Misr University for Science & Technology
Engineering

Metro ticket machine

Students:

Khaled Abdelnasser 80768

Mohamed Ahmed 79573

Omar Ashraf 80601

Assem Yasser 80681

Ahmed Ayman 80836

MECHATRONICS II

MTE504

Supervised by Dr. Mohammed ibrahim

Abstract:

this report explain the automation of metro ticketing system by using the RFID reader and cards so we can identify the card is valid or invalid and indicating it using green led and buzzer for valid cards and red led for invalid cards and servo motor to simulate the entrance of the person through the gate and lcd to indicate the purpose of the gate whether entrance or exit and we use seven segment module to indicate the number of the people in the station that is going to increase when someone go in an decrease when someone go out of the station and we used the SPI communication protocol in the communication between the Arduino mega2560 which is the controller of the system and the RFID reader and Arduino IDE to write the code for the system and upload it to the Arduino from the computer .and we protues to draw the circuit of the system .

Literature review:

1-RFID Based Metro Train Ticketing System:

automated system for ticketing in the Metro Train System which is based on unique passenger identification. This is a user-friendly system, which will automatically identify the passenger and deduct the passenger's fare according to the distance travelled. The Radio Frequency Identification (RFID) card and rfid reader are used to make the identification of passenger and transaction very precise. System thus reduces human errors and efforts. This project is a complete metro train simulator that will be installed in a metro train. The RF Card technology is used that is it has a unique id for every tag. Every person is allotted an RF card with the fixed balance in it. The metro train will keep on passing between the stations and as it stops at the station the person has to swipe its RF card to make an exit. According to distance travelled by the passenger the ticket is deducted from its balance, as the person has to show the RF card while entering and leaving the train station. The MCU is used to perform the above task along with the display unit and RF ID reader. A stepper motor is used to show the door opening and closing on swiping the RF card.

1.1-PROJECT DESCRIPTION:

This project is designed to demonstrate the RFID based ticketing system used in Metro train system. In this technology we make use of an RFID tags and an EM-18 RFID reader which is installed on the metro train. Each RFID tag has its unique identity which provides security to its user. According to the distance travelled by the passenger the ticket is deducted from the balance, as the person has to show the RF card while entering and leaving the train station. Multiple users can enter the train or leave the train simultaneously. The MCU stores the data of each card user and deducts the balance at the destination station based on the distance travelled. A stepper motor is used to open and close the gate each time the card is tapped on the RFID reader. RFID cards can be used in all weather conditions and can be read at a wide range of angles. The RFID does not require visual line of site. It allows transparent automated tracking eliminating human error. New information can be overwritten on the RFID card.

1.2. Block diagram:

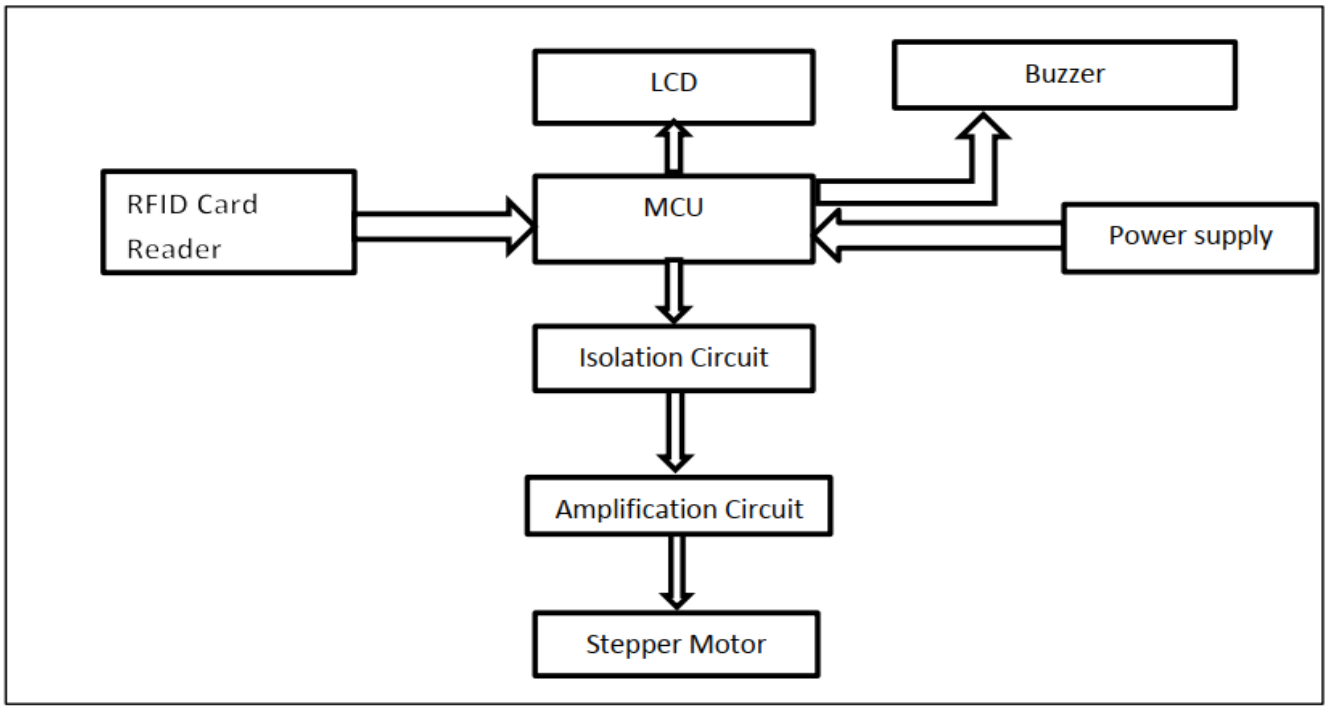


Figure 1

1.3-THE HARDWARE IMPLEMENTATION:

In RFID based metro train ticketing system we use AT89C52 microcontroller which controls the operation of the lcd display, the L293D motor driver IC and RFID reader module. Supply voltage for the IC which is of 5V is given through step down transformer which converts 240V AC supply to 5V which is then passed to rectifiers We make use of a 16*2 lcd display which indicates the station, entry and exit of the passengers, and the amount of money that will be deducted for the journey. The L293D IC controls the operation of the door motor which opens and closes the door during the entry and exit of the passengers. We make use of EM-18 RFID reader module which reads the passenger tags and gives input to the microcontroller which forwards the passenger information to the display. Whenever a card is placed nearby to the reader a buzzer sound indicates successful reading of that card.

1.4-RESULT AND DISCUSSION:

This project helps in Restricting unauthorised travelling of passengers in the metro trains. RFID cards are immune to any kind of damage due to weather conditions. This project also helps in Cashless transaction which is the trend to be followed soon. No queuing is required i.e. to wait in lines for tickets before boarding a train. Passengers can get off at any station as required, no predefined stations and money will get deducted accordingly. The system is time efficient since it helps in saving

time for booking tickets. The RFID system creates a system which is capable of mass identification process, precise location data

recorder and easier and faster contactless payment.

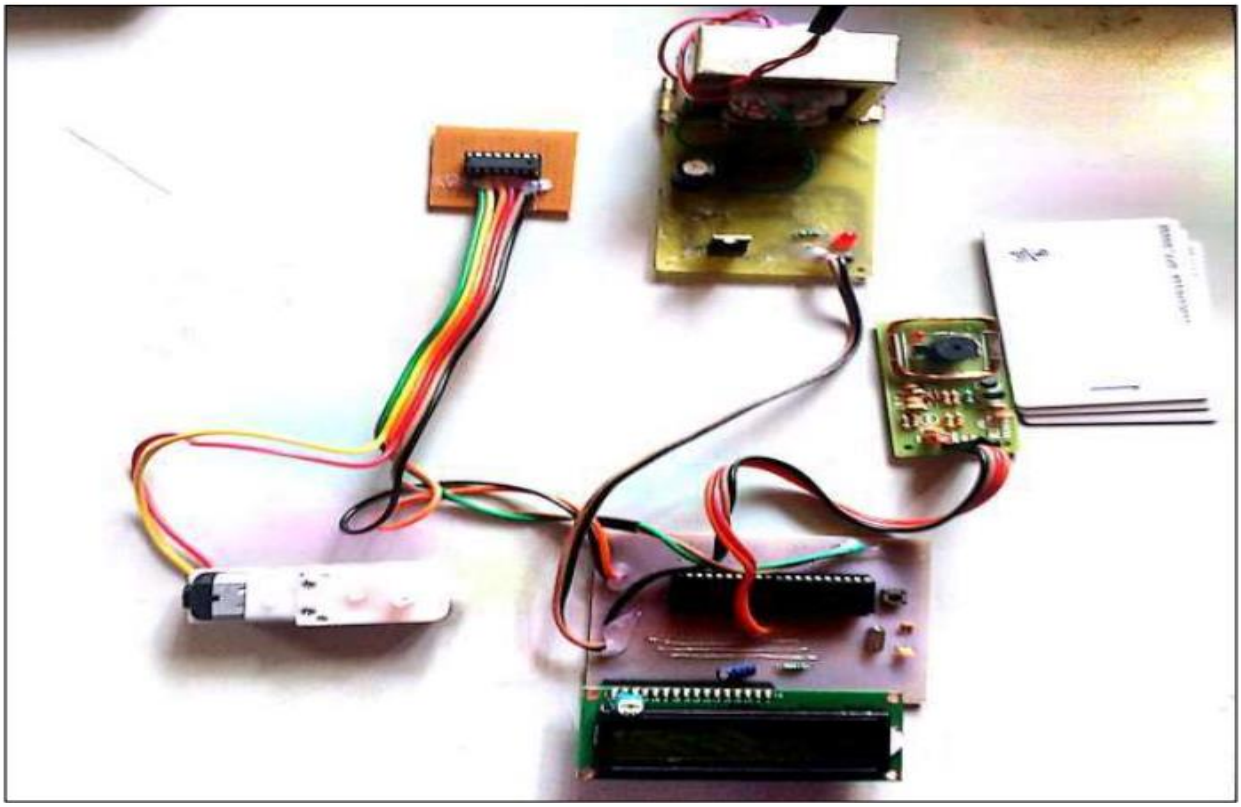


Figure 2

Introduction:

RFID Based Smart Ticketing System for Metro:

The Public transport system is a major source of income in developing countries like Egypt. So, Government Earns lot of revenue from the ticketing But, this public transport system faces several problems. The conductor will face various problems in issuing the tickets. But this new system will provide the tickets automatically and deduct the fare for the distance travelled from the passenger's account. It is also used for passenger identification. RFID has been an emerging technology in recent years. RFID technology can be effectively employed in a number of applications due to its penchant for efficiency. As for its application, it's been a widespread tool for both tracking transit transports. A fundamental system of RFID consists of two primary components: The reader circuit and tag, details of which are discussed later. The main idea behind this project is to collect the fare automatically using the RFID technology and GSM modem. RFID TAG holds information about passengers, the information is about the passenger is contact number and name. RFID READER used to read a RFID TAG and with the help of MICROCONTROLLER programs are declared in secure manner to get a ticket. Tickets contain information about the person.

The idea of the project:

1- We have an entrance gate and exit gate with lcd display the entrance with arrows to indicate the direction of the entrance so the person will have a big indicator to know where the exit or the entrance is.
2- The gate has two LEDs, green led to indicate that the card is valid and goes through the gate and red led to indicate that the card is invalid and can't go through.
3- We have rfid reader to read the cards and identify the passenger.
4- We have seven segments to indicate the number of the people in the station
5- We have Arduino mega 2560 to control the system.

The components of the system:

1-Arduino mega 2560:

The controller of the system. Arduino Mega is based on ATmega2560 Microcontroller, an 8-bit AVR Architecture based MCU from ATMEL.

Arduino Mega Board Layout:

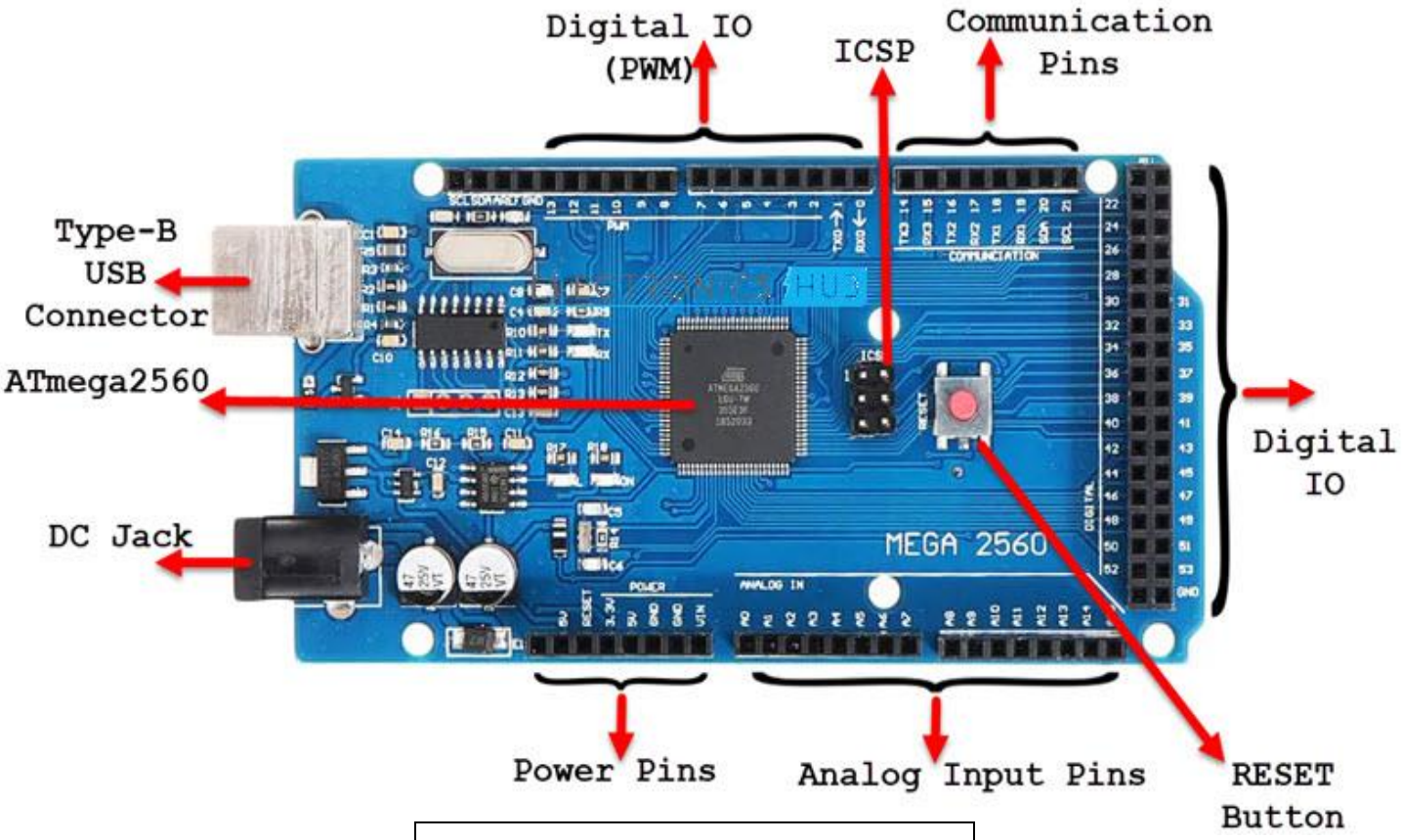


Figure 3

Technical Specifications of Arduino Mega:

MCU	ATmega2560
Architecture	AVR

Operating Voltage	5V
Input Voltage	6V – 20V (limit) 7V – 12V (recommended)
Clock Speed	16 MHz
Flash Memory	256 KB (8 KB of this used by bootloader)
SRAM	8 KB
EEPROM	4 KB
Digital IO Pins	54 (of which 15 can produce PWM)
Analog Input Pins	16

here are three different memories available in ATmega2560. They are:

256 KB of Flash Memory

8 KB of SRAM

4 KB of EEPROM

8 KB of Flash Memory is used by the bootloader code.

Arduino Mega supports three different types of communication interfaces. They are:

Serial ,I2C and SPI

Arduino Mega Pinout:

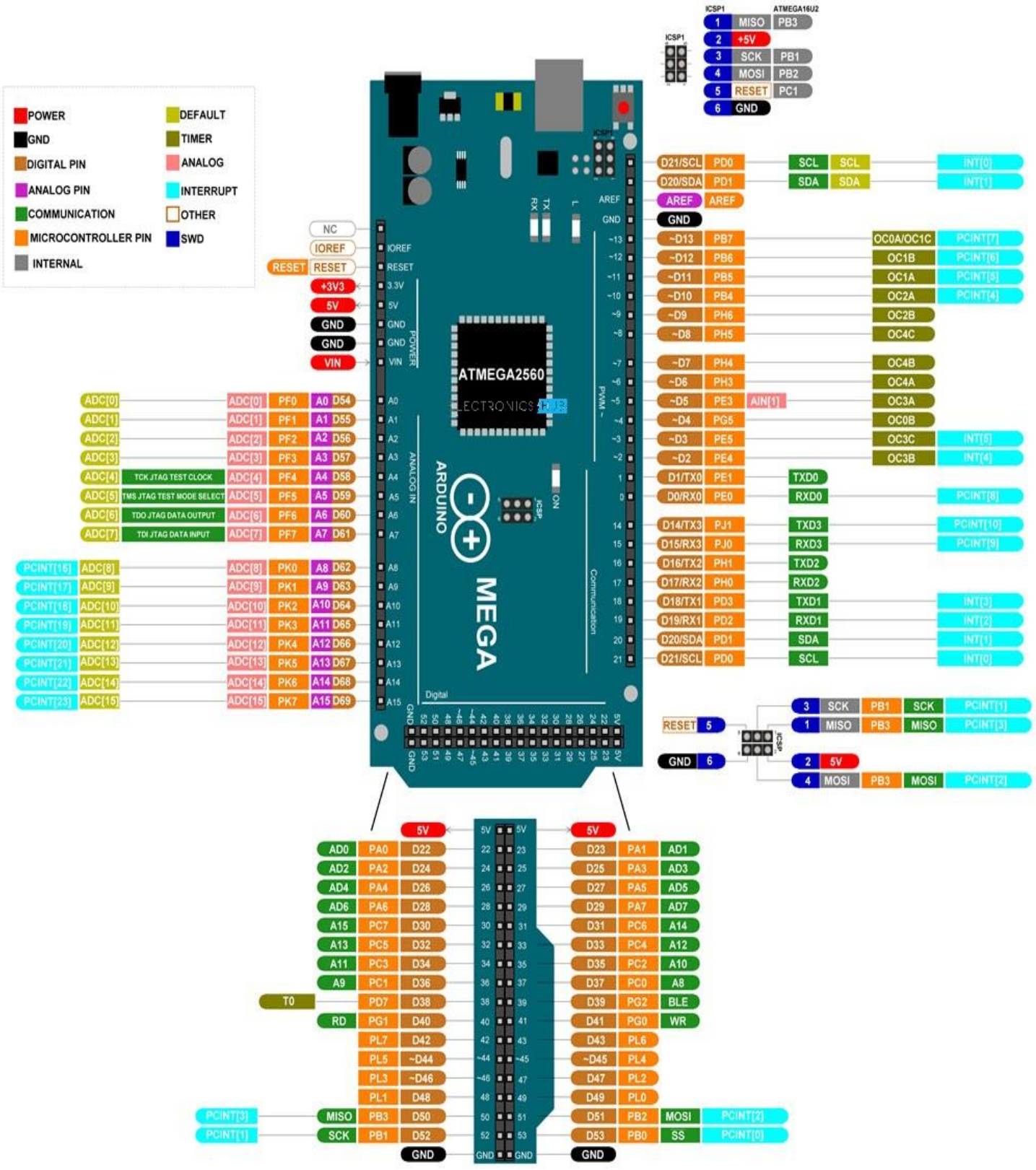


Figure 4

2-LCD:

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments.

The 16×2 LCD pinout is:

- Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
- Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
- Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
- Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
- Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
- Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
- Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
- Pin15 (+ve pin of the LED): This pin is connected to +5V
- Pin 16 (-ve pin of the LED): This pin is connected to GND.

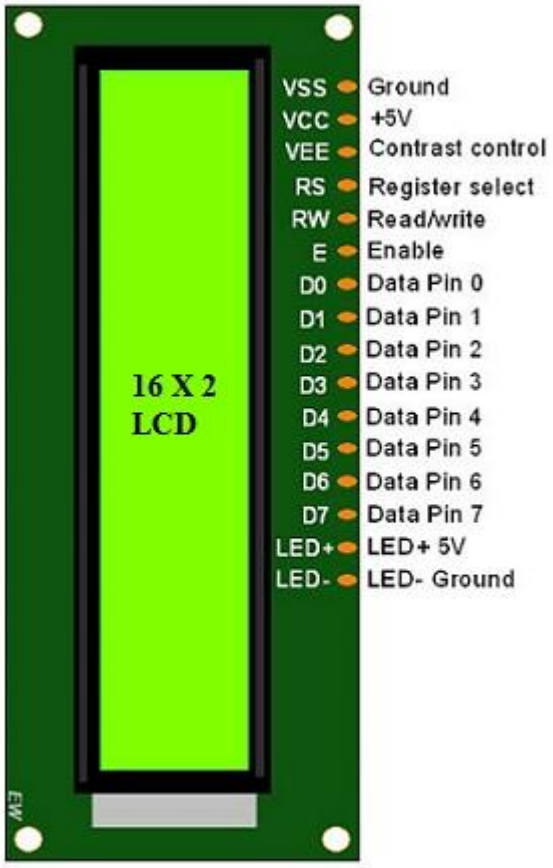


Figure5: LCD-16×2-pin diagram

Features of LCD16x2

The features of this LCD mainly include the following.:

1- The operating voltage of this LCD is 4.7V-5.3V
2- It includes two rows where each row can produce 16 characters.
3- The utilization of current is 1mA with no backlight
4- Every character can be built with a 5×8-pixel box
5- The alphanumeric LCDs alphabets & numbers
6- Is display can work on two modes like 4-bit & 8-bit

3-RFID(RC522):

An RFID or radio frequency identification system consists of two main components, a tag attached to the object to be identified, and a reader that reads the tag.

A reader consists of a radio frequency module and an antenna that generates a high frequency electromagnetic field. Whereas the tag is usually a passive device (it does not have a battery). It consists of a microchip that stores and processes information, and an antenna for receiving and transmitting a signal.

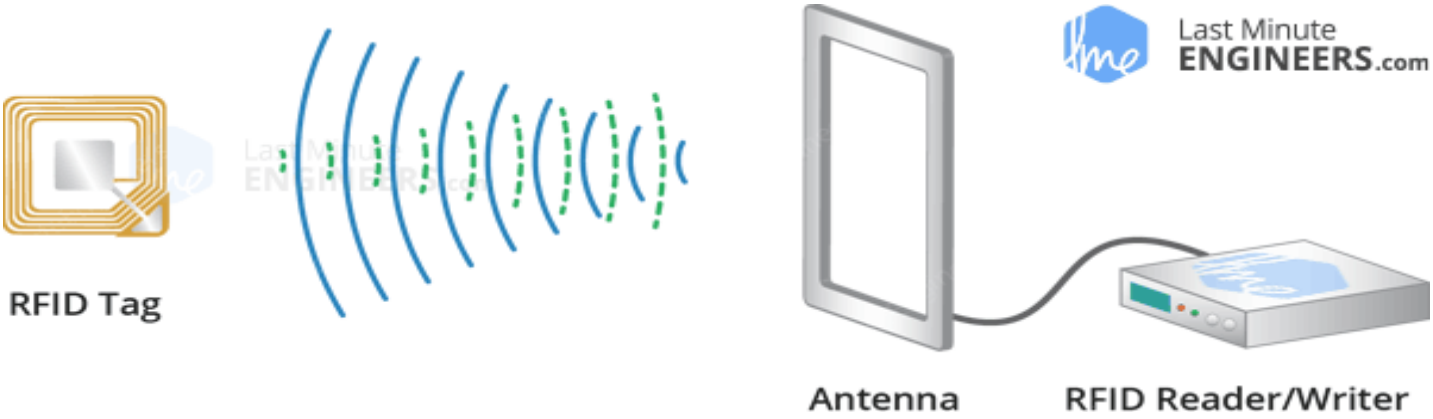


Figure 6

When the tag is brought close to the reader, the reader generates an electromagnetic field. This causes electrons to move through the tag's antenna and subsequently powers the chip.

The chip then responds by sending its stored information back to the reader in the form of another radio signal. This is called a backscatter. The reader detects and interprets this backscatter and sends the data to a computer or microcontroller.

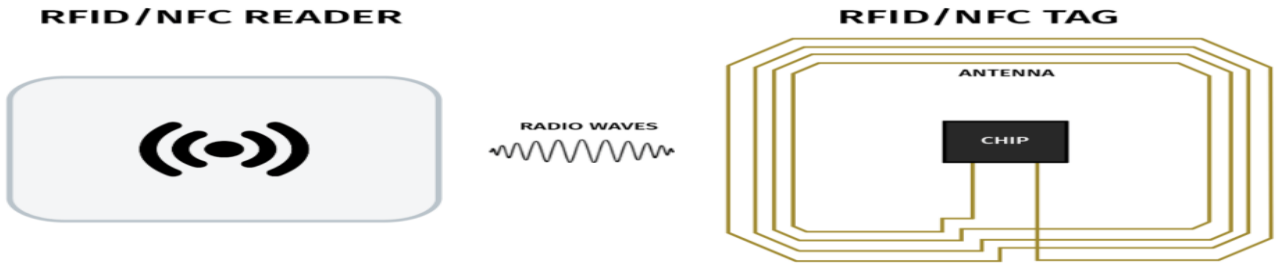


Figure 7

Hardware Overview:

The RC522 RFID module based on the MFRC522 IC from NXP is one of the cheapest RFID options you can get online for less than four dollars. It usually comes with an RFID card tag and a key fob tag with 1KB of memory. And the best part is that it can write a tag that means you can store any message in it.

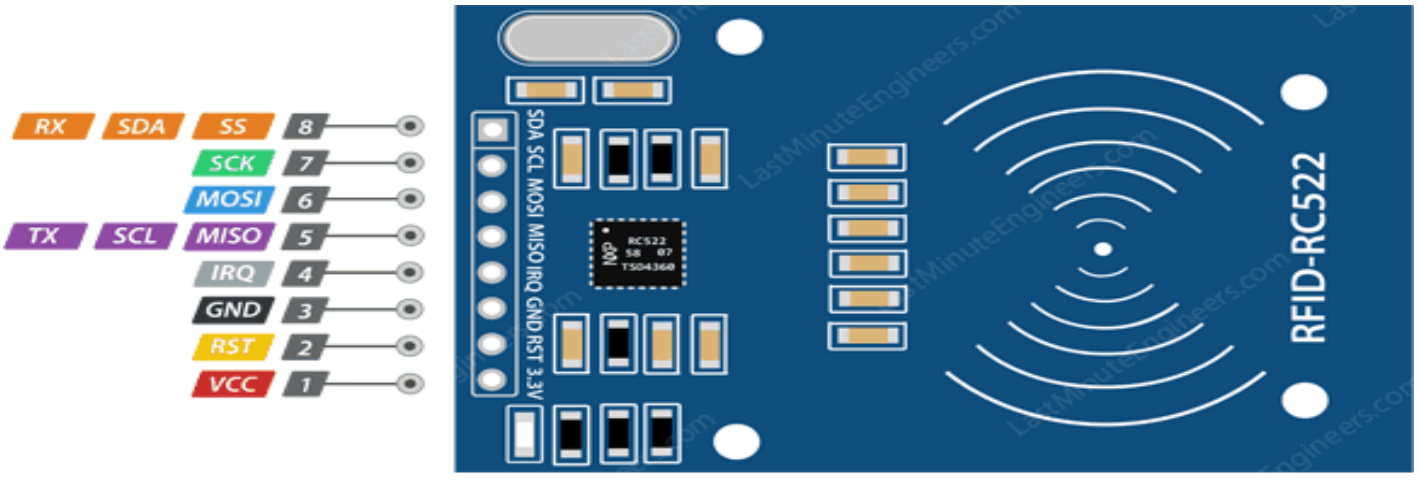
Technical Specifications:

Here are the specifications:

Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26mA
Min. Current(Power down)	10µA
Logic Inputs	5V Tolerant
Read Range	**5 cm**

RC522 RFID Module Pinout:

The RC522 module has a total of 8 pins that connect it to the outside world. The connections are as follows:



VCC: supplies power to the module. This can be anywhere from 2.5 to 3.3 volts. You can connect it to the 3.3V output from your Arduino. But remember that connecting it to the 5V pin will probably destroy your module!

RST: is an input for reset and power-down. When this pin goes low the module enters power-down mode. In which the oscillator is turned off and the input pins are disconnected from the outside world. Whereas the module is reset on the rising edge of the signal.

GND: is the ground pin and needs to be connected to the GND pin on the Arduino.

IRQ: is an interrupt pin that alerts the microcontroller when an RFID tag is in the vicinity.

MISO / SCL / Tx: pin acts as master-in-slave-out when SPI interface is enabled, as serial clock when I2C interface is enabled and as serial data output when the UART interface is enabled.

MOSI (Master Out Slave In): is the SPI input to the RC522 module.

SCK (Serial Clock): accepts the clock pulses provided by the SPI bus master i.e. Arduino.

SS / SDA / Rx: pin acts as a signal input when the SPI interface is enabled, as serial data when the I2C interface is enabled and as a serial data input when the UART interface is enabled. This pin is usually marked by encasing the pin in a square so that it can be used as a reference to identify other pins.

MIFARE Classic 1K Memory Layout

The tag's 1K memory is organized into 16 sectors (from 0 to 15). Each sector is further divided into 4 blocks (blocks 0 to 3). And each block can store 16 bytes of data (from 0 to 15).

This of course tells us that we have:

16 sectors x 4 blocks x 16 bytes of data = 1024 bytes = 1K memory

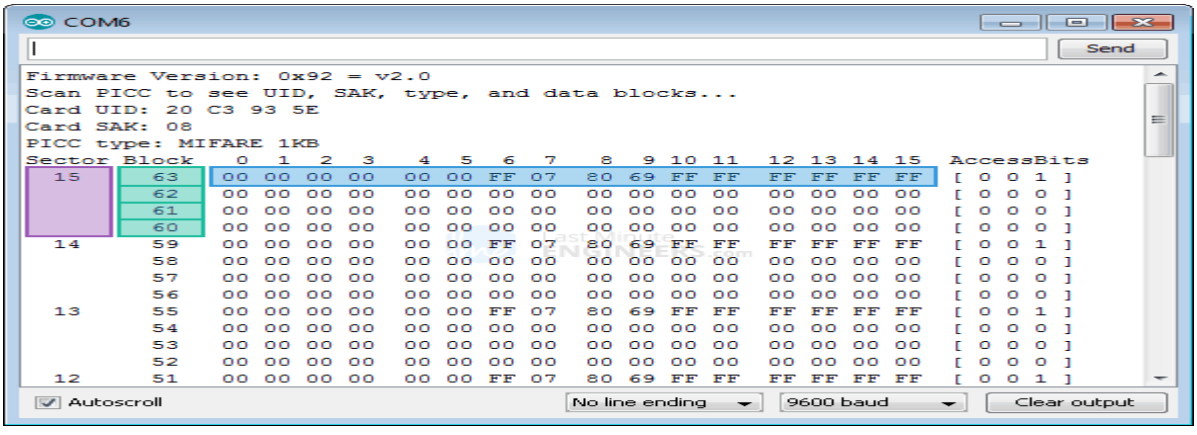


Figure 8: The entire 1K of memory, along with sectors, blocks, and Data.

Here is the 3D representation of the MIFARE Classic 1K memory map layout.

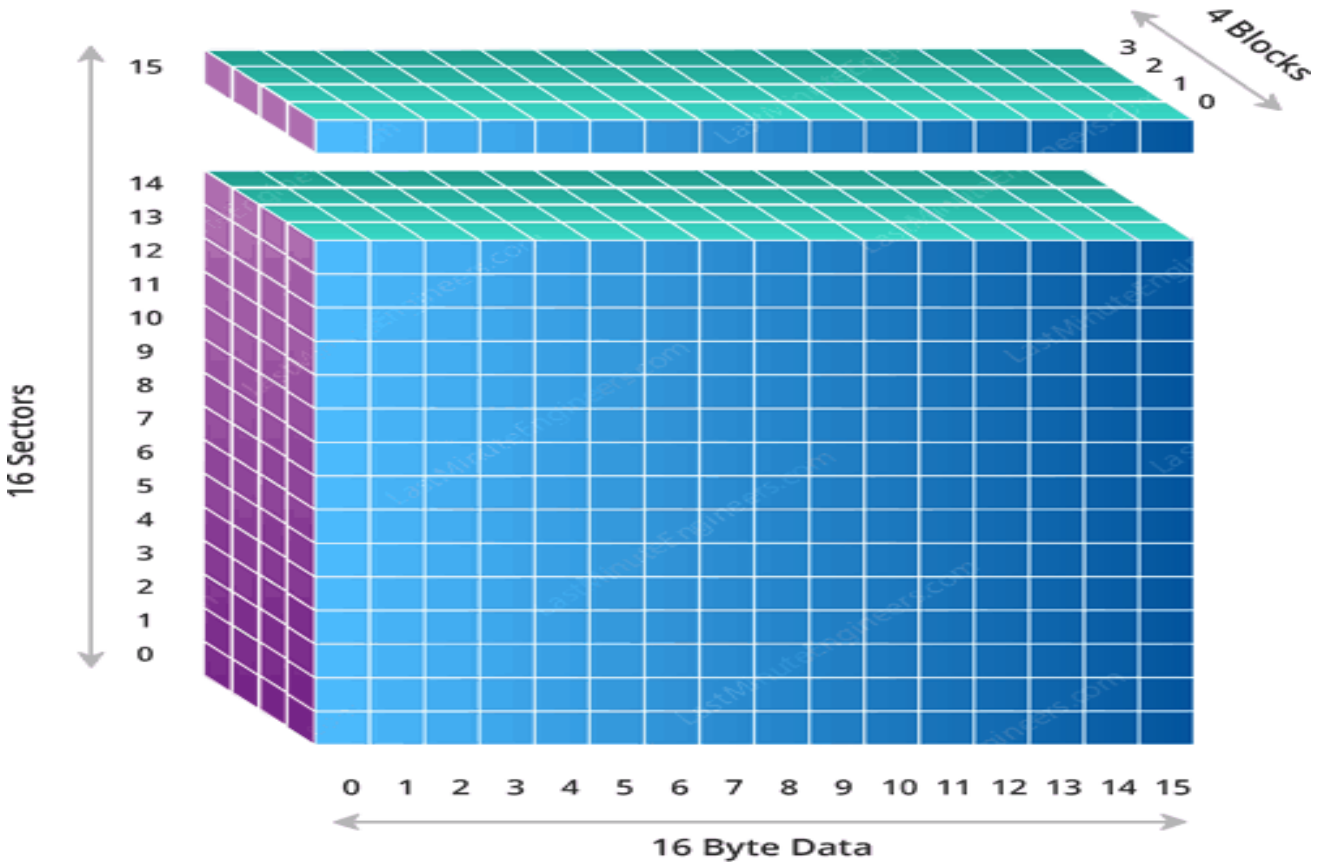


Figure 9: 3D Representation of MIFARE Classic

The last block of each sector is called a Sector Trailer. It contains information called Access Bits that provide read and write access to the remaining blocks in the sector. This means that only 3 blocks of each sector (Blocks #0, #1 and #2) are actually writable, in other words only 48 bytes per sector are available for use. Also Block #0 of Sector #0 is called Manufacturer Block which contains IC Manufacturer data and Unique Identifier (UID). The manufacturer block is highlighted in red.

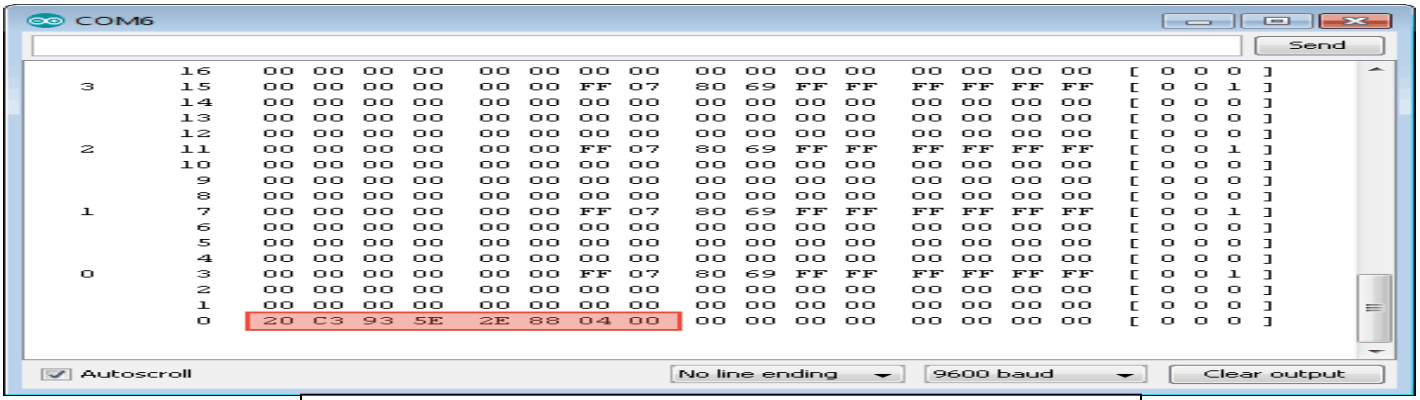


Figure 10: IC Manufacturer data and Unique Identifier (UID)

4-BCD to Seven Segment Decoder:

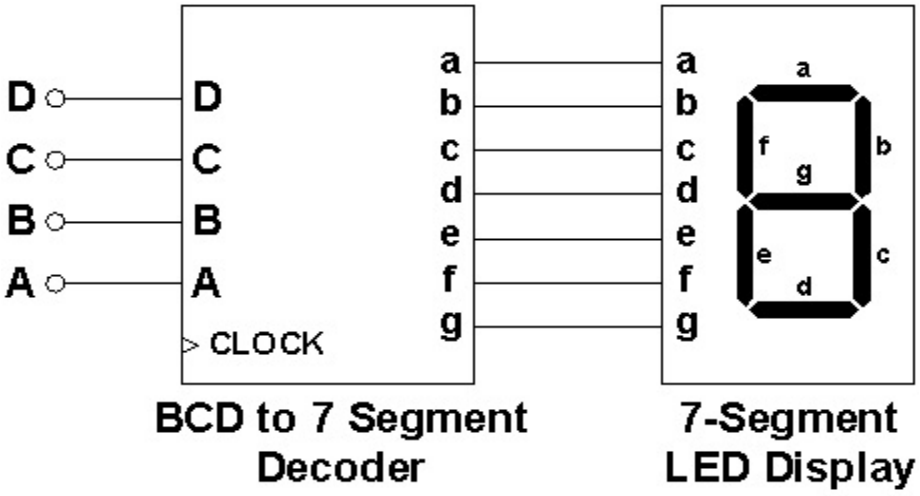


Figure 11

For the display to work, these segments are to be driven by the certain logic level at their input. Depending on this, seven segment displays are found to be of two types viz., common cathode type and common anode type.

A common cathode display has all the cathode terminals of its LED segments tied together (green line). Further, this is grounded and hence is considered to be at logic 0 state. This means that in order to light up an LED, one needs to drive it high. On the other hand, a common anode display shown by Figure 2b has all its anode terminals connected together which is further driven high by connecting it to a positive supply voltage (green line). Hence for this kind of display to work, one has to drive low on the cathode terminals of the individual LED segments.

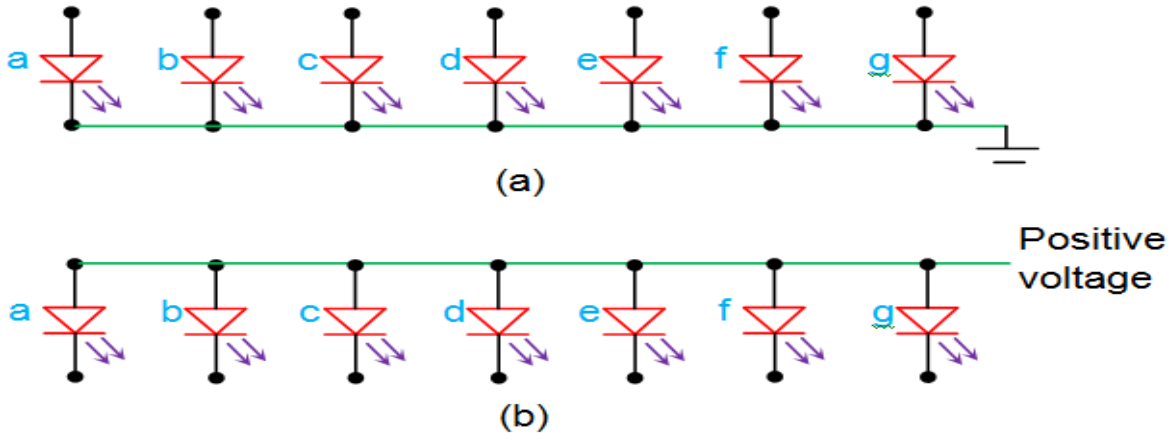


Figure 12

BCD to seven segment decoder is a circuit used to convert the input BCD into a form suitable for the display. It has four input lines (A, B, C and D) and 7 output lines (a, b, c, d, e, f and g) as shown in Figure 3. Considering common cathode type of arrangement, be given as in Table I.

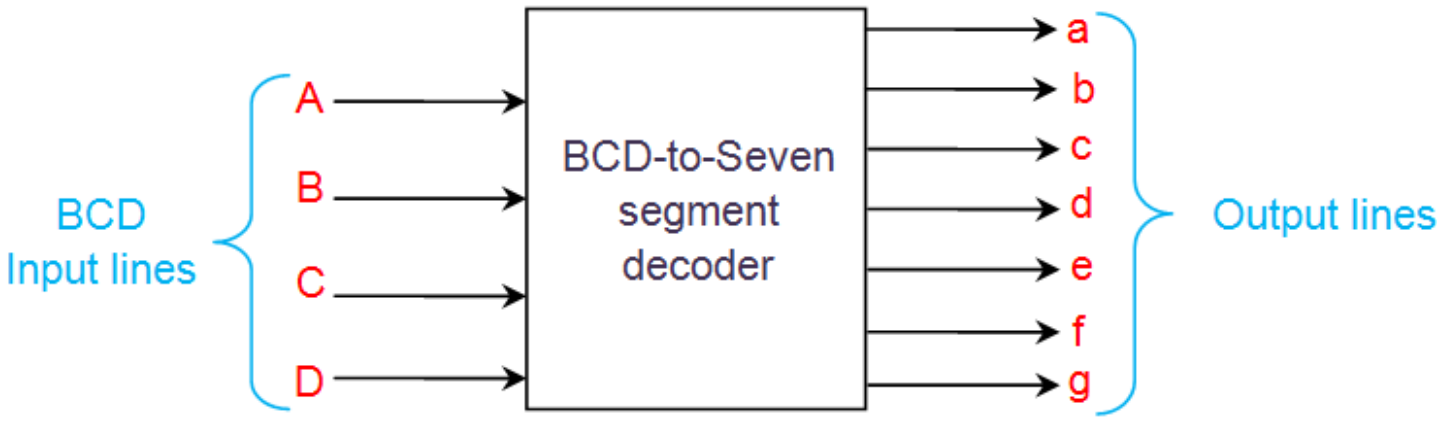


Figure 13

Table I Truth table for common cathode type BCD to seven segment decoders:

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

Figure 14

This table indicates the segments which are to be driven high to obtain certain decimal digit at the output of the seven segment display.

5-servo motor:

A servo motor is a closed-loop system that uses position feedback to control its motion and final position. There are many types of servo motors, and their main feature is the ability to precisely control the position of their shaft.

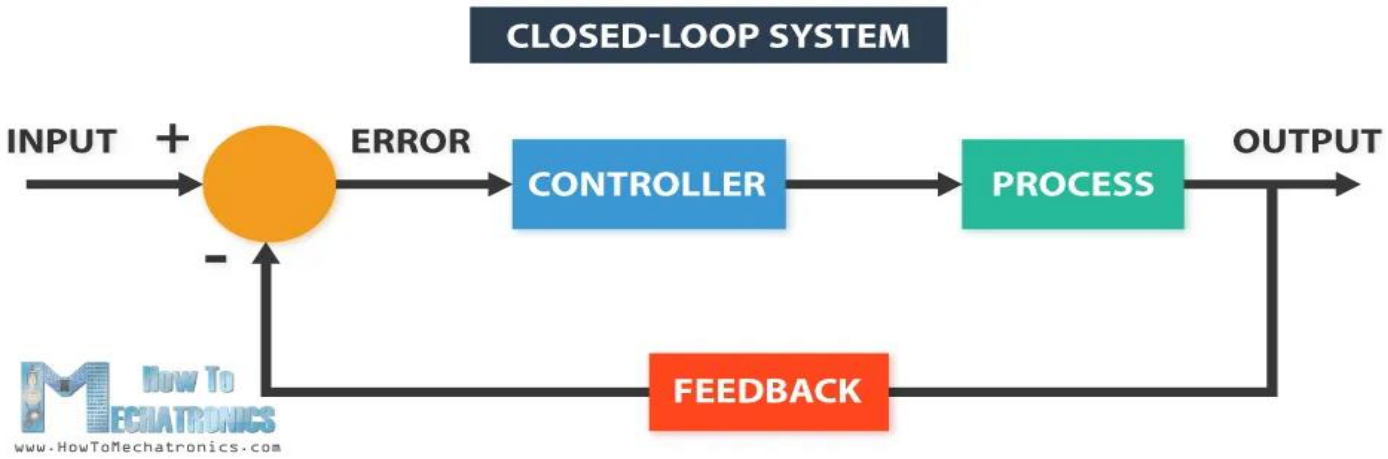


Figure 15

In industrial type servo motors the position feedback sensor is usually a high precision encoder, while in the smaller RC or hobby servos the position sensor is usually a simple potentiometer. The actual position captured by these devices is fed back to the error detector where it is compared to the target position. Then according to the error, the controller corrects the actual position of the motor to match with the target position.

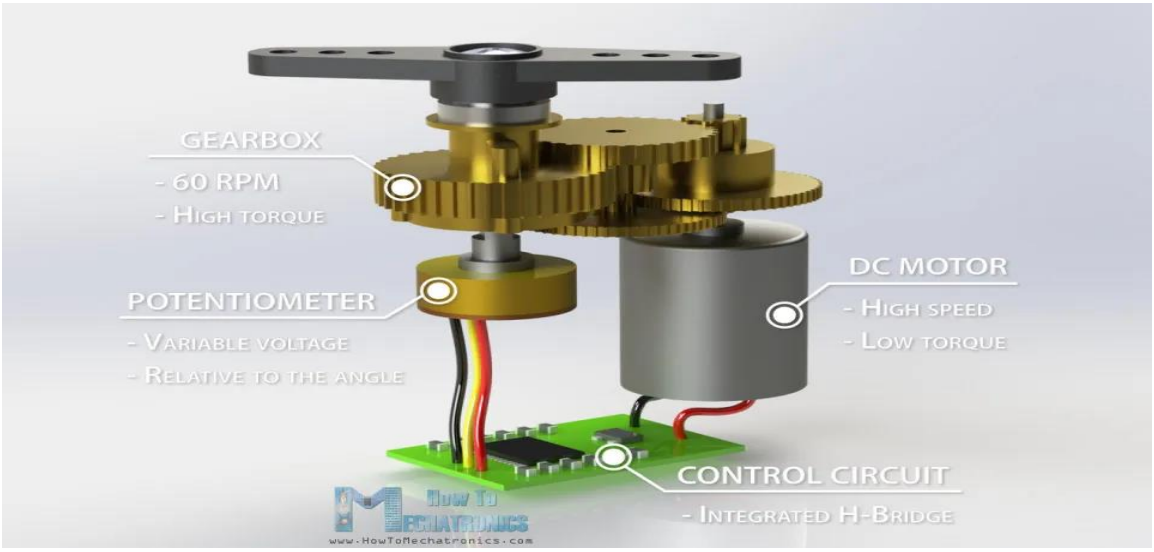


Figure 16

The servo motor working principle:

There are four main components inside of a hobby servo, a DC motor, a gearbox, a potentiometer, and a control circuit. The DC motor is high speed and low torque, but the gearbox reduces the speed to around 60 RPM and at the same time increases the torque. The potentiometer is attached to the final gear or the output shaft, so as the motor rotates the potentiometer rotates as well, thus producing a voltage that is related to the absolute angle of the output shaft. In the control circuit, this potentiometer voltage is compared to the voltage coming from the signal line. If needed, the controller activates an integrated H-Bridge which enables the motor to rotate in either direction until the two signals reach a difference of zero.

A servo motor is controlled by sending a series of pulses through the signal line. The frequency of the control signal should be 50Hz or a pulse should occur every 20ms. The width of pulse determines the angular position of the servo and these types of servos can usually rotate 180 degrees (they have physical limits of travel).

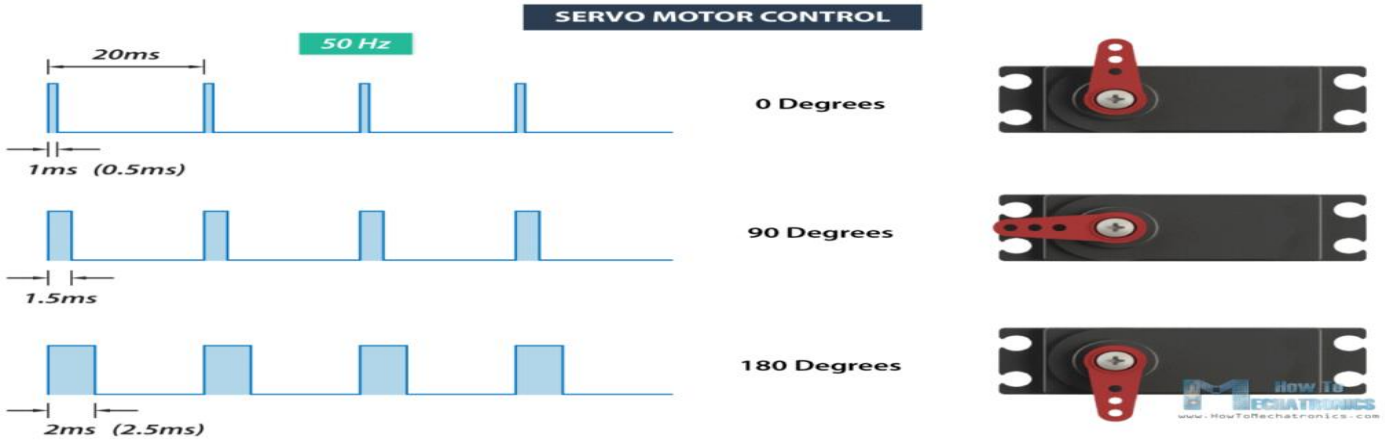


Figure 17

Generally, pulses with 1ms duration correspond to 0 degrees position, 1.5ms duration to 90 degrees and 2ms to 180 degrees. Though the minimum and maximum duration of the pulses can sometimes vary with different brands, and they can be 0.5ms for 0 degrees and 2.5ms for 180 degrees position.

SG90 Micro Servo technical specifications:

Stall Torque	1.2kg·cm @4.8V, 1.6kg·cm @6V,
Operating Voltage	3.5 – 6V
No Load Current	100mA
Stall Current	650mA
Max Speed	60 degrees in 0.12s
Weight	9g

Block diagram for the project:

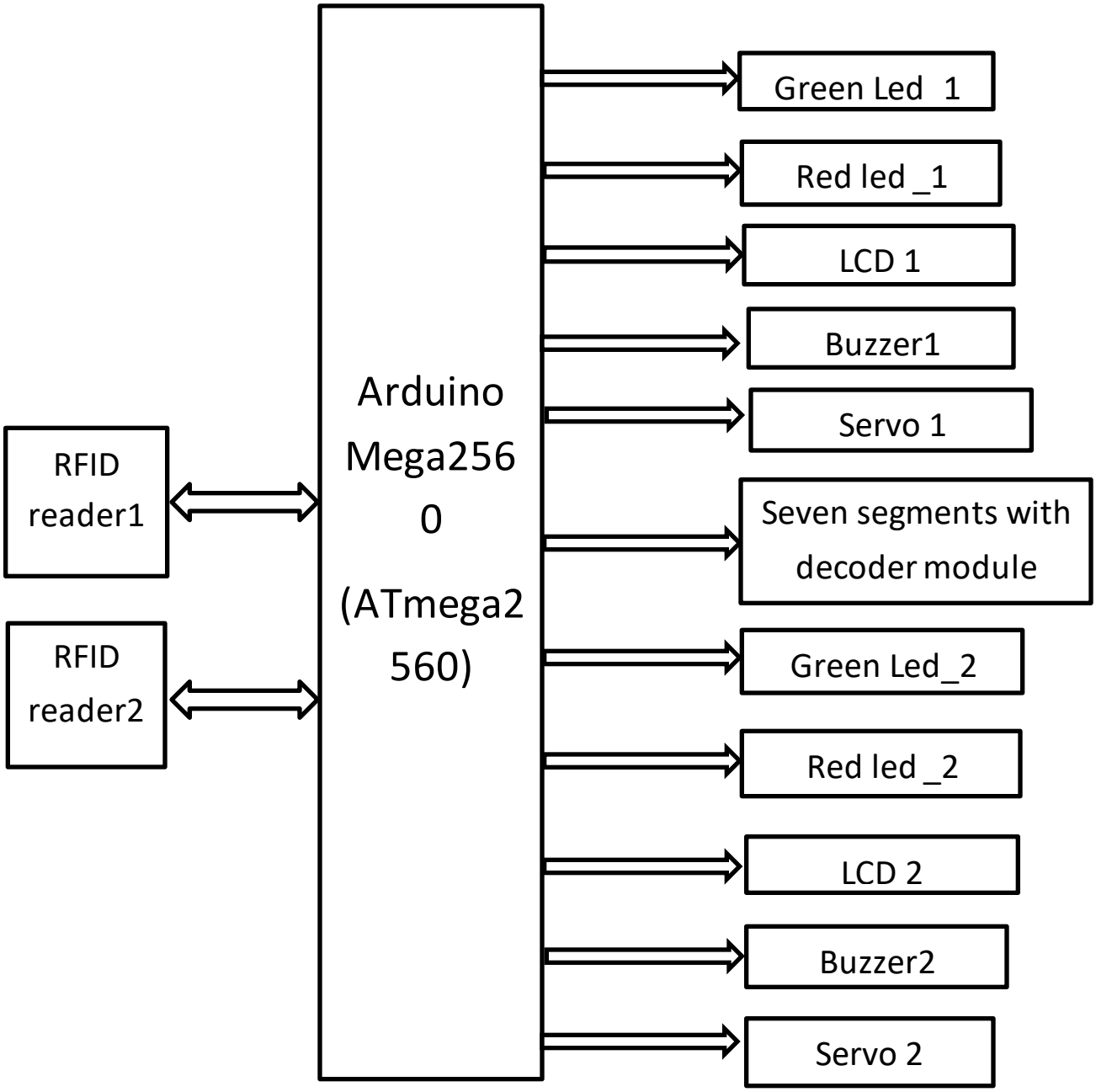


Figure 18

SPI Communication:

SPI is a common communication protocol used by many different devices. For example, SD card reader modules, RFID card reader modules, and 2.4 GHz wireless transmitter/receivers all use SPI to communicate with microcontrollers.

One unique benefit of SPI is the fact that data can be transferred without interruption.

Any number of bits can be sent or received in a continuous stream. With I2C and UART, data is sent in packets, limited to a specific number of bits. Start and stop conditions define the beginning and end of each packet, so the data is interrupted during transmission.

Devices communicating via SPI are in a master-slave relationship. The master is the controlling device (usually a microcontroller), while the slave (usually a sensor, display, or memory chip) takes instructions from the master. The simplest configuration of SPI is a single master, single slave system, but one master can control more than one slave (more on this below).

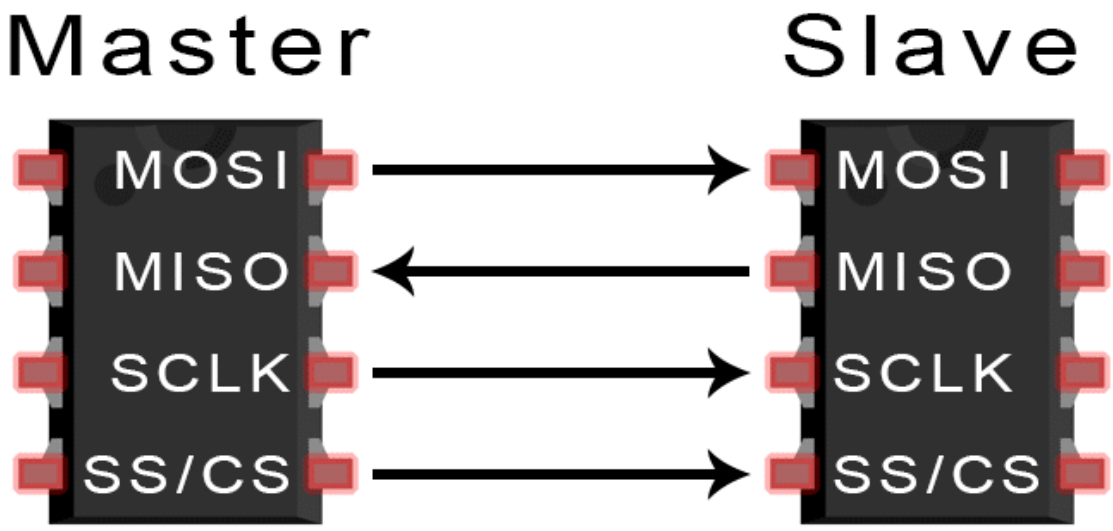


Figure 19

MOSI (Master Output/Slave Input) – Line for the master to send data to the slave.

MISO (Master Input/Slave Output) – Line for the slave to send data to the master.

SCLK (Clock) – Line for the clock signal.

SS/CS (Slave Select/Chip Select) – Line for the master to select which slave to send data to.

MOSI and MISO:

The master sends data to the slave bit by bit, in serial through the MOSI line. The slave receives the data sent from the master at the MOSI pin. Data sent from the master to the slave is usually sent with the most significant bit first.

Steps of SPI Data Transmission

1. The master outputs the clock signal:

Figure 20

2. The master switches the SS/CS pin to a low voltage state, which activates the slave:

Figure 21

3. The master sends the data one bit at a time to the slave along the MOSI line. The slave reads the bits as they are received:

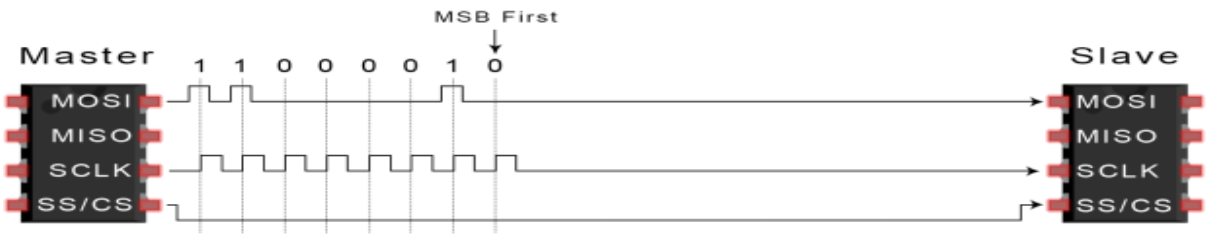


Figure 22

4. If a response is needed, the slave returns data one bit at a time to the master along the MISO line. The master reads the bits as they are received:

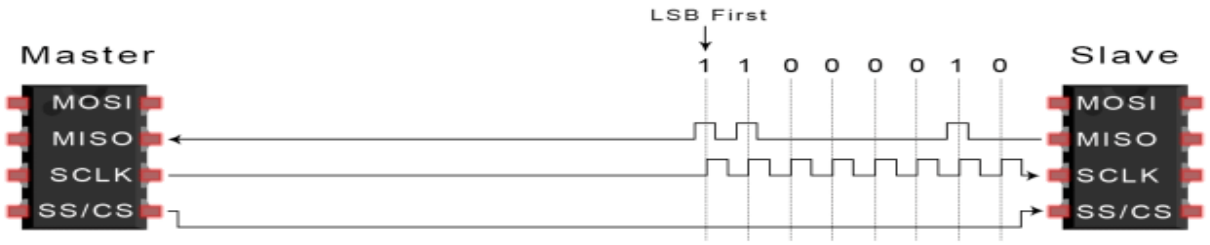


Figure 23

Advantages and Disadvantages of SPI

There are some advantages and disadvantages to using SPI, and if given the choice between different communication protocols**.**

Advantages:

- No start and stop bits, so the data can be streamed continuously without interruption
- No complicated slave addressing system like I2C
- Higher data transfer rate than I2C (almost twice as fast)
- Separate MISO and MOSI lines, so data can be sent and received at the same time.

Disadvantages:

- Uses four wires (I2C and UARTs use two)
- No acknowledgement that the data has been successfully received (I2C has this)
- No form of error checking like the parity bit in UART
- Only allows for a single master

Serial vs. Parallel Communication

Electronic devices talk to each other by sending bits of data through wires physically connected between devices. Bits are transferred from one device to another by quick changes in voltage. In a system operating at 5 V, a 0 bit is communicated as a short pulse of 0 V, and a 1 bit is communicated by a short pulse of 5 V.

The bits of data can be transmitted either in parallel or serial form.

In parallel communication, the bits of data are sent all at the same time, each through a separate wire. The following diagram shows the parallel transmission of the letter "C" in binary (01000011):

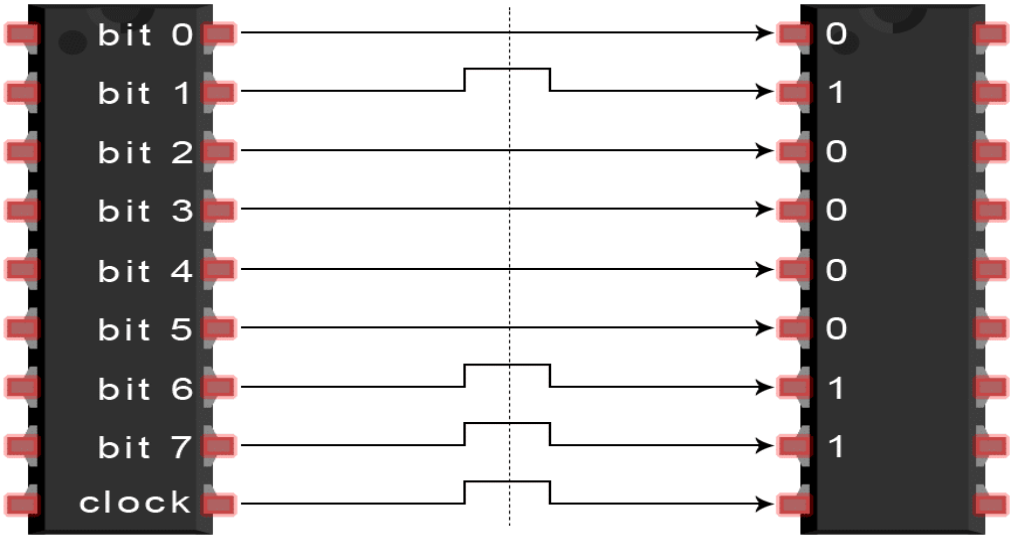


Figure 24

In serial communication, the bits are sent one by one through a single wire. The following diagram shows the serial transmission of the letter "C" in binary (01000011):

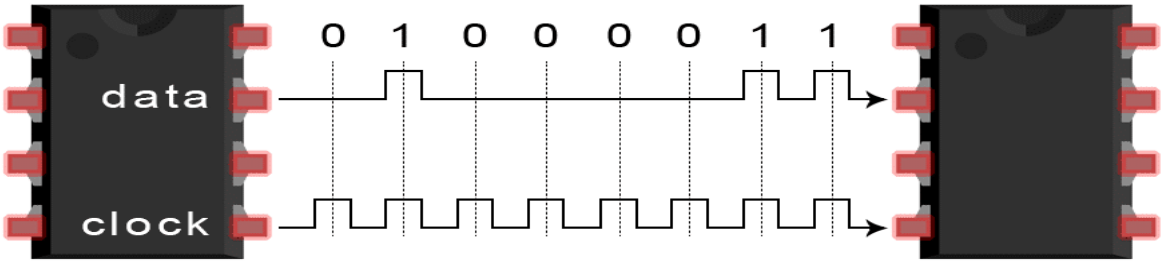


Figure 25

Serial Communication

The serial communication is a simple scheme that uses the **UART** (Universal Asynchronous Receiver/Transmitter) on the Microcontroller. It uses,

- **5V for logic 1 (high)**
- **0V for logic 0 (low)**

For a 3.3V board, it uses

- **3V for logic 1 (high)**
- **0V for logic 0 (low)**

Every message sent on the UART is in the form of 8 bits or 1 byte, where **1 byte = 8 bits.**

The messages sent to the computer from Arduino are **sent from PIN 1 of the Arduino board, called Tx (Transmitter)**. The messages being sent to the Arduino from the computer are **received on PIN 0, called Rx** (**Receiver**).

Arduino Mega has four serial ports. The Tx pins on the Mega board are listed below:

- 1 (TX)
- 18 (TX)
- 16 (TX)
- 14 (TX)

The Rx pins on the Mega port are listed below:

- 0 (RX)
- 19 (RX)
- 17 (RX)
- 15 (RX)

Figure 26

The circuit diagram:

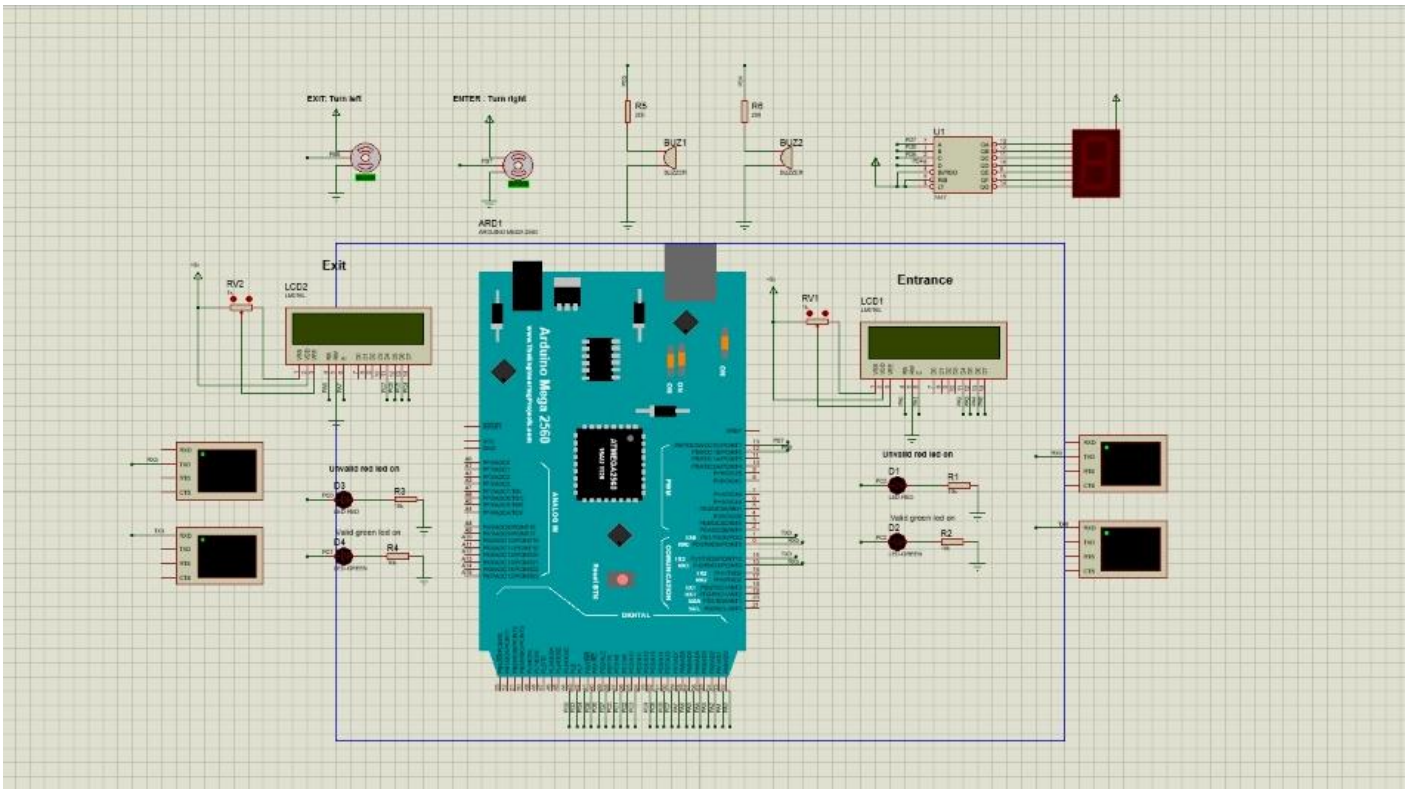


Figure 27

The connections:

Buzzer_1 → pin 2

green_Led_1→ pin 31

red_Led_1 → pin 30

buzzer2 → pin 42

green_Led_2 → pin 44

red_Led_2 → pin 39

lcd→pins (22,23,24,25,26,27)

lcd2→pins (32,33,34,35,36,37)

rf1 → pins (5,28,51,50,53)

rf2 → pins (6,38,51,50,53)

servo_1 → pin (8)

servo_2 → pin(11)

seven_segment → pins(16,17,18,19,20,21)

System program:

```
#include <Servo.h>

#include <LiquidCrystal.h>

#include <SPI.h>

#include <RFID.h>

#define buzzer1 2

#define green_Led_1 31

#define red_Led_1 30

#define buzzer2 42

#define green_Led_2 44

#define red_Led_2 39

LiquidCrystal lcd(22,23,24,25,26,27);

LiquidCrystal lcd2(32,33,34,35,36,37);

RFID rf1(28,5);
```

```
RFID rf2(38,6);

Servo servo_1;

Servo servo_2;

int sernum0[5] = {128,142,175,34,131};

int sernum1[5] = {96,20,112,26,30};

int sernum2[5] = {147,221,110,11,43};

int sernum3[5] = {179,65,106,4,156};

int sernum4_master[5] = {97,123,204,39,241};

int sernum5_master[5] = {128,37,197,34,66};

//int sernum[6][5] =
{{128,142,175,34,131},{96,20,112,26,30},{147,221,110,11,43},{179,65,106,4,156
},{97,123,204,39,241},{128,37,197,34,66}};

int serNum0_1=0;

int serNum1_1=0;

int serNum2_1=0;

int serNum3_1=0;

int serNum4_1=0;

int serNum0_2=0;

int serNum1_2=0;
```

```
int serNum2_2=0;

int serNum3_2=0;

int serNum4_2=0;

int count=0;

byte arrow_up[8] = {

  B00000,

  B11110,

  B11000,

  B10100,

  B10010,

  B00001,

  B00000,

  B00000

};

byte arrow_down[8] = {

  B00000,

  B00000,

  B10000,
```

```
  B01001,
  B00101,
  B00011,
  B01111,
  B00000
};
//seven segment pins
const int A=20;
const int B=19;
const int C=18;
const int D=17;
const int EN2 = 16;
const int EN1 = 21;
void setup()
{   pinMode(53 , OUTPUT);
   Serial.begin(9600);
   SPI.begin();
   rf1.init();
```

```
  rf2.init();
  lcd.begin(16 , 2);
  lcd2.begin(16 , 2);
  lcd.createChar(5,arrow_up);
  lcd2.createChar(6,arrow_down);
  lcd.setCursor(4, 0);
  lcd2.setCursor(6, 0);
  lcd.print("Entrance");
  lcd2.print("Exit");
  lcd.setCursor(3, 1);
  lcd2.setCursor(3, 1);
  for(int i =0 ; i<10 ; i++) {
  lcd.write(5);  }
for(int i =0 ; i<10 ; i++) {
  lcd2.write(6); }
  pinMode(buzzer1, OUTPUT);
  pinMode(green_Led_1, OUTPUT);
  pinMode(red_Led_1, OUTPUT);
```

```
  pinMode(buzzer2, OUTPUT);
  pinMode(green_Led_2, OUTPUT);
  pinMode(red_Led_2, OUTPUT);
  servo_1.attach(8);
  servo_2.attach(11);
  servo_1.write(90);
  servo_2.write(90);
  //seven segment
  pinMode(A, OUTPUT); //LSB
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT); //MSB
    }
void to_BCD(int count)
{  if (count == 0) //write 0000
       {   digitalWrite(A, LOW);
     digitalWrite(B, LOW);
     digitalWrite(C, LOW);
```

```
  digitalWrite(D, LOW);
}
if (count == 1) //write 0001
    {  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);   }
 if (count == 2) //write 0010
    {digitalWrite(A, LOW);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
}
 if (count == 3) //write 0011
    {  digitalWrite(A, HIGH);
  digitalWrite(B, HIGH);
  digitalWrite(C, LOW);
  digitalWrite(D, LOW);
```

```
 }
if (count == 4) //write 0100
      { digitalWrite(A, LOW);
   digitalWrite(B, LOW);
   digitalWrite(C, HIGH);
   digitalWrite(D, LOW); }
    if (count == 5) //write 0101
      {  digitalWrite(A, HIGH);
   digitalWrite(B, LOW);
   digitalWrite(C, HIGH);
   digitalWrite(D, LOW);}
    if (count == 6) //write 0110
      {  digitalWrite(A, LOW);
   digitalWrite(B, HIGH);
   digitalWrite(C, HIGH);
   digitalWrite(D, LOW);    }
    if (count == 7) //write 0111
      { digitalWrite(A, HIGH);
```

```
    digitalWrite(B, HIGH);
    digitalWrite(C, HIGH);
    digitalWrite(D, LOW); }
    if (count == 8) //write 1000
      { digitalWrite(A, LOW);
    digitalWrite(B, LOW);
    digitalWrite(C, LOW);
    digitalWrite(D, HIGH) }
  if (count == 9) //write 1001
      { digitalWrite(A, HIGH);
    digitalWrite(B, LOW);
    digitalWrite(C, LOW);
    digitalWrite(D, HIGH); }
 }
void loop()
{     if(rf1.isCard()){
   if(rf1.readCardSerial()){
    if(serNum0_1!=rf1.serNum[0] &&
```

```
 serNum1_1!=rf1.serNum[1] &&
 serNum2_1!=rf1.serNum[2] &&
 serNum3_1!=rf1.serNum[3] &&
 serNum4_1!=rf1.serNum[4] )
 {   Serial.println("Card ID: ");
 Serial.print(rf1.serNum[0]);
 Serial.print(",");
 Serial.print(rf1.serNum[1]);
 Serial.print(",");
 Serial.print(rf1.serNum[2]);
 Serial.print(",");
 Serial.print(rf1.serNum[3]);
 Serial.print(",");
 Serial.print(rf1.serNum[4]);
 Serial.println(" ");
serNum0_1=rf1.serNum[0];
 serNum1_1=rf1.serNum[1];
 serNum2_1=rf1.serNum[2];
```

```
    serNum3_1=rf1.serNum[3];

    serNum4_1=rf1.serNum[4]

  if((sernum0[0] == rf1.serNum[0]&&sernum0[1]==
rf1.serNum[1]&&sernum0[2]== rf1.serNum[2]&&sernum0[3]==
rf1.serNum[3]&&sernum0[4]== rf1.serNum[4])||

    (sernum1[0] == rf1.serNum[0]&&sernum1[1]==
rf1.serNum[1]&&sernum1[2]== rf1.serNum[2]&&sernum1[3]==
rf1.serNum[3]&&sernum1[4]== rf1.serNum[4])||

    (sernum2[0] == rf1.serNum[0]&&sernum2[1]==
rf1.serNum[1]&&sernum2[2]== rf1.serNum[2]&&sernum2[3]==
rf1.serNum[3]&&sernum2[4]== rf1.serNum[4])||

    (sernum3[0] == rf1.serNum[0]&&sernum3[1]==
rf1.serNum[1]&&sernum3[2]== rf1.serNum[2]&&sernum3[3]==
rf1.serNum[3]&&sernum3[4]== rf1.serNum[4])||

    (sernum4_master[0] == rf1.serNum[0]&&sernum4_master[1]==
rf1.serNum[1]&&sernum4_master[2]== rf1.serNum[2]&&sernum4_master[3]==
rf1.serNum[3]&&sernum4_master[4]== rf1.serNum[4])||

    (sernum5_master[0] == rf1.serNum[0]&&sernum5_master[1]==
rf1.serNum[1]&&sernum5_master[2]== rf1.serNum[2]&&sernum5_master[3]==
rf1.serNum[3]&&sernum5_master[4]== rf1.serNum[4])){

        tone(buzzer1,1000); // Send 1KHz sound signal...

        digitalWrite(green_Led_1 , HIGH);
```

```
        servo_1.write(0);
         count++;
        to_BCD(count);
        digitalWrite(EN1 , HIGH);
         delay(2000);        // ...for 1 sec
        noTone(buzzer1);     // Stop sound...
       digitalWrite(green_Led_1 , LOW);
       servo_1.write(90);
        digitalWrite(EN1 , LOW);
    }
else{
   digitalWrite(red_Led_1,HIGH);
   delay(2000);
   digitalWrite(red_Led_1,LOW);
     }         }           }   }
    rf1.halt();
 if(rf2.isCard()){
        if(rf2.readCardSerial()){
```

```
if(serNum0_2!=rf2.serNum[0] &&
 serNum1_2!=rf2.serNum[1] &&
 serNum2_2!=rf2.serNum[2] &&
 serNum3_2!=rf2.serNum[3] &&
 serNum4_2!=rf2.serNum[4] )
 {     Serial.println("Card ID: ");
 Serial.print(rf2.serNum[0]);
 Serial.print(",");
 Serial.print(rf2.serNum[1]);
 Serial.print(",");
 Serial.print(rf2.serNum[2]);
 Serial.print(",");
 Serial.print(rf2.serNum[3]);
 Serial.print(",");
 Serial.print(rf2.serNum[4]);
 Serial.println(" ");
 serNum0_2=rf2.serNum[0];
 serNum1_2=rf2.serNum[1];
```

```
    serNum2_2=rf2.serNum[2];

    serNum3_2=rf2.serNum[3];

    serNum4_2=rf1.serNum[4];

      if((sernum0[0] == rf2.serNum[0]&&sernum0[1]==
rf2.serNum[1]&&sernum0[2]== rf2.serNum[2]&&sernum0[3]==
rf2.serNum[3]&&sernum0[4]== rf2.serNum[4])||

    (sernum1[0] == rf2.serNum[0]&&sernum1[1]==
rf2.serNum[1]&&sernum1[2]== rf2.serNum[2]&&sernum1[3]==
rf2.serNum[3]&&sernum1[4]== rf2.serNum[4])||

    (sernum2[0] == rf2.serNum[0]&&sernum2[1]==
rf2.serNum[1]&&sernum2[2]== rf2.serNum[2]&&sernum2[3]==
rf2.serNum[3]&&sernum2[4]== rf2.serNum[4])||

    (sernum3[0] == rf2.serNum[0]&&sernum3[1]==
rf2.serNum[1]&&sernum3[2]== rf2.serNum[2]&&sernum3[3]==
rf2.serNum[3]&&sernum3[4]== rf2.serNum[4])||

    (sernum4_master[0] == rf2.serNum[0]&&sernum4_master[1]==
rf2.serNum[1]&&sernum4_master[2]== rf2.serNum[2]&&sernum4_master[3]==
rf2.serNum[3]&&sernum4_master[4]== rf2.serNum[4])||

    (sernum5_master[0] == rf2.serNum[0]&&sernum5_master[1]==
rf2.serNum[1]&&sernum5_master[2]== rf2.serNum[2]&&sernum5_master[3]==
rf2.serNum[3]&&sernum5_master[4]== rf2.serNum[4])){

      tone(buzzer2, 1000); // Send 1KHz sound signal...
```

```
    digitalWrite(green_Led_2 , HIGH);
    servo_2.write(0);
    count--;
    to_BCD(count);
    digitalWrite(EN1 , HIGH);
    delay(2000);        // ...for 1 sec
    noTone(buzzer2);     // Stop sound...
    digitalWrite(green_Led_2 , LOW);
    servo_2.write(90);
    digitalWrite(EN1 , HIGH);       }
else{
 digitalWrite(red_Led_2,HIGH);
delay(2000);
digitalWrite(red_Led_2,LOW);
        }   }   }   }    rf2.halt(); }
```