UNIVERSITY OF GAZIANTEP

RFID Based Electonic Voting System

EEE 499 GRADUATION PROJECT

IN

ELECTRICAL & ELECTRONICS ENGINEERING

BY

CEM KIRAY

2018

ABSTRACT

RFID Based Electronic Voting System

KIRAY, Cem

Graduation Project in Electrical and Electronics Engineering

Supervisor: Prof.Dr.Savaş UÇKUN

January 2018, 22 pages

In this project an RFID based electronic voting system which uses the RFID techonology to identify the voters is designed and constructed. The project can be divided to 2 parts which are computer side and controller side. The main computer stores all the data related to voting and the voters, and provides a GUI that shows voter's name, photo and eligibility to vote. On the controller side, which is the electronic voting circuit, microcontroller (Arduino) gets the data and sends to the computer via serial. This project also has capability to handle a number of people voting simultaneously.

Although electronic voting system is being used in some countries such as USA, this project combines RFID technology and the electronic voting system which is a first in the world.

Keywords: RFID, Electronic Voting

ÖZET

RFID Tabanlı Elektronik Oylama Sistemi

KIRAY, Cem

Mezuniyet Projesi, Elektrik ve Elektronik Mühendisliği Bölümü

Proje yöneticisi: Prof.Dr.Savaş UÇKUN

Ocak 2018, 22 sayfa

Bu projede RFID teknolojisini seçmenlerin tanınması için kullanan bir RFID tabanlı elektronik oylama sistemi tasarlandı ve oluşturuldu. Proje bilgisayar tarafı ve denetleyici tarafı olarak iki kısma ayrılabilir. Ana bilgisayar oylama ve seçmenlerle ile ilgili verileri saklar ve seçmenin ismi, fotoğrafı ve oy verebilme durumunu gösteren bir görsel arayüz sağlar. Denetleyici tarafında ise mikrodenetleyici (Arduino) verileri toplar ve ana bilgisayara serial port aracılığıyla gönderir. Bu proje çok sayıda seçmenin aynı anda oy kullanmasına da olanak sağlar.

Elektronik seçim sisteminin şuanda ABD gibi bazı ülkelerde kullanılıyor olmasına rağmen bu proje elektronik oylama sistemini, RFID teknolojisiyle birleştiriyor. Bu yönüyle mevcut sistemlerden ayrılabilir.

Anahtar kelimeler: RFID, Elektronik Oylama

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof.Dr.Savaş UÇKUN who helped me a lot with his ideas and support while designing the project and throughout the semester.

I also would like to acknowledge my family and friends who supported me while developing this project.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF SYMBOLS

RFID	**R**adio-**F**requency **Id**entification
UID	Unique ID
GUI	Graphical User Interface

1.INTRODUCTION

RFID stands for Radio Frequency Identification. It uses electromagnetic fields to automatically identify and track tags attached to objects. The tags contain electronically stored information. Passive tags collect energy from interrogating radio waves of a nearby RFID reader. On the other hand, active tags have a local power source such as a battery and may operate hundreds of meters from the RFID reader.

In this project, a RFID based electronic voting system is developed. This system combines the usage of RFID technology for identification and the ease and advantages of electronic voting.

Although some countries such as United States, Brazil, India are using electronic voting currently, this project improves the electronic voting system with the help of RFID technology. In this respect, we can say that this project is a first in the world.

This project, RFID based electronic voting system, has the following features and advantages:

- It uses RFID tags in the form of cards (tags can be embed) for identifying the voters.
- This system can be used to control multiple voting booths which means that a number of people can vote simultaneously in the voting booths placed side by side.
- It provides a GUI (Graphical User Interface) which shows the voter's photo, name and eligibility to vote. If the person is eligible to vote, It shows which booth is empty if there is any empty booth or tells that all of them are occupied otherwise.
- With this system, "invalid votes" and "counting mistakes" are prevented which are very common problems in Turkey's current voting system.
- It will speed up the counting of ballots, as soon as the voting ends the results can be seen.

Identification of the voter is solid, security-wise unless the security guards makes a mistake, since he will check if the photo and the name matches with the owner of the card. And there will be no "invalid votes" because in the booth, the voter can select only a single party or select blank vote by using the buttons. In addition, the voter is asked to confirm his selection by pushing the button twice so there is a small possibility of making a wrong selection.

2.RFID BASED ELECTONIC VOTING SYSTEM

2.1. HOW THE SYSTEM WORKS

When a RFID card is shown to the reader, Arduino gets the card UID (Unique ID) number and sends to the computer via serial communication. Python script tries to match the UID of the card with the UIDs stored in the database. If there is a match, and if the voter is eligible to vote the GUI guides him to the available (if available) booth , or if the UID not found in the database or the voter has already voted, it shows a error or warning message on the computer screen.

After the security guard confirms that everything is correct, he clicks on the "OK" button on the GUI and the computer sends a signal to Arduino and the voting process starts.

In the booth, a LCD Display guides the voter during the voting process. The voter chooses a party or a blank vote using pushbuttons and he confirms his choice by pushing the same button again. If he wants to change his selection, he can go back by using the “function button”.

After the vote is confirmed by the voter, vote data is sent to computer. Computer stores the vote data and adds the voter’s card UID to the “voted UIDs” list , which means if he tries to vote again, the computer will detect that the UID is marked as “already voted” and an error message will be shown on the screen.

When the voting ends, by running the Python script with the predefined arguments *(see Section 4)*, the results can be seen. It also shows the statistics such as all UIDs, voted UIDs, number of votes of each party, number of blank votes and the percentage of the attendance of the voters.

3.HARDWARE

3.1.Equipments used in this project

Arduino UNO microcontrollers are used for controlling the identification and the voting process. UNO model is used because of its cost efficiency and availability, and it is capable of handling the requirements of the project.

The power required for microcontrollers are supplied by the main computer via USB cables, and Arduino distributes the power to the connected parts according to their rated values.

MFRC 522 RFID reader is used as the card reader which supports 13.56MHz contactless communication. It is supplied 3.3 Volts by Arduino. For further information about MFRC522, please see Appendices- B.

16X02(16 columns, 2 rows) LCD display is used in the voting circuits to guide the voters and it communicates with the Arduino over I2C bus which simplifies the connections since it only needs 4 pins. It is supplied 5 Volts by Arduino.

Every single booth is controlled with another Arduino board. In the case of a problem in a booth, the other booths will still be available and the voting will not be intercepted.

There are 6 main parts of the system which are:

- Arduino UNO Microcontroller *: Controls the other parts*
- Main Computer *: Arranges the incoming and the outgoing data and provides GUI for identification of the owner of the card*
- MFRC522 RFID Reader *: Reads nearby RFID cards*
- LCD Display *: Guides the voter during the voting*
- Buttons *: Used for selection*
- Buzzer *: Gives an alert when the voting starts*

The basic block diagram of the system can be seen below:

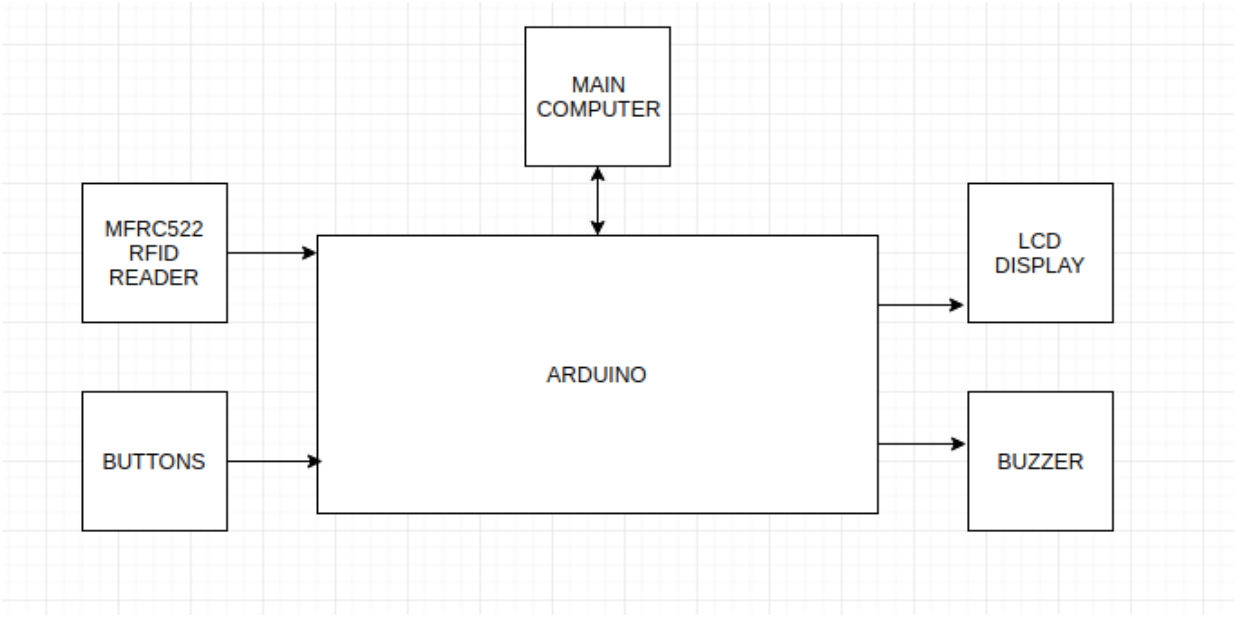


Fig 3.1: Block diagram of the system

3.2.Connections of the voting circuits

The connections for Arduino boards can be seen below:

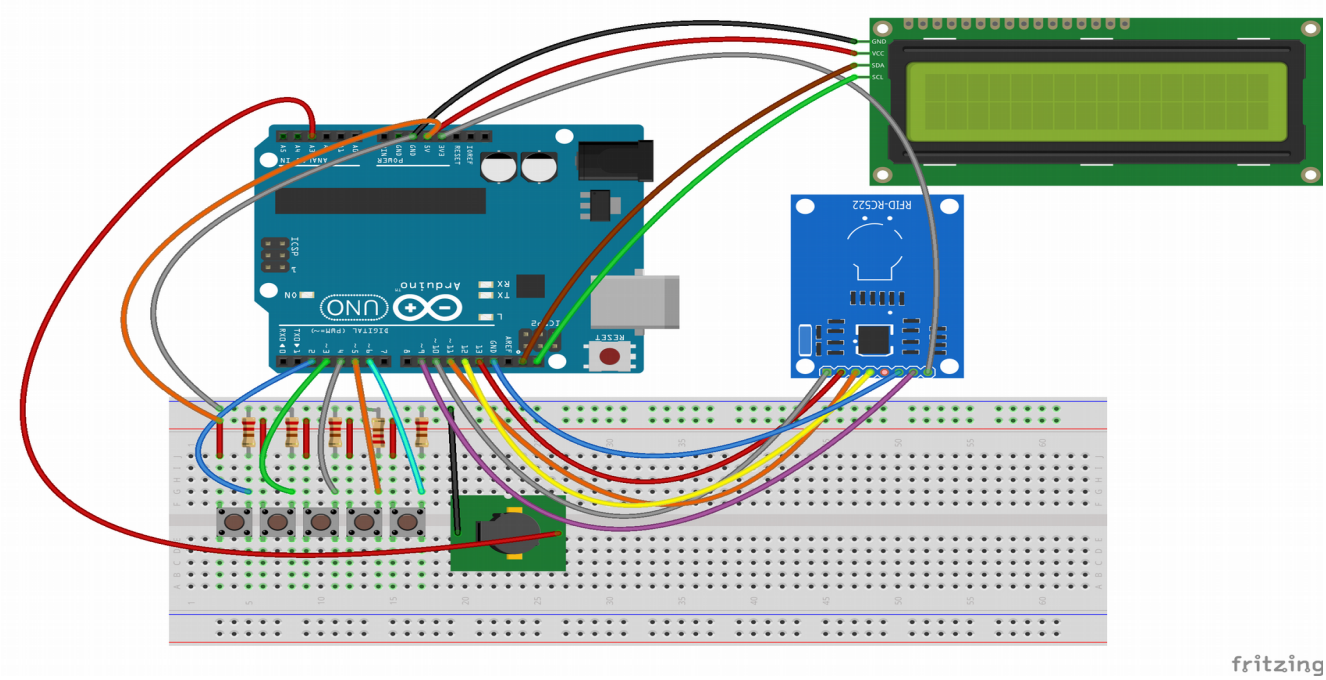


Figure 3.2-a : Connections of the first circuit with the RFID reader

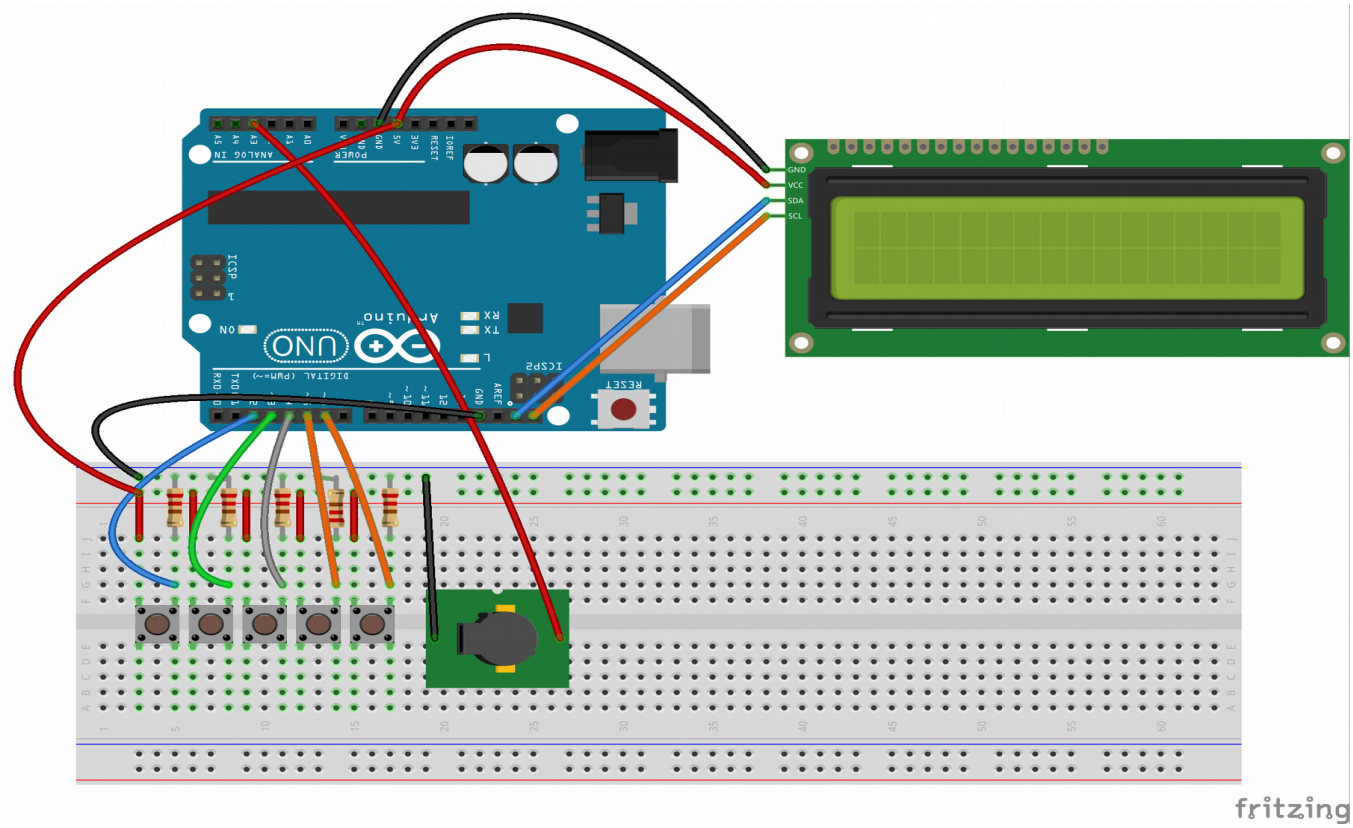


Figure 3.2-b : Connections of the second circuit

4.SOFTWARE

On the computer side, a Python script is developed to arrange the incoming and the outgoing data from/into the Arduino boards and show the GUI. The software consists of 3 main source files: a Python script and two Arduino source files for each voting circuit.

To see the results of the voting, the Python script can be run with the argument '-showDb', for example : 'python evoting.py -showDb'. Or, to reset the database '-reset' argument should be used, but this removes all the saved data.

The Python script can be seen below:

```
1 import serial # serial module is used to communicate with the serial port
2 import time
3 import shelve # shelve module is used to read data from the database
4 import sys
5 from Tkinter import * # Tkinter is used for GUI
6 import tkMessageBox
7
8 try:
9     cmdArg = str(sys.argv[1])
10    if(cmdArg == '-reset'):
11        # this script can be called with the arg. '-reset'
12        # to reset database (THIS REMOVES ALL SAVED DATA !)
```

```
13          dbs = shelve.open('uid_data')
14          dbs['votedUids'] = []
15          dbs['voteCounts'] = {'red':0,'green':0,'blue':0,'blank':0}
16          dbs.close()
17          print "="*10
18          print " ** DATABASE IS RESET **"
19          print "="*10
20          print "arg 1 : " , str(sys.argv[1])
21      elif(cmdArg == '-showDb'):
22          # also can be called with the arg. '-showDb'
23          # to see the results
24          dbs = shelve.open('uid_data')
25          print "="*15
26          print " > VOTED LIST : " , dbs['votedUids']
27          print " > COUNTS     : " , dbs['voteCounts']
28          if(len(dbs['votedUids']) != len(dbs['uidsNames'].keys())):
29              numOfVoted = len(dbs['votedUids'])
30              numOfVoters = len(dbs['uidsNames'])
31              print numOfVoted , "voters have voted out of " , numOfVoters
32              percentOfAttendance = (numOfVoted * 100 / numOfVoters)
33              print percentOfAttendance , "percent of the voters has voted."
34              print (100-percentOfAttendance) , "percent of the voters didn't attend."
35          if(len(dbs['votedUids']) == len(dbs['uidsNames'].keys())):
36              print(" VOTING FINISHED!")
37              redCount = dbs['voteCounts']['red']
38              greenCount = dbs['voteCounts']['green']
39              blueCount = dbs['voteCounts']['blue']
40              blankCount = dbs['voteCounts']['blank']
41              sys.exit(0)
42          print "="*15
43          dbs.close()
44  except IndexError:
45      x=0
46          # do nothing if no argument is written in commandline
47
48
49  # NOTE the user must ensure that the serial port and baudrate are correct
50  serialPort = "/dev/ttyUSB0"
51  serialPort2 = "/dev/ttyUSB1"
52  baudRate = 9600
53  baudRate2 = 19200
54  ser1 = serial.Serial(serialPort, baudRate)
55  # ser1(ttyUSB0) should be connect to the main circuit with the RFID reader
56  # or ser1 and ser2 must be swapped
57  ser2 = serial.Serial(serialPort2 , baudRate2)
58  print "Serial port " + serialPort + " opened  Baudrate " + str(baudRate)
59  print "Serial port " + serialPort2 + " opened  Baudrate " + str(baudRate2)
60
61  votingUids = [] #this list stores the UIDs that are currently voting
62  booth1available = True
63  booth2available = True
64  booth1Uid = 0
65  booth2Uid = 0
66
67  def showGuiMsg(msg , booth , cardUID , voterName):
```

```
    #this fn. shows a GUI message according to
    #the status of the voter such as 'YOU CAN VOTE' , 'YOU ALREADY VOTED' etc.
    master = Tk()
    imgPath = str(cardUID) + ".png"
    name = voterName
    img = PhotoImage(file=imgPath)
    imgPanel = Label(master , image = img)

    def canVoteButton():
        #if the person is eligible to vote
        #print the starting command to the serial of the available circuit
        if(booth == "booth1"):
            ser1.write('S')
        elif(booth == "booth2"):
            ser2.write('<start>')
            print "ser2 write start"
        master.destroy()
        master.quit()

    if(msg=="voted"):
        #if the person has already voted, shows an error message
        nameMsgText = "YOU HAVE ALREADY VOTED\n" + name
        textBg = 'red'
        statusImg = PhotoImage(file = 'error.png')
        okButton = Button(master , text="OK" , command=master.destroy)

    elif(msg=="canVote"):
        nameMsgText = "YOU CAN VOTE\n" + name
        textBg = 'green'
        statusImg = PhotoImage(file = 'ok.png')
        okButton = Button(master , text="OK" , command=canVoteButton)
    elif(msg=="occupied"):
        #if all of the booths are occupied, show a warning
        nameMsgText = "ALL BOOTHS ARE OCCUPIED NOW!\nPLEASE WAIT!"
        textBg = 'blue'
        statusImg = PhotoImage(file = 'error.png')
        okButton = Button(master , text="OK" , command=master.destroy)

    boothMsgText = " "
    #print which booth is available (if any)
    if(booth == "booth1"):
        boothMsgText = "GO TO BOOTH 1"
    elif(booth == "booth2"):
        boothMsgText = "GO TO BOOTH 2"
    boothMsg = Message(master , text=boothMsgText , width =550 ,  anchor = CENTER)
    nameMsg = Message(master , text=nameMsgText , width=550, anchor=CENTER)
    nameMsg.config(bg=textBg ,fg='white', font=( 48 ) , anchor=CENTER )
    statusImgPanel = Label(master, image = statusImg)
    statusImgPanel.pack()
    imgPanel.pack()
    boothMsg.pack()
    nameMsg.pack()


    okButton.pack()
```

```
123
124     master.mainloop()
125
126
127
128 def startBooth(cardUID , voterName):
129     #direct voter to the available booth(if any)
130     #and start the voting circuit
131     global booth1available , booth2available , booth1Uid , booth2Uid
132
133     if(booth1available):
134         booth1Uid = cardUID
135         showGuiMsg("canVote" , "booth1" , booth1Uid , voterName)
136         print " >> GO BOOTH 1"
137         print "="*15
138         booth1available = False #mark booth1 as occupied
139         votingUids.append( cardUID ) #add card UID to currently voting UIDs list
140
141     elif(booth2available):
142         booth2Uid = cardUID
143         showGuiMsg("canVote" , "booth2" , booth2Uid , voterName)
144         print " >> GO BOOTH 2"
145         print "="*15
146         booth2available = False
147         votingUids.append( cardUID )
148
149     else: #if no booths are available
150         print("Both booths are occupied")
151         showGuiMsg("occupied" , "x" , cardUID , "test")
152         print "="*15
153
154
155 def checkForNewCard():
156     global booth1available , booth2available , booth1Uid , booth2Uid
157     if(ser1.inWaiting()): #if there is new data in the serial
158         serLine = ser1.readline() #read the line
159         if(serLine[0] == 'R'): #if the fetched line is uid
160         #'R' is a marker to show that the line contains Rfid card UID number
161             cardUidLine = serLine.split("\n")[0]
162             cardUID = int(cardUidLine[1::])
163             print " >> CARD UID:" , cardUID
164
165             if(cardUID in allUids):
166                 voterName = uidNameDict[cardUID]
167                 print " >> NAME : " , voterName
168
169                 if(cardUID in votingUids): #if the voter is currently voting
170                     print "YOU ARE VOTING NOW."
171                     print "="*15
172                 elif(cardUID in votedUids): #if the voter has already voted
173                     print "YOU HAVE ALREADY VOTED"
174                     showGuiMsg("voted" ,"x" , cardUID , voterName)
175                     print "="*15
176                 elif(cardUID not in votingUids and cardUID not in votedUids):
177                 #if the voter is eligible to vote
```

```
178                     print "START VOTING " , voterName
179                     startBooth(cardUID , voterName)
180                     print "="*15
181             else: #if no match
182                 print "CARD ID NOT FOUND"
183                 print "="*15
184         else:
185             c=0 # do nothing
186
187 def logVote(booth, vote):
188     #this fn. increments the vote counts according to
189     #the selected option
190     global redCount , greenCount , blueCount , blankCount
191     if(vote == 'RED'):
192         print(" >> RED selectedx")
193         redCount += 1
194
195     elif(vote == 'GREEN'):
196         print(" >> GREEN selected")
197         greenCount += 1
198
199     elif(vote == 'BLUE'):
200         print(" >> BLUE selected")
201         blueCount += 1
202
203     elif(vote == 'BLANK'):
204         print(" >> BLANK selected")
205         blankCount += 1
206
207     else:
208         print(" >> SELECTION IS INVALID")
209
210 def checkForVote():
211     #this fn. checks the serial for new
212     #vote data
213     global booth1available , booth2available
214     if(ser1.inWaiting()): #if new data is waiting in ser1
215         serLine1 = ser1.readline()
216         if(serLine1[0] == 'V'): # if the fetched line is vote
217         #'V' is a marker to show that the line contains Vote data
218             voteLine1 = serLine1.split("\n")[0]
219             vote1 = voteLine1[1::]
220             print " >> VOTE 1: " , vote1
221             logVote(booth1available , vote1)
222             print "="*15
223             votedUids.append(booth1Uid)
224             votingUids.remove(booth1Uid)
225             booth1available = True
226             ser1.write('X') # tell arduino to restart
227
228     elif(ser2.inWaiting()): #if new data is waiting in ser2
229         serLine2 = ser2.readline()
230         if(serLine2[0] == 'V'): # if the fetched line is vote
231             voteLine2 = serLine2.split("\n")[0]
232             vote2 = voteLine2[1::]
```

```
            print " >> VOTE 2: " , vote2
            logVote(booth2Uid , vote2)
            print "="*15
            votedUids.append(booth2Uid)
            votingUids.remove(booth2Uid)
            booth2available = True
            #ser2.write('X') # tell arduino to restart
    else:
        x=2 #do nothing



while(1):
    #start an infinite loop to continuously check for new data
    #the program will continue until terminated by the user
    dbShelfFile = shelve.open('uid_data') # open the database(shelve)

    #get the data and store them in variables
    uidNameDict = dbShelfFile['uidsNames']
    allUids = uidNameDict.keys()

    name = dbShelfFile['uidsNames'].values()
    votedUids = dbShelfFile['votedUids']
    voteCounts = dbShelfFile['voteCounts']
    redCount = voteCounts['red']
    greenCount = voteCounts['green']
    blueCount = voteCounts['blue']
    blankCount = voteCounts['blank']

    dbShelfFile.close()

    #main loop
    checkForNewCard() #check if a new card is present
    checkForVote() #get the vote data

    #write new data to the database
    dbShelfFile = shelve.open('uid_data')
    dbShelfFile['votedUids'] = votedUids
    voteCounts['red']=redCount
    voteCounts['green']=greenCount
    voteCounts['blue']=blueCount
    voteCounts['blank']=blankCount
    dbShelfFile['voteCounts'] = voteCounts
    dbShelfFile.close() #close the database
```

The source code for the first booth with the RFID reader:

```
//RFID reader libraries
#include <MFRC522Hack.h>
#include <MFRC522Extended.h>
#include <deprecated.h>
```

```
#include <MFRC522Debug.h>
#include <require_cpp11.h>
#include <MFRC522.h>
//LCD libraries
#include <SPI.h>
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

#define I2C_ADDR    0x27 // I2C address of the LCD display
#define BACKLIGHT_PIN     3
#define En_pin  2
#define Rw_pin  1
#define Rs_pin  0
#define D4_pin  4
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7

//init. LCD display
LiquidCrystal_I2C  lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

#define SS_PIN 10
#define RST_PIN 9

#define BUZ A3 //buzzer
#define G_BUTTON 4 //green button
#define W_BUTTON 5 //white button
#define OK_BUTTON 6 //black button
#define B_BUTTON 2 //blue button
#define R_BUTTON 3 //red button1

char receivedChar; //the received chars from serial port is stored in this variable
boolean newData = false;

MFRC522 mfrc522(SS_PIN , RST_PIN);

void lcdClearLine(int line){
  //this fn. cleans the desired line on the lcd display
  lcd.setCursor(0 , line);
  String blank;
  for(int i = 1 ; i <= 16 ; i++){
    blank += " ";
  }
  lcd.print(blank);
}

void lcdPrint(int line , String text){
  //shortcut fn. to print text on the lcd display
  //it first cleans the line and then prints the text
    lcdClearLine(line);

  lcd.setCursor(0 , line);

  lcd.print(text);
```

```
}

void buzzerOK(){
  //fn. to activate buzzer in a sequence
    digitalWrite(BUZ,HIGH);
    delay(150);
    digitalWrite(BUZ,LOW);
    delay(150);
    digitalWrite(BUZ,HIGH);
    delay(150);
    digitalWrite(BUZ,LOW); }

char recvOneChar() {
  //this fn. is used to receive a single char
  //that is printed on the serial
    if (Serial.available() > 0) {
      //if a new char is received, it is stored in 'receivedChar'
      //and the fn. returns to 'receivedChar'
        receivedChar = Serial.read();
        newData = true;
        return receivedChar;
    }
}


// TODO : Change this function to get HEX UID and store in an array
unsigned long getCardID(int showCardData = 0){
  /* this function gets the uid of the rfid card
   and returns the uid in unsigned long format */
  /* this fn. should be called with argument 1
  if printing the card data to the serial monitor
  is needed. */

  unsigned long UID_unsigned;
  UID_unsigned =  mfrc522.uid.uidByte[0] << 24;
  UID_unsigned += mfrc522.uid.uidByte[1] << 16;
  UID_unsigned += mfrc522.uid.uidByte[2] <<  8;
  UID_unsigned += mfrc522.uid.uidByte[3];

  if(showCardData==1){
    Serial.println("UID Unsigned int");
    Serial.println(UID_unsigned);
    String UID_string =  (String)UID_unsigned;
    long UID_LONG=(long)UID_unsigned;

    Serial.println("UID Long :");
    Serial.println(UID_LONG);
    Serial.println("UID String :");
    Serial.println(UID_string);
  } // print card data to the serial monitor

  int UID_int = (int) UID_unsigned;
  return UID_int;
}

```

```
115 void setup() {
116   pinMode(BUZ       , OUTPUT);
117   pinMode(B_BUTTON  , INPUT);
118   pinMode(G_BUTTON  , INPUT);
119   pinMode(B_BUTTON  , INPUT);
120   pinMode(W_BUTTON  , INPUT);
121   pinMode(OK_BUTTON , INPUT);
122 
123   Serial.begin(9600); // Init. serial comm. with the PC
124   SPI.begin(); // Init. SPI bus for MFRC522
125   mfrc522.PCD_Init(); // Init. MFRC522 card
126 
127   lcd.begin(16,2);
128   lcd.setBacklightPin(BACKLIGHT_PIN , POSITIVE);
129   lcd.setBacklight(HIGH);
130   lcd.home();
131   lcdPrint(0 , "- ELECTION 18' -");
132   lcdPrint(1 , "Waiting for Card ");
133 }
134 
135 char canVote;
136 
137 void checkForNewCard(){
138   //checks if there is an RFID card is present in the field of the reader
139   if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() ){
140     //if a card is present, it gets the card UID and prints to the serial ports
141     int newCardUID = getCardID();
142     Serial.print("R");
143     Serial.print(newCardUID);
144     Serial.print("\n");
145     delay(1500);
146   }
147 }
148 
149 void loop() {
150     // Look for new cards
151   if ( ! mfrc522.PICC_IsNewCardPresent() )
152     return;
153 
154   // Verify if the NUID has been readed
155   if ( ! mfrc522.PICC_ReadCardSerial() )
156     return;
157 
158   int cardUID = getCardID();
159 
160   Serial.print("R");
161   Serial.print(cardUID);
162   Serial.println("\n");
163   delay(1000);
164 
165 do {
166   checkForNewCard();
167 int c; // do nothing
168 } while(recvOneChar() != 'S');
169 
```

```
  lcdPrint(0 , " USE BUTTONS TO ");
  lcdPrint(1 , " SELECT A PARTY ");
  buzzerOK();
  getVote(); //call the fn. to start the voting process
  while(recvOneChar() != 'X'){
    int ac;
  }
}


bool confirmSelection(String vote , int pin){
/*this fn. asks user to confirm his selection
by pushing the same button again, or to cancel
by pushing the black button.*/
  String text = vote + "  -> CONFIRM";
  String text2 = "BLACK -> CANCEL";
  lcdPrint(0 , text);
  lcdPrint(1 , text2);
  bool voteStatus;
  while(1){
    checkForNewCard();
    if(digitalRead(pin) == HIGH){
      lcdPrint(0 , "SUCCESS !");
      lcdPrint(1 , vote + " SELECTED");

      voteStatus = true;
      break;
    }
    else if(digitalRead(OK_BUTTON)==HIGH){
      lcdPrint(0 , " VOTE  CANCELLED");
      lcdPrint(1 , "  SELECT  AGAIN");
      voteStatus = false;
      break;
    }
    else
    continue;
  }
  return voteStatus;

}

void getVote(){
  /* this fn. waits for the button clicks
  after a selection is confirmed, it prints the vote
  information to serial port (sends to python script)*/
  while(1){
    checkForNewCard(); //while waitin for buttons, also check for new cards
    if(digitalRead(R_BUTTON) == HIGH){
      lcdPrint(0 , "RED SELECTED");
      lcdPrint(1 , "Please wait...");
      delay(1000);
      if(confirmSelection("RED" , R_BUTTON) == true){ //if selection is confirmed
      lcdPrint(0 , "VOTE CONFIRMED");
      Serial.println("");
      Serial.print("VRED");
```

```
      Serial.print('\n');
      delay(2000);
      lcdPrint(0 , "");
      //lcdPrint(1 , "");
        break;
      }
      else
      continue;
    }
    else if(digitalRead(G_BUTTON) == HIGH){
      lcdPrint(0 , "GREEN SELECTED");
      lcdPrint(1 , "Please wait...");
      delay(1000);
      if(confirmSelection("GREEN" , G_BUTTON) == true){
     lcdPrint(0 , "VOTE CONFIRMED");
     Serial.println("");
        Serial.print("VGREEN");
       Serial.print('\n');
       delay(2000);
       lcdPrint(0 , "");
       lcdPrint(1 , "");
        break;
      }
      else
      continue;
    }
    else if(digitalRead(B_BUTTON) == HIGH){
      lcdPrint(0 , "BLUE SELECTED");
      lcdPrint(1 , "Please wait...");
      delay(1000);
      if(confirmSelection("BLUE" , B_BUTTON) == true){
     lcdPrint(0 , "VOTE CONFIRMED");
     Serial.println("");
        Serial.print("VBLUE");
       Serial.print('\n');
       delay(2000);
       lcdPrint(0 , "");
       lcdPrint(1 , "");
        break;
      }
      else
      continue;
    }
    else if(digitalRead(W_BUTTON) == HIGH){
      lcdPrint(0 , "BLANK SELECTED");
      lcdPrint(1 , "Please wait...");
      delay(1000);
      if(confirmSelection("BLANK" , W_BUTTON) == true){
     lcdPrint(0 , "VOTE CONFIRMED");
     Serial.println("");
        Serial.print("VBLANK");
       Serial.print('\n');
       delay(2000);
       lcdPrint(0 , "");
       lcdPrint(1 , "");
```

```
280         break;
281       }
282       else
283       continue;
284     }
285     else
286     continue;
287   }
288 }
```

The source code for the second booth:

```
 1 //LCD libraries
 2   #include <Wire.h>
 3   #include <LCD.h>
 4   #include <LiquidCrystal_I2C.h>
 5
 6   #define I2C_ADDR    0x26 // I2C address of the LCD display
 7   #define BACKLIGHT_PIN     3
 8   #define En_pin  2
 9   #define Rw_pin  1
10   #define Rs_pin  0
11   #define D4_pin  4
12   #define D5_pin  5
13   #define D6_pin  6
14   #define D7_pin  7
15
16   //init. LCD display
17   LiquidCrystal_I2C  lcd(I2C_ADDR, En_pin, Rw_pin, Rs_pin, D4_pin, D5_pin, D6_pin, D7_pin);
18
19   #define BUZ A3 //buzzer
20   #define G_BUTTON 4 //green button
21   #define W_BUTTON 5 //white button
22   #define OK_BUTTON 6 //black button
23   #define B_BUTTON 2 //blue button
24   #define R_BUTTON 3 //red button1
25
26
27   char receivedChar; //the received chars from serial port is stored in this variable
28
29
30   const byte numChars = 32;
31   char receivedChars[numChars];
32
33   boolean newData = false;
34
35   void lcdClearLine(int line) {
36     //this fn. cleans the desired line on the lcd display
37     lcd.setCursor(0 , line);
38     String blank;
39     for (int i = 1 ; i <= 16 ; i++) {
40       blank += " ";
```

```
41      }
42      lcd.print(blank);
43    }
44
45    void lcdPrint(int line , String text) {
46      //shortcut fn. to print text on the lcd display
47      //it first cleans the line and then prints the text
48      lcdClearLine(line);
49
50      lcd.setCursor(0 , line);
51
52      lcd.print(text);
53    }
54
55    void buzzerOK() {
56      //fn. to activate buzzer in a sequence
57      digitalWrite(BUZ, HIGH);
58      delay(150);
59      digitalWrite(BUZ, LOW);
60      delay(150);
61      digitalWrite(BUZ, HIGH);
62      delay(150);
63      digitalWrite(BUZ, LOW);
64    }
65
66    char recvOneChar() {
67      //this fn. is used to receive a single char
68      //that is printed on the serial
69      if (Serial.available() > 0) {
70        //if a new char is received, it is stored in 'receivedChar'
71        //and the fn. returns to 'receivedChar'
72        receivedChar = Serial.read();
73        newData = true;
74        return receivedChar;
75      }
76    }
77
78    void recvWithStartEndMarkers() {
79      //this fn. is used to receive multiple character messages
80      //which is enclosed by '<' and '>'
81       static boolean recvInProgress = false;
82       static byte ndx = 0;
83       char startMarker = '<';
84       char endMarker = '>';
85       char rc;
86
87       while (Serial.available() > 0 && newData == false) {
88           rc = Serial.read();
89
90           if (recvInProgress == true) {
91               if (rc != endMarker) {
92                   receivedChars[ndx] = rc;
93                   ndx++;
94                   if (ndx >= numChars) {
95                       ndx = numChars - 1;
```

```
 96                 }
 97             }
 98             else {
 99                 receivedChars[ndx] = '\0'; // terminate the string
100                 recvInProgress = false;
101                 ndx = 0;
102                 newData = true;
103             }
104         }
105 
106         else if (rc == startMarker) {
107             recvInProgress = true;
108         }
109     }
110 }
111 
112  void setup() {
113    pinMode(BUZ   , OUTPUT);
114    pinMode(B_BUTTON , INPUT);
115    pinMode(G_BUTTON , INPUT);
116    pinMode(B_BUTTON , INPUT);
117    pinMode(W_BUTTON , INPUT);
118    pinMode(OK_BUTTON , INPUT);
119 
120    Serial.begin(19200); // Init. serial comm. with the PC
121 
122    lcd.begin(16, 2);
123    lcd.setBacklightPin(BACKLIGHT_PIN , POSITIVE);
124    lcd.setBacklight(HIGH);
125    lcd.home();
126    lcd.print(0 , "- ELECTION 18' -");
127    lcdPrint(1 , "Waiting for Card ");
128  }
129 
130  char canStart;
131 
132  void loop() {
133 
134 
135    while (true){
136      recvWithStartEndMarkers();
137      String message(receivedChars);
138      //  lcdPrint(0 , receivedChars);
139 
140      if ( message == "start" ){
141        // receivedChars[0] = '0';
142         break;
143      }
144 
145      else{
146        lcd.setCursor(0,0);
147        lcd.print("- ELECTION 18' -");
148        lcd.setCursor(0,1);
149        lcd.print("Waiting for card");
150        continue;
```

```
151
152       }
153
154     }
155
156
157       lcdPrint(0 , " USE BUTTONS TO ");
158       lcdPrint(1 , " SELECT A PARTY ");
159       buzzerOK();
160       getVote();
161       lcdPrint(0 , "- ELECTION 18' -");
162       lcdPrint(1 , "Waiting for Card ");
163       newData = false;
164       receivedChars[0] = 0;
165   }
166
167   bool confirmSelection(String vote , int pin) {
168
169     String text = vote + "  -> CONFIRM";
170     String text2 = "BLACK -> CANCEL";
171     lcdPrint(0 , text);
172     lcdPrint(1 , text2);
173     bool voteStatus;
174     while (1) {
175       if (digitalRead(pin) == HIGH) {
176         lcdPrint(0 , "SUCCESS !");
177         lcdPrint(1 , vote + " SELECTED");
178         voteStatus = true;
179         break;
180       }
181       else if (digitalRead(OK_BUTTON) == HIGH) {
182         lcdPrint(0 , " VOTE  CANCELLED");
183         lcdPrint(1 , "  SELECT  AGAIN");
184         voteStatus = false;
185         break;
186       }
187       else
188         continue;
189     }
190     return voteStatus;
191
192   }
193
194   void getVote() {
195     while (1) {
196       if (digitalRead(R_BUTTON) == HIGH) {
197         lcdPrint(0 , "RED SELECTED");
198         lcdPrint(1 , "Please wait...");
199         delay(1000);
200         if (confirmSelection("RED" , R_BUTTON) == true) {
201           lcdPrint(0 , "VOTE CONFIRMED");
202           Serial.print("VRED");
203           Serial.print('\n');
204           delay(2000);
205           lcdPrint(0 , "");
```

```
206           lcdPrint(1 , "");
207           break;
208         }
209         else
210           continue;
211       }
212       else if (digitalRead(G_BUTTON) == HIGH) {
213         lcdPrint(0 , "GREEN SELECTED");
214         lcdPrint(1 , "Please wait...");
215         delay(1000);
216         if (confirmSelection("GREEN" , G_BUTTON) == true) {
217           lcdPrint(0 , "VOTE CONFIRMED");
218           Serial.print("VGREEN");
219           Serial.print('\n');
220           delay(2000);
221           lcdPrint(0 , "");
222           lcdPrint(1 , "");
223           break;
224         }
225         else
226           continue;
227       }
228       else if (digitalRead(B_BUTTON) == HIGH) {
229         lcdPrint(0 , "BLUE SELECTED");
230         lcdPrint(1 , "Please wait...");
231         delay(1000);
232         if (confirmSelection("BLUE" , B_BUTTON) == true) {
233           lcdPrint(0 , "VOTE CONFIRMED");
234           Serial.print("VBLUE");
235           Serial.print('\n');
236           delay(2000);
237           lcdPrint(0 , "");
238           lcdPrint(1 , "");
239           break;
240         }
241         else
242           continue;
243       }
244       else if (digitalRead(W_BUTTON) == HIGH) {
245         lcdPrint(0 , "BLANK SELECTED");
246         lcdPrint(1 , "Please wait...");
247         delay(1000);
248         if (confirmSelection("BLANK" , W_BUTTON) == true) {
249           lcdPrint(0 , "VOTE CONFIRMED");
250           Serial.print("VBLANK");
251           Serial.print('\n');
252           delay(2000);
253           lcdPrint(0 , "");
254           lcdPrint(1 , "");
255           break;
256         }
257         else
258           continue;
259       }
260       else
```

```
261             continue;
262         }
263     }
264 
```

5.CONCLUSION

The main purposes of this project were to develop a voting system which is easier and faster than current voting methods. With the help of RFID technology, identification of the voters will be easier and also thanks to electronic voting, as soon as the voting ends, we will be able to see the results and there will be no “invalid votes” since it is not possible to vote incorrectly in this system. Although some countries such as USA are currently using electronic voting, this project improves it by combining the electronic voting and RFID technology.

A biometric fingerprint scanner for identification can be added to the system for further security concerns in addition to RFID reader.

REFERENCES

- **http://forum.arduino.cc/index.php?topic=271097.0 (Arduino Control with Python)**
- **http://forum.arduino.cc/index.php?topic=396450.0 (Serial Input Basics)**
- **http://forum.arduino.cc/index.php?topic=261445.0 (Planning and Implementing an Arduino Program)**
- **https://www.devdungeon.com/content/gui-programming-python#fullscreen (GUI Programming with Python)**
- **https://github.com/miguelbalboa/rfid (RFID Library for Arduino IDE)**
- **https://www.sunfounder.com/learn/Sensor-Kit-v2-0-for-Arduino/lesson-1-display-by-i2c-lcd1602-sensor-kit-v2-0-for-arduino.html (LCD1602 I2C Connection)**
- **https://arduino.stackexchange.com/questions/30644/how-do-i-change-the-i2c-address-on-the-lcd-backpack (Changing I2C Address of the LCD1602)**
- **https://www.cprogramming.com/ (C programming)**
- **https://en.wikipedia.org/wiki/Electronic_voting (Electronic Voting)**

APPENDICES

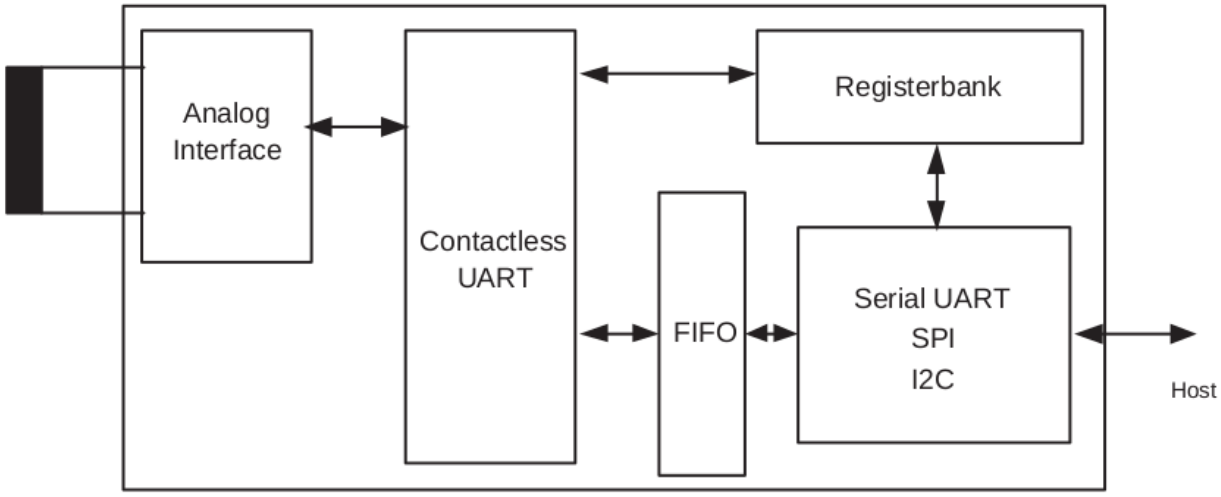
A) COST ANALYSIS

Amount	Item	Cost ($)	Cost (TL)
2	Arduino UNO (Clone)	6.8 (3.4 * 2)	23.8
1	MFRC522 RFID Reader	1.55	5.42
12	12x12mm Pushbuttons	3.96 (0.33 * 12)	12.67
2	Buzzer	0.2 (0.1 * 2)	0.7
2	16x02 LCD I2C Display	3.64 (1.82 * 2)	12.77
~	Various Resistors	~ 0.5	~ 1.75
~	Varios Jumper Wires	~ 1.5	~ 5.25
	TOTAL COST	**18.15**	**63.52**

B) MFRC522 Contactless Reader IC

The MFRC522 is a highly integrated reader/writer for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO 14443A / MIFARE® mode. The MFRC522's internal transmitter part is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443A/MIFARE ® cards and transponders without additional active circuitry. The receiver part provides a robust and efficient implementation of a demodulation and decoding circuitry for signals from ISO/IEC 14443A/MIFARE ® compatible cards and transponders. The digital part handles the complete ISO/IEC 14443A framing and error detection (Parity & CRC).The MFRC522 supports MIFARE ® Classic (e.g. MIFARE ® Standard) products. The MFRC522 supports contactless communication using MIFARE ® higher transfer speeds up to 848 kbit/s in both directions.
Various host interfaces are implemented:

- SPI interface
- serial UART (similar to RS232 with voltage levels according pad voltage supply)
- I2C interface.



(Simplified MFRC522 Block Diagram)

UART stands for “Universal Asynchronous Receiver / Transmitter

C)GANTT CHART

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Research														
Designing														
Purchase														
Constructing the circuits														
Programming														
Testing														