**PROJECT**

**ON**

**“PASSWORD BASED DOOR LOCK SYSTEM”**



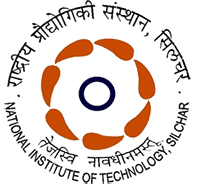
**BY**

**PRACHI MATHUR**

**BRANCH: ELECTRONICS AND COMMUNICATION ENGINEERING**

**SCHOLAR NO: 15-14-081**

**NATIONAL INSTITUTE OF TECHNOLOGY, SILCHAR**



**CONTENTS**

1. Aim
2. Introduction to embedded system
3. Microprocessor and Microcontroller
4. Difference between Microprocessor and Microcontroller
5. Types of microcontrollers
6. AVR microcontroller
   1. History
   2. Architecture
7. Embedded C
   1. Issues using C language
   2. Features
   3. Main header files used
   4. Structure of embedded C
8. Description of Components Used
   1. IC Atmega16
      1. Features
      2. Basis of Architecture
      3. Types of Atmega16 microcontroller
      4. Details of pins on IC
      5. Pin Diagram
   2. LCD
      1. Advantages of LCD
      2. Types of LCD
      3. Series of LCD
      4. Pin diagram
      5. Description of LCD
      6. Few predefined Commands
      7. Interfacing of LCD
   3. Keypad
9. Acknowledgements

**AIM**

The aim of this project is to make a “Password Based Door Lock System” using embedded systems.

Atmega 16 microcontroller of the AVR family is used. LCD is used for display and the keypad is used for inputting the password.

The scope of this project is

* Designing circuit using Atmega16
* Developing embedded C program
* Capturing schematic and simulating using Proteus
* Study of various components used in the design.

**INTRODUCTION TO EMBEDDED SYSTEM**

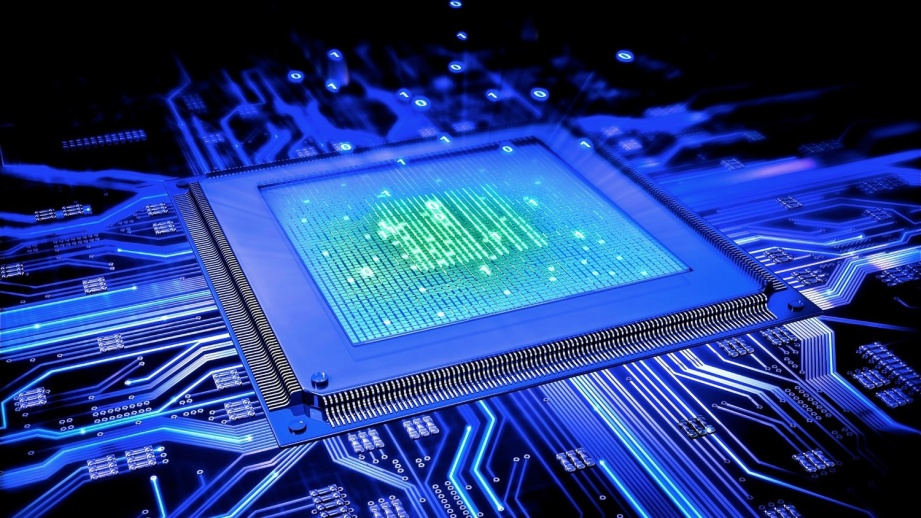
In simple words, it is a device which is used to perform a specific task.  Embedded systems are application specific & single functioned. The programs are executed repeatedly.

Embedded systems are typically designed to meet real time constraints; a real time system reacts to stimuli from the controlled object/operator within the time interval dictated by the environment.

Embedded systems often interact with external world through sensors and hence are typically reactive systems; a reactive system is in continual interaction with the environment and executes at a pace determined by that environment.

Efficiency is of paramount importance for embedded systems. They are optimized for energy, code size, execution time, weight, dimensions and cost.

They generally have minimal or no user interface.



**MICROPROCESSOR**

The microprocessor, also known as the Central Processing Unit (CPU), is the brain of all computers and many household and electronic devices.

A microprocessor is an electronic device that is used by a computer to do its work. It is a central processing unit on a single integrated circuit chip containing millions of very small components including transistors, resistors and diodes that work together.

**MICROCONTROLLER**

Microcontroller can be termed as a single on chip computer which includes number of peripherals like RAM, EEPROM, timers etc. required to perform some predefined and certain specific tasks. E.g. switching the AC off automatically when room temperature drops to a certain defined limit and again turning it ON when temperature rises above the defined limit.

There are a number of popular families of microcontrollers which are used in different applications as per their capabilities and feasibility to perform a desired task. Some of the common microcontrollers are-

1. 8051
2. PIC
3. AVR
4. ARM

This is discussed to understand the basic reason of AVR being a microcontroller and not a microprocessor.

**TYPES OF MICROCONTROLLERS**

**1. 8051 Microcontroller**

* It’s 8-bit microcontroller belonging to the MCS-51 family built by Intel in the year 1979.
* It does not have the property to erase.
* It has 128 byte RAM, 4 KB ROM and 64 KB external ROM.
* It is used for small applications.

**2.PIC Microcontroller**

* Abbreviation of Peripheral interface controller, it is manufactured by Microchip in the year 1985.
* It is 8-bit, 16-bit and 32 bit microcontroller.
* Early models of PIC had read-only memory (ROM) or field-programmable EPROM for program storage, some with provision for erasing memory. All current models use [Flash memory](https://en.wikipedia.org/wiki/Flash_memory) for program storage, and newer models allow the PIC to reprogram itself.
* It is mainly used in home appliances.

**3.AVR Microcontroller**

* It was manufactured by Atmel in the year 1996.
* It is an 8-bit microcontroller which means it can transmit and receive 8-bit data.AVR family controllers have registers based architecture which means that both the operands for an operation are stored in a register.
* It is an 8-bit microcontroller belonging to the family of Reduced Instruction Set Computer(RISC). In RISC architecture the instruction set of the computer are not only fewer but are simpler and faster in operation.
* It is mainly used for robotics.

**4.ARM**

* Arm also called Advanced RISC Machine was manufactured in 1985 by Philips.
* It is 32 bit and 64 bit microprocessor.
* It is used for mobile applications.

**For this project AVR is being used. 8051, PIC being old families has not been considered for the project. ARM is the next generation of microcontrollers which is currently out of scope of this project.**

**DIFFERENCE BETWEEN MICROPROCESSOR AND MICROCONTROLLER**

|  |  |
| --- | --- |
| Microprocessor | Microcontroller |
| All peripherals are connected externally. | All the peripherals like RAM, ROM, Counter/Timer, Interrupt, Input/output Ports, ADC, I2C, SPI etc are all on a single chip. |
| Large in size and expensive.  Since they find applications where tasks are unspecific and input and output relationship is not defined thus they need high amount of resources like RAM, ROM, I/O ports etc.  . | Small in size and cheaper.  Since the applications are very specific, they need small resources like RAM, ROM, I/O ports etc and hence can be embedded on a single chip. This in turn reduces the size and the cost. |
| Less portable. | More portable. |
| More complex. | Fewer complexes. |
| It is used for a general purpose task. | It is used for performing a specific task. |
| Very high clock speed about 1 GHz. | Slow clock speed about 30-50 MHz. |

**AVR MICROCONTROLLER**

**History:**

AVR was developed in the year 1996 by Atmel Corporation. The architecture of AVR was developed by Alf-Egil and Vegard Wollen. AVR derives its name from its developers and stands for Advanced Virtual RISC machine.

AVR microcontrollers are available in three categories-

1. TinyAVR-less memory, small size, suitable for small applications.
2. MegaAVR- these are most popular ones having good amount of memory (upto 256 KB), higher number of inbuilt peripherals and suitable for moderate to complex applications.
3. XmegaAVR- used commercially for complex applications which require large program memory and high speed. The following tables compares the above mentioned AVR series of microcontrollers-

|  |  |  |  |
| --- | --- | --- | --- |
| Series Name | Pins | Flash memory | Special features |
| Tiny AVR | 6-32 | .5-8 KB | Small in size |
| Mega AVR | 28-100 | 4-256 KB | Extended peripherals |
| XMEGA AVR | 44-100 | 16-384 KB | DMA, Event System included |

**Architecture:**

The AVR Microcontrollers are based on the advanced RISC architecture consist of 32×8 bit general purpose work registers.

Within one single clock cycle, AVR can take input from two general purpose registers and put them to ALU for carrying requested operation, and transfer back the result to an arbitrary register.

AVR follows ‘Harvard Architecture format’ in which the processor is equipped with separate memories and buses for Program and the Data information. Here

while an instruction is being executed, the next instruction is pre-fetched from the program memory.

CPU

PROGRAM MEMORY

DATA MEMORY

Since AVR can perform single cycle execution, the higher is the operating frequency of the controller the higher is the processing speed.

**EMBEDDED C**

Embedded C is a generic term given to a programming language written in C, which is associated with particular hardware architecture. Embedded C is an extension to the C language with some additional header files. These header files may change from controller to controller.

Embedded C is the most popular embedded software language in the world. Most embedded software is written in Embedded C.

**Issues in using C language:**

1. Big and inefficient code generation
2. Fat code for the standard IO routines (printf, scanf, strcpy, etc…)
3. Data declaration in RAM and ROM
4. Difficulty writing Interrupt Service Routines
5. Compiler optimizations

**Features of Embedded C:**

1. Embedded C, unlike low level assembly languages, is portable.
2. It can run on a wide variety of processors, regardless of their architecture.
3. Unlike high level languages, Embedded C requires less resources to run and isn’t as complex.
4. Limited ROM
5. Limited RAM
6. Hardware oriented programming
7. Critical timing (Interrupt Service Routines, tasks).
8. Many different pointer kinds (far / near / rom / uni / paged).
9. Special keywords and tokens (@, interrupt, tiny)
10. Control structures same as in C.
11. Scanf() and printf() are removed as inputs are scanned from sensors and output are given to the ports.

**Main Header files used:**

1. #include<avr/io.h>

This contains the appropriate input output definition.

1. #include<util/delay.h>

This introduces delay of milliseconds or nanoseconds.

**Structure of C program:**

#include<avr/io.h>

#include<util/delay.h>

Int main()

{

While(1)

{

Code

}

Return 0;

}

**DESCRIPTION OF COMPONENTS USED**

1. IC Atmega 16
2. LCD
3. Keypad

* **IC Atmega 16**

We will be working on Atmega 16 microcontroller, which is a 40-pin IC and belongs to the mega AVR category of the AVR family.



**Features:**

Some of the major features of Atmega16 are:

* 16 KB of flash memory
* 1 KB of Static RAM
* 512 bytes of EEPROM
* Available in 40-pin DIP
* 8-Channel 10-bit ADC
* Two 8 bit timers/counters
* One 16 bit timer/counter
* Serial USART
* 131 powerful instructions executed in one machine cycle
* Maximum frequency of 16 MHz

**Basis of Architecture:**

Following points explain the building blocks of Atmega 16 architecture:

* I/O Ports: Atmega 16 has four (PORTA, PORTB, PORTC and PORTD) 8-bit input/output ports.
* Internal calibrated oscillator: Atmega16 is equipped with an internal calibrated oscillator for driving the clock. By default it operates at a frequency of 1 MHz. The maximum frequency of the internal calibrated oscillator is 8 MHz. Atmega16 can be operated using an external crystal oscillator with a maximum frequency of 16 MHz.
* ADC Interface: Atmega16is equipped with an 8 channel ADC with a resolution of 10 bits.
* Timer/Counters: Atmega16 consists of 2 8 bit timers and one 16 bit timer.
* Watchdog Timer: It is present with internal oscillator. Watchdog timer continuously monitors and resets the controller if the code gets stuck at any execution action for more than definite time interval.
* Interrupts : Atmega 16 consists of 21 interrupt sources out of which four are external. The remaining are internal interrupts which support the peripherals like USART,ADC, Timers etc.
* USART(Universal Synchronous and Asynchronous Receiver and Transmitter): It is used for interfacing with external device which are capable of serial communication.

**Types of Atmega16 microcontroller:**

1. Atmega16: Operating frequency range is 0-16 MHz.
2. Atmega16L: Operating frequency range 0-8 MHz.

**Details of Pins on IC:**

1. It has four ports- PORTA, PORTB, PORTC and PORTD.
   1. All ports are capable of performing dual functions.
   2. They have read-modify-write functionality.
2. Each port pin has three registers-
   1. DDRx(x can be A, B, C and D) - data direction register to specify whether a particular port pin will act as input or output pin.

If DDRx is configured as logic high it acts as output

Eg. DDRA=0xff

If DDRx is configured as logic low it acts as input

Eg. DDRA=0x00

* 1. PORTx(x can be A, B, C and D) – port output
  2. PINx(x can be A, B, C and D) – port input

If PORTx is written logic one when the pin is configured as an output pin the port pin is driven high.

If PORTx is written logic zero when the pin is configured as an output pin, the port pin is driven low.

Eg. DDRB=0xff;

PORTB=0xff; //output high

//given delay

PORTB=0x00; //output low

**Pin Diagram:**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

PB0

PB1

PB2

PB3

PB4

PB5

PB6

PB7

RESET

VCC

GROUND

XTAL2

XTAL1

PD0

PD1

PD2

PD3

PD4

PD5

PD6

40

39

38

37

36

35

34

33

32

31

30

29

28

27

26

25

24

23

22

21

PA0

PA1

PA2

PA3

PA4

PA5

PA6

PA7

VREF

GROUND

VCC

PC7

PC6

PC5

PC4

PC3

PC2

PC1

PC0

PD7

* **LCD**

Liquid Crystal Display (LCD) consists of rod-shaped tiny molecules sandwiched between a flat piece of glass and an opaque substrate. These rod-shaped molecules in between the plates align into two different physical positions based on the electric charge applied to them. When electric charge is applied they align to block the light entering through them, where as when no-charge is applied they become transparent.

Light passing through makes the desired images appear.

It consists of an 8-bit microcontroller.

It is an embedded product i.e. it has RAM, ROM etc.

LCD have single line packaging (pins on only one side).



**Advantages of LCDs:**· Consumes less power and generates less heat.  
· Saves lot of space compared picture tubes due to LCD's flatness.  
· Due to less weight and flatness LCDs are highly portable.  
· No flicker and fewer screens glare in LCDs to reduce eyestrain.

**Types of LCD:**

* Graphical:  It has pixels in rows and columns. By energizing set of pixels any character can be displayed.
* Text:  is an electronic [alphanumeric](https://en.wikipedia.org/wiki/Alphanumeric) [display device](https://en.wikipedia.org/wiki/Display_device) that is mainly or only capable of showing [text](https://en.wikipedia.org/wiki/Plain_text), or extremely limited [graphic characters](https://en.wikipedia.org/wiki/Graphic_character).

**Series of LCD:**

* 16×1, 16×2, 16×4(1 ,2 and 4 lines having 16 characters respectively)
* 20×1, 20×2, 20×4(1 ,2 and 4 lines having 20 characters respectively)
* 40×1, 40×2(1 and 2 lines having 40 characters respectively)

**Pin Diagram:**

;

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

GROUND VCC VEE RS RW EN D0 D1 D2 D3 D4 D5 D6 D7 VCC GROUND

**Description of Pins:**

* LCD has two registers:

1. Command Register (To control the data. Eg. Clearing the screen, spacing etc)
2. Data Register (To write the data)

|  |  |  |
| --- | --- | --- |
| Pin no | Name | Purpose |
| 1 | GROUND | Ground voltage |
| 2 | VCC | +5 volts |
| 3 | VEE | Variable resistance to control the contrast of LCD |
| 4 | RS | Stands for “Register Select” control line.   * When RS is low (0), the data is to be treated as a command or special instruction (such as clear screen, position cursor, etc.). * When RS is high (1), the data being sent is text data which should be displayed on the screen. |
| 5. | RW | Stands for “Read/Write” control line.   * When RW is low (0), the information on the data bus is being written to the LCD. * When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction ("Get LCD status") is a read command. All others are write commands--so RW will almost always be low. |
| 6 | EN | Stands for “enable” control line.’   * This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines. * When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet, and end by bringing it low (0) again. |
| 7. | D0 | Data bit 0(LSB) |
| 8. | D1 | Data bit 1 |
| 9. | D2 | Data bit 2 |
| 10. | D3 | Data bit 3 |
| 11. | D4 | Data bit 4 |
| 12. | D5 | Data bit 5 |
| 13. | D6 | Data bit 6 |
| 14. | D7 | Data bit 7 |
| 15. | VCC | Used for background LED. |
| 16. | GROUND | Ground voltage. |

**Few Predefined commands of LCD:**

1. 0x01= Clear Display Screen
2. 0x0E=Display on and cursor blinking
3. 0x0F=Display on and cursor static
4. 0x80= Force cursor to the beginning of the first line
5. 0xC0= Force cursor to the beginning of the second line
6. 0x90= Force cursor to the beginning of the third line
7. 0xD0= Force cursor to the beginning of the fourth line
8. 0x02=Set 4-bit mode
9. 0x28=Select 5×7 matrix for 4-bit mode
10. 0x38= Select 5×7 matrix for 8-bit mode
11. 0x06= Increment Cursor
12. 0x04= Decrement Cursor
13. 0x18= Move entire Display left
14. 0x1C= Move entire display to right.

**Interfacing of LCD:**

**Modes of LCD:**

There are two modes of interfacing the LCD with IC atmega16. These two modes differ in the way the data and commands are send to the LCD.

1. 8-bit mode-

In 8 Bit mode character data (as 8 bit ASCII) and LCD command are sent through the data lines D0 to D7.

1. 4-bit mode-

* In 4 Bit mode uses only 4 data lines D4 to D7.
* In this 8 bit data is divided into two parts and are sent sequentially through the data lines.
* The idea of 4 bit communication is introduced to save pins of microcontroller.
* 4 bit communication is bit slower than 8 bit but this speed difference has no significance as LCDs are slow speed devices. Thus 4 bit mode data transfer is most commonly used.

**1.8-bit mode:**

1. Basic commands-0x038, 0x0E, 0x01, 0x80, 0x06
2. Functions in 8 bit mode
   1. Command function

void cmd()

{

PORTA=0B00000100;

\_delay\_ms(200);

PORTA=0x00000000;

}

* 1. Display function

void display()

{

PORTA=0b00000101;

\_delay\_ms(200);

PORTA=0b00000001;

}

**2.4-bit mode:**

1. Basic Commands: 0x02,0x28,0x0E,0x01,0x80,0x06
2. Functions in 4-bit mode
   1. Command function

Void cmd()

{

PORTB&=~(1<<PB0);

PORTB&=~(1<<PB1);

PORTB&=|(1<<PB2);

\_delay\_ms(200);

PORTB&=~(1<<PB2);

}

* 1. Display function

void display()

{

PORTB&=|(1<<PB0);

PORTB&=~(1<<PB1);

PORTB&=|(1<<PB2);

\_delay\_ms(200);

PORTB&=~(1<<PB2);

}

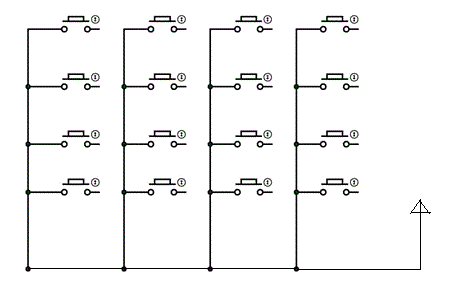
* **KEYPAD**

Keypad is an input device which is used to send data to the microcontroller.

In this project we are going to use a 4×4 matrix.

It is of two types:

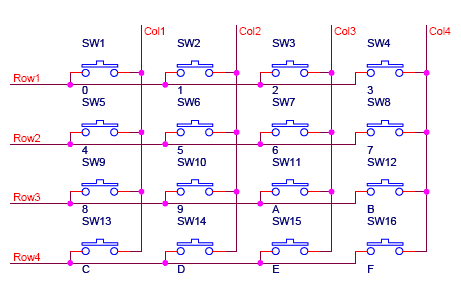
1. Non-Multiplexing keypad: It requires same number of input/output ports as that of keys on the keypad



VCC

1. Multiplexing keypad: it reduces the number of pins. Eg. A 6 x 2 matrix, this would require only eight (6+2) I/O pins, or if you altered the approach to a 3 x 4 you would be down to seven I/O pins.

PORTC4 PORTC5 PORTC6 PORTC7



PORTC33

PORTC2

PORTC1

PORTC0

**CODING**

#include <avr/io.h>

#include<util/delay.h>

//functions

void lcdstring( char \*p)

{

while(\*p)

{

PORTB=(\*p);

display();

p++;

}

}

//command function of LCD

void cmd()

{

PORTA=0B00000100;

\_delay\_ms(200);

PORTA=0B00000000;

}

//display function of LCD

void display()

{

PORTA=0B00000101;

\_delay\_ms(800);

PORTA=0B00000001;

}

//command array

unsigned char arr1[]={0x38,0x0e,0x01,0x80,0x06};

unsigned char arr2[]={"Enter new pwd"};

unsigned char ar5[]={"Password based"};

unsigned char arr5[]={"Door Lock System"};

unsigned char arr6[]={"correct pwd"};

unsigned char arr7[]={"wrong pwd"};

unsigned char arr8[]={"enter password"};

//array to store new password

unsigned char e[6];

//array to store the password to be verified

unsigned char e1[6];

int main(void)

{

int i,j,k=0,l=0;

//PORTA & PORTB made output ports

//PORTC is made the input port

DDRB=0xff;

DDRA=0xff;

DDRC=0x00;

//to send commands to the LCD

for(i=0;i<=4;i++)

{

PORTB=arr1[i];

cmd();

}

//to display to the LCD

for(j=0;j<1;j++)

{

for(i=0;i<=13;i++)

{

PORTB=ar5[i];

display();

}

PORTB=0xc0;

cmd();

for(i=0;i<=15;i++)

{

PORTB=arr5[i];

display();

}

}

\_delay\_ms(2000);

PORTB=0x01;

cmd();

//to display text on the LCD

for(i=0;i<=13;i++)

{

PORTB=arr2[i];

display();

}

PORTB=0xc0;

cmd();

//to enter characters from the keypad

while(k<=4)

{

if (PINC=0x01)

{

if(PINC & 0X10)

{

PORTB='A';

e[k]=PORTB;

\_delay\_ms(500);

//to display characters as asterisks

lcdstring("\*");

k++;

}

if(PINC & 0X20)

{

PORTB='B';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X40)

{

PORTB='C';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X80)

{

PORTB='D';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

}

if (PINC=0x02)

{

if(PINC & 0X10)

{

PORTB='E';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X20)

{

PORTB='F';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X40)

{

PORTB='G';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X80)

{

PORTB='H';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

}

if (PINC=0x04)

{

if(PINC & 0X10)

{

PORTB='I';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X20)

{

PORTB='J';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X40)

{

PORTB='K';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X80)

{

PORTB='L';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

}

if (PINC=0x08)

{

if(PINC & 0X10)

{

PORTB='M';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X20)

{

PORTB='N';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X40)

{

PORTB='O';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

if(PINC & 0X80)

{

PORTB='P';

e[k]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

k++;

}

e[k]='\0';

}

}

PORTB=0x01;

cmd();

PORTB=0x80;

cmd();

for(i=0;i<=13;i++)

{

PORTB=arr8[i];

display();

}

PORTB=0xc0;

cmd();

while(l<=4)

{

if (PINC=0x01)

{

if(PINC & 0X10)

{

PORTB='A';

e1[l]=PORTB;

\_delay\_ms(500);

//to display characters as asterisks

lcdstring("\*");

l++;

}

if(PINC & 0X20)

{

PORTB='B';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X40)

{

PORTB='C';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X80)

{

PORTB='D';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

}

if (PINC=0x02)

{

if(PINC & 0X10)

{

PORTB='E';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X20)

{

PORTB='F';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X40)

{

PORTB='G';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X80)

{

PORTB='H';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

}

if (PINC=0x04)

{

if(PINC & 0X10)

{

PORTB='I';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X20)

{

PORTB='J';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X40)

{

PORTB='K';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X80)

{

PORTB='L';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

}

if (PINC=0x08)

{

if(PINC & 0X10)

{

PORTB='M';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X20)

{

PORTB='N';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X40)

{

PORTB='O';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

if(PINC & 0X80)

{

PORTB='P';

e1[l]=PORTB;

\_delay\_ms(500);

lcdstring("\*");

l++;

}

e1[l]='\0';

}

}

PORTB=0x01;

cmd();

PORTB=0x80;

cmd();

//to verify whether the password entered matches the inbuilt password

if(((strncmp(e,e1,5))==0))

{

for(i=0;i<=10;i++)

{

PORTB=arr6[i];

display();

}

}

else

{

for(i=0;i<=8;i++)

{

PORTB=arr7[i];

display();

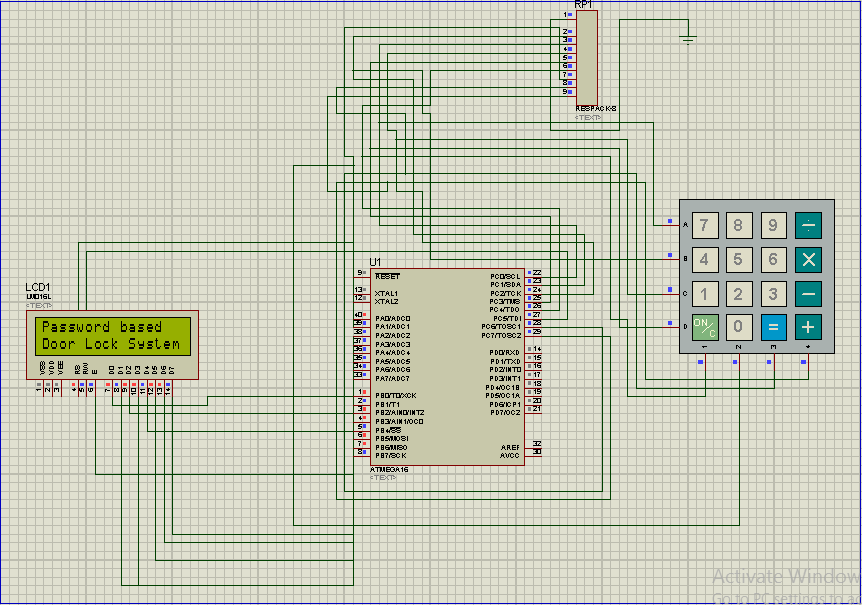
}

}

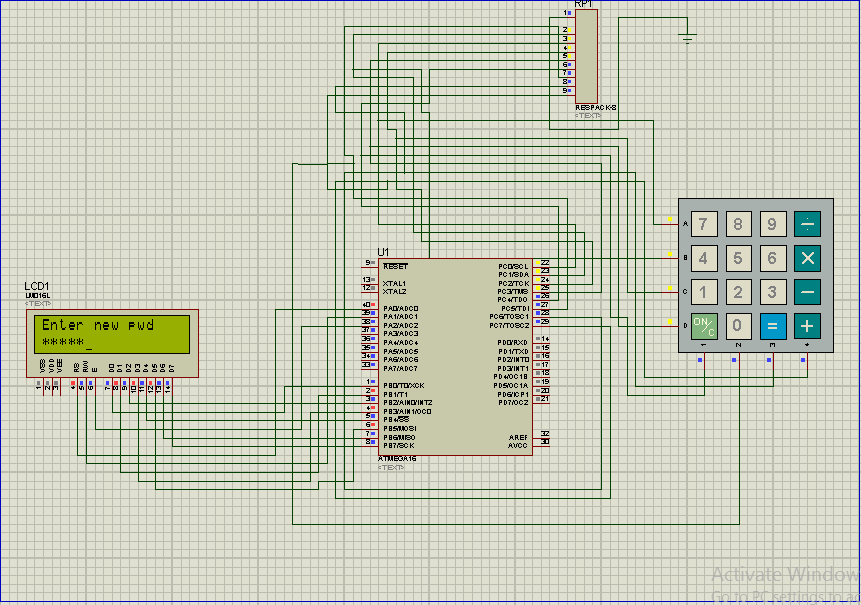
}

**OUTPUT**

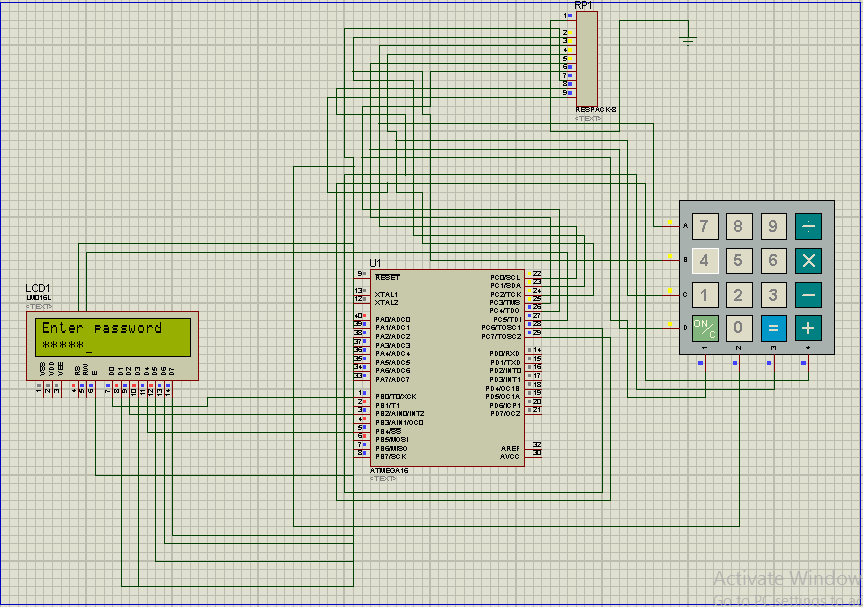
Connections:



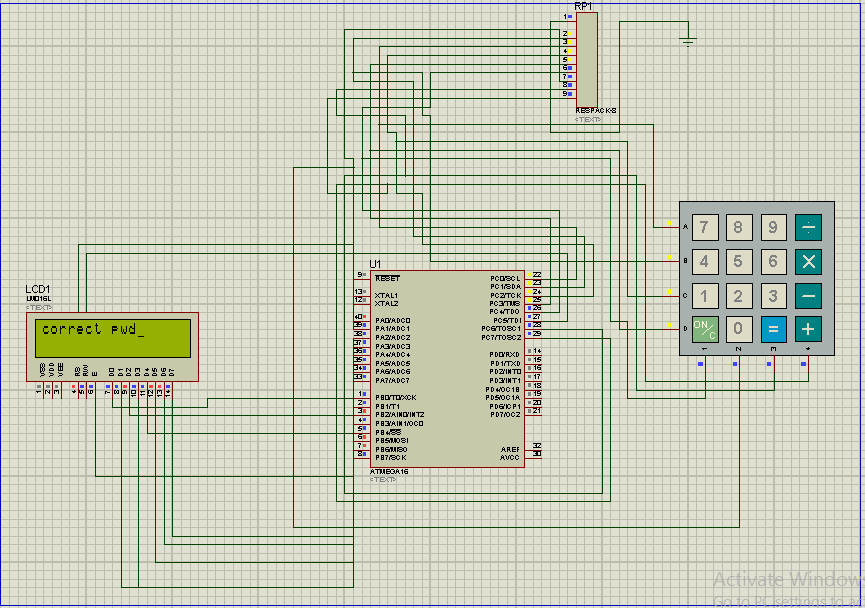
When a new password is entered:



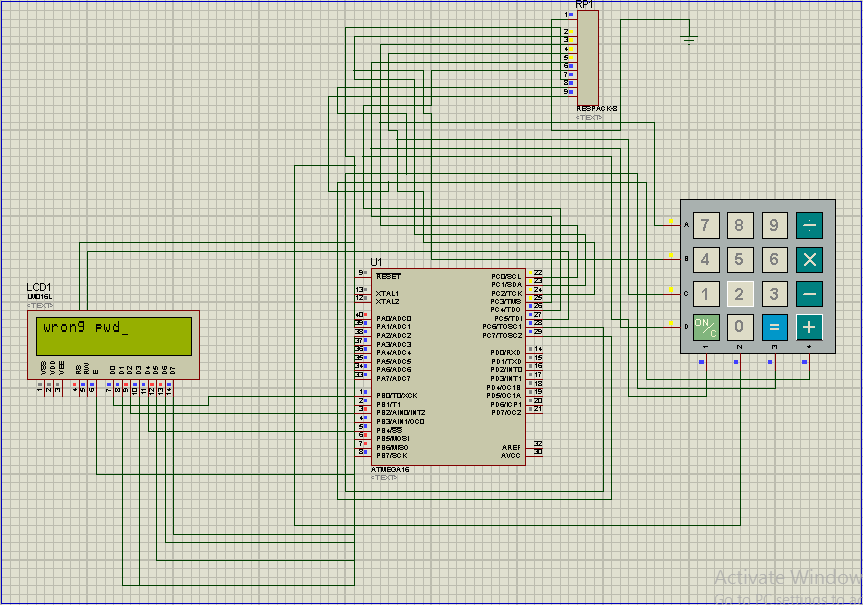
When password is entered to verify the previously entered password:



When correct password is entered:



When wrong password is entered:



**ACKNOWLEDGEMENT**

I wish to express my sincere gratitude to Cetpa Infotech Pvt. Ltd. for providing me the opportunity to do the project titled ‘PASSWORD BASED DOOR LOCK SYSTEM’.

I am very thankful to Ms. Medhavi for her valuable guidance and suggestions at various stages of my project and training period.

I would also like to thank my parents for their constant support and encouragement to make this project a success.