VIT UNIVERSITY, CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

March 2018

Electronic Voting Machine using arduino

by

AASTHA SOOD

16BCE1104

MEGHNA LOHANI

16BCE1395

SRUTHI SHIVA

16BCE1064

A project report submitted to

Dr NIRAJ KUMAR

SCHOOL OF ELECTRONICS ENGINEERING (SENSE)

For the course of

CSE2006 –MICROPROCESSOR AND INTERFACING

BONAFIDE CERTIFICATE

This is to certify that the project work entitled "Electronic Voting Machine using arduino" that is being submitted for CAL in B.Tech Microprocessor and Interfacing (CSE2006) is a record of bonafide work done under my supervision. The contents of this Project work have not been submitted for any other CAL course.

Dr NIRAJ KUMAR

SCHOOL OF ELECTRONICS ENGINEERING (SENSE)

ACKNOWLEDGEMENTS

Firstly, I would like to thank Dr Niraj Kumar, professor, Microprocessor and Interfacing (CSE2006), Winter semester 2017-18 for his guidance and which made this PBL project possible. Finally, I would acknowledge VIT University, Chennai Campus for providing me with the opportunity and facilities which ensured the project's completion.

MEGHNA LOHANI 16BCE1395

SRUTHI SHIVA 16BCE1064

AASTHA SOOD 16BCE1104

ABSTRACT

- The highlighting feature of this voting machine using 8051 microcontroller is that the results of the poll can be viewed instantly also can be monitored from time to time over the period of the poll.
- The poll Poll 1 to Poll 8 switches is scanned from the Microcontroller for casting vote to the respective candidates. But at first Poll control switch must be pressed to make the Poll switches available for voting. So every time a vote is casted the control switch must be pressed in order to allow the next candidate to vote. This avoids multiple votes from the pollers. The Poll control switch was monitored by means of external interrupts.
- A LCD was used to display the status of the voting machine i.e whether it is ready to take the next vote or needs to be initialized by the poll control. Also it will display the candidate name once a vote is casted**.**

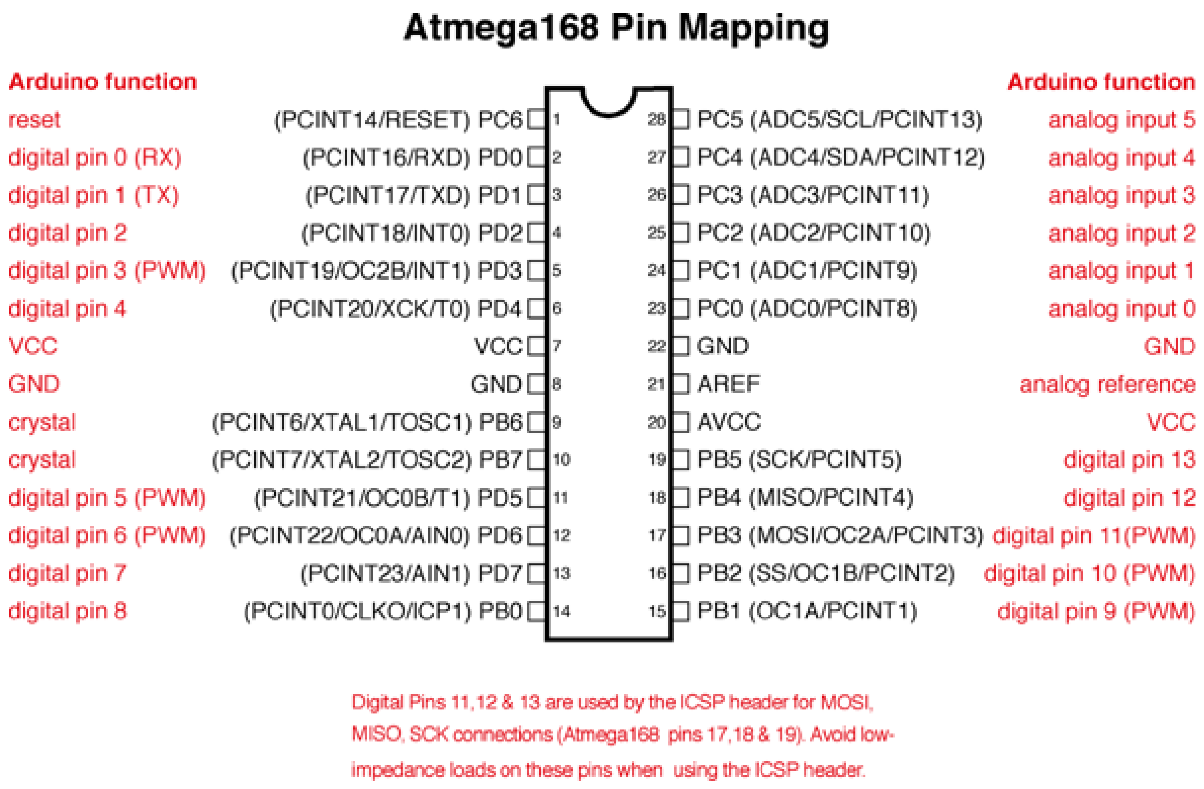
ALGORITHM TO CODE

- Initialize the serial connection in the Arduino.
- Wait for password input and check the input password match with default one once it is entered.
- If input password is correct grant access for the user to give the command input.
- If a particular character is pressed start scanning to detect the vote input.
- Allow only one single input by the voter and again wait for further command.
- If a particular key is pressed send the poll tally to the serial monitor from Arduino.

Arduino Uno

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.



Summary

Microcontroller ATmega328 Operating Voltage 5V Input Voltage (recommended) 7-12V Input Voltage (limits) 6-20V Digital I/O Pins 14 (of which 6 provide PWM output) Analog Input Pins 6 DC Current per I/O Pin 40 mA DC Current for 3.3V Pin 50 mA Flash Memory 32 KB (ATmega328) of which 0.5 KB used by bootloader SRAM 2 KB (ATmega328) EEPROM 1 KB (ATmega328) Clock Speed 16 MHz

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN**. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V**.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3**. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND**. Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms.

Functions

pinMode()

[Digital I/O]

Description

Configures the specified pin to behave either as an input or an output. See the description of (digital pins) for details on the functionality of the pins.

As of Arduino 1.0.1, it is possible to enable the internal pullup resistors with the mode INPUT_PULLUP. Additionally, the INPUT mode explicitly disables the internal pullups.

Syntax

`pinMode(pin, mode)`

Parameters

`pin`: the number of the pin whose mode you wish to set

`mode`: `INPUT`, `OUTPUT`, or `INPUT_PULLUP`. (see the (digital pins) page for a more complete description of the functionality.)

Returns

Nothing

digitalWrite()

[Digital I/O]

Description

Write a `HIGH` or a `LOW` value to a digital pin.

If the pin has been configured as an `OUTPUT` with `pinMode()`, its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for `HIGH`, 0V (ground) for `LOW`.

If the pin is configured as an `INPUT`, `digitalWrite()` will enable (`HIGH`) or disable (`LOW`) the internal pullup on the input pin. It is recommended to set the `pinMode()` to `INPUT_PULLUP` to enable the internal pull-up resistor. See the digital pins tutorial for more information.

If you do not set the `pinMode()` to `OUTPUT`, and connect an LED to a pin, when calling `digitalWrite(HIGH)`, the LED may appear dim. Without explicitly setting `pinMode()`, `digitalWrite()` will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

Syntax

`digitalWrite(pin, value)`

Parameters

`pin`: the pin number

`value`: `HIGH` or `LOW`

Returns

Nothing

digitalRead()

[Digital I/O]

Description

Reads the value from a specified digital pin, either `HIGH` or `LOW`.

Syntax

`digitalRead(pin)`

Parameters

`pin`: the number of the digital pin you want to read

Returns

`HIGH` or `LOW`

In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- **External Interrupts: 2 and 3**. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the **analogReference()** function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the Wire library.

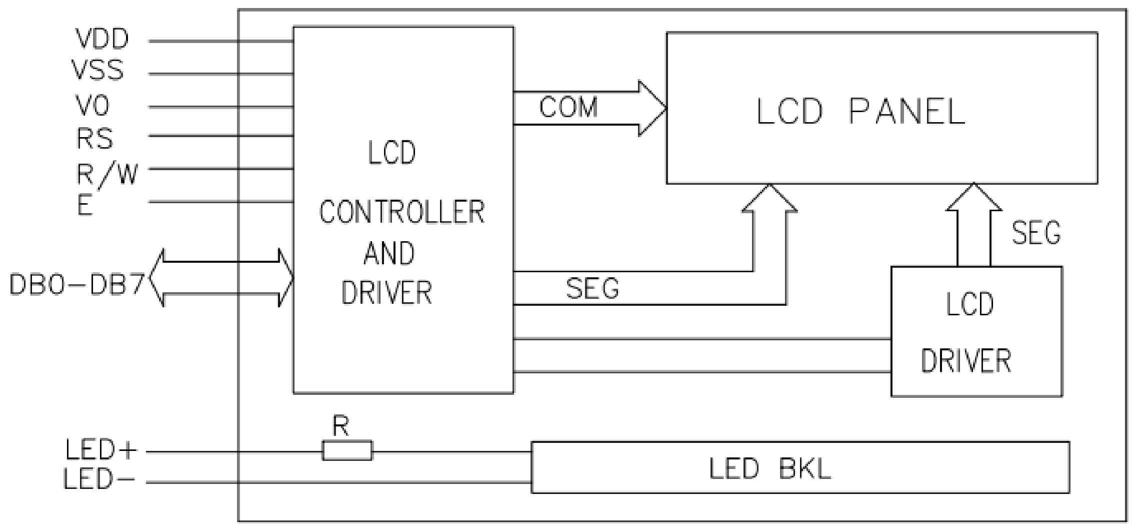
Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials. The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by: • On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. • On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

LCD

5. Block diagram

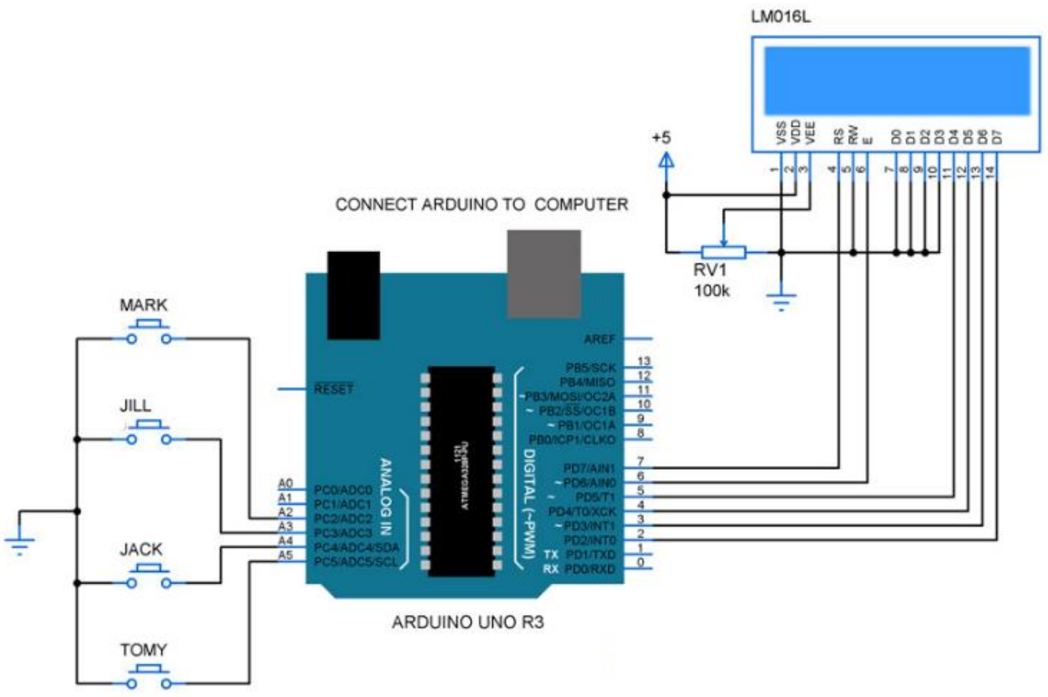


6. Interface pin description

Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL

Circuit

IGN:



Code

```
#include <LiquidCrystal.h>
int votes[4]={0,0,0,0};
char inbyte;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //LCD connections
  String pwd="VOTE";                    //Default Password
  String inpt="";
  boolean flag=false;
  boolean securitygranted=false;
  int i;
void setup() {
   lcd.setCursor(0, 1);
  pinMode(A2, INPUT_PULLUP);          //Setting pins as input
  pinMode(A3, INPUT_PULLUP);
  pinMode(A4, INPUT_PULLUP);
  pinMode(A5, INPUT_PULLUP);
  lcd.begin(16, 2);
  lcd.display();

  Serial.begin(9600);                 //Begin serial communication
  Serial.println("ENTER PASSWORD");
}
void loop() {
   lcd.setCursor(0, 1);
  while(flag==true)                   //Check flag for "V" command
```

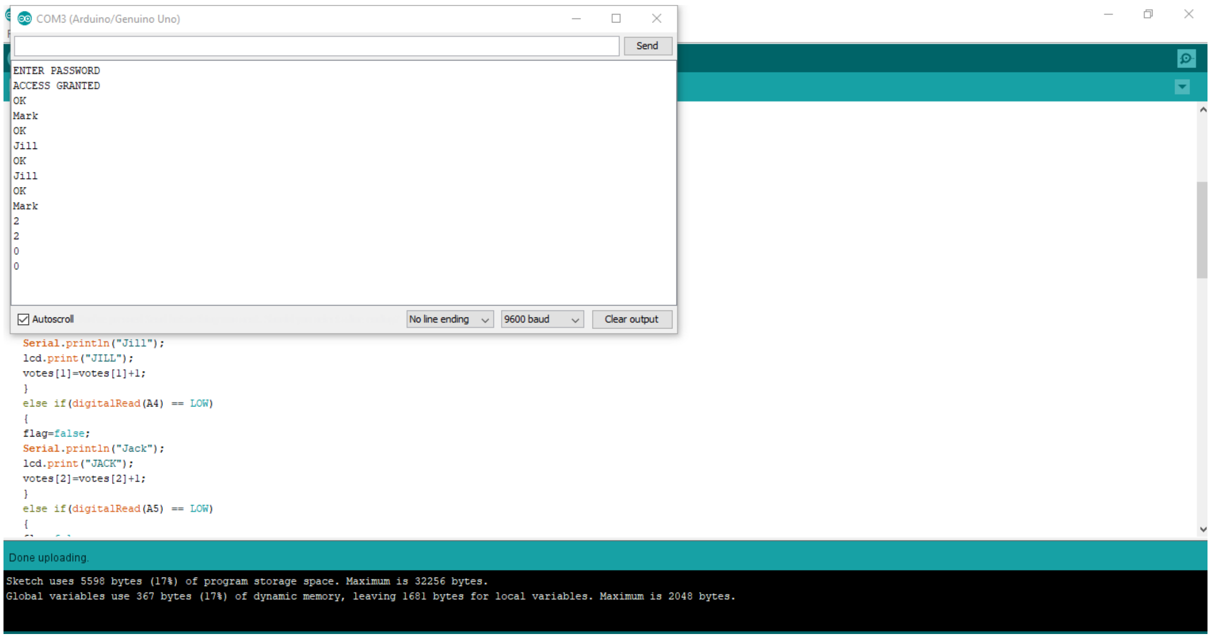
```
 {
 if(digitalRead(A2) == LOW)
 {
 flag=false;
 Serial.println("Mark");
 lcd.print("MARK");              //Example candidate name
 votes[0]=votes[0]+1;
 }
 else if(digitalRead(A3) == LOW)
 {
 flag=false;
 Serial.println("Jill");
 lcd.print("JILL");
 votes[1]=votes[1]+1;
 }
 else if(digitalRead(A4) == LOW)
 {
 flag=false;
 Serial.println("Jack");
 lcd.print("JACK");
 votes[2]=votes[2]+1;
 }
 else if(digitalRead(A5) == LOW)
 {
 flag=false;
 Serial.println("Tomy");
 lcd.print("TOMY");
 votes[3]=votes[3]+1;
 }
 }
}
void serialEvent()
{
 while(Serial.available())
 {
  inbyte=(char)Serial.read();               //Reading incoming character
  if(securitygranted==false)
  {
   inpt += inbyte;
   inbyte='\n';
  }
 }
 commandcheck();
}
void commandcheck()
{
 switch (securitygranted)              //See for the security permissions
```

```
{
 case false: {
  if(inpt==pwd)                    //Checking for password match
   {
    securitygranted=true;
    inbyte='\n';
    inpt="";
    Serial.println("ACCESS GRANTED");
   /* char TestData;
    if(Serial.available())
    {
     TestData=Serial.read();
     lcd.print(TestData);
    }
    lcd.print("hello, world!");*/
   }
  else if((inpt!=pwd)&&(inpt.length()>3))      //Mismatch
  {
    Serial.println(inpt);
    inpt="";
    inbyte='\n';
    Serial.println("WRONG PASSWORD");
    Serial.println("ENTER PASSWORD");
    Serial.end();
    delay(100);
    Serial.begin(9600);
  }
  break;
       }

 case true: {
  if(inbyte=='V')
   {
    flag=true;                     //Allowing voter to cast a single vote
    Serial.println("OK");
    lcd.clear();
    inbyte='\n';
   }
  else if(inbyte=='D')
   {
    for(i=0;i<=3;i++)                //Displaying vote tally
    Serial.println(votes[i]);
    inbyte='\n';
   }
  else
   {Serial.println("UNKNOWN COMMAND");}
   break;
```

```
        }
  }
}
```

SCREENSHOTS



```
ENTER PASSWORD
ACCESS GRANTED
OK
Mark
OK
Jill
OK
Jill
OK
Mark
2
2
0
0
```

No line ending
9600 baud

```
  Serial.println("Jill");
  lcd.print("JILL");
  votes[1]=votes[1]+1;
  }
  else if(digitalRead(A4) == LOW)
  {
  flag=false;
  Serial.println("Jack");
  lcd.print("JACK");
  votes[2]=votes[2]+1;
  }
  else if(digitalRead(A5) == LOW)
  {
```

Done uploading.

```
Sketch uses 5598 bytes (17%) of program storage space. Maximum is 32256 bytes.
Global variables use 367 bytes (17%) of dynamic memory, leaving 1681 bytes for local variables. Maximum is 2048 bytes.
```