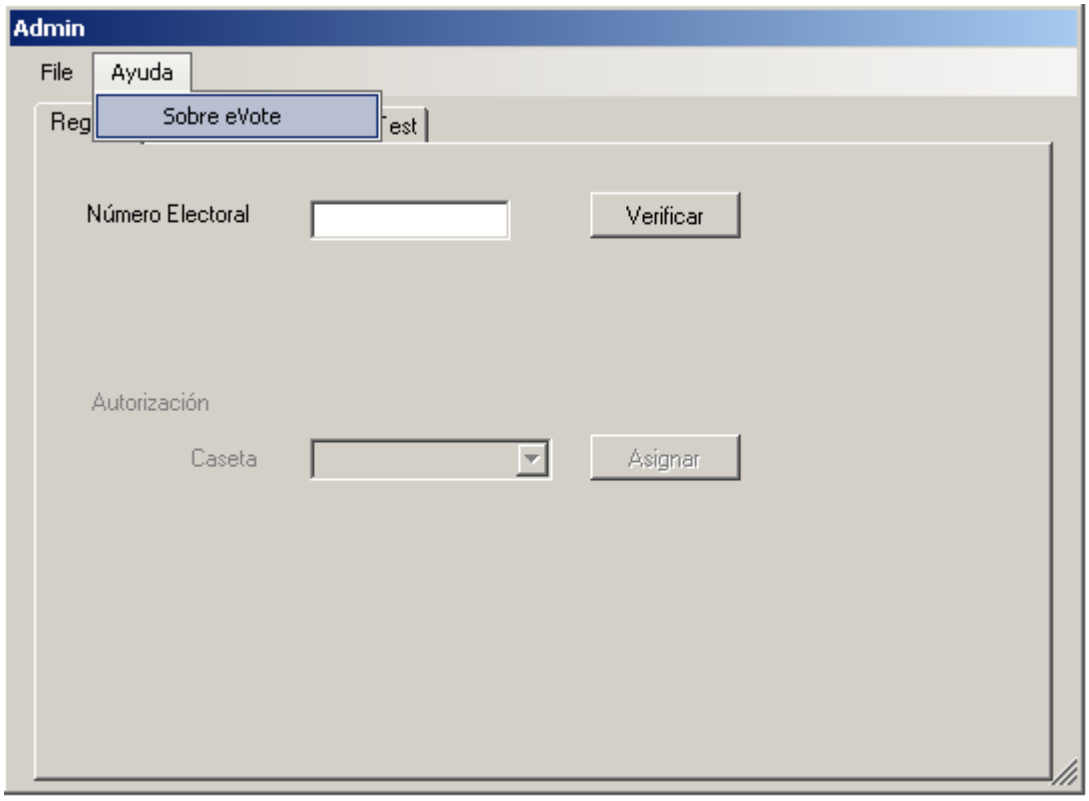



Figure 21 - Help Menu

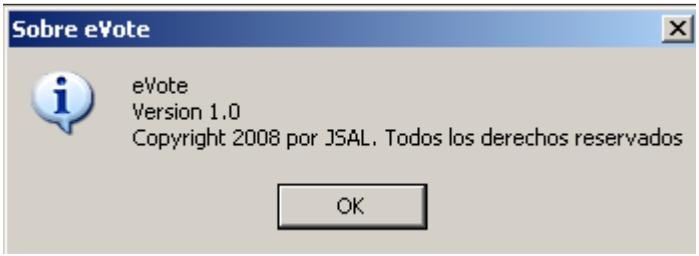



Figure 22 - About e-vote Message Window

A temporary tab has been added to the Admin application; the printing tab. The printing tab allows simulating the voter options and creates a document which contains the results. The document is automatically generated, can be printed and automatically closed without saving any information. The automation of the document allows having a secure and anonymous vote.

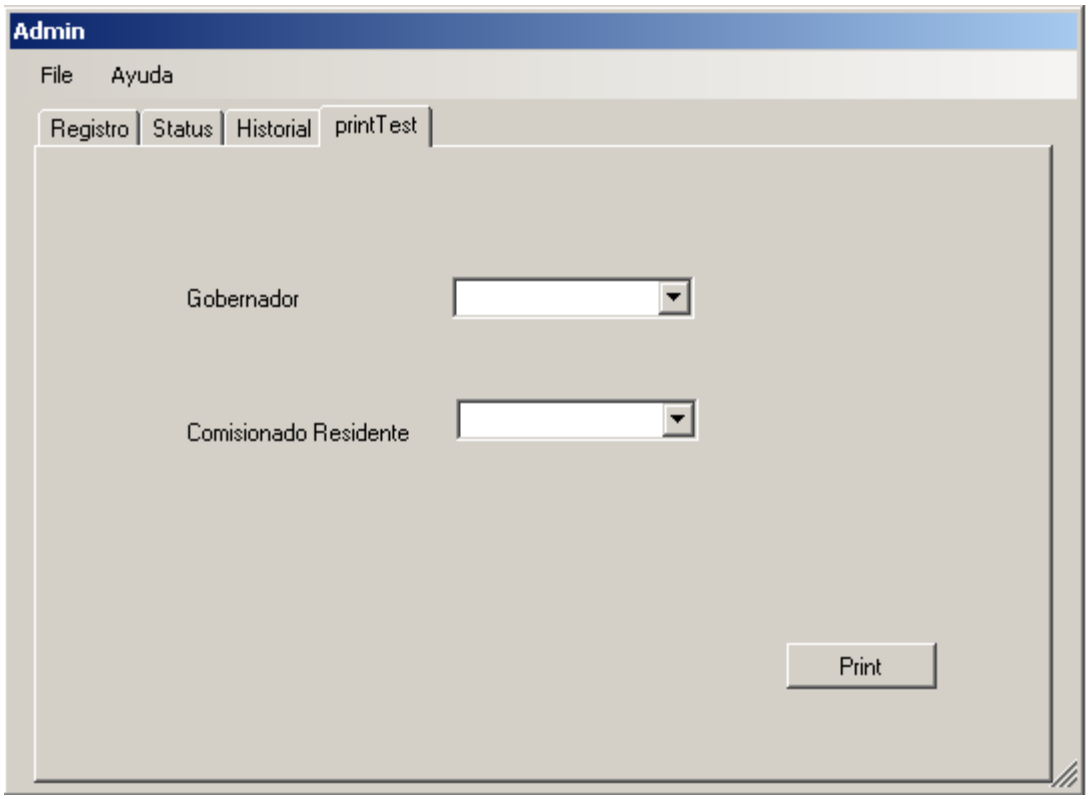



Figure 23 - Temporary Print Test Tab

## 8.2. Phase II

### 8.2.1. Administrator Form

The print test tab was updated with the correct commands to automate a Microsoft Word 2007 document. These changes were needed since last week the computers were changed and the Office version was updated. In order to optimize the behavior of the application and its memory consumption, there were changes implemented on the code to close the Microsoft Application (not only closing the document) after the results document is sent to the printer.

The print tab was also updated to simulate a more real situation of the integration with the hardware. Instead in having drop downs, it has text box that receive strings. The hardware will send three strings to the application, the political party, the governor and the resident commissioner selected by the voter. It will generate the document with the results, print it and closes the application. As now, the application has a thread that works as a timer and after the timer is passed the application is closed. This has to be changed by a process that waits for a response from the printer and after that closes the application and update the database with the votes.

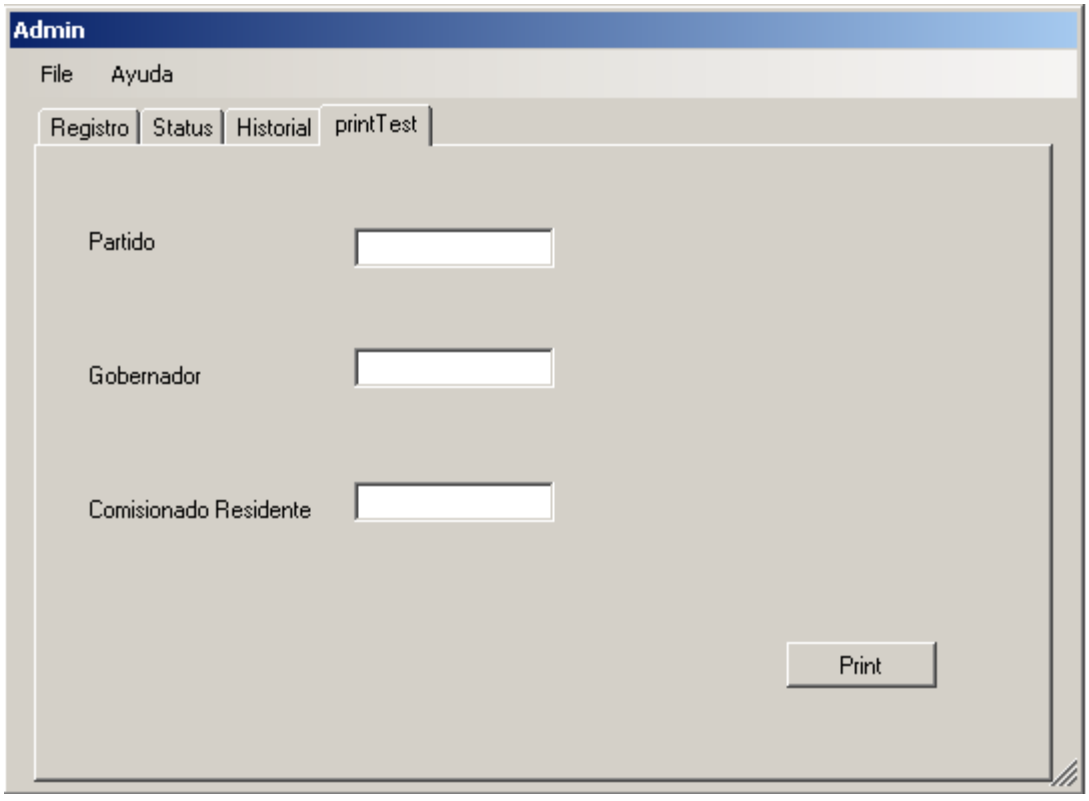



Figure 24 - Print Test tab View

The History Tab is completed and it is being tested. History tab allows the user to access a list of the voters who had used a particular kiosk. If for some reason the kiosk was not working properly, and the votes summated there were not counted, then there is a way of looking to the persons whose vote has not been counted.

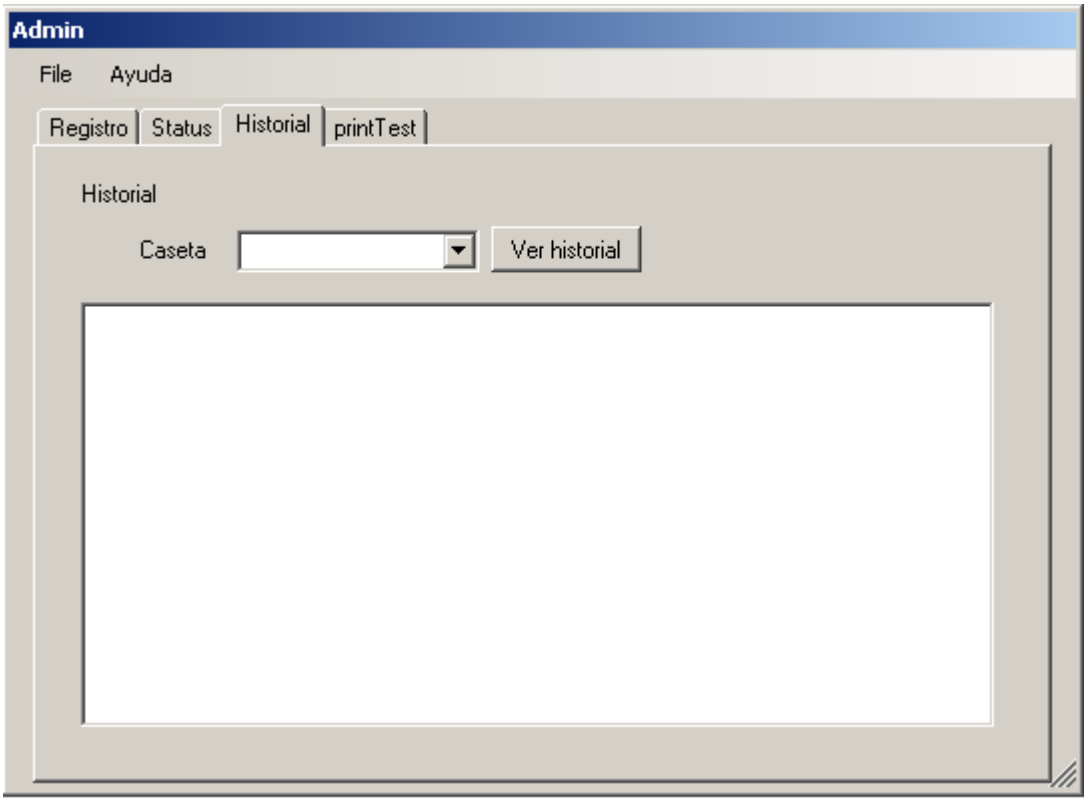



Figure 25 - History Tab View

The status tab requires the UART connection of the device to be completed, this connection will be implemented at the beginning of the third phase and therefore this tab will be implemented for the third phase of this project.

### 8.2.2. Navigation Keypad

The keypad used has 14 pins, from which only 6 are being used. Four of the pins are used for navigation directions (up, down, left and right), one pin is used as a selection button (OK button) and the other pin is used for power voltage. The keypad was tested to identify the behavior and the output voltages. This voltages need to be considerate when integrating the key pad with the micro. A 3.3 volts

voltage regulator was connected in order to regulate the voltage entering the micro. The key pad code is currently being implemented. A pseudo-code of the behavior of the voting program is implemented. Interrupts were considered to implement the behavior of the voting process, but since the interrupts are complicate structures, a logic structure was chosen. Actual code is currently in process. The code for navigating in the language selection screen is already implemented (See Sparrow Flowchart image). As now, there is a problem presenting the cursor, but it is being worked.

The Sparrow flowchart depicts the possible navigational choices that can be taken as well as the possible election choices. When the voter selects mixed, they will have the option to select a party and then a candidate, or if they only wish to vote for the candidate, they can simply choose none of the provided parties and continue on to the candidate selection.

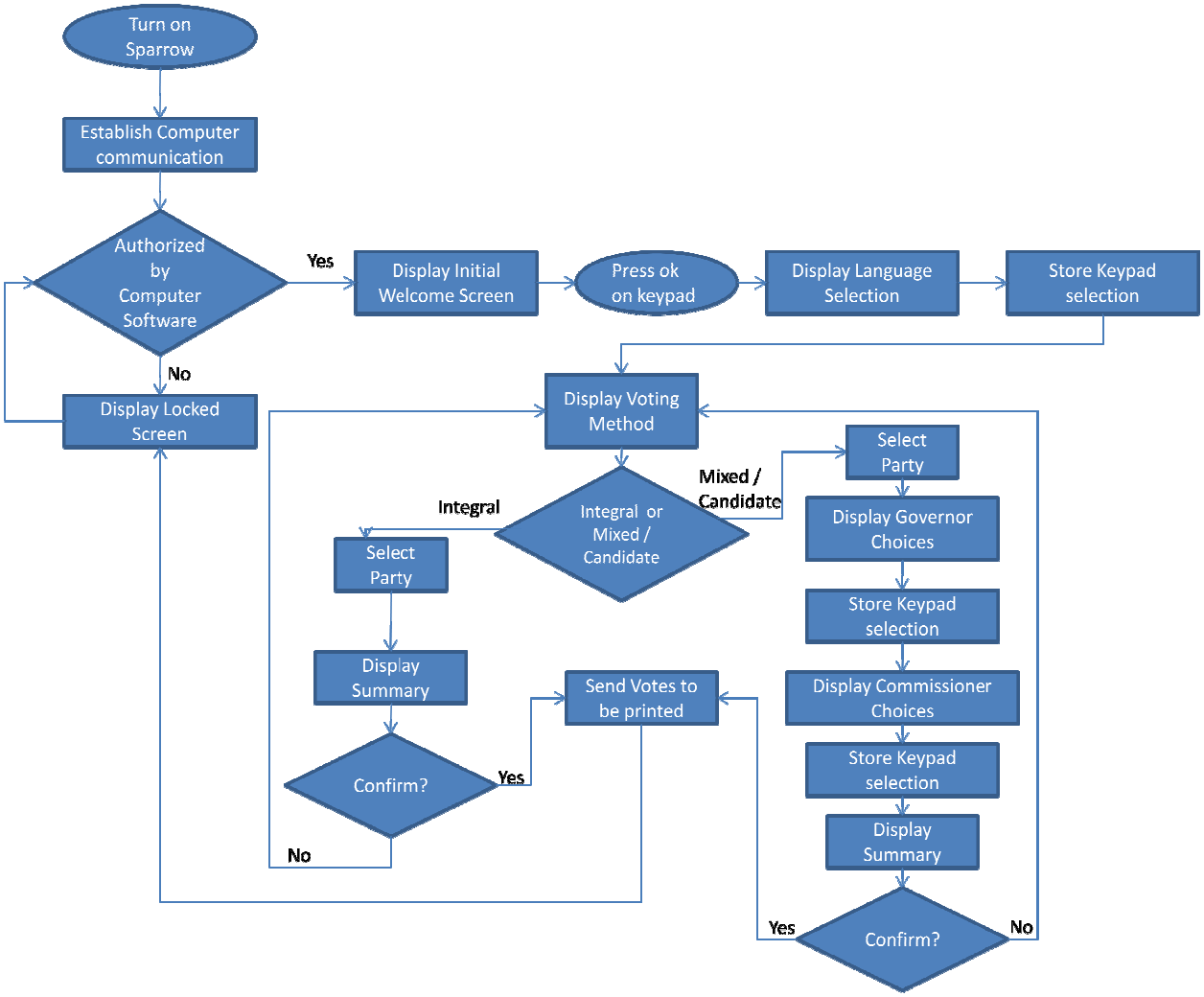



Figure 26 - Sparrow Flowchart

# 9. Laura Cruz Rodriguez

## 9.1. Phase I

### 9.1.1. Website

The JSAL website focuses on providing quick information related to the JSAL team and the evote project. Each page has the team mission and links menu to the left and the page content to the right. The top of the page contains the title of the site and navigation tabs.The site contains five main sections that are accessed through the navigation tabs: home, team, media, progress, and contact. The "home" page contains brief information related to the two versions of the evote project. The "media" page contains an up to date slide show of pictures taken inside the capstone lab. The "progress" page is updated after every significant milestone. Finally, the contact page contains the upr emails for all the team members.

The website was built using a free template from freecsstemplates.com and was edited in Microsoft Frontpage. The site is built to look only the content to the right changes when a tab is clicked. This enables smooth navigation through the site. However, all pages are independent, and clicking a tab link will direct the user to a new site that looks very similar to the previous one. This effect could also be achieved by using HTML frames, but since this template was built by a third party, editing it to such an extent can cause damages to the look of the template. An advantage of using a css template is that if we wanted to change the color theme of the page, or other theme elements like fonts, we only have to edit the css file located in the directory and the effect could be seen throughout the site.

A flickr account was made in order to store all the pictures that will appear on the media page. Flickr is a very popular online tool for photo storage and photo sharing. An online tool was found that was able to access any flickr account and showcase the pictures online.

### 9.1.2. Changes in Database Design due to change in programming language

During the planning stage of the project it was envisioned that all the software for the project would be written in JAVA. This was the programming language of choice since all four of the team members had more experience in Java than with any other programming language. Also, both members that had experience with databases had done all database related programming with Java.

The tool the team had been planning to use for Java development was Eclipse. An RCP plug in exists in order to make UI development considerably easier in Java. However, during the installation of the

plug in and while browsing tutorials the team decided to consider C# as programming language. From the program manager's expierence, the team learned that C# was considerably easy to use when designing an user interface and making the UI respond to user actions.

An interface is required in order to define how an application will connect to a database. Changing the programming language had its implication regarding the interface required between the database and the application accessing the database. In the case of connecting Java the required interface was JBDC. In the case of C# one of the available interfaces is ODBC(open database connectivity).

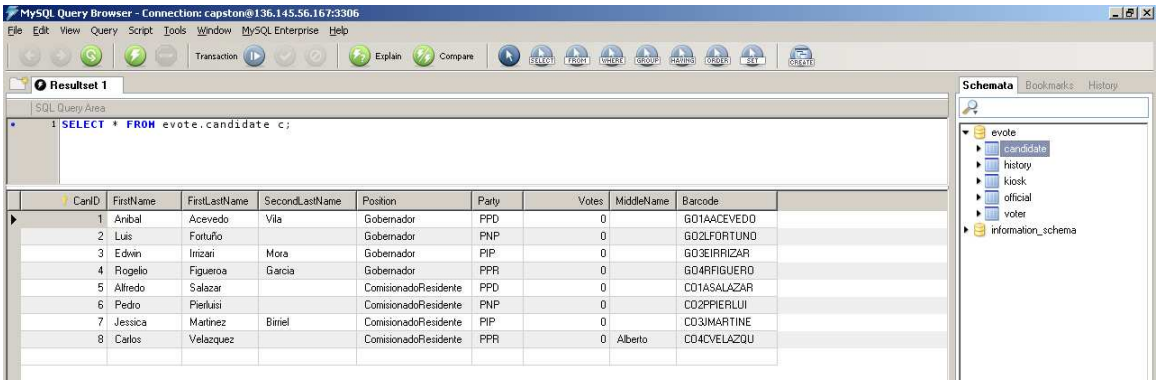
After researching ODBC and seeing that it was not in any way more complicated to install and use than JDBC, it was decided that switching to C# presented more benefits than obstacles.

### 9.1.3. Database Design Decisions

The database being used in this project follows the Entity-Relationship model. With this model we associate database entities with real word "objects" and establish their relationships through the different identity fields.

Throughout the project several changes were made to the original database design. These changes are documented in the eVote Change Document found in the appendix.

### 9.1.4. Database Tables




| CanID | FirstName | FirstLastName | SecondLastName | Position | Party | Votes | MiddleName | Barcode |
|---|---|---|---|---|---|---|---|---|
| 1 | Anibal | Acevedo | Vila | Gobernador | PPD | 0 | | GO1AACEVEDO |
| 2 | Luis | Fortuño | | Gobernador | PNP | 0 | | GO2LFORTUNO |
| 3 | Edwin | Irrizari | Mora | Gobernador | PIP | 0 | | GO3EIRRIZAR |
| 4 | Rogelio | Figueroa | Garcia | Gobernador | PPR | 0 | | GO4RFIGUERO |
| 5 | Alfredo | Salazar | | ComisionadoResidente | PPD | 0 | | CO1ASALAZAR |
| 6 | Pedro | Pierluisi | | ComisionadoResidente | PNP | 0 | | CO2PPIERLUI |
| 7 | Jessica | Martinez | Birriel | ComisionadoResidente | PIP | 0 | | CO3JMARTINE |
| 8 | Carlos | Velazquez | | ComisionadoResidente | PPR | 0 | Alberto | CO4CVELAZQU |

Elections Candidates Table – Figure 27

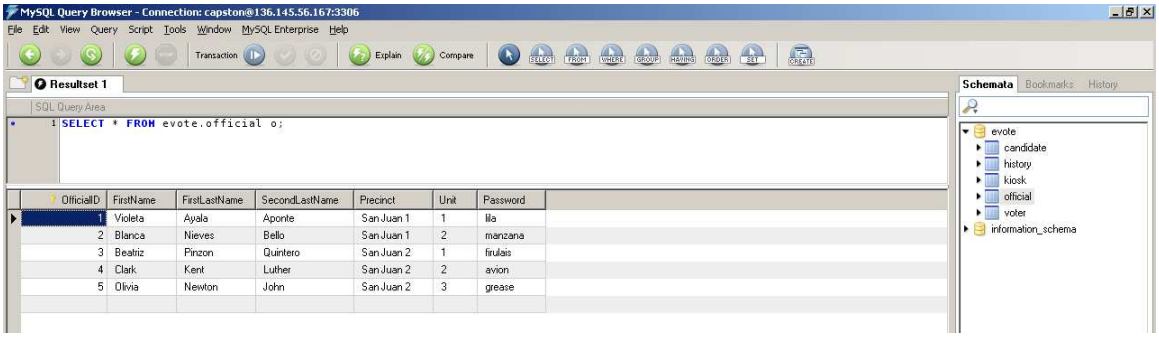



Figure 28 - Elections Official Table

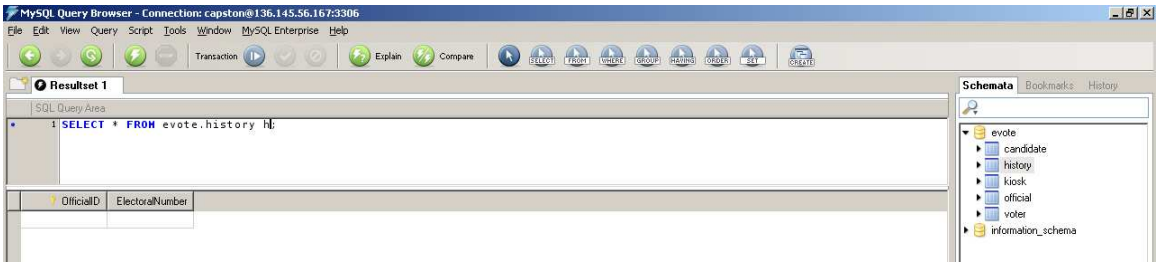



Figure 29 – History Table

### 9.1.5. LogIn Form

The login form is small application that the user (official in charge of voter registration) will use in order to enter the administrator form. The form populates a Precinct list by quuerying the database. Once the user has selected a Precinct, a query is sent to the database in order to acquire a list of Units for that selected precinct. The user then selects a unit and the list of officials is populated. The user should then select his or her name in the list and then type the correct password for his account.

The following images show the completed LogIn form.


Figure 30 - LogIn: User selects Precinct


Figure 31 - LogIn: User selects Unit


Figure 32 - LogIn: User selects their name


Figure 33 - LogIn: User enters password

## 9.2. Phase II

### 9.2.1. Write-In

After the first oral exam, the team received several recommendations. Among them: the write-in LCD feature and assigning votes to a party.

The write in screen is composed of a title that will display “Write-In” or “Nominacion Directa” depending on the language selected, a blank in which the typed name will appear as it is being edited, an alphabet with additional “space” and “backspace” keys, and a “Fin”/”End” key that will allow the voter to cast his or write in vote.

Originally, the alphabet would be displayed in three rows, therefore allowing the user to scroll up, down, left and right through the letters quickly. However, when coding began for this layout, it was realized that the code was becoming increasingly intricate and lengthy. In order to make the program smaller in size, and avoid the chance of running out of space in the microprocessor, the layout was changed to a single line. While this scrolling through this layout is slightly tedious, the team is aiming for functionality over optimization. With the three row layout, it was more probable to encounter errors in programming logic.

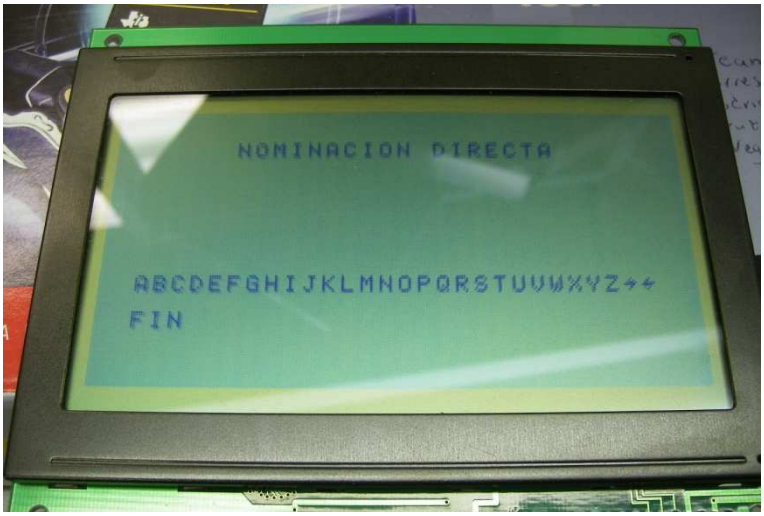



Figure 34 - "Nominacion Directa" Screen


Figure 35 - "Write-In" Screen

According to the Electoral Law of Puerto Rico a political party will online by register for the following elections if voters cast a sign under the political party insignia in at least 7% of the votes for the election. In order to verify this in the eVote system, we created a database table that will allocate a vote to each party when a voter casts an integral or mixed vote and votes for a party.

# 10. Angel Vega Cortes

## 10.1. Phase I

### 10.1.1. Database Tables

The voter table includes the following fields:

- Electoral number
- First name
- Middle name
- First last name
- Second last name
- Already voted (Boolean value)
- Precinct
- Unit
- kID (The ID of the assigned kiosk)

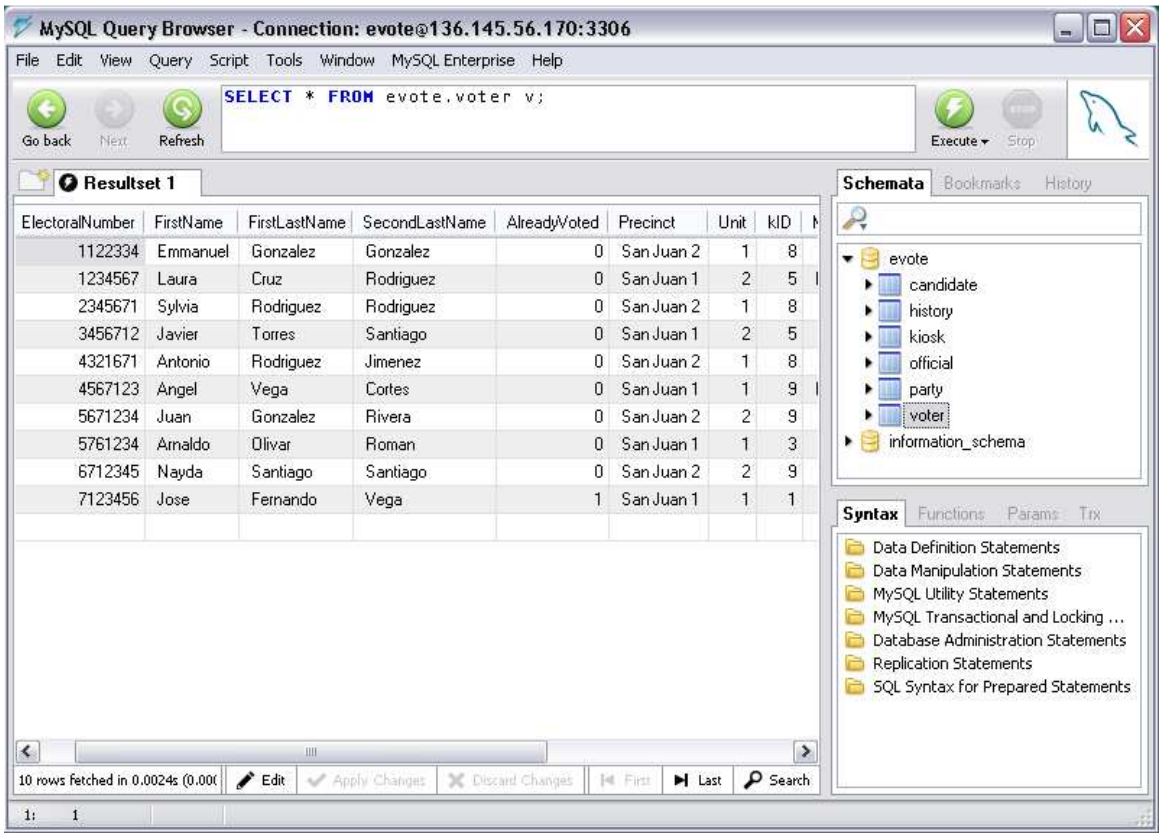



Figure 36 - Screenshot of Voter Table

The kiosk table includes the following fields:

- kID (The unique ID of the kiosk)
- Unit
- Precinct
- Occupied (Indicates if the kiosk is being currently used or not)
- Local ID (The local ID of the kiosk)

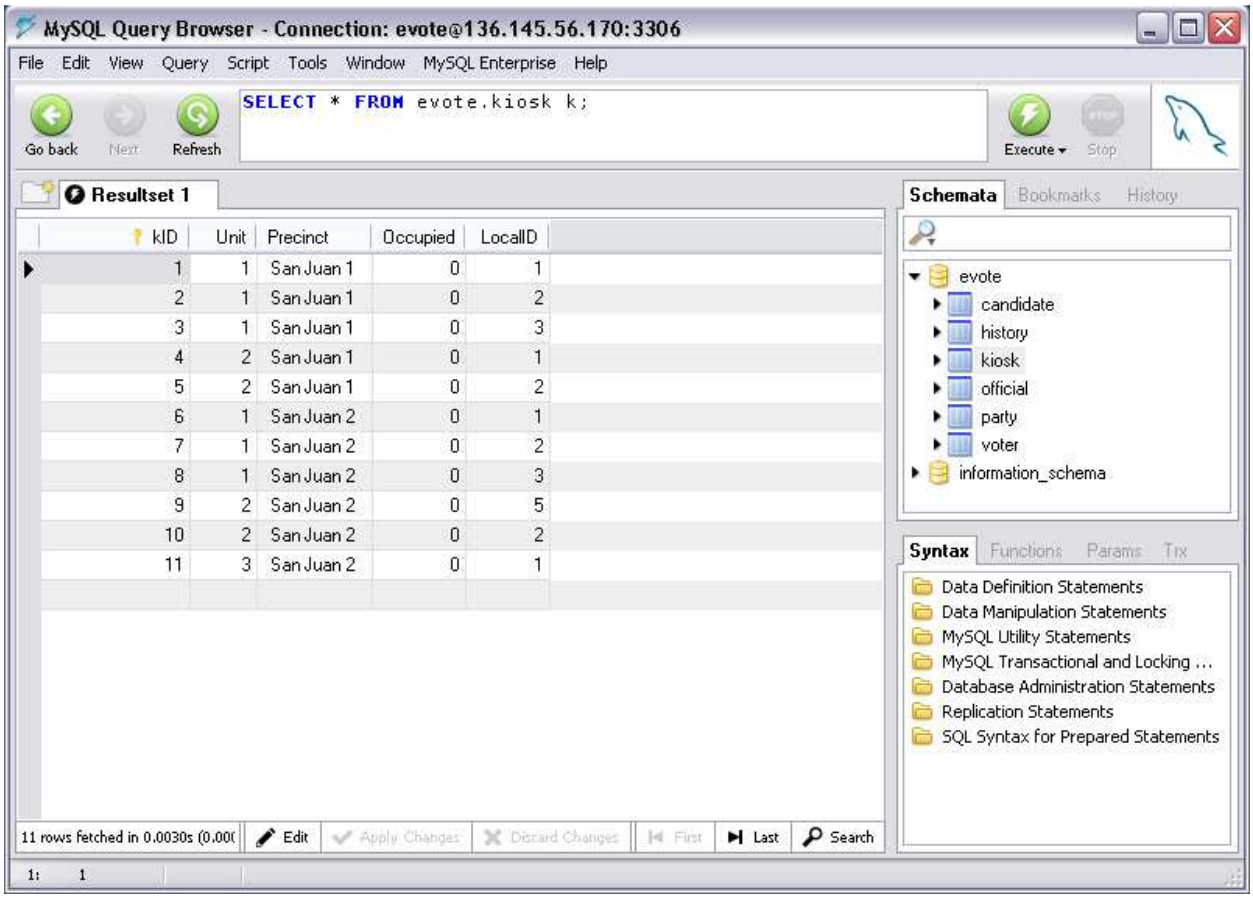



Figure 37 - Screenshot of Kiosk Table

### 10.1.2. Database Connection Test

The database connection test application's purpose was to establish a connection with the MySQL database and test simple queries (i.e. SELECT and UPDATE). Once completed, it was used as an aid in the development for the main application. Presented below are: the screenshots of the test application, output results reflected in our MySQL database, and the C# source code.

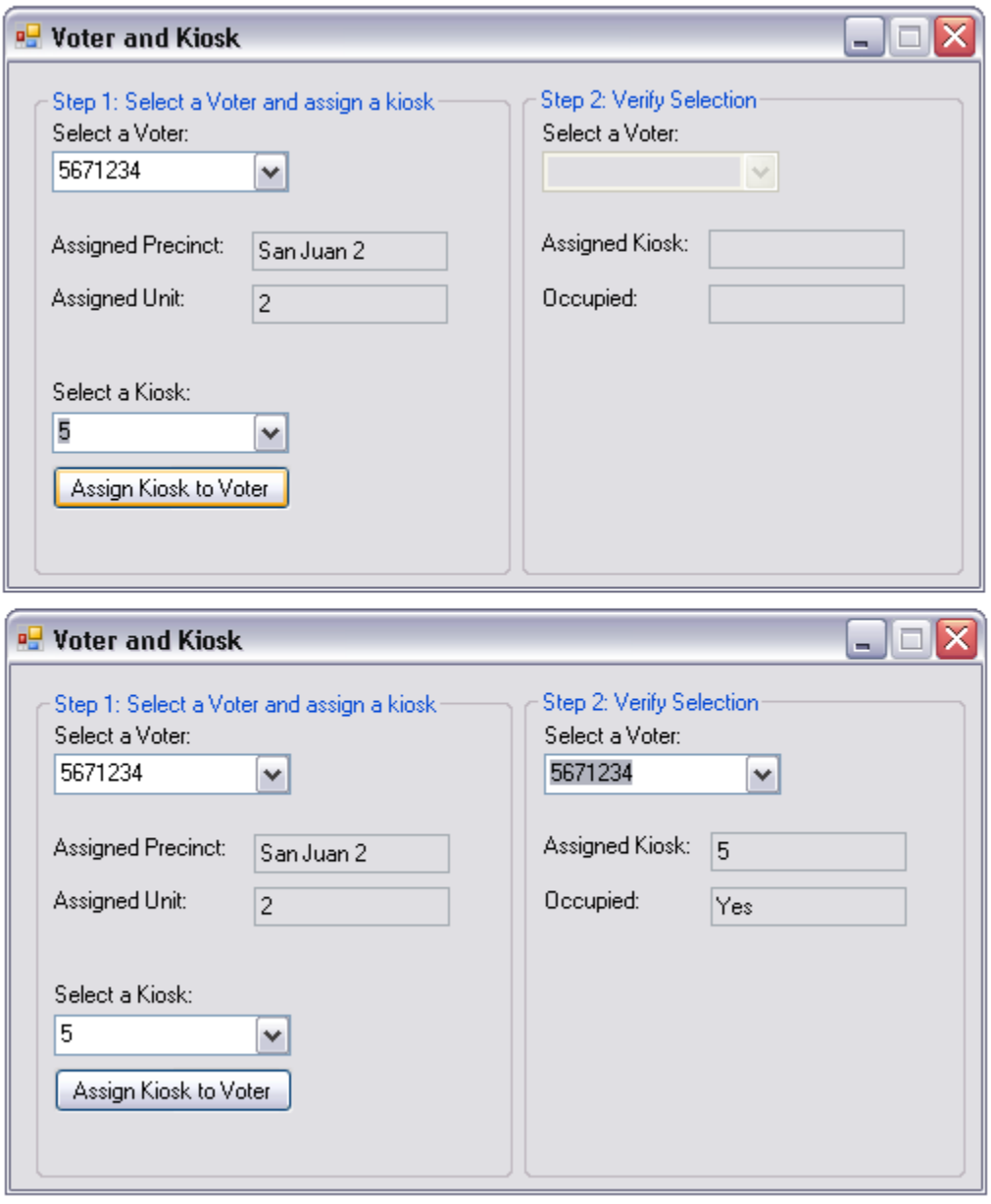



Figure 38 - Screenshot of Database Connection Test Application


Figure 39 - Screenshot of eVOTE database before test run.


Figure 40 - Screenshot of eVOTE database after test run.

## 10.2. Phase II

### 10.2.1. Sparrow ER Diagram

The following is the updated entity-relationship diagram that corresponds to our prototype a.k.a. “Sparrow”.

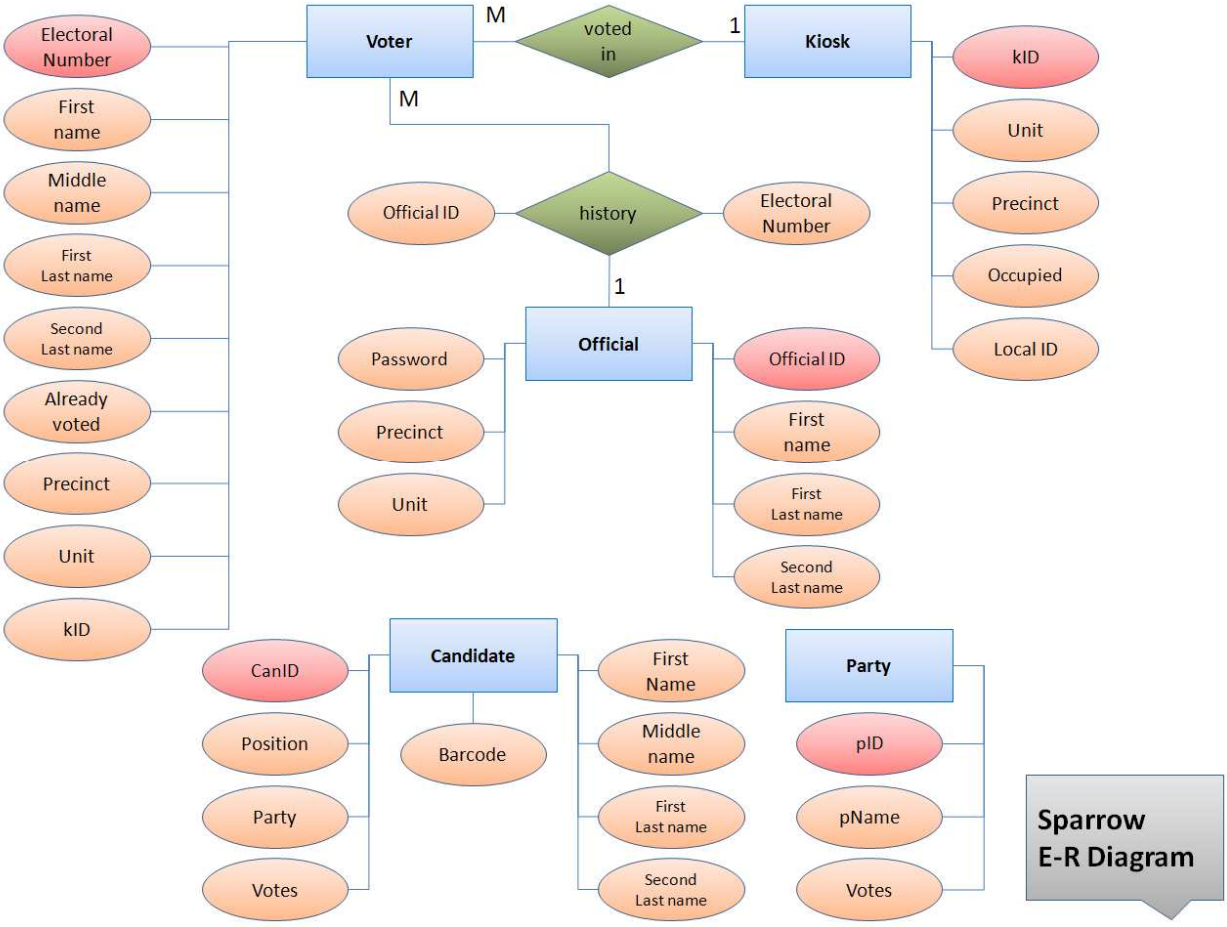



Figure 41 - Sparrow ER Diagram

### 10.2.2. Coordination of installing necessary software

In the beginning of the development of phase 1, the team noticed that some necessary programs and drivers were not installed on our computers. This was just a minor setback in our schedule that was resolved by coordinating with the ECE Help Department, this was done by Laura Cruz and me. Here are the following programs and drivers that were installed:

- Microsoft Visual Studio 2008
- MySQL
- MySQL ODBC 5.1 Driver

## 10.3. Phase II

### 10.3.1. Data Encryption

Since our project manages sensitive data (i.e. voters' personal information, current votes per candidate), encrypting the data in our database is mandatory. The Advanced Encryption Standard (AES) algorithm, also known as the Rijndael algorithm, was chosen because it's the most popular algorithm used in symmetric key cryptography. Finally, it is also approved by the National Security Agency [1] and the National Institute of Standards and Technology [2].

We currently are using several encryption functions that are already integrated in MySQL; listed below are the functions in current use with their respective description taken from the MySQL 5.0 Reference Manual [3].

- PASSWORD(*string*)
  - Calculates and returns a password string from the plaintext password *string* and returns a binary string or **NULL** if the argument was **NULL**.
- AES_ENCRYPT(*string* , *key_string*) and AES_DECRYPT(*encrypted_string* , *key_string*)
  - They allow the encryption and decryption of data using the official AES algorithm, previously known as "Rijndael". Encoding with a 128-bit key length is used, due to the fact that it is much faster and it is secure enough for most purposes. AES_ENCRYPT() encrypts a string and returns a binary string. AES_DECRYPT() decrypts the encrypted string and returns the original string. The input arguments may be any length. If either argument is NULL, the result of this function is also NULL. Because AES is a block-level algorithm, padding is used to encode uneven length strings and so the result string length may be calculated using this formula:

$$16 \times \left[ trunc\left( \frac{string\_length}{16} \right) + 1 \right]$$

If AES_DECRYPT() detects invalid data or incorrect padding, it returns NULL. However, it is possible for AES_DECRYPT() to return a non-NULL value if the input data or the key is

invalid. AES_ENCRYPT() and AES_DECRYPT() can be considered the most cryptographically secure encryption functions currently available in MySQL.

Finally, we conclude with screenshots of the testing done so far using the AES algorithm.

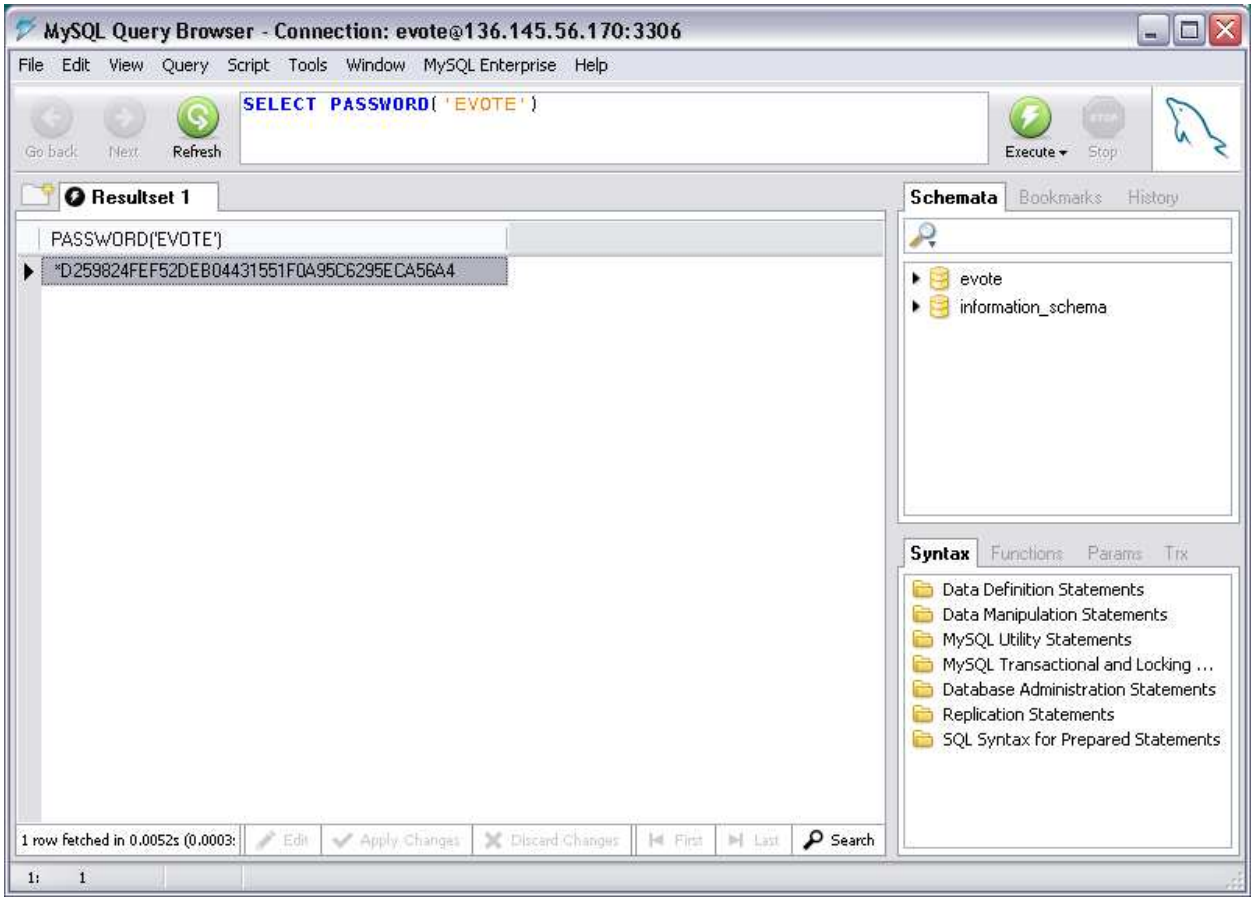



Figure 42 - Demonstrating the PASSWORD(*string*) function.

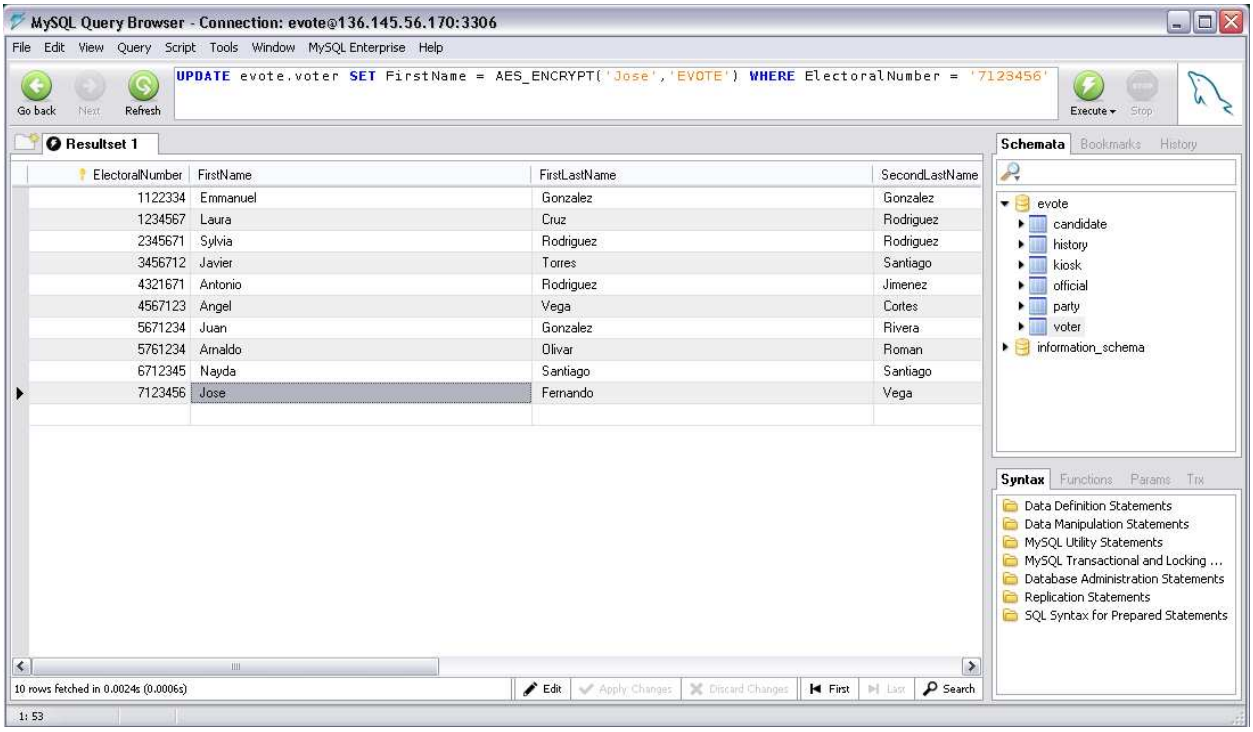



**Figure 43 - Before executing the AES_ENCRYPT(*string* , *key_string*) function.**

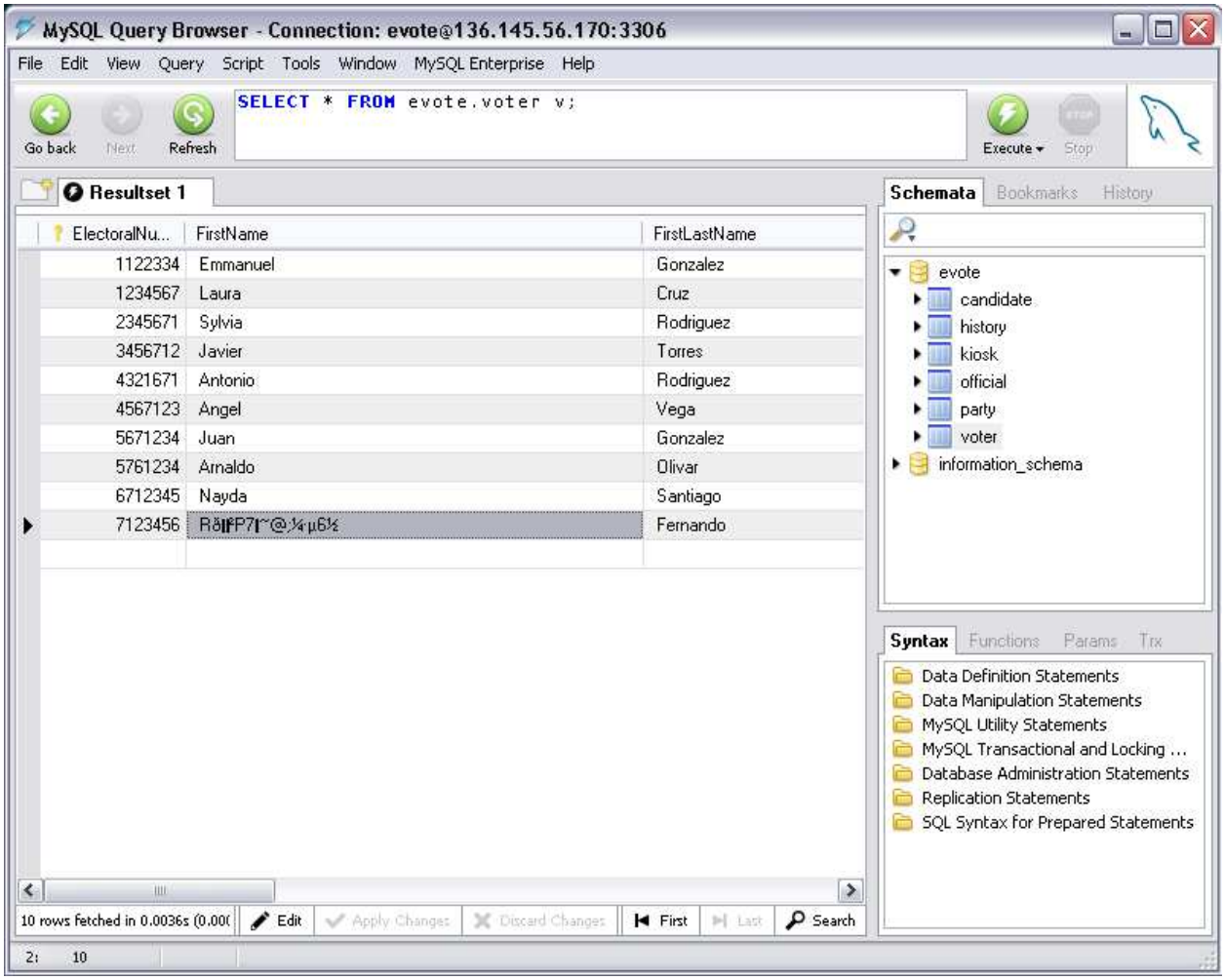



**Figure 44 - After executing the AES_ENCRYPT(*string* , *key_string*) function.**

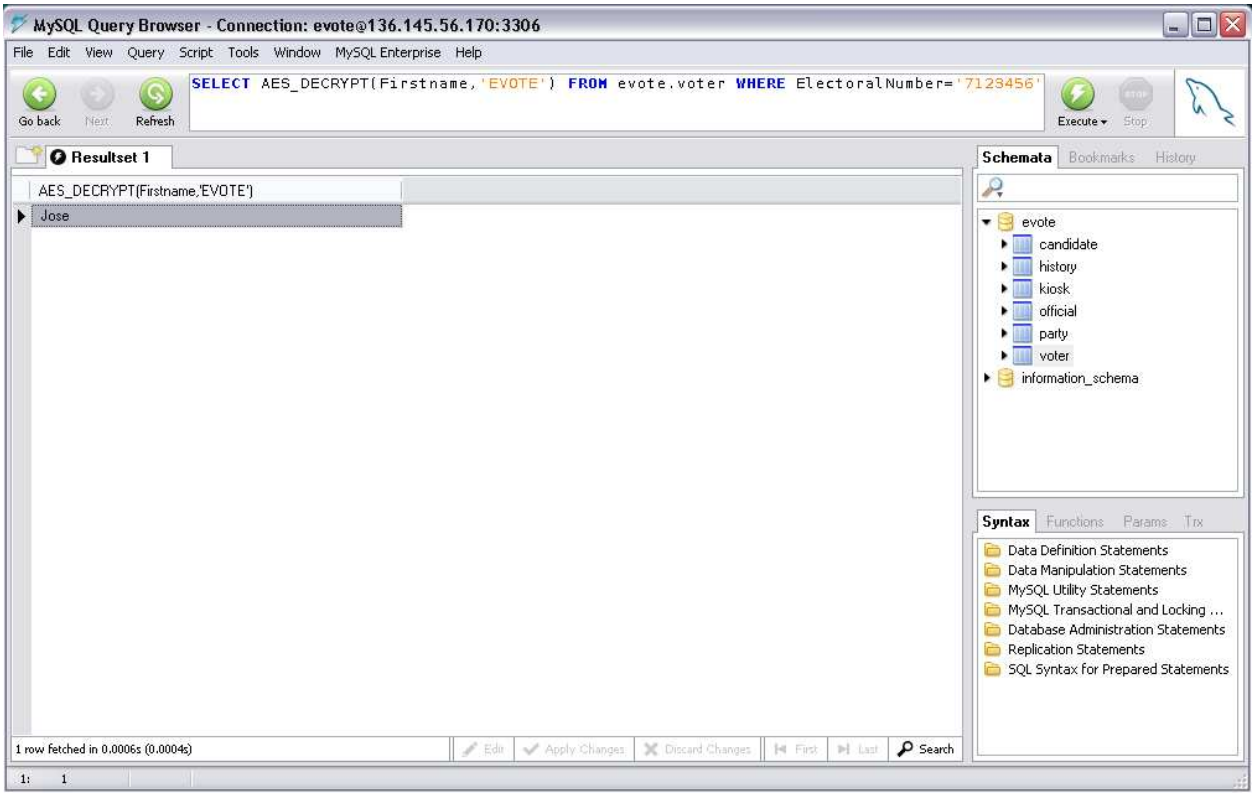



**Figure 45 - After executing the AES_DECRYPT(*encrypted_string* , *key_string*) function.**

## 11. References

[1] "Advanced Encryption Standard." Wikipedia. 24 Oct 2008. Wikipedia. 27 Oct 2008 <http://en.wikipedia.org/wiki/Advanced_Encryption_Standard>.

[2] "Advanced Encryption Standard." Computer Security Division: Computer Security Resource Center. Nov 2001. NIST. 27 Oct 2008 <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[3] "Encryption and Compression Functions." MySQL 5.0 Reference Manual. 2008. MySQL. 27 Oct 2008 <http://dev.mysql.com/doc/refman/5.0/en/encryption-functions.html#function_aes-encrypt>.

"AES encryption and MySQL." MSDN Forums. 24 Jan 2008. MSDN. 27 Oct 2008 <http://social.msdn.microsoft.com/Forums/en-US/netfxbcl/thread/0e45bd3b-8171-40ce-9cd7-894ea511208c>.

"How to use AES encryption and decryption?." MicrosoftASP.net. 02 July 2007. MicrosoftASP.net. 27 Oct 2008 <http://en.wikipedia.org/wiki/Advanced_Encryption_Standard#Security_of_AES>.

“LCD Initialization and Inverter Design”.<http://www.sover.net/~snowleop/gdisp3/>.

"Database Access (C# vs Java)." MSDN. 2008. <http://msdn.microsoft.com/en-us/library/ms228366(VS.80).aspx>

[2] "Msp430FG4619." Datasheet for Msp430xG461x. April 2006. <*www.**ti-estore.com***>.

# 12. Appendix

## 12.1. Copyright

The voltage inverter design is copyrighted in 2006 by Snowleopard Labs and is used in this project with permission.

## 12.2. Gantt Chart

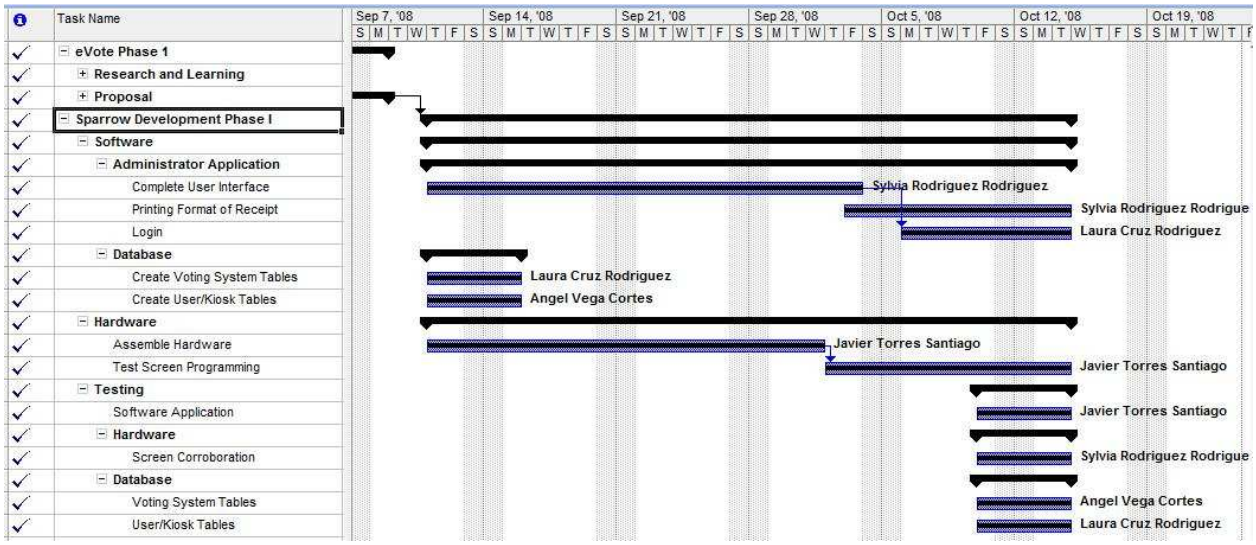



Figure 45 – First Phase Gantt Chart

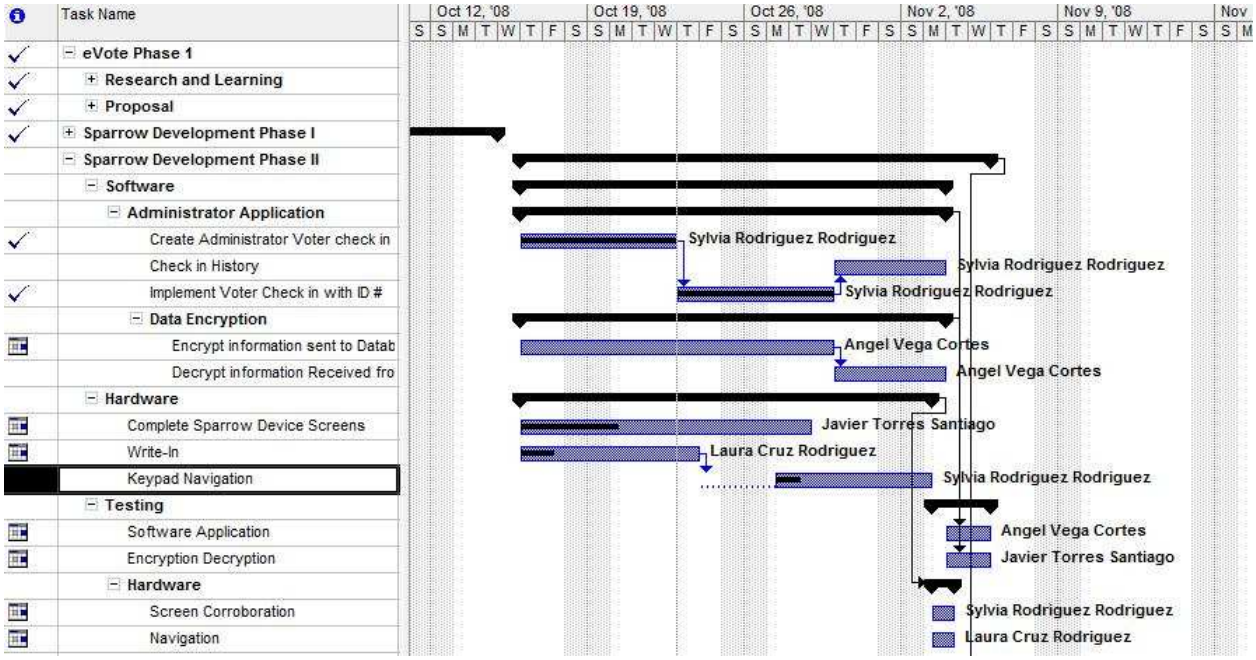



Figure 46 – Second Phase Gantt Chart

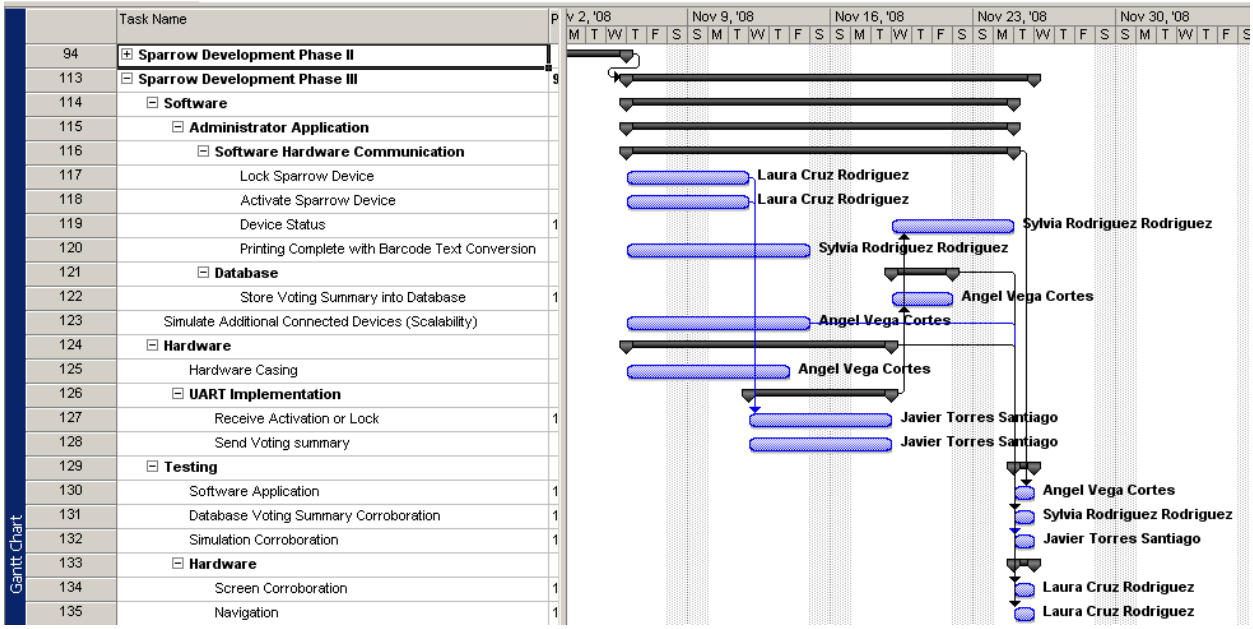



Figure 47 – Third Phase Gantt Chart

## 12.3. eVote Screenshots

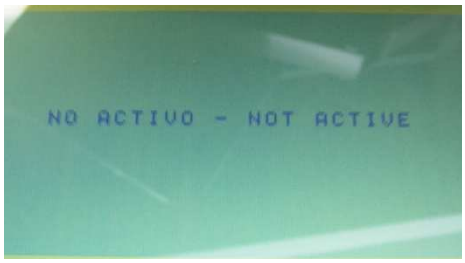



Figure 48 – Not Active Screen

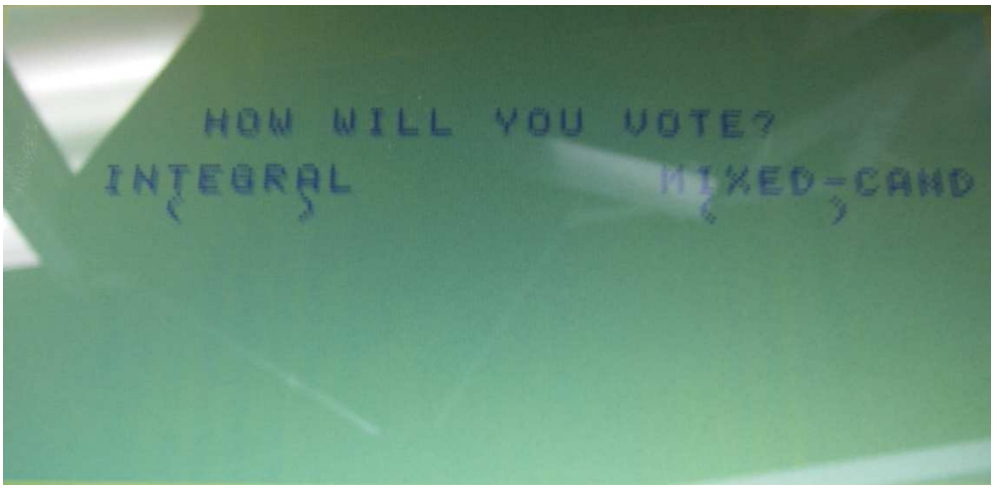



Figure 49 – Vote Method selection Screen

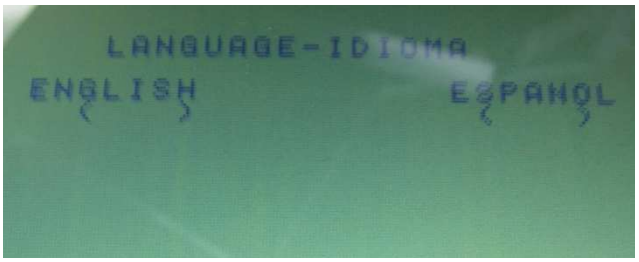



Figure 50 – Language Selection Screen

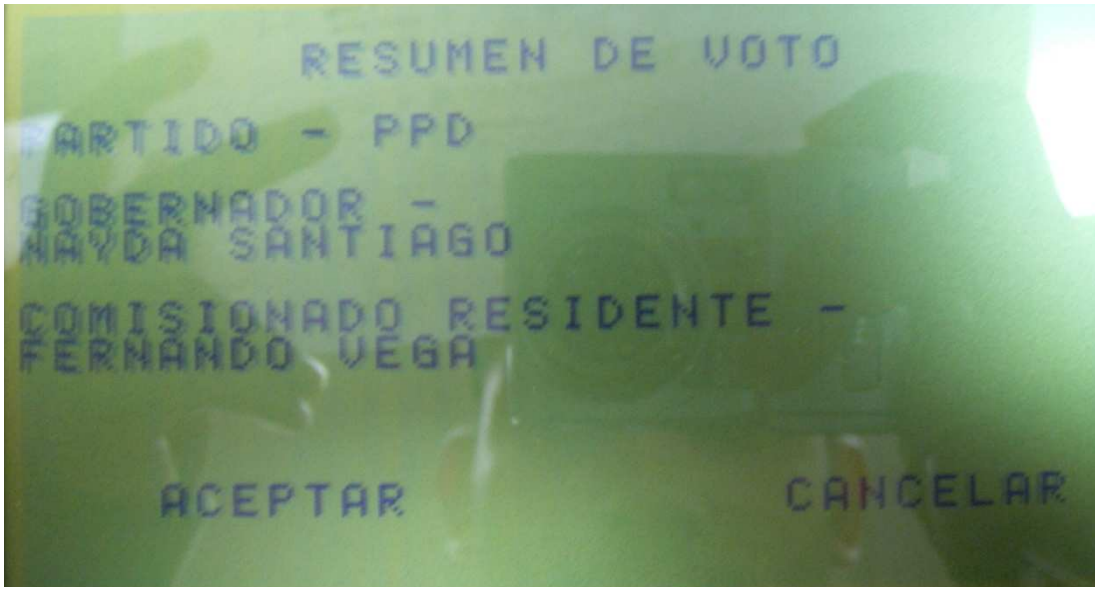



Figure 51 – Voting Summary

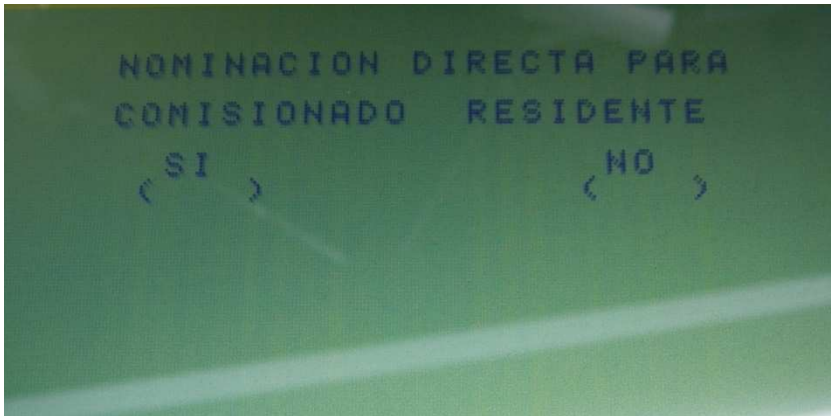



Figure 52 – Write-in Screen for Resident Commissioner

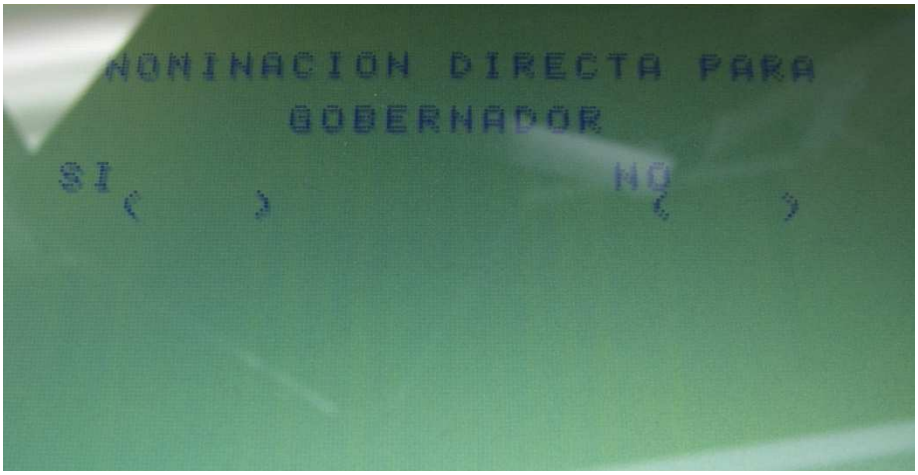



Figure 53 – Write-in Option for governor

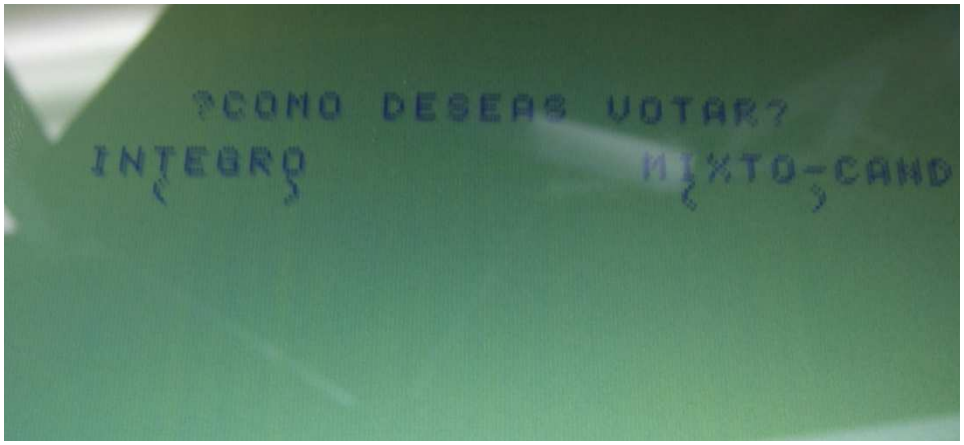



Figure 54 – Voting method in Spanish

## 12.4. Firmware Code

```
/*
Firmware Code eVote
*/

#include "msp430.h"
#include "msp430xG46x.h"
```

```
//**************************************
#define reset 0x20;     //pin 5 of port 10
#define read 0x10;      //pin4 of port 10
#define write 0x08;     //pin 3 of port 10
#define chipSel 0x04;   //pin 2 of port 10
#define dataSel 0x02;   //pin1 of port 10 A0
#define allPins 0xFF    //activate all pins equal to 1
#define displayOff 0x58  //display toggle off, the last bit is 0 negated
                //turning off the LCD with 1
#define displayOn 0x59   //display toggle on, the last bit is 1 negated turning
                //on the LCD with 0
//**************************************

//**********************Sylvia Definitions
#define systemSet 0x40// Initialize device and display
#define sleepIn 0x53  // Enter standby mode
#define scroll 0x44   //Set display start address and display regions
#define csrForm 0x5D  // Set cursor type
#define cgramAdr 0x5C // Set start address of character generator RAM
#define crsDir1 0x4C  // Set direction of the cursor movement
#define crsDir2 0x4D  // Set direction of the cursor movement
#define crsDir3 0x4E  // Set direction of the cursor movement
#define crsDir4 0x4F  // Set direction of the cursor movement
#define hdotScr 0x5A  // Set horizontal scroll position
#define ovlay 0x5B    // Set display overlay format
#define csrw 0x46     // Set cursor address
#define csrr 0x47     // Read cursor address
#define mwrite 0x42   // Write to display memory
#define mread 0x43    // Read from display memory
```

```
//**********************Sylvia Defintions

//*******************Functions Declarations
  void writeCom(void);
  void writeData(void);
  void initLCD(void);
  void delay10(void);
  void clearP9(void);
  void delayWr(void);
  void delayRes(void);
  void clearLCD(void);
  void locked(void);  //locked will lock screen until computer activates eVote
                //With UART
  void ballot1(unsigned int); //first ballot for governor
  void ballot2(unsigned int); //second ballot for resident commisioner
  void confirm(unsigned int); //confirm previous choice
  void sumDisp(unsigned int, unsigned char gov[], unsigned int,
           unsigned char com[], unsigned int,unsigned int);  //final
           //confirmation before printing and UART communications
  void wrScr(unsigned int, unsigned int);
  int enabled(void);
  void typeText(unsigned char a[], unsigned int);
  void language(void);
  void setPos(unsigned char, unsigned char);
  void clearText(void);
  void voteSel(unsigned int);
  void initKeyPad(void);
  void keyPress(unsigned int);
  void wrDisp(unsigned int);
```

```
//*******************Functions Declarations

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;  // Stop watchdog timer

 // initKeyPad();
  initLCD();  //aproximately 15 seconds of startup

locked();
wrDisp(0);
clearText();
wrDisp(1);
clearText();
language();
clearText();
confirm(0);
clearText();
voteSel(1);
clearText();
voteSel(0);
clearText();
wrScr(0,0);
clearText();
wrScr(0,1);
clearText();
sumDisp(0,"javier_torres",13,"pikachu",7,3);
```

```
}
    //display locked screen

//********************Functions


void eVote()
{
  //show first screen
}

int enabled()
{
  //UART acquire information of unlock from computer for phase 3
  return 1;
}

//Sylvia's Code
void initKeyPad()
{
  //hacer los interrupts del teclado

}

void keyPress(unsigned int screen)
{
  //if screen, menu de opciones de teclas
```

```
  // que pasa cuando presiona un boton
  //P8IN   0 000 00  00
  //si quiero verificar   que el P8.4 esta on
  //0000 0100
  int key = 0;
  while(1)
  {

  while(1)
  {
    if(P8IN == 0x04)
   {
     //ok
     key = 3;
     break;
   }
  }

  //execute key command in switch
  }

}


void initLCD()
{
  //Initialize LCD
  P9OUT &= 0;        //Sets P9 to low
  P10OUT &= 0;       //Sets P10 to low
```

```
P10OUT |= reset;    //make reset start off high to not have it active
P10OUT |= read;     //set rd to 1 to disable
P10OUT |= write;    //initialize write to disable

P10DIR |= allPins;  //makes P10 all output using define
P9DIR |= allPins;   //all of P9 is output direction

//add delays if code malfunctions
delayRes();

/*
P9OUT = displayOff;  //turn off display
writeCom();         //Write the command to LCD
*/
P9OUT = systemSet;   //Send system Set command to initialize LCD
writeCom();

P9OUT = 0x30;       //Since the command is set, sending data will
writeData();       //be understood as data for this specific command
                   //unless a new command is issued.  0x30 is for no invert

P9OUT = 0x87;      //Character width   1000 XXXX   changed from 87
writeData();       //8 Sets a two frame (16 line inverts after 16 lines
                   //not desired the 7 represents the character
                   //width (7 = 8 pixels wide)

P9OUT = 0x07;       //Vertical Character height  = 0000 XXXX
writeData();       //same as width, but this value can
```

```
            //get up to 0F unlike the height

P9OUT |= 0x1F;     //Set C/R, address range for one line of the
writeData();      //display from 0-239 ---  32 bytes

P9OUT = 0x23;     //Set line range horizontally (screen edges for line) TC/R
writeData();

P9OUT = 0x7F;     //Set Height of the screen in lines L/F 128 lines in height
writeData();

P9OUT = 0x20;    //Virtual Screen address range   //Low
writeData();
P9OUT = 0;      //High
writeData();     //The data is received as 0020  with the low
            //first and then high.

P9OUT = scroll;  //Send Command for scroll
writeCom();

P9OUT = 0;      //clears the data and begins at
writeData();    //0000h address for layer 1 (text)  low = 00

P9OUT = 0;      //High half of the address so it can have 16 bits of info
writeData();

P9OUT = 0x7F;    // 128 lines according to datasheet table
writeData();
```

```
P9OUT = 0;      //start layer 2 on address 1000 --- Low
writeData();
P9OUT = 0x10;   //high portion of addres
writeData();
//*************************
P9OUT = 0x7F;    // 128 lines for layer 2 (graphics)
writeData();

P9OUT = hdotScr;  //Horizontal Scroll Command
writeCom();
P9OUT = 0x00;     //no change
writeData();

P9OUT = ovlay;   //Overlay
writeCom();

P9OUT = 0;
writeData();

clearLCD();        //function to erase everything on screen

P9OUT = csrForm;   //Command CSForm to adjust cursor properties
writeCom();

P9OUT = 0x04;      //Set width 0000XXXX 5 pixels
writeData();

P9OUT = 0x86;      //Set Height 1000XXXX   7 pixels
writeData();
```

```
  P9OUT = displayOn; //turn on display
  writeCom();

  P9OUT = 0x14;     //turn off layers that are not to be used 0001 0100
  writeData();      //0x14 turns on first 2 layers and disactivates the 3rd
                    //and 4th.  Turns cursor on or off.  14 turns cursor off.
                   //turns on the cursor 16

  P9OUT = cgramAdr; //CGRAM, Starts address of Character generator
  writeCom();

  P9OUT = 0x00;     //send address in high and low start at 0400 ---Low
  writeData();
  P9OUT = 0x04;    //high
  writeData();
}

void delay10()
{
  //delay 10ms for A0 CS high and low activations
  //1ms = 8000  8000*10 = 10 milliseconds
     for(int i = 100; i>0; i--);
}
void delayWr()
{
  //delay for writing 220ms required
  for(int i = 10; i>0;i--)
  {
```

```
   //delay 250 nano seconds
   //since compare and decrement consume 2 instructions in total.
  // delay10();  //call delay 10 22 times for 220milliseconds
  }
}
void delayRes()
{
  for(int i =5000; i>0; i--)
  {
    //each cycle is 125 nanoseconds.
   delay10(); //for 50 milliseconds to reset
  }



}

//accepts coordinates on where to start typing or drawing.
void setPos(unsigned char posHigh,unsigned char posLow)
{
P9OUT = csrw;
writeCom();

//00 is at the beggining, F0 moves the text to the middle and more to the right
P9OUT =posLow;
writeData();

P9OUT =posHigh;
writeData();
}
```

```
//writes the command that is in on the data port P9
void writeCom()
{
  //if statement to reduce delays
  P10OUT |= dataSel;
  //delay since the Ao needs time to change
  P10OUT &= ~chipSel;
  delay10();
  P10OUT &= ~write;
  delayWr();
  P10OUT |= write;
  P10OUT |= chipSel;
  delay10();
  clearP9();
}


//Display write in screen

void wrDisp(unsigned int lang)
{
if(lang==1){
//Show screen title in english
setPos(0x00,0x4C);
typeText("write-in",8);
}
else{
//Show screen title in spanish
```

```
setPos(0x00,0x47);
typeText("nominacion directa",18);
}

//Show Alphabet and Space on screen
setPos(0x01,0x42);
typeText("abcdefghijklmnopqrstuvwxyz><",28);

if(lang==1){
//Show "END" option
setPos(0x01,0x82);
typeText("end",3);

}
else{
//Show "FIN" option
setPos(0x01,0x82);
typeText("fin",3);
}

}
//writes data from the P9 port
void writeData()
{
  P10OUT &= ~dataSel;  //turn off Ao
  P10OUT &= ~chipSel;
  delay10();
  P10OUT &= ~write;
  delayWr();
```

```
  P10OUT |= write;
  P10OUT |= chipSel;
  delay10();
  clearP9();
}

//clears the LCD of initial random data present when the LCD is turned on
//and can also be used to clear the screen
void clearLCD()
{
  P9OUT |= crsDir1;  //Auto Cursor Increment
  writeCom();
  clearText();
  P9OUT = 0;          //put zeros into the CGRAM to clear out trash
                    //28*256 iterations to clear all 7168 spaces in CGRAM
  for(int i=896;i>0;i--) //clear CGRAM 3584
  {
    writeData();
    writeData();
    writeData();
    writeData();
  }
  for(int j=1024; j>0; j--)//Clear Graphics 4096
  {
    writeData();
    writeData();
    writeData();
    writeData();
  }
```

```
}

//clears the information in port 9 to zero
void clearP9()
{
  P9OUT &= 0x00;
}

//erase text on screen
void clearText()
{
  P9OUT |= csrw;     //Drawing Control CSRW
  writeCom();
  P9OUT = 0;        //Address 0000  ---Low
  writeData();
  P9OUT = 0;        //--High
  writeData();
  P9OUT |= mwrite;   //write to display memory
  writeCom();
  for(int i=256; i>0 ; i--)  //512 spaces for characters 4*128
  {
    P9OUT |=0x20;  //ascii space from character map in Datasheet
    writeData();
    P9OUT |=0x20; //ascii space from character map in Datasheet
    writeData();
  }
}

//locked screen showned when eVote is inactive
```

```
void locked()
{
  clearText();
  setPos(0x00,0xE5);
  typeText("no activo - not active", 22);
}

//select language screen
void language()
{
setPos(0x00,0x87);
typeText("language_-",10);
typeText("idioma", 6);
setPos(0x00,0xC4);
typeText("english",7);
setPos(0x00,0xD5);
typeText("espanol",7);

setPos(0x00,0xE6);
 typeText("(   )",5);
//typeParL();
//setPos(0x00,0xEA);
//typeParR();

//in between on F8
setPos(0x00,0xF6);
typeText("(   )",5);
}
```

```
void voteSel(unsigned int lang)  //screen for showing the choice of mixed or integral votes
{

if(lang == 0)
{
//if Spanish
setPos(0x00,0x87);
typeText("?como deseas", 12);
typeText(" votar?",7);
setPos(0x00,0xC4);
typeText("integro",7);
setPos(0x00,0xD5);
typeText("mixto-cand", 10);   //adjust if possible to spell candidatura
}
else if(lang == 1)
{
//if english
setPos(0x00,0x87);
typeText("how will you", 12);
typeText(" vote?",6);
setPos(0x00,0xC4);
typeText("integral",8);
setPos(0x00,0xD5);
typeText("mixed-cand", 10);   //adjust if possible to spell candidatura
}

//Display parenthesis regardless of language
//in keyboard place cursor inbetween on E8
setPos(0x00,0xE6);
```

```
typeText("(   )",5);

//in between on F8
setPos(0x00,0xF6);
typeText("(   )",5);
}

//write in choice screen for governor and resident commisioner
void wrScr(unsigned int lang, unsigned int screen)
{
 if(lang == 0)  //spanish
 {
  setPos(0x00,0x45);
  typeText("nominacion directa para",23);
  setPos(0x00,0x8B);
  if(screen == 0 )
  {
 typeText("gobernador", 10);
 }
 else
 {
  typeText("comisionado residente", 10);
 }
 //show yes no options
 setPos(0x00,0xC4);
 typeText("si",2);
 setPos(0x00,0xD5);
 typeText("no", 2);
```

```
 //Parenthesis for selection
setPos(0x00,0xE6);
typeText("(   )",5);

setPos(0x00,0xF6);
typeText("(   )",5);
}
else if(lang == 1)
{
//english version
    setPos(0x00,0x45);
    if(screen == 0)
    {
   typeText("write-in for governor",21);
    }
    else
    {
      typeText("write-in for",12);
      setPos(0x00,0x67);
      typeText("resident commissioner",21);
    }
 //show yes no options
 setPos(0x00,0xC4);
 typeText("yes",2);
 setPos(0x00,0xD5);
 typeText("no", 2);

 //Parenthesis for selection
 setPos(0x00,0xE6);
```

```
 typeText("(   )",5);

 setPos(0x00,0xF6);
  typeText("(   )",5);

 }
}

 void confirm(unsigned int lang)
 {
  if(lang == 0) //spanish
  {
    setPos(0x00,0x45);
    typeText("?confirmar voto?", 16);
    setPos(0x00,0xC4);
   typeText("si",2);
   setPos(0x00,0xD5);
   typeText("no", 2);
  }
  else if(lang == 1)
  {
   setPos(0x00,0x45);
    typeText("confirm vote?", 13);
     //show yes no options
   setPos(0x00,0xC4);
   typeText("yes",2);
   setPos(0x00,0xD5);
   typeText("no", 2);
  }
```

```
  //Parenthesis for selection
  setPos(0x00,0xE6);
typeText("(   )",5);

 setPos(0x00,0xF6);
typeText("(   )",5);

}

void sumDisp(unsigned int lang, unsigned char gov[], unsigned int govLen,
          unsigned char com[], unsigned int comLen, unsigned int par)
{
 //acquire choices from sylvias keyboard code, as choices are made, store them
 //and when arriving to this screen, send as parameters.
 if(lang == 0) //spanish
 {
  setPos(0x00,0x28);
  typeText("resumen de voto", 16);
  setPos(0x00,0x60);
    typeText("partido - ",10);
   //ppd = 1 pnp = 2 pip = 3 ppr = 4
  if(par == 0)
  {
    typeText("ninguno", 7);
  }
  else if(par == 1)
  {
   typeText("ppd", 3);
```

```
 }
 else if(par == 2)
 {
 typeText("pnp", 3);
 }
 else if(par == 3)
 {
  typeText("pip", 3);
 }
 else if(par == 4)
 {
 typeText("ppr", 3);
 }
 setPos(0x00,0x80);
 typeText("gobernador -",12);
 setPos(0x00,0xC0);
 typeText(gov,govLen);
 setPos(0x00,0xE0);
 typeText("comisionado residente -", 23);
 setPos(0x01,0x20);
 typeText(com,comLen);
 setPos(0x01,0xA4);
 typeText("aceptar",7);
 setPos(0x01,0xB5);
 typeText("cancelar", 8);
}
else if(lang == 1)
{
  setPos(0x00,0x28);
```

```
typeText("voting summary", 14);
setPos(0x00,0x60);
typeText("party -", 7);
 //ppd = 1 pnp = 2 pip = 3 ppr = 4
if(par == 0)
{
   typeText("none", 4);
}
else if(par == 1)
{
  typeText("ppd", 3);
}
else if(par == 2)
{
  typeText("pnp", 3);
}
else if(par == 3)
{
  typeText("pip", 3);
}
else if(par == 4)
{
  typeText("ppr", 3);
}
setPos(0x00,0x80);
typeText("governor -",10);
setPos(0x00,0xC0);
typeText(gov,govLen);
setPos(0x00,0xE0);
```

```
    typeText("resident commissioner -", 23);
    setPos(0x01,0x20);
    typeText(com,comLen);
    setPos(0x01,0xA4);
    typeText("accept",6);
    setPos(0x01,0xB5);
    typeText("cancel", 6);
   }


 }


void ballot1(unsigned int lang)
{

}

void ballot2(unsigned int lang)
{

}

void partido()
{

}
```

```
void typeText(unsigned char a[], unsigned int size)
{

  int len = size;

  P9OUT |= mwrite;   //write to display memory
  writeCom();

for(int i = 0; i<len ;i++)
{

  switch (a[i])
  {

  case 'a' :
  P9OUT = 0x41;
  writeData();
  break;

  case 'b' :
  P9OUT = 0x42;  //hex code for letter B
  writeData();
  break;

  case 'c' :
  //hex code for letter C
  P9OUT = 0x43;
  writeData();
  break;
```

```
case 'd' :
//hex code for letter D
P9OUT = 0x44;
writeData();
break;

case 'e' :
//hex code for letter E
P9OUT = 0x45;
writeData();
break;

case 'f' :
//hex code for letter F
P9OUT = 0x46;
writeData();
break;

case 'g' :
//hex code for letter G
P9OUT = 0x47;
writeData();
break;

case 'h' :
//hex code for letter H
P9OUT = 0x48;
writeData();
```

```
break;

case 'i' :
//hex code for letter I
P9OUT = 0x49;
writeData();
break;

case 'j' :
//hex code for letter J
P9OUT = 0x4A;
writeData();
break;

case 'k' :
//hex code for letter K
P9OUT = 0x4B;
writeData();
break;

case 'l' :
//hex code for letter L
P9OUT = 0x4C;
writeData();
break;

case 'm' :
//hex code for letter M
P9OUT = 0x4D;
```

```
writeData();
break;

case 'n' :
//hex code for letter N
P9OUT = 0x4E;
writeData();
break;

case 'o' :
//hex code for letter O
P9OUT = 0x4F;
writeData();
break;

case 'p' :
//hex code for letter P
P9OUT = 0x50;
writeData();
break;

case 'q' :
//hex code for letter Q
P9OUT = 0x51;
writeData();
break;

case 'r' :
//hex code for letter R
```

```
P9OUT = 0x52;
writeData();
break;

case 's' :
//hex code for letter S
P9OUT = 0x53;
writeData();
break;

case 't' :
//hex code for letter T
P9OUT = 0x54;
writeData();
break;

case 'u' :
//hex code for letter U
P9OUT = 0x55;
writeData();
break;

case 'v' :
//hex code for letter V
P9OUT = 0x56;
writeData();
break;

case 'w' :
```

```
//hex code for letter W
P9OUT = 0x57;
writeData();
break;

case 'x' :
//hex code for letter X
P9OUT = 0x58;
writeData();
break;

case 'y' :
//hex code for letter Y
P9OUT = 0x59;
writeData();
break;

case 'z' :
//hex code for letter Z
P9OUT = 0x5A;
writeData();
break;

case '*' :
//hex code for letter *
P9OUT = 0x2A;
writeData();
break;
```

```
case '(' :
//hex code for letter (
P9OUT = 0x28;
writeData();
break;

case ')' :
//hex code for letter )
P9OUT = 0x29;
writeData();
break;

case '&' :
//hex code for letter '  apostrophe
P9OUT = 0x27;
writeData();
break;

case '-' :
//hex code for letter -
P9OUT = 0x2D;
writeData();
break;

case '?' :
//hex code for a ?
P9OUT = 0x3F;
writeData();
break;
```

```
  case ' ' :
  //hex code for a space
  P9OUT = 0x20;
  writeData();
  break;

  case '>' :
  //writein space symbol
  P9OUT = 0x7E;
  writeData();
  break;

  case '<' :
  P9OUT = 0x7F;
  writeData();
  break;

  default :
  break;
  }
}
}
```

/*

P1IN&0x10  Verifies the input of Port 1 with 0x10 by executing an And if they match

then it enters the if statement, if not it deviates into the else statement

^= is exclusive or

For USART check usart 430 examples

Select  (verify what this does

P7SEL |= 0x0E;  // P7.3,2,1 option select

for (i = 50000; i; i--);  // Delay

*/

## 12.5. Log In Form code

Login Form Code

```
//LogIn Form
//by Laura M. Cruz
//802-03-1797
//Icom 5047 Sec 031

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Windows.Forms;
//using MySql.Data;

namespace LogIn
{
    public partial class Form1 : Form
    {
        private System.Data.Odbc.OdbcConnection OdbcCon;

        private System.Data.Odbc.OdbcCommand OdbcCom;

        private System.Data.Odbc.OdbcDataReader OdbcDR;

        private string ConStr;

        private string query;

        private string precinct;

        private string unit;

        private string firstname;

        private string lastname1;

        private string lastname2;

        private string psswd;



        public Form1()
        {
```

```
            InitializeComponent();

            // Build the connection string

            ConStr = "DRIVER={MySQL ODBC 5.1
Driver};SERVER=136.145.56.170;PORT=3306;DATABASE=evote;UID=evote;PWD=evote;OP
TION=3";

            //Create connection
            OdbcCon = new System.Data.Odbc.OdbcConnection(ConStr);


            //When form is initialized, fill precinct dropdown box
(comboBox1) with precinct list
            try
            {
                if (OdbcCon.State == ConnectionState.Closed)
                {
                    OdbcCon.Open();
                    query = "SELECT DISTINCT Precinct FROM evote.official
o;";
                    //Create ODBC Command with necessary query and odbc
connection
                    OdbcCom = new System.Data.Odbc.OdbcCommand(query,
OdbcCon);
                    //ExecuteReader is used when query results in more than
one row
                    OdbcDR = OdbcCom.ExecuteReader();
                    //Adds all available precincts to dropdown list
                    while (OdbcDR.Read())
                    {
                        comboBox1.Items.Add(OdbcDR[0]);
```

```
                    }
                    //Always close reader when not in use
                    OdbcDR.Close();
                }

            }

            catch (System.Data.Odbc.OdbcException Ex)
            {
                //Thrown when connection to database is not possible
                MessageBox.Show("Could not access the database.\r\nPlease
make sure you completed the fields with the correct information and try
again.\r\n\r\nMore       details:\r\n" + Ex.Message, "Database connection
error", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }


        }
        //When precinct is selected or changed
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            //Clear all related dropdown boxes (units and officials)
            comboBox2.Items.Clear();
            comboBox3.Items.Clear();

            //Sets "precinct" to selected precinct name
            precinct = comboBox1.SelectedItem.ToString();

            query = "SELECT DISTINCT Unit FROM evote.official o WHERE
Precinct ='" + precinct + "';";
            //query = "SELECT DISTINCT Unit FROM evote.official o WHERE
Precinct LIKE ? ;";
            OdbcCom.CommandText = query;
```

```
            //MessageBox.Show(query);
            //OdbcParam = new System.Data.Odbc.OdbcParameter();
            //OdbcParam.DbType = DbType.String;
            //OdbcParam.Value = precinct;
            //OdbcCom.Parameters.Add(OdbcParam);

            //OdbcCom.ExecuteNonQuery();

            OdbcDR = OdbcCom.ExecuteReader();
            //Adds units in precinct to dropdown box (comboBox2)
            while (OdbcDR.Read())
            {
                comboBox2.Items.Add(OdbcDR[0]);
            }

            //Always close reader when it is not in use
            OdbcDR.Close();

            //Enable unit dropdown when precinct is selected
            comboBox2.Enabled = true;
        }

        //When Ok button is clicked, form verifies provided information
        private void button1_Click(object sender, EventArgs e)
        {
            //Stored variables are used in verification query
            query = "SELECT Password FROM evote.official o WHERE FirstName='"
+ firstname + "' AND FirstLastName='" + lastname1 + "' AND SecondLastName='"
+ lastname2 + "' AND Precinct ='" + precinct + "' AND Unit='" + unit + "';";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();

            //If record is found in query result
            if (OdbcDR.Read())
            {
```

```
                //compares provided password with password in database
                psswd = textBox1.Text;

                //If passwords match
                if (psswd.Equals(OdbcDR[0] + ""))
                {


                    //Creates new Admin form and passes selected precinct and
unit
                    //Form Administrator = new Admin(precinct, unit);
                    //Administrator.Show();
                    //this.Hide();
                }
                else
                {
                    //Shown when incorrect information is provided
                    MessageBox.Show("Incorrect Information.");

                }
            }
            else
            {
                //Shown were query returned no results
                MessageBox.Show("No record was found.");
            }


            //Always close reader when it is not in use
            OdbcDR.Close();
        }

        private void groupBox1_Enter(object sender, EventArgs e)
        {
```

```
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void comboBox2_SelectedIndexChanged_1(object sender,
EventArgs e)
        {
            //When unit is selected or changed, clear official drop down
(comboBox3)
            comboBox3.Items.Clear();
            //Sets "unit" to selected unit name
            unit = comboBox2.SelectedItem.ToString();
            query = "SELECT FirstName,FirstLastName,SecondLastName FROM
evote.official o WHERE Precinct='" + precinct + "' AND Unit ='" + unit +
"';";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();

            //Adds First Name, First Last Name , and Second Last Name to
official dropdown list
            if (OdbcDR.Read())
            {
                comboBox3.Items.Add(OdbcDR[0] + " " + OdbcDR[1] + " " +
OdbcDR[2]);
                firstname = (string)OdbcDR[0];
                lastname1 = (string)OdbcDR[1];
                lastname2 = (string)OdbcDR[2];
```

```
            }

            //Always close reader when it is not in use
            OdbcDR.Close();
            //Enables official dropdown list after names have been added
            comboBox3.Enabled = true;




        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        //When official (comboBox3) selection is made password textbox is
enabled
        private void comboBox3_SelectedIndexChanged(object sender, EventArgs
e)
        {
            textBox1.Enabled = true;
        }
    }
}
```

## 12.6. Admin Form Code

Admin Form Code

```
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.Threading;

namespace WindowsFormsApplication1
{
    public partial class Admin : Form
    {
        //Contains the Database Address
        private String dbAddress = "DRIVER={MySQL ODBC 5.1
Driver};SERVER=136.145.56.170;PORT=3306;DATABASE=evote;UID=evote;PWD=evote;OP
TION=3";
        private String UnidadActiva; //Stores the unit selected by the
employee at the login
        private String PrecintoActivo; //Stores the precinct selected by the
employee at the login
        private String numElectoral; // Stores the ID of the current voter to
be registered
        private Form LoginForm; // Contains the reference of the Login Form

        /**
         *  Method used to initialize an Admin Form with the parameters
selected in the login form
         **/
        public Admin(String Precinto, String Unidad, Form Log)
        {
            InitializeComponent();
            UnidadActiva = Unidad;
            PrecintoActivo = Precinto;
```

```
            LoginForm = Log;
            populateCBCasetaHTab();
        }


        /**
         *  Calls the Verification method
         **/
        private void ButtonVerificar_Click(object sender, EventArgs e)
        {
            Verification();
        }

        /**
         *    Verifies if the voter has not voted already and if the ID is
assgined to the current Precinct and Unit
         *    If the has not voted the asignment field will be activated.
         *    If the voter has already voted or not assigned to that
precinct/unit a message will appear indicating the issue.
         **/
        private void Verification()
        {
            CBCasetaRTab.Items.Clear();
            System.Data.Odbc.OdbcConnection OdbcConnect; //Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object
            String UnidadAsignadaQ; //Query to get the assigned unit of the
voter
            String PrecintoAsignadoQ; // Query to get the assigned precinct
to the voter
            String  alreadyVotedQ; // Query to get if the voter has or not
voted
            String alreadyVoted; // Stores if the voter has voted or not.
Equals to 0 if has not voted, and equals to 1 if already voted
```

```
            String UnidadAsignada; //Stores the assigned unit of the voter
            String PrecintoAsignado; //Stores the assigned precinct of the
voter

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {


                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Opens the database connection
                    OdbcConnect.Open();

                    // Reads the Numero Electotal ID to be verfied
                    numElectoral = TBNumeroElectoral.Text;


                            if (NumeroElectorValido())
                                {
                                    // Looks in the tabase to determine if
the voter has already voted
                                    alreadyVotedQ = "SELECT AlreadyVoted FROM
voter v Where ElectoralNumber='" + numElectoral + "';";
                                    OdbcComm = new
System.Data.Odbc.OdbcCommand(alreadyVotedQ, OdbcConnect);
                                    OdbcDR = OdbcComm.ExecuteReader();
                                    OdbcDR.Read();
                                    alreadyVoted = OdbcDR[0].ToString();
                                    OdbcDR.Close();

                                    // Looks in the database the Precinct to
which the voter is assigned
```

```
                                    PrecintoAsignadoQ = "SELECT Precinct FROM
voter v Where ElectoralNumber='" + numElectoral + "';";
                                    OdbcComm = new
System.Data.Odbc.OdbcCommand(PrecintoAsignadoQ, OdbcConnect);
                                    OdbcDR = OdbcComm.ExecuteReader();
                                    OdbcDR.Read();
                                    PrecintoAsignado = OdbcDR[0].ToString();
                                    OdbcDR.Close();


                                    // Looks in the database the Unit to
which the voter is assigned
                                    UnidadAsignadaQ = "SELECT Unit FROM voter
v Where ElectoralNumber='" + numElectoral + "';";
                                    OdbcComm = new
System.Data.Odbc.OdbcCommand(UnidadAsignadaQ, OdbcConnect);
                                    OdbcDR = OdbcComm.ExecuteReader();
                                    OdbcDR.Read();
                                    UnidadAsignada = OdbcDR[0].ToString();
                                    OdbcDR.Close();



                                    //Verifies if the voter has not voted
already and if the voter is assigned to the current unit and precinct
                                    if (alreadyVoted.Equals("0") &&
UnidadActiva.Equals(UnidadAsignada) &&
PrecintoActivo.Equals(PrecintoAsignado))
                                    {
                                        LAutorizacionRTab.Enabled = true;
                                        LCasetaRTab.Enabled = true;
                                        CBCasetaRTab.Enabled = true;
                                        ButtonAsignar.Enabled = true;
                                        populateCBCasetaRTab();
```

```
                                    }

                                    // If the voter has already voted, a
warning screen will appear.
                                    else if (alreadyVoted.Equals("1"))
                                    {
                                        MessageBox.Show("Elector ya ejerció
su derecho al voto en estas elecciones.", "Advertencia",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                        TBNumeroElectoral.Clear(); // Refresh
the Numero Electoral text box
                                    }
                                    // If the voter has not voted but is
assigned to another Unit or Precinct a notification screen will appear
                                    else
                                    {
                                        MessageBox.Show("Elector no esta
asignado a esta unidad o precinto.", "Advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                                        TBNumeroElectoral.Clear();// Refresh
the Numero Electoral text box
                                    }
                                }

                            else
                                MessageBox.Show("Número Electoral no es
válido.", "Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Error);

                        //Closes the DB Connection
                        OdbcConnect.Close();

                    }
                }

                catch
```

```
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }

        private void CBCasetaRTab_SelectedIndexChanged(object sender,
EventArgs e)
        {
        }

        /**
         *  Call the metohd to populate the Caseta combo box
         **/
        private void CBCasetaRTab_SelectedIndexChanged_1(object sender,
EventArgs e)
        {
            populateCBCasetaRTab();
        }


        /**
         * Assigns a kiosk to the voter
         **/
        private void ButtonAsignar_Click(object sender, EventArgs e)
        {

            try
            {
                System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
                System.Data.Odbc.OdbcCommand OdbcComm; // Database command
object
```

```
                System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader
object

                //Initialize the connection to the database
                OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Opens the database connection
                    OdbcConnect.Open();

                    try
                    {
                        String casetaAsignada =
CBCasetaRTab.SelectedItem.ToString(); //Stores the kiosk asigned to the voter

                        //Set the status of the assgined kiosk as occupied
                        String casetaAsignadaQ = "UPDATE evote.kiosk k SET
Occupied = '1' where Unit ='" + UnidadActiva +
                            "' and Precinct='" + PrecintoActivo + "' and
LocalID='" + casetaAsignada + "'";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(casetaAsignadaQ, OdbcConnect);
                        OdbcComm.CommandText = casetaAsignadaQ;
                        OdbcComm.ExecuteNonQuery();

                        //Updates the voter record with the assigned kiosk
                        String LugarDondeVotoQ = "SELECT kID FROM evote.kiosk
k Where Precinct='" + PrecintoActivo +
                            "' and LocalID='"+ casetaAsignada+"';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(LugarDondeVotoQ, OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        String LugarDondeVoto = OdbcDR[0].ToString();
```

```
                        OdbcDR.Close();
                        String SetLugarDondeVotoQ = "UPDATE evote.voter v SET
kID='" + LugarDondeVoto + "'where ElectoralNumber='" + numElectoral + "';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(SetLugarDondeVotoQ, OdbcConnect);
                        OdbcComm.ExecuteNonQuery();

                        //Updates the voter record with the already voted
status
                        String hasVotedQ = "UPDATE evote.voter v SET
AlreadyVoted= '1'where ElectoralNumber='" + numElectoral + "';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(hasVotedQ, OdbcConnect);
                        OdbcComm.ExecuteNonQuery();

                        //Shows a message to the employee notifying that the
authorization is completed
                        MessageBox.Show("Autorización Procesada",
"Autorización", MessageBoxButtons.OK, MessageBoxIcon.None);

                        //Refresh the Numero Electoral ID Text Box
                        TBNumeroElectoral.Clear();

                        //Refresh the Kiosk Selection
                        CBCasetaRTab.Items.Clear();

                        //Disables the authorization fields
                        LAutorizacionRTab.Enabled = false;
                        LCasetaRTab.Enabled = false;
                        CBCasetaRTab.Enabled = false;
                        ButtonAsignar.Enabled = false;

                        //Closes the database connection
                        OdbcConnect.Close();
                    }
```

```
                    catch
                    {
                        MessageBox.Show("Favor seleccionar una caseta",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }


                }

            }

             catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }

        /**
         *  Verifies if the Electoral Number is a valid ID by searching for a
record with that ID
         **/
        private bool NumeroElectorValido()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
```

```
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for a record with the ID
                    String IDQ = "SELECT * FROM voter v Where
ElectoralNumber='" + numElectoral + "';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(IDQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();
                    if (OdbcDR.HasRows)
                    {
                        OdbcConnect.Close();
                        return true;
                    }
                    else
                    {
                        OdbcConnect.Close();
                        return false;
                    }

                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }


            return false;
```

```
        }


        /**
         *  Populates the Caseta Combo Box with the unoccupied kiosks
         **/
        private void populateCBCasetaRTab()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {
                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for unoccupied kiosks
                    String casetaQ = "SELECT LocalID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo + "' and Occupied = '0';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();

                    //Reads the casetaQ query and populates the combo box
with available (unoccupied) kiosks.
                    while (OdbcDR.Read())
```

```
                    {
                        CBCasetaRTab.Items.Add(OdbcDR[0]);

                    }
                    // Show a notice if all the kiosks are occupied
                    if (CBCasetaRTab.Items.Count.ToString().Equals("0"))
                        MessageBox.Show("Todas las casetas estan ocupadas en
este momento", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    OdbcDR.Close();
                }

            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }


        private void CBCasetaHTab_SelectedIndexChanged(object sender,
EventArgs e)
        {
            LBHistorialHTab.Items.Clear();
        }

        /**
         *  Populates the list of the voters that have voted on the selected
kiosk
         **/
        private void populateLBHTab(String NumCaseta)
        {
            String casetaSeleccionada = NumCaseta; // local id
```

```
            String idCaseta; // global id at the database
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {
               if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for the global ID
                    String casetaQ = "SELECT kID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo + "' and localID
='"+casetaSeleccionada+"';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();
                    OdbcDR.Read();
                    idCaseta = OdbcDR[0].ToString();
                    OdbcDR.Close();

                    String voters = "SELECT ElectoralNumber FROM evote.voter
v where kID='" +idCaseta + "';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(voters,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();
```

```
                    //Reads the casetaQ query and populates the combo box
with available kiosks.
                    while (OdbcDR.Read())
                    {
                        LBHistorialHTab.Items.Add(OdbcDR[0]);

                    }

                    OdbcDR.Close();
                }
             }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }

        private void populateCBCasetaHTab()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
```

```
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for kiosks
                    String casetaQ = "SELECT LocalID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo +"';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();

                    //Reads the casetaQ query and populates the combo box
with available kiosks.
                    while (OdbcDR.Read())
                    {
                        CBCasetaHTab.Items.Add(OdbcDR[0]);
                    }

                    OdbcDR.Close();
                }

            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void CBComisionadoPTab_SelectedIndexChanged(object sender,
EventArgs e)
        {
        }
```

```
        /**
         *   Ends the session and returns to the login screen
         **/
        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Visible = false;
            LoginForm.Visible = true;
            UnidadActiva = "";
            PrecintoActivo = "";
        }

        /**
         *  Provides infomation about the product
         **/
        private void sobreEVoteToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            MessageBox.Show("eVote \r\nVersion 1.0 \r\nCopyright 2008 por
JASL. Todos los derechos reservados", "Sobre eVote", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }

        /**
         *  Generates and prints a document with the results
         **/
        private void BPrintPTab_Click(object sender, EventArgs e)
        {
            System.Data.Odbc.OdbcConnection OdbcConnect;// Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);
```

```
            try
            {
                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Reads the selection for "Gobernador" and "Comisionado
Residente"
                    String partido;
                    String gobernador; // Stores the selection for the
"Gobernador" position
                    String comisionado; // Stores  the selection for the
"Comisionado Residente" position
                    partido = PartidoText.Text;
                    gobernador = GobernadorText.Text;
                    comisionado = ComisionadoText.Text;

                    String partidoCode = " ";
                    String gobernadorCode = " ";
                    String comisionadoCode = " ";

                    // If the gobernador is selected from one of the provided
options looks for the code assigned for the barcode
                    if (partido.Equals("PPD") || partido.Equals("PNP") ||
partido.Equals("PIP") || partido.Equals("PPR"))
                    {
                        String partidoQ = "SELECT Barcode FROM
`evote`.`party`where pName= '" + partido+ "';";
                        OdbcComm = new System.Data.Odbc.OdbcCommand(partidoQ,
OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        partidoCode = OdbcDR[0].ToString();
                        OdbcDR.Close();
```

```
                    }

                    if (gobernador.Equals("Anibal Acevedo
Vila")||gobernador.Equals("Luis Fortuno")||
                        gobernador.Equals("Edwin Irrizary
Mora")||gobernador.Equals("Rogelio Figeroa Garcia"))
                    {
                        char[] space = { ' ' };
                        string[] gobernadorA = new string[2];
                        gobernadorA = gobernador.Split(space);
                        String gobernadorQ = "SELECT Barcode FROM
`evote`.`candidate`where FirstName = '" + gobernadorA[0] +
                            "' and FirstLastName = '" + gobernadorA[1] +
"';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(gobernadorQ, OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        gobernadorCode = OdbcDR[0].ToString();
                        OdbcDR.Close();
                    }

                    if (comisionado.Equals("Alfredo
Salazar")||comisionado.Equals("Pedro Pierluisi")||
                        comisionado.Equals("Jessica Martinez
Birriel")||comisionado.Equals("Carlos Alberto Velazquez"))
                    {
                        char[] space = { ' ' };
                        string[] comisionadoA = new string[2];
                        comisionadoA = comisionado.Split(space);
                        String comisionadoQ = "SELECT Barcode FROM
`evote`.`candidate`where FirstName = '" + comisionadoA[0] +
                            "' and FirstLastName = '" + comisionadoA[1] +
"';";
```

```
                    OdbcComm = new
System.Data.Odbc.OdbcCommand(comisionadoQ, OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();
                    OdbcDR.Read();
                    comisionadoCode = OdbcDR[0].ToString();
                    OdbcDR.Close();
                }

                //Reference objects
                object oMissing = System.Reflection.Missing.Value;
                object oEndOfDoc = "\\endofdoc";  //end of file,
predifined bookmark

                //Starts Word Application and create a new document.
                Microsoft.Office.Interop.Word.Application oWord = new
Microsoft.Office.Interop.Word.Application();
                Microsoft.Office.Interop.Word.Document oDoc = new
Microsoft.Office.Interop.Word.Document();
                //Word._Application oWord; //2003 reference
                //Word._Document oDoc;//2003 reference
                //oWord = new Word.Application();
                oWord.Visible = true; // hides the document, runs in back
without anyone seeing what it contains
                oDoc = oWord.Documents.Add(ref oMissing, ref oMissing,ref
oMissing, ref oMissing);

                //Insert a text line with paragraph spacing at the
beginning of the document.
                Microsoft.Office.Interop.Word.Paragraph oPara1;
                //Word.Paragraph oPara1;//2003 reference
                oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);//Adds
a new paragraph to the contents of the document
                oPara1.Range.Text = "Resultados de su votación";//Adds
text to the paragraph
                oPara1.Range.Font.Bold = 1; // Set the text font as BOLD
```

```
                    oPara1.Range.Font.Size = 12; // Set the font size to 12
                    oPara1.Format.SpaceAfter = 24;    //Space left between
lines, 24pt
                    oPara1.Range.InsertParagraphAfter();

                    //Sets the range of the document using the end of file
bookmark
                    object oRng = oDoc.Bookmarks.get_Item(ref
oEndOfDoc).Range;

                    //Insert a 7 x 3 table to enter the results
                    Microsoft.Office.Interop.Word.Table oTable;
                    // Word.Table oTable; //Table object //2003 reference
                    Microsoft.Office.Interop.Word.Range wrdRng =
oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
                    //Word.Range wrdRng = oDoc.Bookmarks.get_Item(ref
oEndOfDoc).Range; //Sets the range of the text for the table using the end of
file bookmark //2003 reference
                    oTable = oDoc.Tables.Add(wrdRng, 7, 3, ref oMissing, ref
oMissing); // Adds the table to the document
                    oTable.Range.ParagraphFormat.SpaceAfter = 10;//Space left
between line after the text block


                    //Insert text into the table
                    oTable.Cell(1, 1).Range.Text = "BOLETO ESTATAL";
                    oTable.Cell(3, 1).Range.Text = "PARTIDO";
                    oTable.Cell(3, 2).Range.Text = "NOMBRE DEL PARTIDO";
                    oTable.Cell(3, 3).Range.Text = "CODIGO DE BARRA";
                    oTable.Cell(5, 2).Range.Text = "NOMBRE DEL CANDIDATO";
                    oTable.Cell(5, 3).Range.Text = "CODIGO DE BARRA";
                    oTable.Cell(6, 1).Range.Text = "GOBERNADOR";
                    oTable.Cell(7, 1).Range.Text = "COMISIONADO RESIDENTE";

                    //Set the font of some of the cell to be BOLD
```

```
                    oTable.Cell(1, 1).Range.Font.Bold = 1;
                    oTable.Cell(3, 1).Range.Font.Bold = 1;
                    oTable.Cell(3, 2).Range.Font.Bold = 1;
                    oTable.Cell(3, 3).Range.Font.Bold = 1;
                    oTable.Cell(5, 2).Range.Font.Bold = 1;
                    oTable.Cell(5, 3).Range.Font.Bold = 1;
                    oTable.Cell(6, 1).Range.Font.Bold = 1;
                    oTable.Cell(7, 1).Range.Font.Bold = 1;

                    //Insert the selection into the table
                    oTable.Cell(3, 2).Range.Text = partido;
                    oTable.Cell(3, 2).Range.Font.Italic = 1;
                    oTable.Cell(3, 3).Range.Text = partidoCode;
                    oTable.Cell(3, 3).Range.Font.Name =
"IDAutomationHC39M";//set the font of the cell to be barcode
                    oTable.Cell(6, 2).Range.Text = gobernador;
                    oTable.Cell(6, 2).Range.Font.Italic = 1;
                    oTable.Cell(6, 3).Range.Text = gobernadorCode;
                    oTable.Cell(6, 3).Range.Font.Name =
"IDAutomationHC39M";//set the font of the cell to be barcode
                    oTable.Cell(7, 2).Range.Text = comisionado;
                    oTable.Cell(7, 2).Range.Font.Italic = 1;
                    oTable.Cell(7, 3).Range.Text = comisionadoCode;
                    oTable.Cell(7, 3).Range.Font.Name =
"IDAutomationHC39M";//set the font of the cell to be barcode




                    //Print the document
                    object copies = "1";
                    object pages = "1";
                    object range =
Microsoft.Office.Interop.Word.WdPrintOutRange.wdPrintCurrentPage;
```

```
                    //object range = Word.WdPrintOutRange.wdPrintCurrentPage;
//2003 reference
                   // object items =
Word.WdPrintOutItem.wdPrintDocumentContent;
                    object items =
Microsoft.Office.Interop.Word.WdPrintOutItem.wdPrintDocumentContent;
                    //object pageType = Word.WdPrintOutPages.wdPrintAllPages;
//2003 reference
                    object pageType =
Microsoft.Office.Interop.Word.WdPrintOutPages.wdPrintAllPages;
                    object oTrue = true;
                    object oFalse = false;

                    oDoc.PrintOut(ref oTrue, ref oFalse, ref range, ref
oMissing, ref oMissing, ref oMissing,
                        ref items, ref copies, ref pages, ref pageType, ref
oFalse, ref oTrue,
                        ref oMissing, ref oFalse, ref oMissing, ref oMissing,
ref oMissing, ref oMissing);

                    //Closes the document without saving it
                    object doNotSaveChanges =
Microsoft.Office.Interop.Word.WdSaveOptions.wdDoNotSaveChanges;
                    //object doNotSaveChanges =
Word.WdSaveOptions.wdDoNotSaveChanges; //2003 reference
                    //oDoc.Close(ref doNotSaveChanges, ref oMissing, ref
oMissing); //2003 reference
                    oDoc.Close(ref oFalse, ref oMissing, ref oMissing);

                    //Pause to send the information to the printer before the
application is closed
                    Thread.Sleep(15000);

                   //Closes the Microsoft Word Application
```

```
                    oWord.Quit(ref doNotSaveChanges, ref oMissing, ref
oMissing);
                }
            }
            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        /**
         *  Disables the close button to avoid the user closing the
application with logging off
         **/
        private void Admin_Load(object sender, EventArgs e)
        {
            this.ControlBox = false;
        }


        /**
         *   Reads the selected kiosk and populate its history
         **/
        private void button1_Click(object sender, EventArgs e)
        {
            LBHistorialHTab.Items.Clear();
            populateLBHTab(CBCasetaHTab.SelectedItem.ToString());
        }
    }
}
```

## 12.7. Database Test Code

START OF SOURCE CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
/**
 * A class that tests the connectivity with our mySQL database. Implements
basic queries such as
 * SELECT and UPDATE.
 * by Angel L. Vega Cortés
**/
namespace DatabaseConnectionTest
{
    public partial class Form1 : Form
    {
        // ODBC Connection Object
        private System.Data.Odbc.OdbcConnection OdbcCon;
        // ODBC Command Object
        private System.Data.Odbc.OdbcCommand OdbcCom;
        // ODBC Data Reader
        private System.Data.Odbc.OdbcDataReader OdbcDR;
        // Connection String
        private string ConStr;
        // Query String
        private string query;
        // Voters' electoral number
        private string electoralNumber;
```

```
        // Voters' assigned unit
        private string unit;
        // Voters' assigned precinct
        private string precinct;
        // Local ID for the Kiosk
        private string kioskLocalID;
        // Unique ID for the Kiosk
        private string kioskUniqueID;

        /**
         * Initiliazes the connection with the respective database.
         * Loads the list of registered voters in the component "comboBox1".
         *
        **/
        public Form1()
        {
            InitializeComponent();

            // Build the connection string
            ConStr = "DRIVER={MySQL ODBC 5.1
Driver};SERVER=136.145.56.170;PORT=3306;DATABASE=evote;UID=evote;PWD=evote;OP
TION=3";
            OdbcCon = new System.Data.Odbc.OdbcConnection(ConStr);

            try
            {
                if (OdbcCon.State == ConnectionState.Closed)
                {
                    OdbcCon.Open();
                    query = "SELECT DISTINCT ElectoralNumber FROM evote.voter
v";
                    OdbcCom = new System.Data.Odbc.OdbcCommand(query,
OdbcCon);
                    OdbcDR = OdbcCom.ExecuteReader();
                    while (OdbcDR.Read())
```

```
                    {
                        comboBox1.Items.Add(OdbcDR[0]);
                    }
                    OdbcDR.Close();
                }
            }
            catch (System.Data.Odbc.OdbcException Ex)
            {
                MessageBox.Show("Could not access the database.\r\nPlease
make sure you completed the fields with the correct information and try
again.\r\n\r\nMore       details:\r\n" + Ex.Message, "Database connection
error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        /**
         * Contains the list of registered voters. Once a voter is selected
the
         * following components will be populated with their respective
results:
         * Component # 1: textBox1 = the assigned precinct of the voter
         * Component # 2: textBox2 = the assigned unit of the voter
         * Component # 3: comboBox2 = contains the respective kiosk local IDs
corresponding to that precinct and unit
         *
        **/
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            electoralNumber = comboBox1.SelectedItem.ToString();
            query = "SELECT DISTINCT Precinct, Unit FROM evote.voter o WHERE
ElectoralNumber = '" + electoralNumber + "'";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();
            OdbcDR.Read();
            precinct = (String)OdbcDR[0];
```

```
            unit = OdbcDR[1] + " ";
            textBox1.Text = precinct;
            textBox2.Text = unit;
            OdbcDR.Close();

            query = "SELECT DISTINCT LocalID FROM evote.kiosk o WHERE
Precinct ='" + precinct + "' and Unit = '" + unit + "'";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();
            comboBox2.Items.Clear();
            comboBox2.ResetText();

            while (OdbcDR.Read())
            {
                comboBox2.Items.Add(OdbcDR[0]);
            }
            OdbcDR.Close();

        }
        /**
         *  Will update the record of the selected voter in the database.
This component will become
         *  ENABLED once the user has selected a voter and a kiosk from the
drop down lists.
         *  Finally, the component "comboBox3" will become ENABLED.
         *
        **/
        private void button1_Click(object sender, EventArgs e)
        {
            if (!(kioskUniqueID.Equals("") || electoralNumber.Equals("")))
            {
                query = "UPDATE evote.voter SET kID = '" + kioskUniqueID + "'
WHERE ElectoralNumber = '" + electoralNumber + "'";
                OdbcCom.CommandText = query;
                OdbcCom.ExecuteNonQuery();
```

```
                query = "UPDATE evote.kiosk SET Occupied = '1' WHERE kID = '"
+ kioskUniqueID + "'";
                OdbcCom.CommandText = query;
                OdbcCom.ExecuteNonQuery();

                query = "SELECT DISTINCT ElectoralNumber FROM evote.voter v";
                OdbcCom.CommandText = query;
                OdbcDR = OdbcCom.ExecuteReader();
                comboBox3.Items.Clear();

                while (OdbcDR.Read())
                {
                    comboBox3.Items.Add(OdbcDR[0]);
                }
                OdbcDR.Close();
                comboBox3.Enabled = true;
            }
        }
        /**
         * Contains the local IDs of the kiosk. Once one is selected the
component "button1" will become
         * ENABLED.
        **/
        private void comboBox2_SelectedIndexChanged(object sender, EventArgs
e)
        {
            kioskLocalID = comboBox2.SelectedItem.ToString();

            query = "SELECT DISTINCT kID FROM evote.kiosk o WHERE Precinct
='" + precinct + "' and Unit = '" + unit + "' and LocalID = '" + kioskLocalID
+ "'";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();
            OdbcDR.Read();
```

```
            kioskUniqueID = OdbcDR[0] + " ";
            OdbcDR.Close();
            button1.Enabled = true;
        }
        /**
         * Contains the list of registered voters. Once a voter is selected
the
         * following components will be populated with their respective
results:
         * Component # 1: textBox3 = the assigned kiosk of the voter
         * Component # 2: textBox4 = if the kiosk is occupied or not
         *
        **/
        private void comboBox3_SelectedIndexChanged(object sender, EventArgs
e)
        {
            String temp1 = comboBox3.SelectedItem.ToString();
            query = "SELECT DISTINCT LocalID,kID FROM evote.kiosk WHERE kID =
(SELECT DISTINCT kID FROM evote.voter o WHERE ElectoralNumber = '" + temp1 +
"')";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();
            OdbcDR.Read();
            textBox3.Text = OdbcDR[0] + " ";
            String temp2 = OdbcDR[1] + " ";
            OdbcDR.Close();

            query = "SELECT DISTINCT Occupied FROM evote.kiosk WHERE kID = '"
+ temp2 + "'";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();
            OdbcDR.Read();
            int temp3 = int.Parse(OdbcDR[0] + "");
            if (temp3 == 0)
                textBox4.Text = "No";
```

```
            else
                textBox4.Text = "Yes";
            OdbcDR.Close();
        }
        /**
         * Displays the assigned precinct of the voter.
        **/
        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
        /**
         * Displays the assigned unit of the voter.
        **/
        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }
        /**
         * Displays the assigned kiosk of the voter.
        **/
        private void textBox3_TextChanged(object sender, EventArgs e)
        {

        }
        /**
         * Displays whether the kiosk is occupied or not.
        **/
        private void textBox4_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

END OF SOURCE CODE

## 12.8. Change Control Document

# Change Control Process

# for

# eVote

**Version 1.0 draft 1**

**Prepared by Javier Torres**

**Sylvia Rodriguez**

**Laura Cruz**

**Angel Vega**

# Contents



# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| eVote Write-In Feature, Database modifications and Scalability simulation | Oct 26,2008 | Additional Client features desired and database modifications made. Also adding to simulate additional devices connected to an eVote system. | 1.0 |
| | | | |

## Introduction

**Purpose**

This document represents changes made to the proposal document given to the client on September 10, 2008. The changes include a write-in feature to allow the eVote system to type in a name using an on screen keyboard with an external keypad. This additional requirement causes an impact to the budget as well as resources for eVote and is specified in this document.

**Scope**

- Write-In feature specifications
- Database design modifications
- Scalability simulation
    - Removal of UART driver

## Write-In

Due to recent developments, the write-in feature is now desired for implementation for the eVote project. The write-in will be implemented by providing an onscreen alphabet that is navigated through with the keypad that was already involved in the navigational design. The keypad will allow the user to cycle through the alphabet and press the "ok" button once they have the desired letter selected. The write-in will only support a maximum of 28 characters so that the name can be viewed on one row on

screen. The alphabet will be displayed in a single row as well with a backspace key and a finish key which will be represented by "fin" allowing the user to complete their write-in.

### Impact

Additional lines of code are required to implement this, and due to the microprocessor limitations, there is a possibility that the entire code for eVote may not fit into one microchip of the microprocessor currently being used (msp430fg4619). If this is to occur, then the demonstration will be divided into segments of code, demonstrating the individual functionality of the different functions of the eVote firmware.

More resources are required, but due to the limited size of the team, task reassignment has been done to allow for completion of this feature. Angel will now be in charge of completing the encryption and decryption modules for eVote. Laura who was initially assigned the task of decryption will now divert all her time to completing the write-in feature. Due to this more hours will be clocked in, therefore a budget increase may be necessary.

The increase in the budget would be $1153.80 due to two additional weeks needed for designing and implementing this feature at this stage of eVote. This value is calculated by using added two weeks of Laura's salary to the main budget and it may be necessary to push the end date of the eVote project. The total budget would then total $64,711.67 as opposed to the initial budget request.

For the moment everything is going according to the schedule and this impact has been mitigated by distributing the tasks upon the employees of JSAL. The budget increase is a possible impact if the design and implementation of this feature affects the main schedule of eVote.

## Database modifications

As the project progressed some changes were necessary in the database design. Initially a table called "Position" listed all the positions candidates were running for in the current elections. Each position had a unique ID and in the Candidate table, each Candidate was related to a Position ID, therefore, relating a candidate with a position. However, Laura realized that the relationship would be better understood if a candidate simply had a field that indicated in text the position he or she was running for. For this reason, the Position table was dropped and the Position field in the Candidate table was edited to specify in text the position the candidate was running for.

In addition to this, a History table was added in order to associate an Official with a Voter. This way, an official can easily be identified if a voter has any sort of complaint, or vice versa. Another change

was the addition of a Bar Code field in the Candidate table. This barcode is needed at the time of printing the voting receipt. Finally, each kiosk was identified a local ID order to identify them with ease locally (in their respective units).

## Scalability Simulation

After realizing that C# includes driver classes for UART devices (universal asynchronous receiver/transmitter), this task is no longer required for the 3rd phase of the eVote implementation. The new task intended to replace the driver is with a simulation of additional connected devices. The simulation will basically verify if there is another device connected, but instead of an additional eVote device, a pen drive will be used. The pen drive will contain additional voting information that belongs to voters of the kiosk that the pen drive will simulate. C# code will implement additional threads to read text files with the voting summaries located within the pen drives. The threads will prove that more than one device can be connected at once to the eVote system. This task will be assigned to Angel due to the driver implementation was originally his to complete.

## Screens Modifications

The proposal indicated that when an item is selected a rectangle will select the highlighted item and in the ballot's screen, a circle will fill when highlighting a specific row. This has been modified as follows:

1. The confirmation screens that have yes or no choices will have parenthesis below them and the highlighted item will have a cursor flashing between the parentheses of the yes or no choice. As seen below.

Yes no

( ||| ) ( )

1. The ballots screen will also implement the parenthesis and cursor highlighting methods.

## Contact Information

For any additional information please contact the JSAL project manager.

Javier Torres

Cel. 787-248-5262

Email: Javier.Torres@ece.uprm.edu