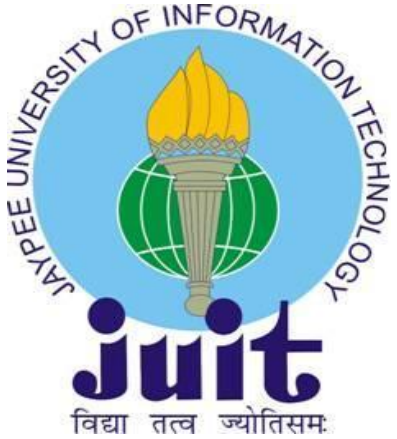
# BLUETOOTH DIGITAL LOCK

Submitted in partial fulfillment of the Degree of

Bachelor of Technology




May -15

Under the Supervision of

**Mr. Pardeep Garg**
Assistant Professor

Submitted By

**Siddharth Singh-111019**

**Gaurav Maheshwari-111112**

**Ashish Ahuja-111115**

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT,SOLAN,H.P.

# CERTIFICATE

This is to certify that project report entitled **"BLUETOOTH DIGITAL LOCK"**, submitted by **Siddharth Singh (111019), Gaurav Maheshwari (111112) and Ashish Ahuja (111115)** in partial fulfilment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

**Date:**

**Supervisor's Name:** Mr. Pardeep Garg

**Designation:** Assistant Professor
Department of ECE

**Signature:**

# ACKNOWLEDGEMENT

We owe a great many thanks to a great many people who have helped and supported us during this project. Our deepest thanks to **Mr. Pardeep Garg (Assistant Professor)**, our project guide for his exemplary guidance, monitoring and constant encouragement throughout the course of this project work. He has taken great pains to go through the project and make necessary corrections as and whenever needed. We are also grateful to **Mr. Mohan (ECE Project lab**) for his practical help and guidance. We would also like to thank all the faculty members of ECE department without whom the progress of this project would have been a distant reality. We also extend our heartfelt thanks to our family and well-wishers.

**Date:**

**Name of the students:**

Siddharth Singh (111019)

Gaurav Maheshwari (111112)

Ashish Ahuja (111115)

# List of Figures

# List of Symbols

| S No. | Symbol | Meaning |
|---|---|---|
| 1 | K | Kilo |
| 2 | MHz | Mega Hertz |
| 3 | V | Volt |
| 4 | uF | Micro Farad |
| 5 | M | Metres |
| 6 | mA | Milli Ampere |
| 7 | KB | Kilo Byte |
| 8 | Mm | Milli Metre |
| 9 | G | Gram |
| 10 | Hz | Hertz |
| 11 | GHz | Giga Hertz |
| 12 | dBm | Decibel Metre |
| 13 | Mbps | Mega Byte per Second |
| 14 | Kbps | Kilo Byte per Second |
| 15 | In | Inch |
| 16 | Oz | Ounce |
| 17 | Kg-cm | Kilogram Centimetre |
| 18 | C | Centigrade |

# Abstract

The goal of this project is to develop a software version of Digital Door Lock using 8051 Microcontroller in first module and a working model of Bluetooth Digital Lock using Arduino Uno in its second module. This advancement in technology facilitates the security mechanism in an efficient manner and can be used to replace the existing door locks.

In our software implementation in the first module of the project, the microcontroller based digital lock is an access control system that allows only authorized persons to access any restricted division. The major components include a keypad, LCD, DC motor and the micro controller AT89C51 which belongs to the 8051 series of micro controllers. The system is fully controlled by the 8 bit microcontroller AT89C51 which has a 4 KB of ROM for the program memory. The electronic control assembly allows the system to unlock and lock the device with a password. A four digit predefined password needs to be specified by the user. A 4x3 matrix keypad and a 16x2 LCD have been used here to set the password which is stored in the microcontroller's internal memory. While unlocking and locking, if the entered password from keypad matches with the stored password, then the lock opens and closes respectively and a message is displayed on LCD.

In our working model of the door lock we have used Arduino Uno to make the effective use of the present and fast growing technology. The main aim of this project is to make the easy and secure access control, to make it more compatible to its user we have also used Bluetooth Module so that the user can lock and unlock it with the help of the Bluetooth Terminal Application installed in his Smartphone.

The main aim of our project is to understand the 8051 microcontroller architecture and its applications, Arduino Uno and Bluetooth technology better which has many applications in the near future. Also we have been able to comprehend with the working of Digital Door Lock system through the development of our working model of the same.

# Table of Content

# 1. Introduction

## 1.1 Objective:

Security is a prime concern in our day-today life. Everyone wants to be as much secure as possible. The microcontroller based digital lock for Doors is an access control system that allows only authorized persons to access a restricted area. It can also be accessed using Bluetooth Kit in its working model using Arduino Uno .The software module is fully controlled by the 8 bit microcontroller AT89C51 which has a 4KB of ROM for the program memory. The system has a keypad by which the password can be entered through it. When the entered password matches with the password stored in the memory then the motor starts rotating to unlock the door. In working model, Arduino Uno is used to control the working of LCD And the servo motor. Bluetooth Kit is embedded in it in order to make it work with the help of the Bluetooth Terminal Application installed in the Smartphone of the user.

## 1.2 What is Digital security code lock?

'Digital Security' gives individuals the freedom to embrace the digital lifestyle confidently engage in everyday interactions across all digital devices. According to Olivier Piou, CEO of global digital security company Gemalto, "digital security has a key role to play in the digital revolution. Yet fear of fraud, identity theft, and other concerns are holding people back from making the most of (what the digital revolution has to offer). They need to feel that the wealth of devices and services available are both convenient to use and trustworthy. "Digital security affects all aspects of the digital lifestyle, which, among others, comprises computers and the internet, telecommunications, financial transactions, transportation, healthcare and secure access. With the number of contemptible acts like robberies, ransacking and vandalism, the absence of competent security solutions is an open invitation to trouble. Digital cameras are extremely useful devices that ensure the safety of your property when you are away at work on or vacation. The same is the case with Digital Lock. It also helps by securing things such as home doors, locker etc.

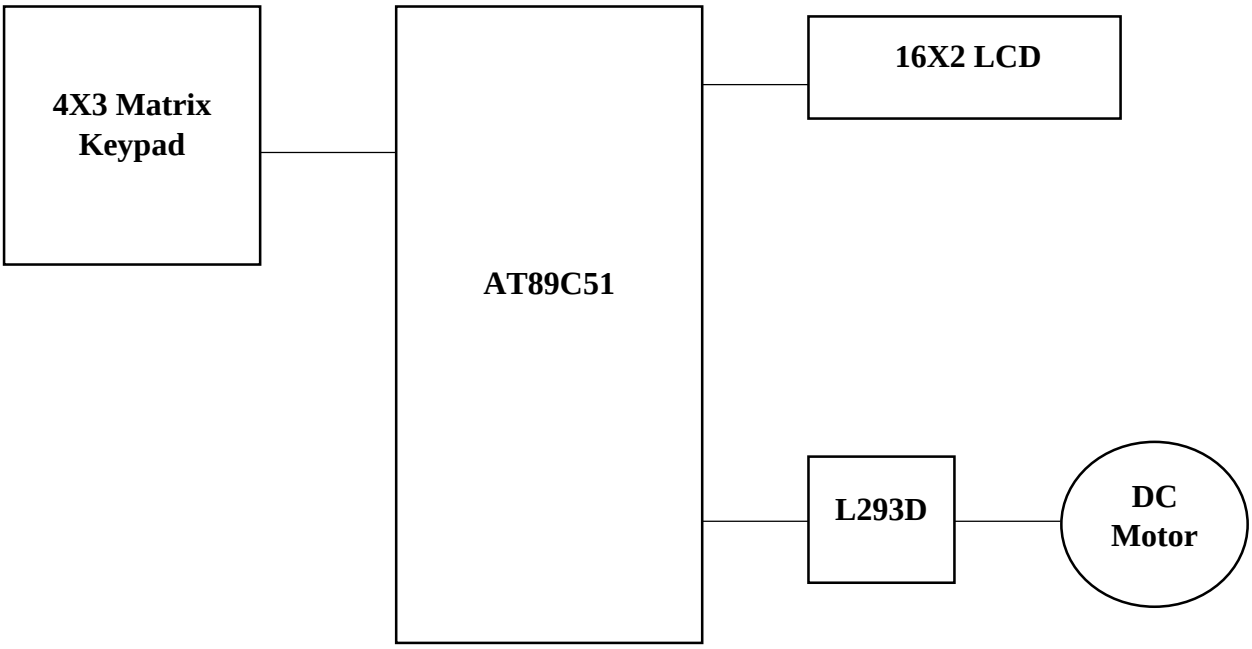
# Module First

## 2 Technical Details

### 2.1 Hardware Required:

| S. No. | Components Used | Quantity |
|---|---|---|
| 1 | AT89C51 Microcontroller (base + IC) | 1 |
| 2 | Diode (IN4007, .7v) | 1 |
| 3 | Motor Driver L293D (base + IC) | 1 |
| 4 | DC Motor | 1 |
| 5 | Potentiometer | 1 |
| 6 | Crystal oscillator (12MHz frequency) | 1 |
| 7 | DC Adapter | 1 |
| 8 | Capacitor(10,470,0.1)uF | 1 |
| 9 | 2,5-Pin connector | 2 |
| 10 | LM7805 Voltage regulator | 1 |
| 11 | 16X2 LCD | 1 |
| 12 | 4X3 Matrix Keypad | 1 |
| 13 | Cello tape (for electrical use) | 1 |
| 14 | Supply wire | 2m |
| 15 | Switches | 2 |

## 2.2 Block Diagram:



## 2.3 Circuit Diagram:

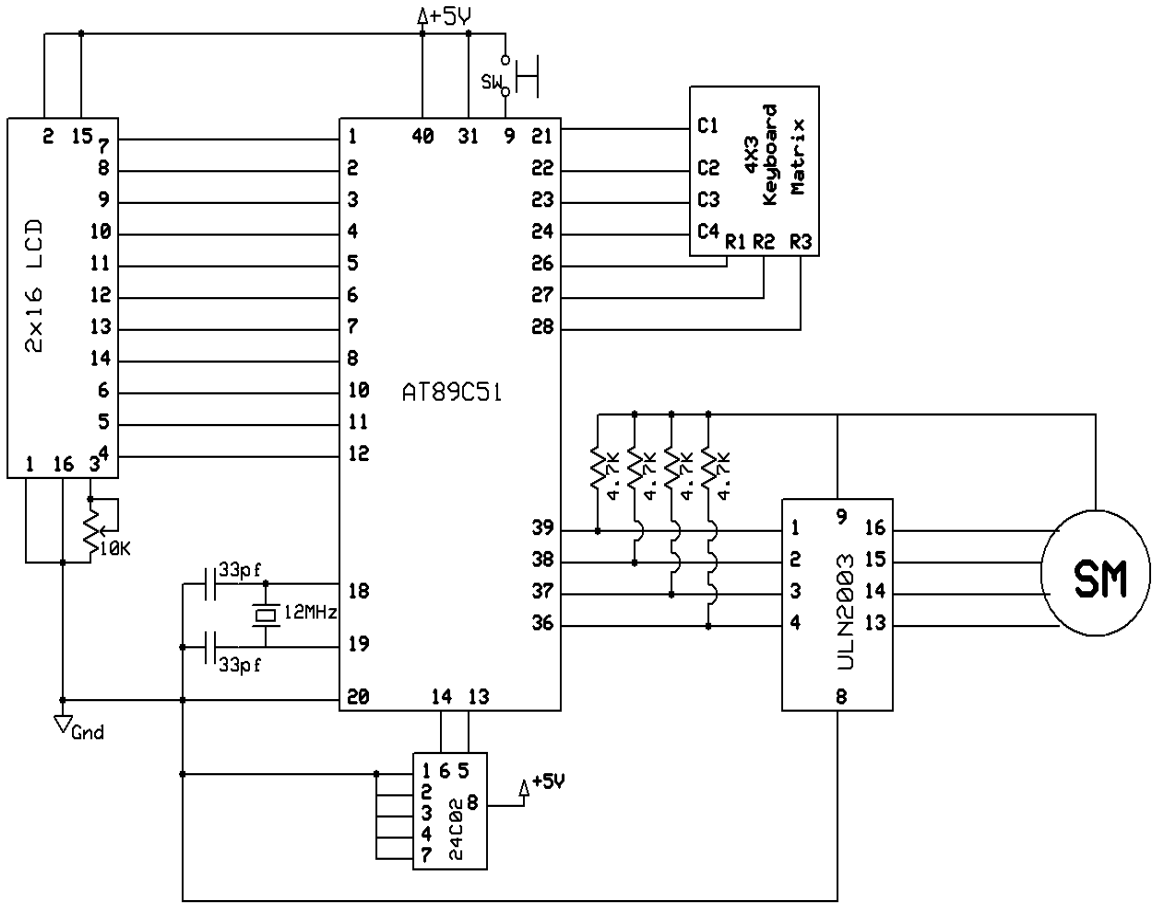



Fig. 1: Circuit diagram of Digital Lock using AT89C51 Microcontroller

# 3 Hardware Description

## 3.1 AT89C51 Microcontroller:

**Definition:**

An embedded microcontroller is chip which has a computer processor with all its support functions (clock & reset), memory (both program and data), and I/O (including bus interface) built in to the device. These built in function minimize the need for external circuits and devices to be designed in the final application.

**Features of AT89C51:**

• Compatible with MCS-51 Products

• 4K Bytes of In-System Reprogrammable Flash Memory

• Fully Static Operation: 0 Hz to 24 MHz

• Three-level Program Memory Lock

• 128 x 8-bit Internal RAM

• 32 Programmable I/O Lines

• Two 16-bit Timer/Counters

• Six Interrupt Sources

• Programmable Serial Channel

• Low-power Idle and Power-down Modes

**Description:**

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4KB of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density non-volatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pin out. The on-chip flash allows the program memory to be reprogrammed in-system or by a conventional non-volatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

**Pin Diagram of 8051:**

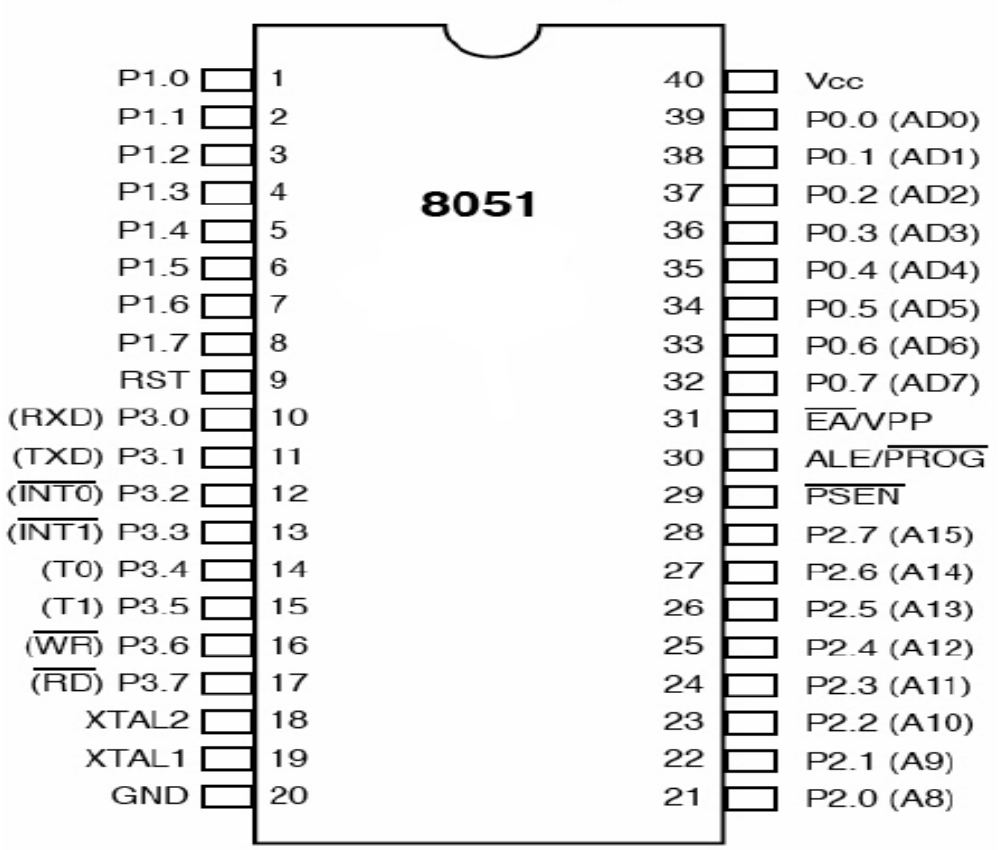



Fig. 2: Pin diagram of At89C51 Microcontroller

**Pin Description:**

**Pins 1-8:** Port 1 – Each of these pins can be configured as an input or an output.

**Pin 9:** RS- A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning.

**Pins10-17:** Port 3- Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions:

**Pin 10:** RXD- Serial asynchronous communication input or Serial synchronous communication output.

**Pin 11:** TXD- Serial asynchronous communication output or Serial synchronous communication clock output.

**Pin 12:** INT0- Interrupt 0 input.

**Pin 13:** INT1- Interrupt 1 input.

**Pin 14:** T0- Counter 0 clock input.

**Pin 15:** T1- Counter 1 clock input.

**Pin 16:** WR- Write to external (additional) RAM**.**

**Pin 17:** RD- Read from external RAM.

**Pin 18, 19:** X2, X1- Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually connected to these pins. Instead of it, miniature ceramics resonators can also be used for frequency stability. Later versions of microcontrollers operate at a frequency of 0 Hz up to over 50 Hz.

**Pin 20:** GND- Ground.

**Pin 21-28:** Port 2- If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64KB is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs.

**Pin 29:** PSEN- If external ROM is used for storing program then a logic zero (0) appears on it every time the microcontroller reads a byte from memory.

**Pin 30:** ALE- Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the ALE pin, the external register (usually 74HCT373 or 74HCT375 add-on chip) memorizes the state of P0 and uses it as a memory chip address. Immediately after that, the ALU pin is returned to its previous logic state and P0 is now used as a Data Bus. As seen, port data multiplexing is performed by means of only one additional (and cheap) integrated circuit. In other words, this port is used for both data and address transmission**.**

**Pin 31:** EA- By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed. By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exists).

**Pin 32-39:** Port 0- Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).

**Pin 40:** VCC +5V power supply.

### 3.2 16X2 LCD:

**Definition:**

The dot-matrix liquid crystal display controller and driver LSI displays alphanumeric, characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4 or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver. A single HD44780U can display up to two 8-character lines (16 x 2).

**Pin Diagram:**

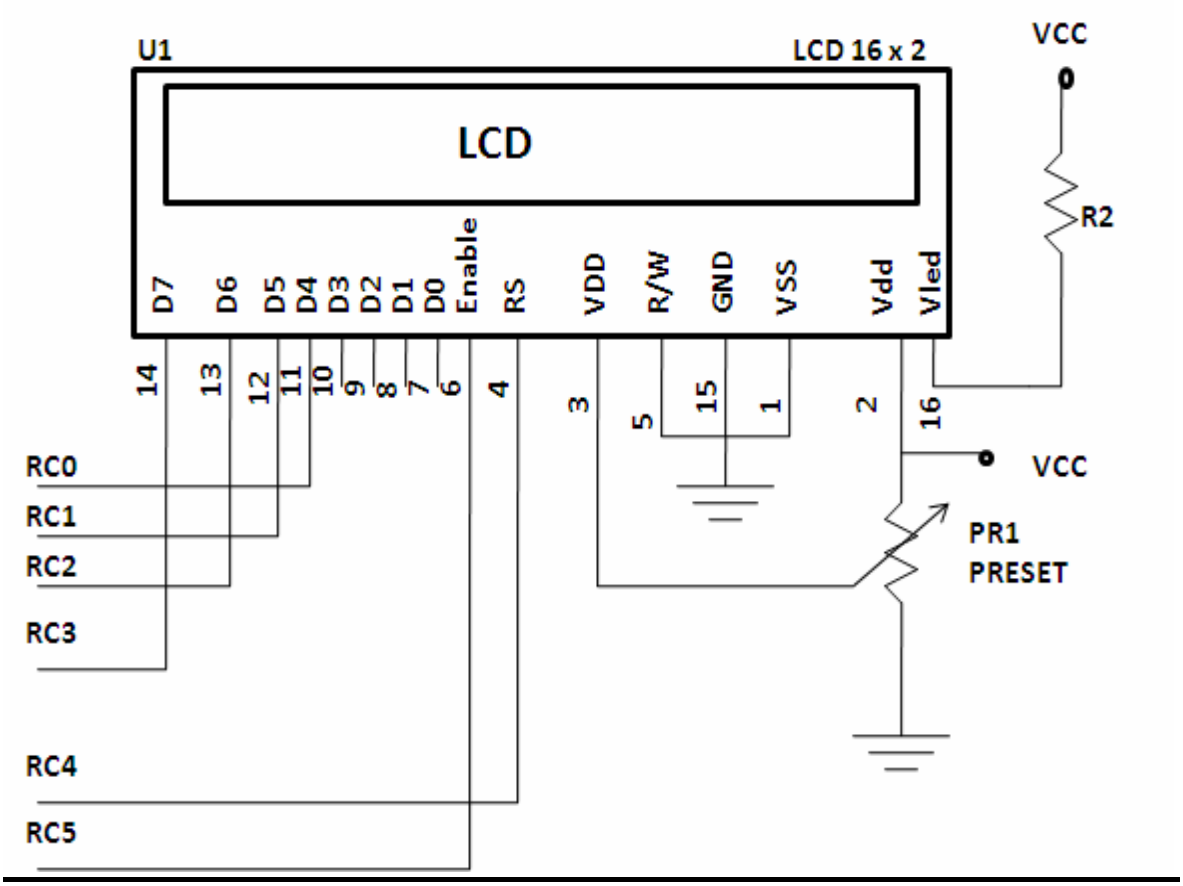



Fig. 3: Pin diagram of LCD

**Descriptions:**

A 16 x 2 line LCD module to display user information. Micro controller send the data signals through pin 11 through 18(RC0-RC3) and control signal through 4, 6 and 7 of the micro controller. Pin no 3 of the LCD is used to control the contrast by using pre-set PR1.

**LCD Command Codes**

| Code (Hex) | Command to LCD Instruction Register |
|---|---|
| 1 | Clear display screen |
| 2 | Return home |
| 4 | Decrement cursor (shift cursor to left) |
| 6 | Increment cursor (shift cursor to right) |
| 5 | Shift display right |
| 7 | Shift display left |
| 8 | Display off, cursor off |
| A | Display off, cursor on |
| C | Display on, cursor off |
| E | Display on, cursor blinking |
| F | Display on, cursor blinking |
| 10 | Shift cursor position to left |
| 14 | Shift cursor position to right |
| 18 | Shift the entire display to the left |
| 1C | Shift the entire display to the right |
| 80 | Force cursor to beginning to 1st line |
| C0 | Force cursor to beginning to 2nd line |
| 38 | 2 lines and 5x7 matrix |

Fig. 4: LCD command codes

### 3.3 4X3 Matrix Keypad:

**Definition:**

A keypad is the most widely used input devices of a microcontroller. In order to use it effectively, we need a basic understanding of them. At the lowest level, keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and column through ports; therefore, with a port of microcontroller, a 4X3 matrix of keys can be connected. When a key pressed, a row and column make a connection; otherwise, there is no connection between row and column.

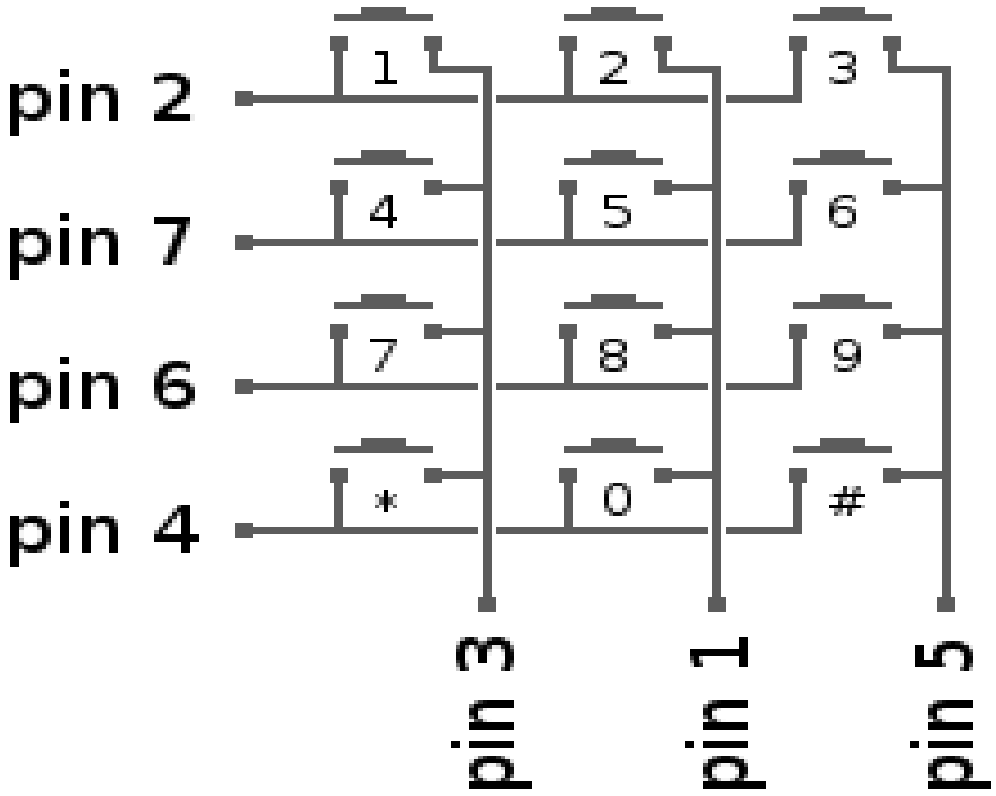



Fig. 5: Pin diagram of 4X3 matrix keypad

### 3.4 DC Motor:

**Definition:**

A DC motor relies on the fact that like magnet poles repel and unlike magnetic poles attract each other. A coil of wire with a current running through it generates an electromagnetic field aligned with the centre of the coil.

**Interfacing DC Motor to 8051:**

The maximum output current of microcontroller pin is 15mA at 5V. But the power requirements of most of DC motors is out of reach of the microcontroller and even the back emf (electro motive force) which is produced by the motor may damage the microcontroller. Hence it is not good to interface DC motor directly to the controller. So, motor driver circuit is used in between of DC motor and controller.

Here, we are using L293D motor driver IC to drive DC motors. Using this IC, we can drive 2 DC motors at a time. For this IC motor supply is variable 4.5 to 36V and it provides maximum current of 600mA.

The motor driver input pins IN1, IN2 are connected to the P3.0 and P3.1 respectively to control the motor directions. DC motor is connected to output terminals of L293D. EN1 pin is connected to the 5V DC to drive the motor. Switches are connected to the P2.0 and P2.1 in pull down configuration. First switch rotates the motor in clockwise direction and second switch rotates the motor in anti-clockwise direction. $8^{th}$ pin of motor driver is connected to the battery directly.

**L293D Motor Driver:**

L293D is a quadruple H- bridge motor driver, as the name suggests it used to drive the DC motors. This IC works based on the concept of H- Bridge. H-bridge is a circuit which allows the voltage in either direction to control the motor direction.

There are 4 input pins for L293D. Motors directions depends on the logic inputs applied at this pins. EN1 and EN2 must be high to drive the 2 DC motors.

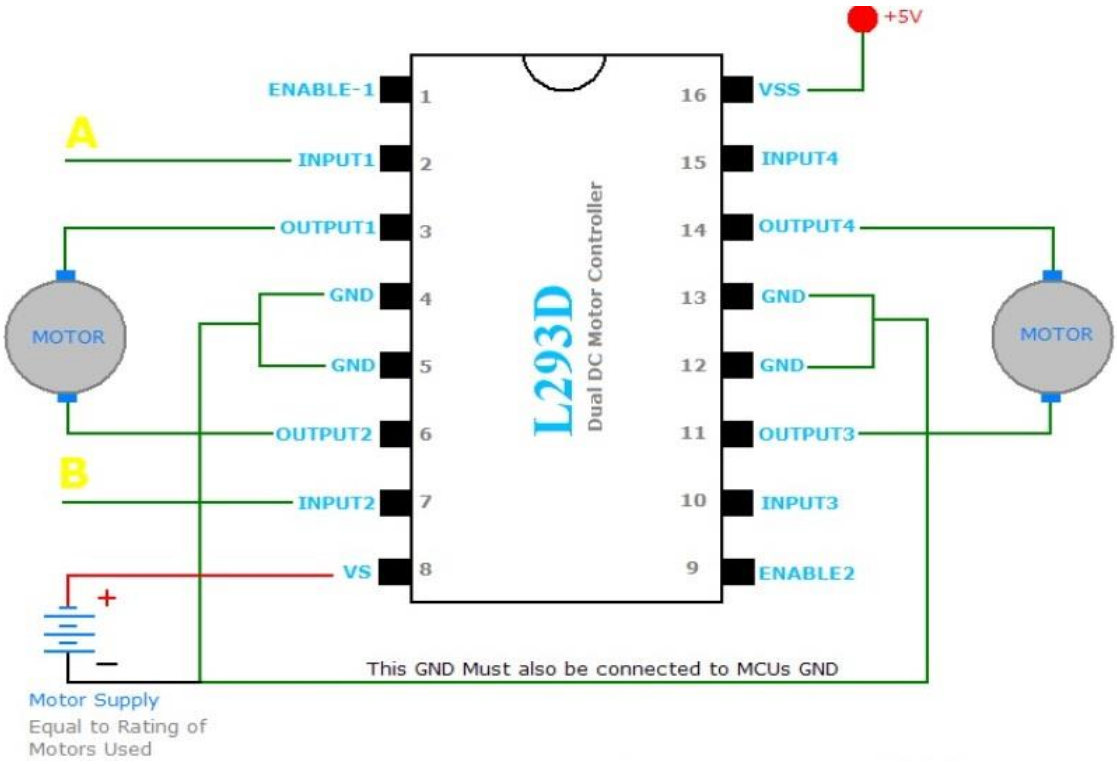
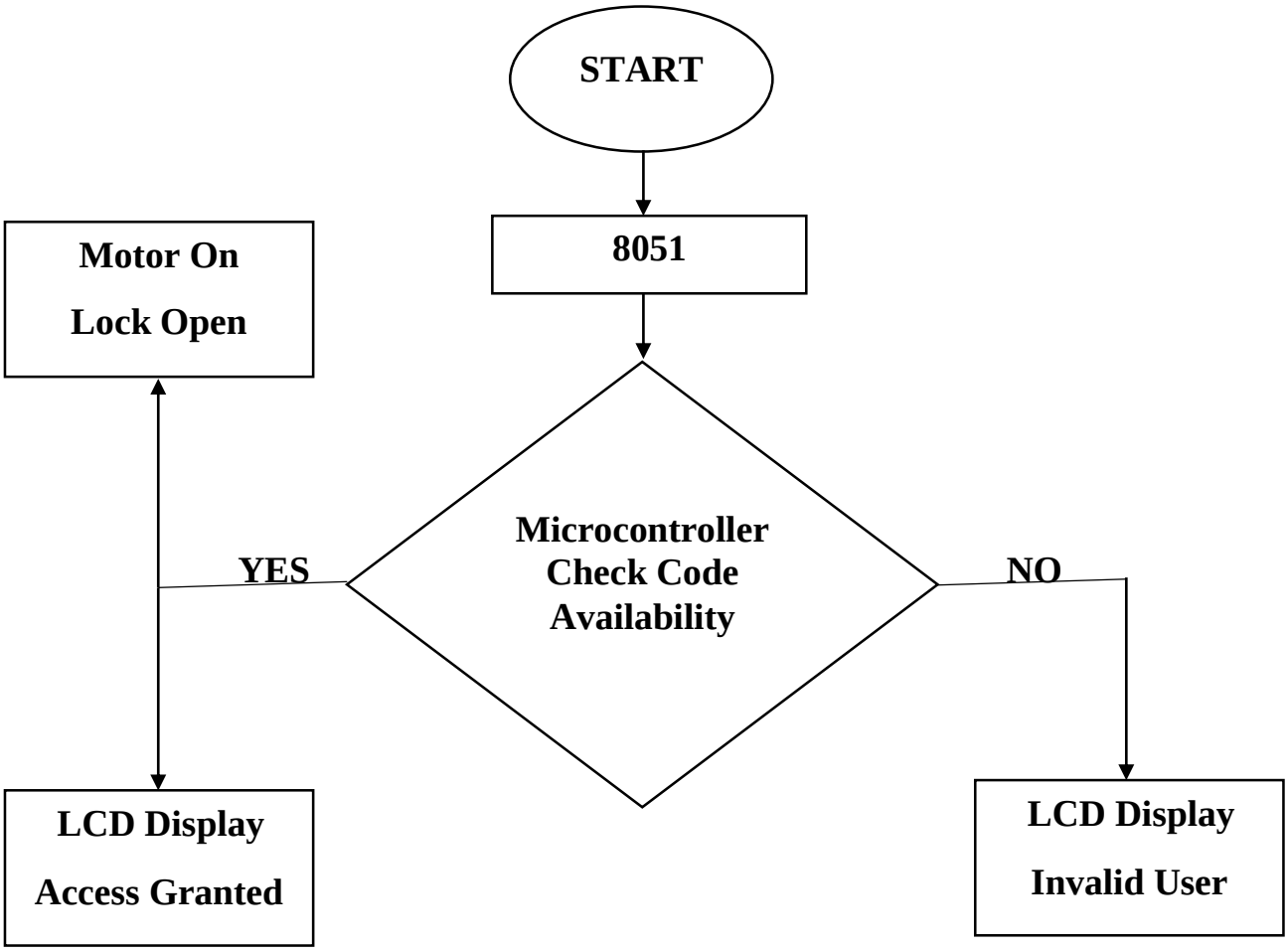



Fig. 6: Pin diagram of L293D DC motor driver

- IN1=0 and IN2=0 -> Motor1 idle
- IN1=0 and IN2=1 -> Motor1 Anti-clock wise direction
- IN1=1 and IN2=0 -> Motor1 Clock wise direction
- IN1=1 and IN2=1 -> Motor1 idle
- IN3=0 and IN4=0 -> Motor2 idle
- IN3=0 and IN4=1 -> Motor2 Anti-clock wise direction
- IN3=1 and IN4=0 -> Motor2 Clock wise direction
- IN3=1 and IN4=1 -> Motor2 idle

# 4 Methodology

## 4.1 Process Chart:



### 4.2 Software Set-up on Proteus:

• The power supply is provided first to the software and through voltage regulator, the circuit components receive their proper supply voltage.

• After that, the LCD displays the required BCD code and asks to the person to show the tag.

• The code is received by the module and checks the code availability.

• For correct code, LCD displays the CORRECT PASSWORD

• For wrong code, LCD displays INVALID PASSWORD.

### 4.3 Source Code:

```
#include <reg51.h>
#include <string.h>
sbit row1=P1^0;
sbit row2=P1^1;
sbit row3=P1^2;
sbit row4=P1^3;
sbit col1=P1^4;
sbit col2=P1^5;
sbit col3=P1^6;
sbit pin3_2=P3^2;
sbit pin3_3=P3^3;
sbit pin3_5=P3^5;
sbit rs=P3^0;
sbit en=P3^1;
```

```
sbit check=P3^7;
void delay(int t);
void readswitch();
void lcdcmd(unsigned char cmd);
void lcddata(unsigned char *p);
void readpassword();
unsigned char msg1[]={“1:UNLOCK “};
unsigned int ch=0,count=0;
char pswd[4];
void main()
  {
   while(1)
  {
  strcpy(pswd,NULL);
  P1=0x0FF;
  if(ch==0)
  {
  lcdcmd(0x30);
  lcdcmd(0x80);
  lcddata(“1:UNLOCK”);
  ch=1;
  }
  while(ch==1)
```

```
{
col1=0;
col2=1;
readswitch();
col2=0;
col1=1;
readswitch();
}
if(ch==2)
readpassword();
}
}
void readpassword()
{
count=0;
lcdcmd(0x01);
lcdcmd(0x30);
lcdcmd(0x80);
lcddata("1:ENTER PASSWORD");
lcdcmd(0x38);
lcdcmd(0xC0);
ch=4;
while(count<=(4-1))
```

```
{
col1=0;
col2=1;
col3=1;
readswitch();
delay(8);
col1=1;
col2=0;
col3=1;
readswitch();
delay(8);
col1=1;
col2=1;
col3=0;
readswitch();
delay(8);
}
lcdcmd(0x01);
lcdcmd(0x30);
if(strcmp(“1234”,pswd)!=0)
{
count=0;
lcddata(“WRONG PASSWORD”);
```

```
main();
//delay(1000);
//readpassword();
}
else
{
lcdcmd(0x01);
lcdcmd(0x30);
lcdcmd(0x80);
lcddata("Press 0: TO LOCK");
pin3_2=1;
pin3_3=0;
delay(5);
pin3_3=1;
ch=5;
}
while(ch==5)
{
col2=0;
readswitch();
 }
 }
//Lcd work
```

```
void lcdcmd(unsigned char cmd)
{
P2=cmd;
rs=0;
en=1;
delay(5);
en=0;
}
void lcddata(unsigned char *p)
{
unsigned char i=0;
while(*(p+i)!=NULL)
{
lcdcmd(0x0F);
P2=*(p+i);
rs=1;
en=1;
delay(1);
en=0;
i=i+1;
}
}
//lcd work end here
```

```
void delay( int time)
 {
 int I,j;
 for(i=0;i<time;i++)
 for(j=0;j<1275;j++);
 }
 void readswitch()
 {
 //For Unlock and Password Alter
 if(ch==1)
 {
 if(col1==0)
 {
 if(row1==0)
 ch=2;
 }
 if(col2==0)
 {
 if(row1==0)
 ch=3;
 }
 }
 //End Here Unlock and Password Alter
```

```
//For password read
 else if(ch==4)
 {
if(col1==0)
{
if(row1==0)
{
pswd[count]=’1’;
lcddata(“*”);
count++;
}
else if(row2==0)
{
pswd[count]=’4’;
lcddata(“*”);
count++;
}
else if(row3==0)
{
pswd[count]=’7’;
lcddata(“*”);
count++;
}
```

```
else if(row4==0)
{
pswd[count]='*';
lcddata("*");
count++;
}
}
else if(col2==0)
{
if(row1==0)
{
pswd[count]='2';
lcddata("*");
count++;
}
else if(row2==0)
{
pswd[count]='5';
lcddata("*");
count++;
}
else if(row3==0)
{
```

```
pswd[count]='8';
lcddata("*");
count++;
}
else if(row4==0)
{
pswd[count]='0';
lcddata("*");
count++;
}
}
else if(col3==0)
{
if(row1==0)
{
pswd[count]='3';
lcddata("*");
count++;
}
else if(row2==0)
{
pswd[count]='6';
lcddata("*");
```

```
count++;
}
else if(row3==0)
{
pswd[count]='9';
lcddata("*");
count++;
}
else if(row4==0)
{
pswd[count]='#';
lcddata("*");
count++;
}
}
}
if(ch==5)
{
if(col2==0)
{
if(row4==0)
{
lcdcmd(0x01);
```

```
lcdcmd(0x30);

lcdcmd(0x80);

lcddata(“LOCKED”);

pin3_2=0;

pin3_3=1;

delay(10);

lcdcmd(0x01);

pin3_2=1;

ch=0;

}

}

}

}
```

## 4.4 Result of Module First:

The project works for the door access by checking the password and displays the results accordingly.

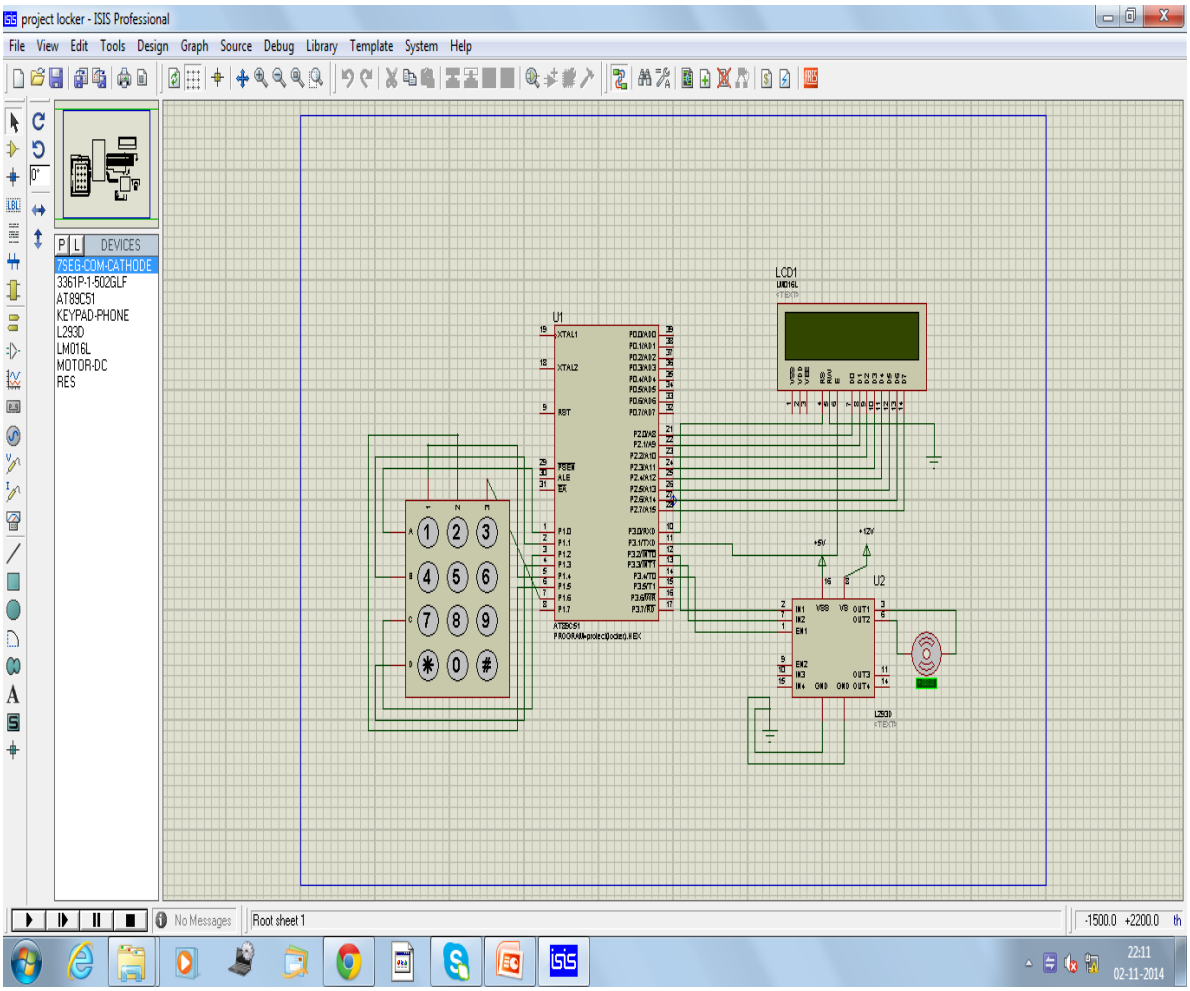



Fig. 7: Result of Module First Proteus schematic

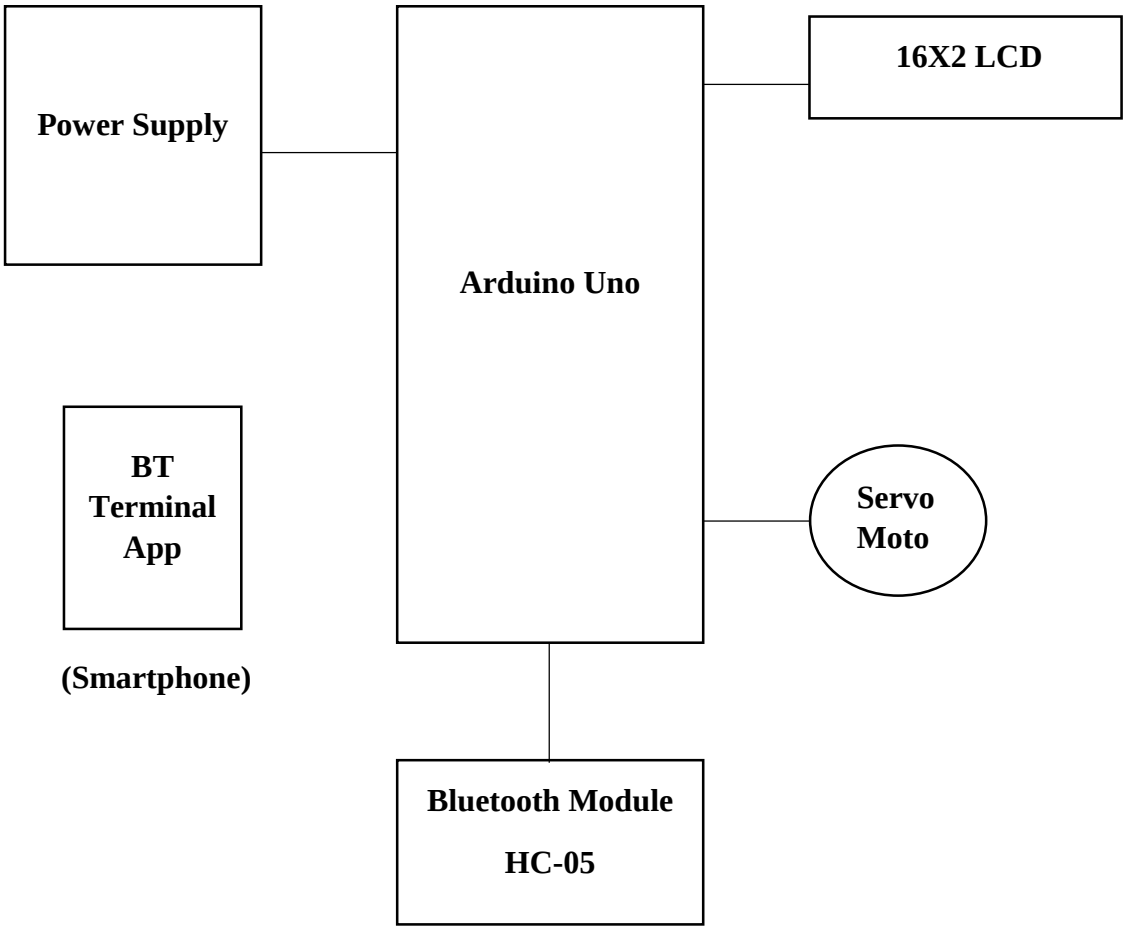
# Module Second

## 5 Technical Details

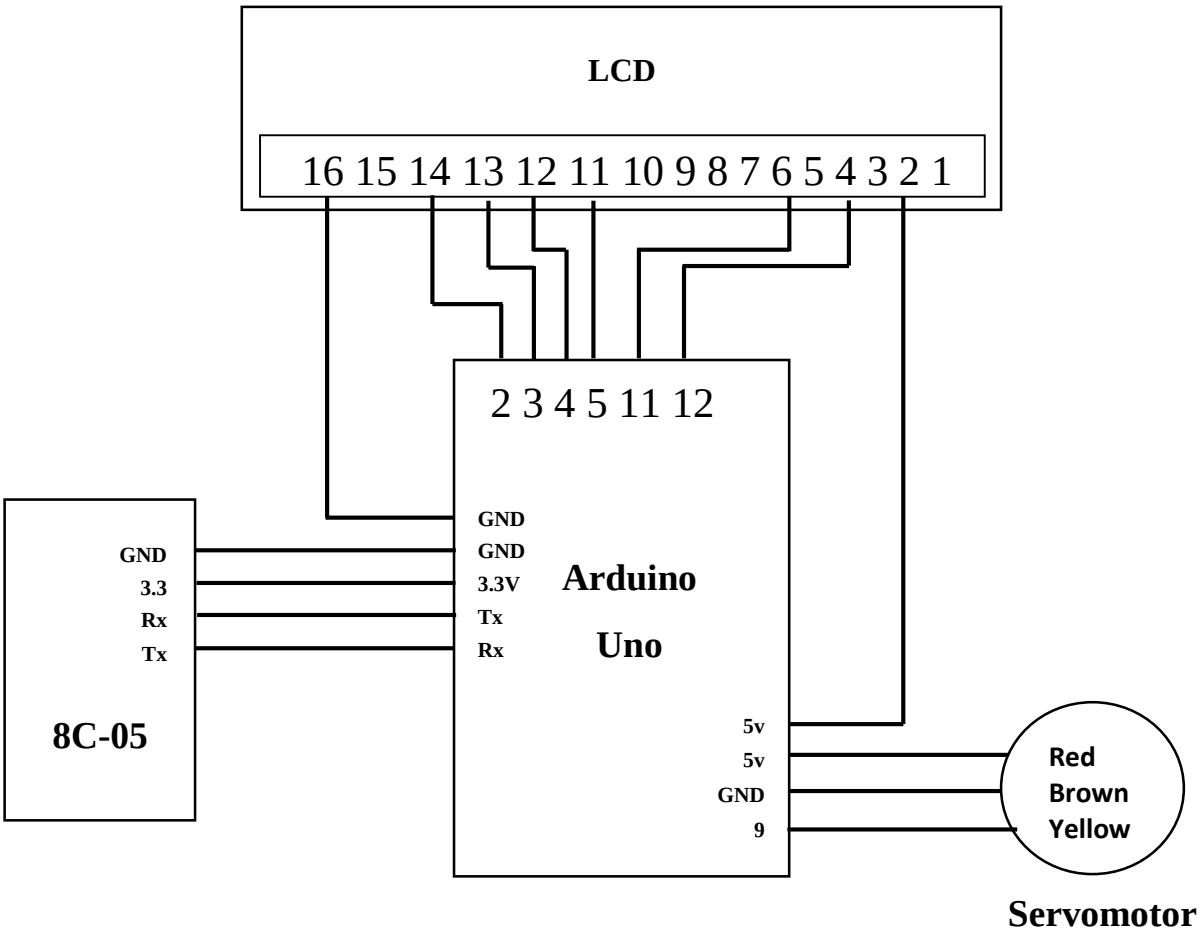
### 5.1 Hardware Details:

| S. No. | Components Used | Quantity |
|---|---|---|
| 1. | Arduino Uno | 1 |
| 2. | Bluetooth Module HC-05 | 1 |
| 3. | Servo Motor | 1 |
| 4. | LCD | 1 |
| 5. | BT Terminal App installed in Smartphone | 1 |
| 6. | LED(Red, Green) | 1,1 |
| 7. | USB Cable | 1 |
| 8. | Connecting Wires | 2m |
| 9. | Bread Board | 1 |
| 10. | Electric Tape | 1 |
| 11. | Potentiometer | 10k |
| 12. | Resistance | 1 |

## 5.2 Block Diagram:



## 5.3 Circuit Diagram



# 6 Hardware Description

## 6.1 Arduino Uno

**Definition:**

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 an AC-to-DC adapter or battery to get started.

**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) |
| SRAM | 2 KB (Atmega328) |
| EEPROM | 1 KB (Atmega328) |
| Clock Speed | 16 MHz |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

**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 centre-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 20V. 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 12V.

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 5V 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 3.3V supply generated by the on-board regulator. Maximum current draw is 50 mA.
**GND**: Ground pins.
**IOREF**: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

**Memory:**

The Atmega328 has 32 KB (with 0.5 KB used for the boot loader). 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 5V. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 k-Ohms. 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, labelled 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 it is 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.

There are a couple of other pins on the board:
**AREF:** Reference voltage for the Analog inputs. Used with analogReference().
**Reset:** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

**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 Software Serial 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. Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board).

The Atmega328 on the Arduino Uno comes pre-burned 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.

We can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

**USB Overcurrent Protection:**

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

**Pin Diagram:**

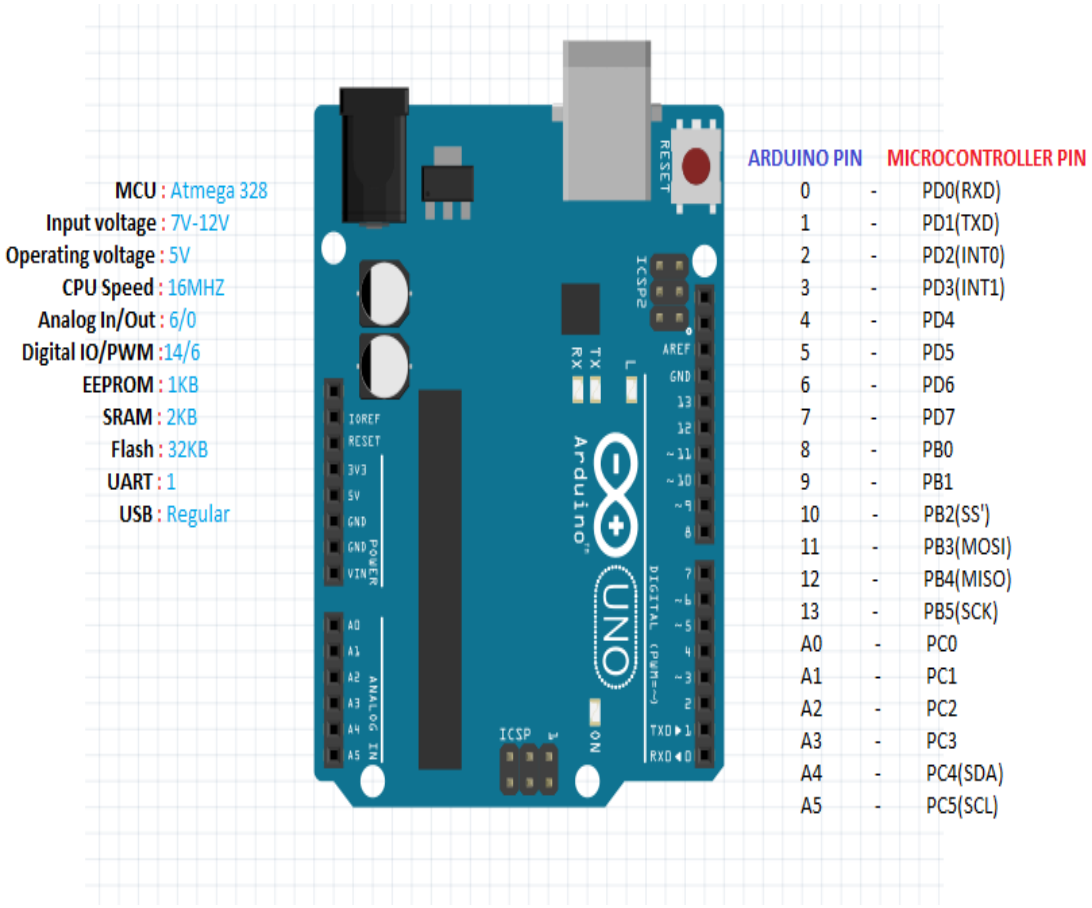



Fig. 8: Pin diagram Arduino Uno

**Back and Front view:**


Fig. 9: Arduino Uno back view

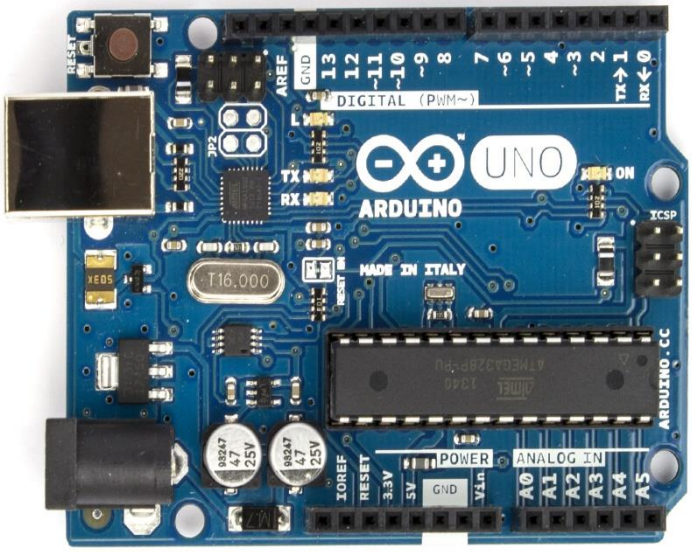



Fig. 10: Arduino Uno front view

## 6.2 Bluetooth Module HC-05

**Description:**

It is a class-2 Bluetooth module with Serial Port Profile, which can configure as either Master or slave. A Drop-in replacement for wired serial connections, transparent usage. You can use it simply for a serial port replacement to establish connection between MCU, PC to your embedded project and etc.

**HC-05 Specification:**

| | |
|---|---|
| Bluetooth protocol: | Bluetooth Specification v2.0+EDR |
| Frequency: | 2.4GHz ISM band |
| Modulation: | GFSK (Gaussian Frequency Shift Keying) |
| Emission power: | ≤4dBm, Class 2 |
| Sensitivity: | ≤-84dBm at 0.1% BER |
| Speed: Asynchronous: | 2.1Mbps (Max) / 160 kbps, Synchronous: 1Mbps/1Mbps |
| Security: | Authentication and encryption |
| Profiles: | Bluetooth serial port |
| Power supply: | +3.3VDC 50mA |
| Working temperature: | -20 ~ +75 C |
| Dimension: | 26.9mm x 13mm x 2.2 mm |

**Pin Diagram and Interfacing with Arduino:**

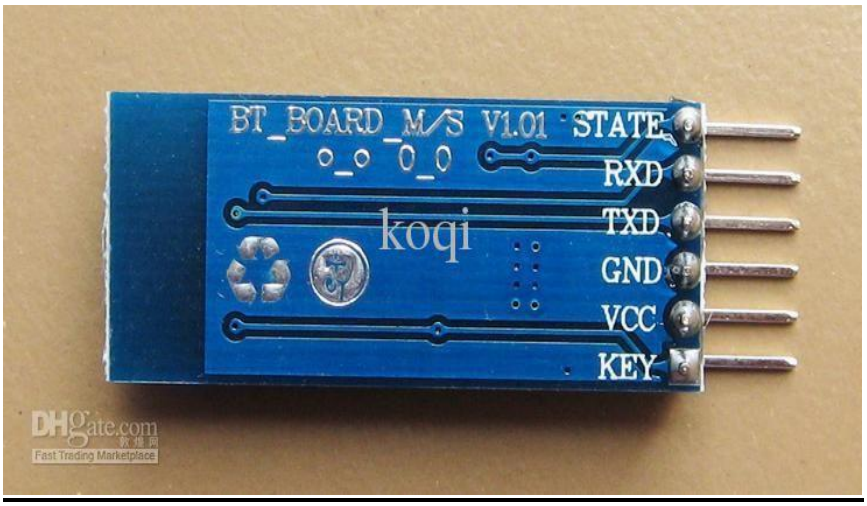



Fig. 11: Pin diagram of HC-05 Bluetooth Module

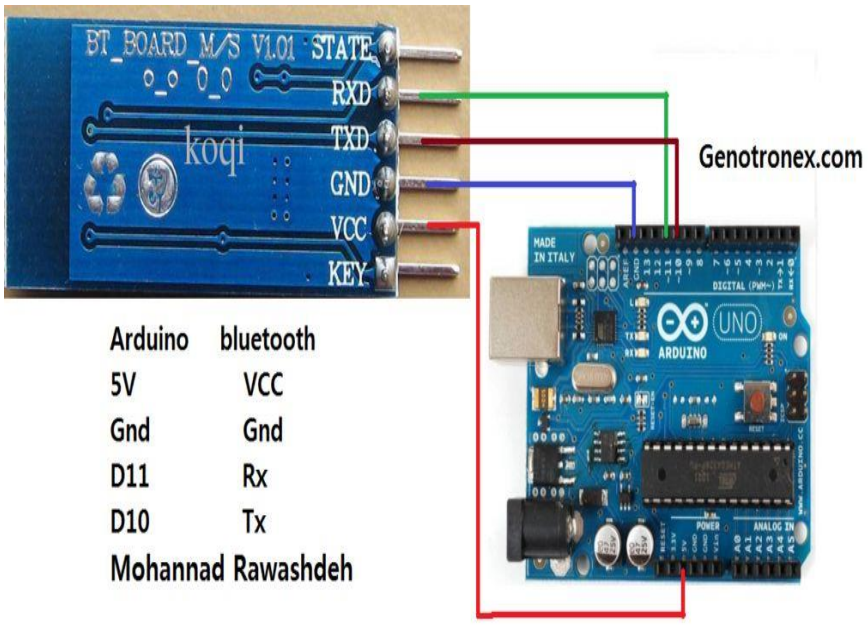



Fig. 12: HC-05 interfacing with Arduino Uno

## 6.3 Servomotor:

**Definition:**

A servomotor is a rotary actuator that allows for precise control of angular position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

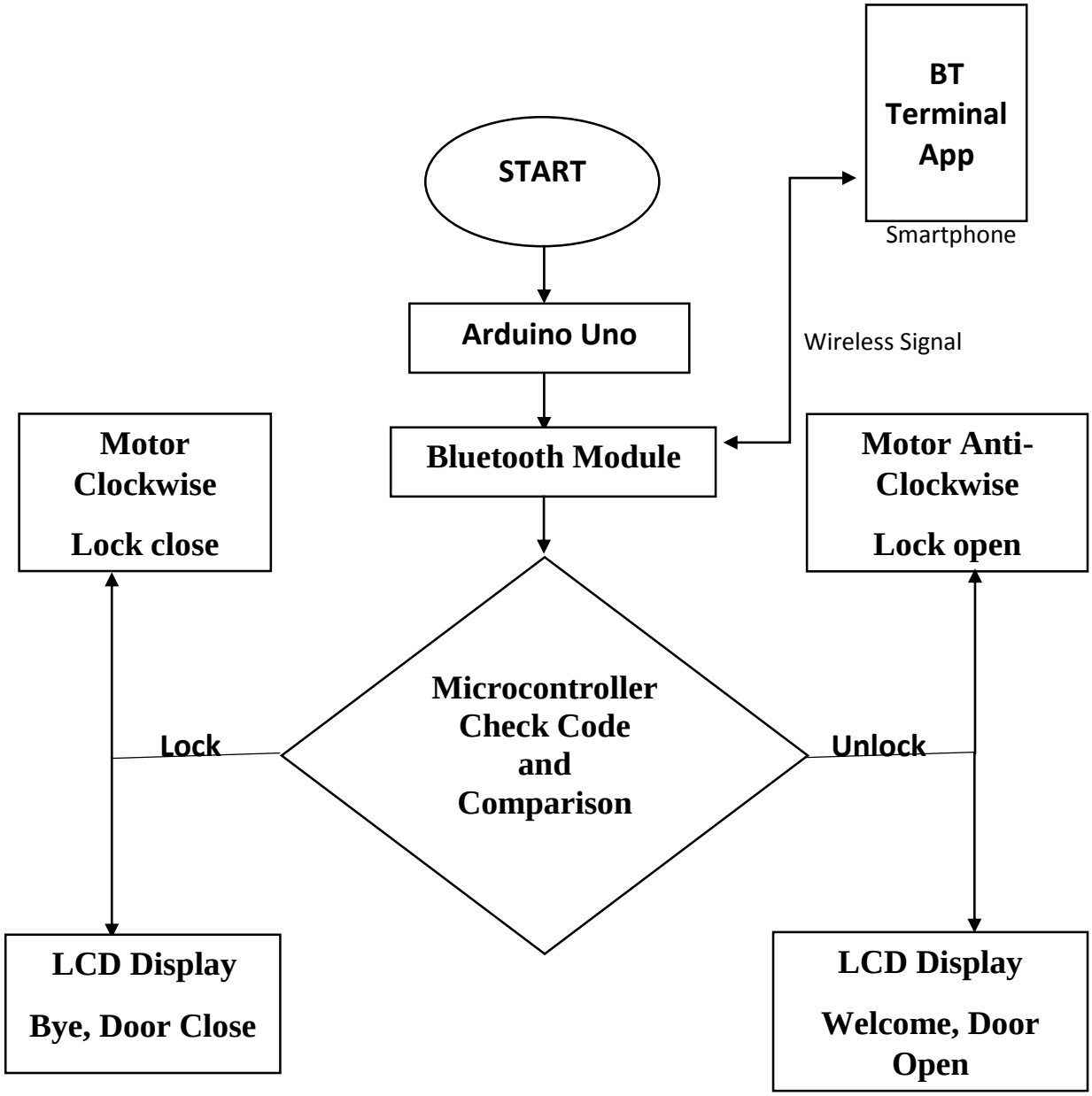
**Basic Information (Tower Pro SG90 – Micro Servo):**

| | |
|---|---|
| Modulation: | Analog |
| Torque: | 4.8V: 25.0 oz-in (1.80 kg-cm) |
| Speed: | 4.8V: 0.10 sec/60° |
| Weight: | 0.32 oz (9.0 g) |
| Dimensions: | Length: 0.91 in (23.1 mm) |
| Width: | 0.48 in (12.2 mm) |
| Height: | 1.14 in (29.0 mm) |
| Motor Type: | 3-pole |
| Gear Type: | Plastic |
| Rotation/Support: | Bushing |


Fig. 13: Tower Pro SG90 Servomotor

# 7 Methodology

## 7.1 Process Chart:



## 7.2 Hardware Setup:

- After the successful compilation of the source code on Arduino 1.6.3 software, upload it on the Arduino Uno board.
- Make all the connection according to the circuit diagram provided.
- Turn ON the HC-05 Bluetooth module.
- Connect the Bluetooth Module with the BT Terminal Application installed in your Smartphone.
- After successful connection, send the respective password from your BT terminal to open or close the lock.
- LCD will display the current status of the lock either ‘OPEN’ or ‘CLOSE’ with respect to the command given from the BT Terminal.
- When the command is to ‘OPEN’ the lock then servomotor will rotate 180 degree anticlockwise and when the command is to ‘CLOSE’ the lock then servomotor will rotate 180 degree clockwise.

### 7.3 Source Code:

```
String inString = “”;    // string to hold input
#include <Servo.h>
#include <LiquidCrystal.h>
Servo myservo;  // create servo object to control a servo
  // twelve servo objects can be created on most boards
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int pos = 0;
void setup() {
 myservo.attach(9);
 lcd.begin(16, 2);
  // Open serial communications and wait for port to open:
 Serial.begin(9600);
 pinMode(13,OUTPUT);
 Serial.println(“Enter pass:”);
 lcd.print(“BLUETOOTH LOCK”);
}
void loop() {
  // Read serial input:
  while (Serial.available() > 0) {
   int inChar = Serial.read();
    if (isDigit(inChar)) {
      // convert the incoming byte to a char
      // and add it to the string:
      inString += (char)inChar;
    }
    // if you get a newline, print the string,
    // then the string’s value:
      if (inChar == ‘f’) {
      if(inString==”1234”)
      {
      digitalWrite(13,1);
```

```
    for(pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
   myservo.write(pos);              // tell servo to go to position in variable ‘pos’
                        // waits 15ms for the servo to reach the position
  }
  Serial.println(“DOOR OPEN”);
  lcd.clear();
  lcd.print(“DOOR OPEN”);
     }
   else if(inString==”4321”)
       {
     digitalWrite(13,0);
     for(pos = 180; pos >=0; pos -= 1) // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
   myservo.write(pos);              // tell servo to go to position in variable ‘pos’
                        // waits 15ms for the servo to reach the position
  }
 Serial.println(“DOOR CLOSE”);
  lcd.clear();
 lcd.print(“DOOR CLOSE”);
     }
     else{
       lcd.clear();
      Serial.println(“WRONG PASS”);
     // clear the string for new input:
  lcd.print(“WRONG PASS”);
  }
       inString = “”;
    }
  }
}
```

### 7.4 Result of the Module Second:

The hardware of the Bluetooth Digital Lock is successfully implemented and it is in working condition.


Fig. 14: Result of Module Second

# 8 Challenges

The following challenges may be faced by the user at the time of using the Bluetooth digital Lock:

**Hardware:** Implementing the hardware by connecting all the components is a hard task. Any loose connection may result in the improper working of it.

**Forgetful:** You may forget the lock code in case when you have changed the code in hurry or rush.

**Keep the PIN code safe and the lock clean:** Only tell the code to people who you trust, as you don't want a code to your property to be local news. When the lock has been used a few too many times, the coating may start to come away or mucky fingerprints may start to occur on the buttons. Keep the lock maintained and clean to stop unwanted people finding out the code.

**Power Failure:** Some digital door locks are powered by electricity, if your house or building has a power failure, then the door lock will not work which restricts you from entering the building.

# 9 References

**Books:**


1. Programming in ANSI C: E BALAGURUSAMY
2. 8051 Microcontroller and Embedded systems: MUHAMMAD ALI MAZIDI and JANICE GILLISPIE MAZIDI


**Websites:**


1. www.atmel.com
2. www.engineersgarage.com
3. www.arduino.cc

[1] http://www.next.gr/uploads/58/LCD-Interfacing-with-8051-using-Keil-C-4-Bit-Mode-Circuit-Diagram.jpg

[2] http://3.bp.blogspot.com/nWa4TEfV2Gg/UjH0cKjiqAI/AA

[3] http://www.learningaboutelectronics.com/images/Arduino-HD44780-circuit-schematic.png

[4] 8051 Microcontroller and Embedded systems: MUHAMMAD ALI MAZIDI and JANICE GILLISPIE MAZIDI

[5] http://hackyourmind.org/public/images/keypad12keys_circuit.png

[6] http://stab-iitb.org/resources/images/e/ee/L293d.jpg

[7] Proteus

[8] http://www.arduino.cc/en/uploads/Main/ArduinoUno.jpg

[9] http://www.arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg

[10] http://www.arduino.cc/en/uploads/Main/ArduinoUno_R3_Back.jpg

[11] http://cdn.instructables.com/F9L/Q8LA/HH2VNS6J/F9LQ8LH2VNS6J.LARGE.jpg

[12] http://www.instructables.com/file/FK9L6ZAHH2VNS6Y

[13] http://www.servodatabase.com/servo/towerpro/sg90

[14] Images of Bluetooth Digital Lock Model