

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING



LAB MANUAL

SUBJECT CODE : ET3491

SUBJECT NAME : EMBEDDED SYSTEMS AND IOT DESIGN

LABORATORY

YEAR / SEM : 03 / 06

REGULATION : 2021

COURSE COORDINATOR :

COURSE FACULTY :

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE VISION	COLLEGE MISSION	DEPARTMENT VISION	DEPARTMENT MISSION
To be an academic institute of continuous excellence towards education and research in rural regime and provide service to nation in terms of nurturing potentially higher social, ethical and engineering companion graduands.	<p>To foster and promote technically competent graduands by imparting the state of art Engineering education in rural regime.</p> <p>To enunciate research assisted scientific learning by dissemination of knowledge towards science, agriculture, industry and national security.</p>	To promote Ethical and Innovative Electronics and communication engineers through excellence in teaching, training and research to contribute to the advancement of the rural society and mankind.	<p>To impart high quality technical education and exposure to recent trends in the industry, to ensure that the students are moulded into competent Electronics and communication engineers.</p> <p>To inculcate research capabilities and exemplary professional conduct to lead and to use technology in agriculture, industry and national security for the progress of our country.</p>

06- Semester - ELECTRONICS AND COMMUNICATION ENGINEERING

ET3491 - EMBEDDED SYSTEMS AND IOT DESIGN

LABORATORY MANUAL

PREPARED BY

APPROVED BY

**Assistant Professor
Department of Electronics and
Communication Engineering**

Revision no.

Date :

COURSE OBJECTIVES :

- Learn the architecture and features of 8051.
- Study the design process of an embedded system.
- Understand the real – time processing in an embedded system.
- Learn the architecture and design flow of IoT.
- Build an IoT based system.

PRACTICAL EXERCISES**30 PERIODS****Experiments using 8051:**

1. Programming Arithmetic and Logical Operations in 8051.
2. Generation of Square waveform using 8051.
3. Programming using On – Chip ports in 8051.
4. Programming using Serial Ports in 8051.
5. Design of a Digital Clock using Timers/Counters in 8051.

Experiments using ARM:

1. Interfacing ADC and DAC.
2. Blinking of LEDs and LCD.
3. Interfacing keyboard and Stepper Motor.

Miniprojects for IoT :

1. Garbage Segregator and Bin Level Indicator
2. Image Processing based Fire Detection
3. Vehicle Number Plate Detection
4. Smart Lock System

COURSE OUTCOMES:

- Explain the architecture and features of 8051.
- Develop a model of an embedded system.
- List the concepts of real time operating systems.
- Learn the architecture and protocols of IoT.
- Design an IoT based system for any application.

EXPERIMENTS USING 8051

EX.NO:		PROGRAMMING ARITHMETIC AND LOGICAL OPERATIONS IN 8051
DATE :		

Aim :

To write an assembly language program to perform the Arithmetic and Logical Operations in 8051.

Apparatus & Software Required :

1. Personal Computer
2. Keilµ Vision5 software.

Theory:

In 8051 assembly language programming, arithmetic and logical operations are fundamental for manipulating data and controlling program flow. Instructions like ADD, SUB, AND, OR, XOR, and NOT are used to perform arithmetic and logical operations on data stored in registers or memory locations. These operations are crucial for tasks such as arithmetic calculations, bitwise manipulation, and decision making in embedded systems. Mastery of these operations enables efficient programming for various applications in the 8051 microcontroller, enhancing its versatility and functionality.

Procedure :

1. Follow the steps to create a New project.
2. Type the below code and save it with the name (anyname.asm).
3. Follow the steps to create a New Project to compile and build the program.
4. To verify the output in the terminal.

Program :

ORG 0H ; Start of program memory

MOV A, #0AH ; Load accumulator A with value 0AH (hexadecimal)

MOV B, #05H ; Load register B with value 05H (hexadecimal)

ADD A, B ; Addition operation: $A = A + B$

SUBB A, B ; Subtraction operation: $A = A - B$

MUL AB ; Multiplication operation: $A = A * B$

DIV AB ; Division operation: $A = A / B$

ANL A, B ; Logical AND operation: $A = A \text{ AND } B$

ORL A, B ; Logical OR operation: $A = A \text{ OR } B$

XRL A, B ; Logical XOR operation: $A = A \text{ XOR } B$

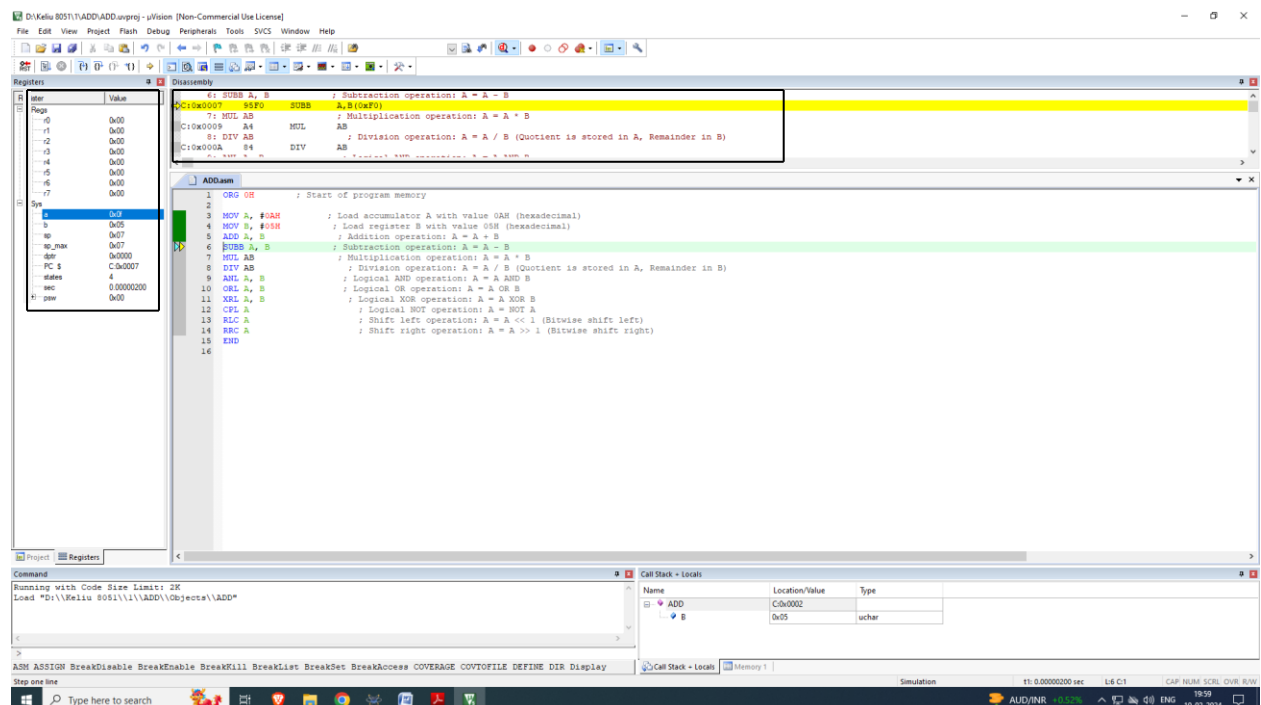
CPL A ; Logical NOT operation: $A = \text{NOT } A$

RLC A ; Shift left operation: $A = A \ll 1$ (Bitwise shift left)

RRC A ; Shift right operation: $A = A \gg 1$ (Bitwise shift right)

END ; End of program

Output :



Result :

The 8051 microcontroller successfully executed arithmetic and logical operations, showcasing its capability to perform mathematical computations and logical evaluations accurately and efficiently.

EX.NO:		GENERATION OF SQUARE WAVEFORM USING 8051
DATE :		

Aim :

To write an C language program to generate a square wave using 8051.

Apparatus & Software Required :

3. Personal Computer
4. Keilµ Vision5 software.

Theory:

Generating a square waveform with 8051 involves toggling an output pin at a certain frequency, achieved by alternately setting and clearing the pin within a specified time interval. This is typically accomplished by implementing a delay loop to control the on-off timing of the pin, thus creating a square wave with equal high and low durations.

Procedure :

1. Follow the steps to create a New project.
2. Type the below code and save it with the name (anyname.c).
3. Follow the steps to create a New Project to compile and build the program.
4. To verify the output in the Logic Analyzer.

Program :

```
#include <REG51xD2.H>
```

```
void delay(unsigned int x)
```

```
/* delay routine */
```

```
{
```

```
for (;x>0;x--);
```

```
}
```

```
main ()
```

```
{
```

```
unsigned char on = 0xFF,Off=0x00;
```



```

P0 = on;

delay(100);

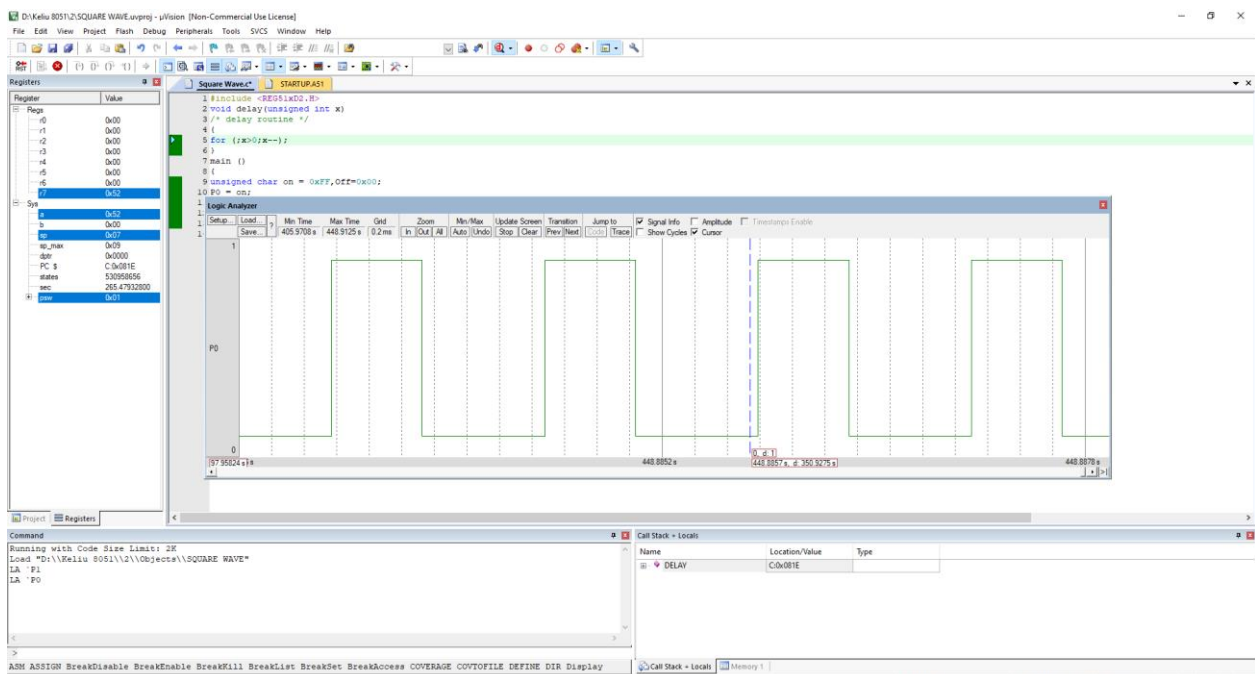
P0 = Off;

delay(100);

}

```

Output :



Result:

The 8051 microcontroller successfully generated a square wave with a defined frequency and duty cycle, observed through a logic analyzer.

EX.NO:		PROGRAMMING USING ON – CHIP PORTS IN 8051
DATE :		

Aim :

To write an ALP to toggle the content of port 0 continuously using timer delay in between.

Apparatus & Software Required :

1. Personal Computer
2. Keilu Vision5 software.

Theory:

Programming on-chip ports in the 8051 microcontroller involves utilizing specific instructions to manipulate the data stored in the ports registers, thereby controlling the input and output operations. By configuring the port registers, the microcontroller can interact with various peripheral devices connected to the ports, enabling tasks such as control in embedded systems.

Procedure :

1. Follow the steps to create a New project.
2. Type the below code and save it with the name (anyname.asm).
3. Follow the steps to create a New Project to compile and build the program.
4. Open the port 1 terminal.
5. To run the data from the registers.

Program :

ORG 0000H

MOV A, #01H ; Load accumulator A with value 01H (binary 0000 0001)

HERE: MOV P1,A ; Move the content of accumulator A to port P1

ACALL DELAY ; Call the delay subroutine

RL A ; Rotate left accumulator A through carry flag

SJMP HERE ; Jump back to HERE label to repeat the process

DELAY:

MOV R2,#255 ; Load register R2 with value 255 for delay loop

LABEL:

PUSH ACC ; Push the content of accumulator to the stack

POP ACC ; Pop the content of accumulator from the stack

NOP ; No operation (1 cycle delay)

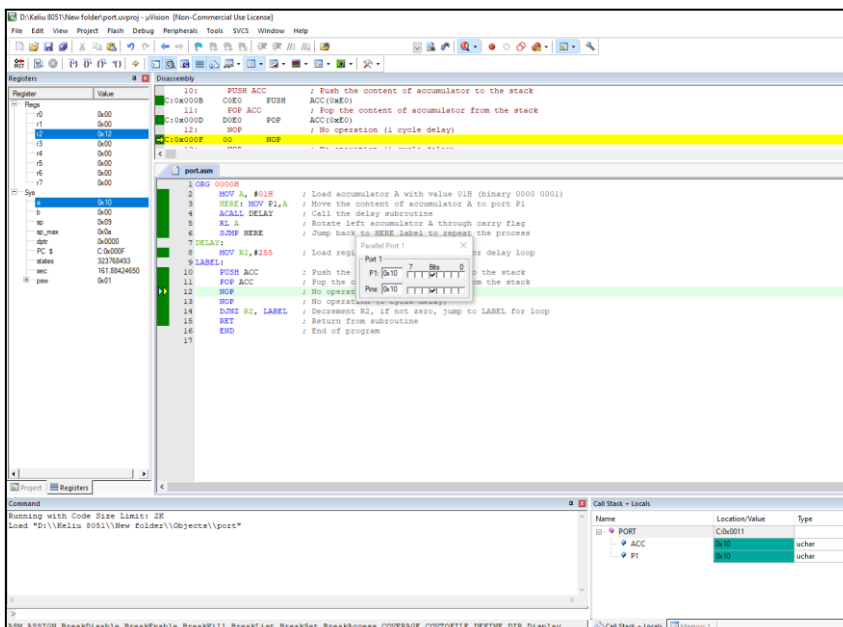
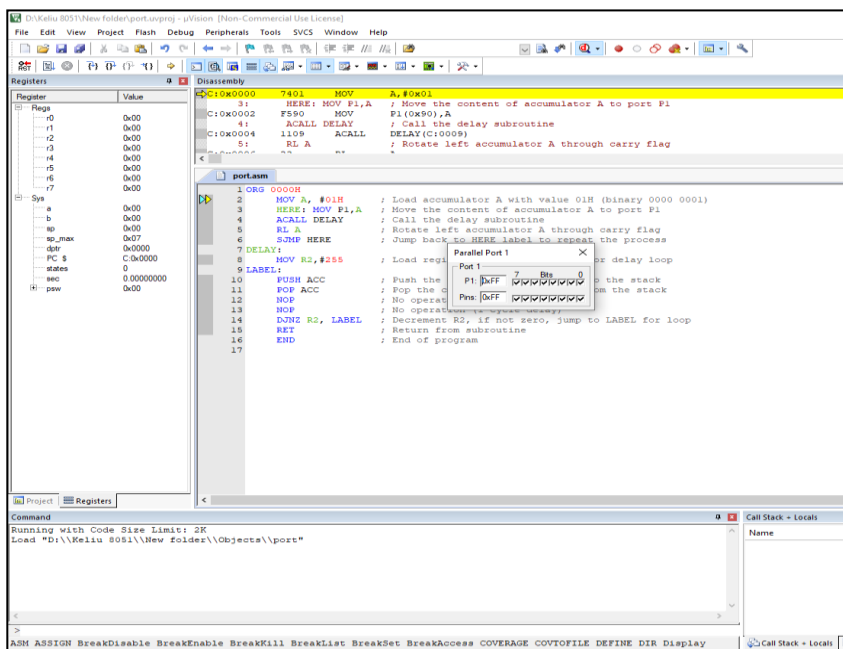
NOP ; No operation (1 cycle delay)

DJNZ R2, LABEL ; Decrement R2, if not zero, jump to LABEL for loop

RET ; Return from subroutine

END ; End of program

Output :



Result :

The program successfully interfaces with the On-Chip ports of the 8051 microcontroller, controlling the logic levels of the pins as intended. The output is verified using a logic analyzer, confirming the expected behavior of the hardware interfacing.

EX.NO:		PROGRAMMING USING SERIAL PORTS IN 8051
DATE :		

Aim :

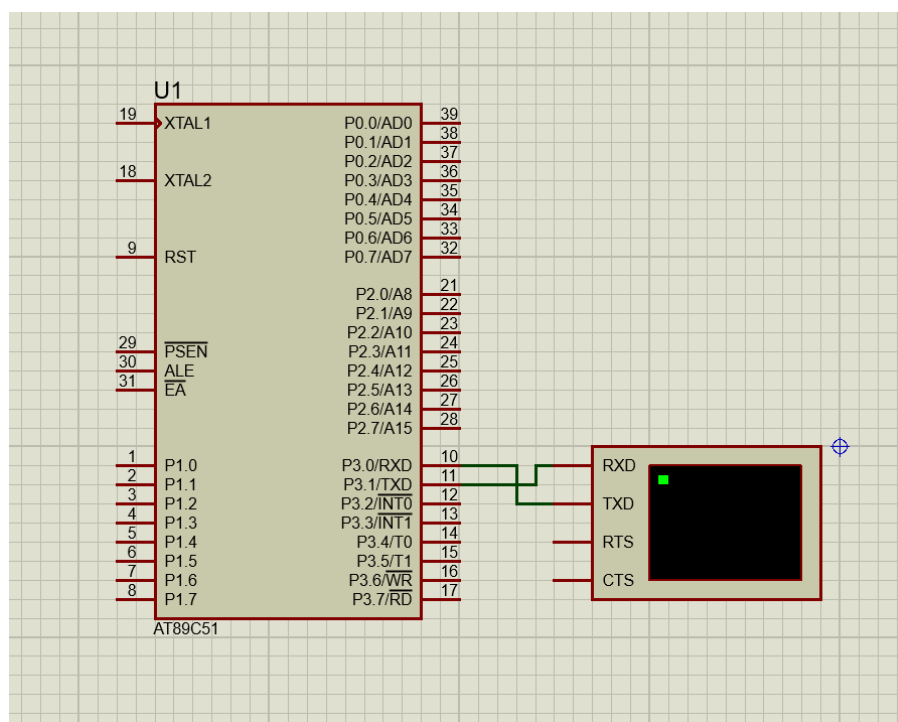
To implement serial communication using 8051 microcontroller and facilitating data transfer between the microcontroller and external devices through serial ports.

Apparatus & Software Required :

1. Personal Computer
2. Keil μ Vision5 software.
3. Proteus Software

Theory:

Serial communication in 8051 involves sending and receiving data one bit at a time over a single wire. The microcontroller's UART (Universal Asynchronous Receiver/Transmitter) module is commonly used for serial communication, with the TXD (Transmit Data) and RXD (Receive Data) pins for transmitting and receiving data, respectively. The communication parameters such as baud rate, data bits, stop bits, and parity must be configured to ensure proper data transfer.



Procedure :

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Proteus Software.
5. To design the circuit Proteus Software.
6. To Observe the Virtual terminal Output of the Proteus Software.

Program :

```
#include<reg51.h>
```

```
void send(char x);    // Function prototype declaration for send function
```

```
void main(void)
```

```
{
```

```
    TMOD = 0x20;        // Set Timer 1 mode for 8-bit auto-reload
```

```
    TH1 = 0xFD;         // Set Timer 1 reload value for baud rate generation
```

```
    SCON = 0x50;        // Configure serial port for mode 1 (8-bit UART, variable baud rate)
```

```
    TR1 = 1;            // Start Timer 1
```

```
    send('R');          // Send character 'R'
```

```
    send('O');          // Send character 'O'
```

```
    send('H');          // Send character 'H'
```

```
    send('I');          // Send character 'I'
```

```
    send('N');          // Send character 'N'
```

```
    send('I');          // Send character 'I'
```

```
    send('C');          // Send character 'C'
```

```
    send('L');          // Send character 'L'
```

```
    while(1);          // Infinite loop
```

```
}
```

```
void send(char x)
```

```
{
```

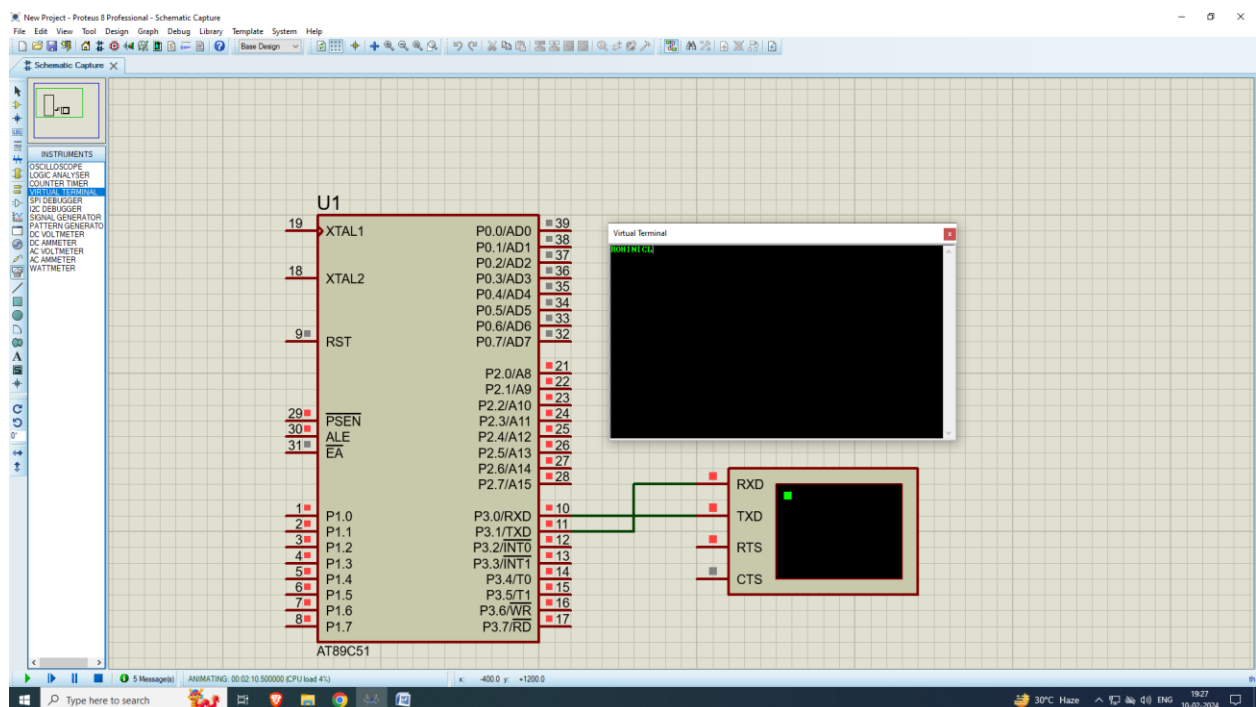
```
    SBUF = x;           // Load character to be sent into the Serial Buffer Register (SBUF)
```

```
    while(TI == 0);     // Wait until transmission is completed (TI flag set)
```

```
    TI = 0;             // Clear the transmit interrupt flag (TI) to indicate transmission completion
```

```
}
```

Output :



Result :

Successfully implement the 8051 microcontroller can transmit and receive data serially, enabling communication with external devices such as Virtual terminal .

EX.NO:		DESIGN OF A DIGITAL CLOCK USING TIMERS/COUNTERS IN 8051
DATE :		

Aim :

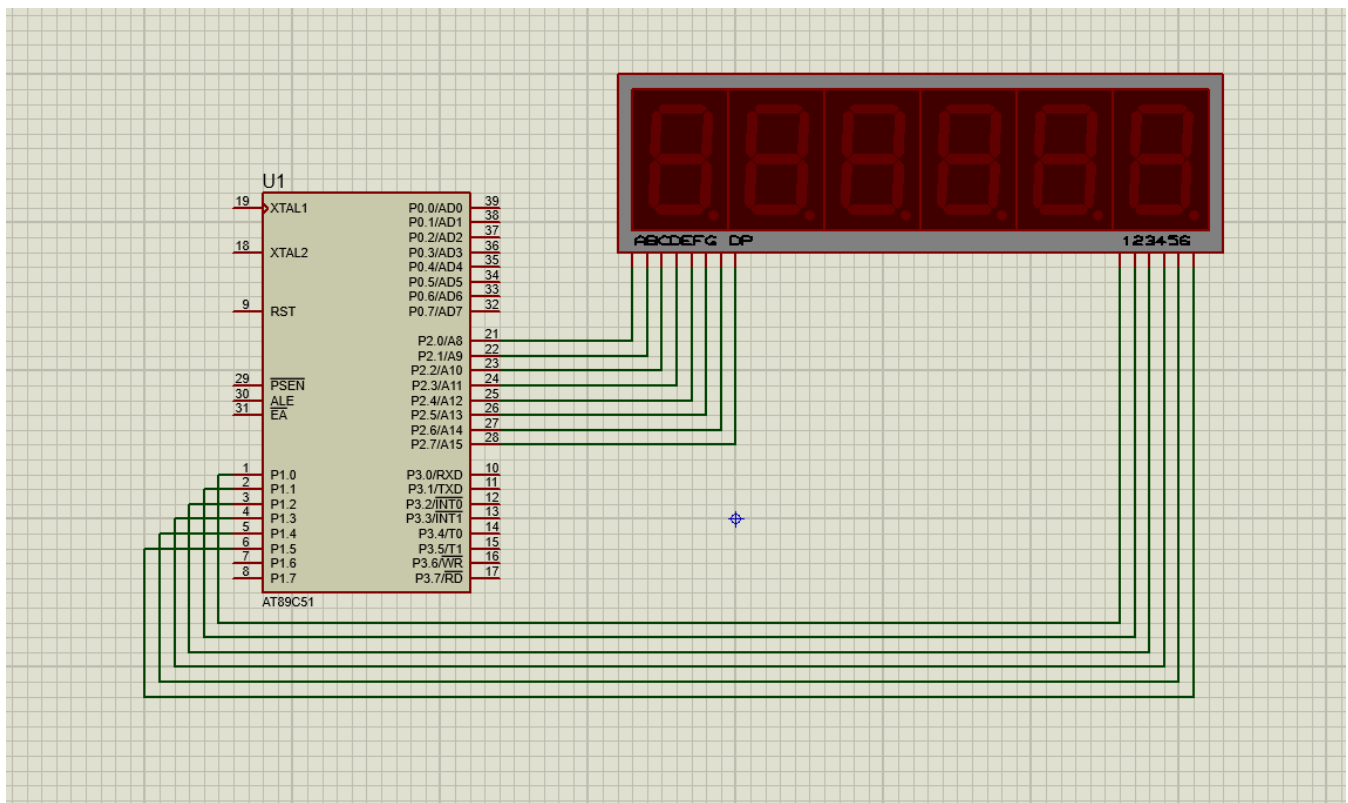
To implement a digital clock using timers/counters in the 8051 microcontroller for accurate timekeeping and display.

Apparatus & Software Required :

1. Personal Computer
2. Keilµ Vision5 software.
3. Proteus Software

Theory:

Digital clocks utilize timers/counters in the 8051 microcontroller to generate precise time intervals. Timer 0 and Timer 1 are commonly used for timekeeping applications, with Timer 0 often used for generating accurate time base interrupts. By configuring timer modes and setting appropriate timer values, the 8051 can maintain accurate timekeeping functionality. Displaying the time typically involves interfacing with a 7-segment display or an LCD to present the time in a human-readable format.



Procedure :

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Proteus Software.
5. To design the circuit Proteus Software.
6. To Observe the Virtual Output of the Proteus Software.

Program :

```
#include <reg51.h>
```

```
#define msec 1
```

```
// Array to store 7-segment display patterns for digits 0-9
```

```
unsigned int arr[10] = {0x40, 0xF9, 0x24, 0x30, 0x19, 0x12, 0x02, 0xF8, 0x00, 0x10};
```

```
// Define 7-segment display control pins
```

```
sbit d4 = P1^0;
```

```
sbit d3 = P1^1;
```

```
sbit d2 = P1^2;
```

```
sbit d1 = P1^3;
```

```
sbit d0 = P1^4;
```

```
sbit d = P1^5;
```

```
// Variables to hold values for each digit of the clock
```

```
unsigned int v1, v2, v3, v4, v0, v5, v6;
```

```
// Function to generate delay
```

```
void delay(unsigned int count) {
```

```
    unsigned int j, k;
```

```
    for (j = 0; j <= count; j++)
```

```
        for (k = 0; k <= 5; k++);
```

```

}

void main() {

    // Initialize all digit values to 0

    v1 = v2 = v3 = v4 = v0 = v5 = v6 = 0;

    // Infinite loop to continuously update clock display

    while (1) {

        // Increment units of seconds

        v0 = v0 + 1;

        if (v0 == 130) {

            v0 = 0;

            v1 = v1 + 1;

        }

        // Display the tens digit of hours

        P2 = 0xFF;

        d = 1;

        d3 = d2 = d4 = d0 = d1 = 0;

        P2 = arr[v1];

        delay(msec);

        // Increment units of minutes

        if (v1 == 10) {

            v1 = 0;

            v2 = v2 + 1;

        }

        // Display the ones digit of hours

        P2 = 0xFF;

        d0 = 1;

        d4 = d3 = d1 = d = d2 = 0;

```

```
P2 = arr[v2];  
delay(msec);  
  
// Increment tens of minutes  
  
if (v2 == 6) {  
    v2 = 0;  
    v3 = v3 + 1;  
}  
  
// Display the tens digit of minutes  
  
P2 = 0xFF;  
d1 = 1;  
d2 = d4 = d3 = d = d0 = 0;  
P2 = arr[v3];  
delay(msec);  
  
// Increment units of minutes  
  
if (v3 == 10) {  
    v3 = 0;  
    v4 = v4 + 1;  
}  
  
// Display the ones digit of minutes  
  
P2 = 0xFF;  
d2 = 1;  
d3 = d4 = d1 = d = d0 = 0;  
P2 = arr[v4];  
delay(msec);  
  
// Increment tens of seconds  
  
if (v4 == 6) {  
    v4 = 0;
```

```

    v5 = v5 + 1;
}

// Display the tens digit of seconds

P2 = 0xFF;

d3 = 1;

d0 = d2 = d1 = d = d4 = 0;

P2 = arr[v5];

delay(msec);

// Increment units of seconds

if (v5 == 10) {

    v5 = 0;

    v6 = v6 + 1;

}

// Display the ones digit of seconds

P2 = 0xFF;

d4 = 1;

d3 = d2 = d1 = d = d0 = 0;

P2 = arr[v6];

delay(msec);

// Reset the clock when it reaches 12:00:00

if (v6 == 1 && v5 == 2) {

    v1 = 0;

    v2 = 0;

    v3 = 0;

    v4 = 0;

    v5 = 0;

    v6 = 0;

```

```

    }

    delay(msec);

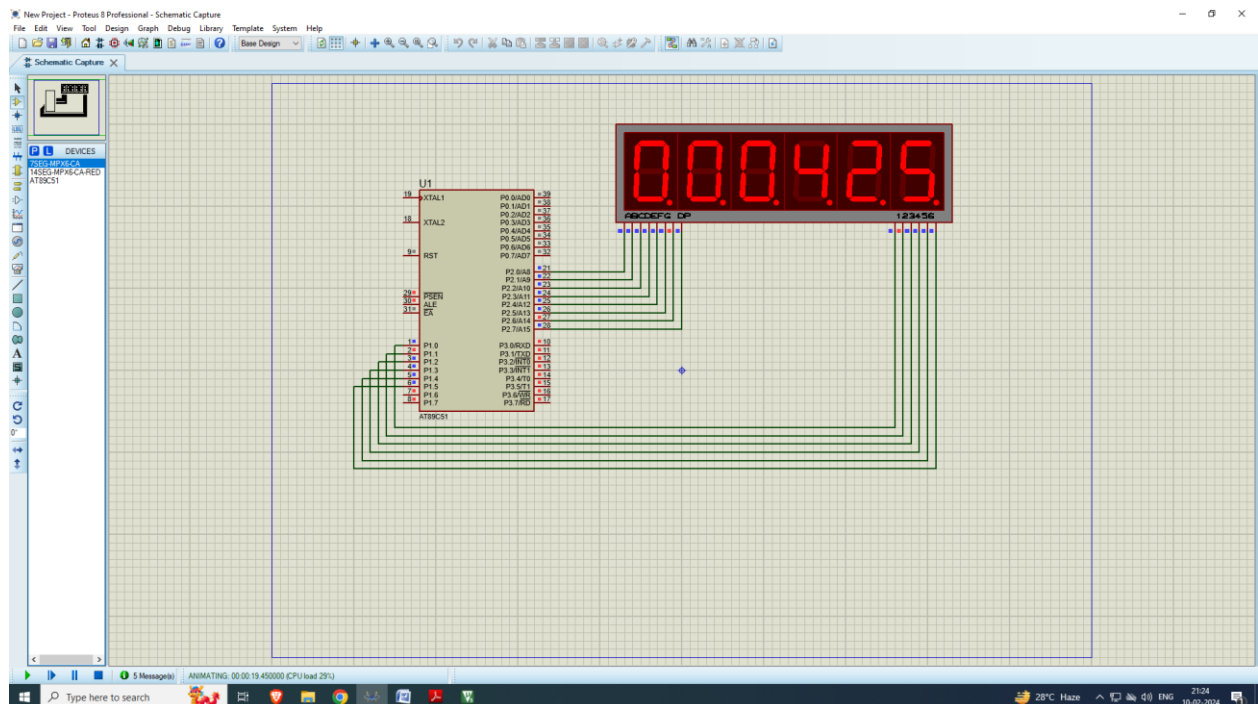
    P2 = 0xFF;

    }

}

```

Output :



Result :

The digital clock implemented using timers/counters in the 8051 microcontroller successfully displayed accurate time, verified by observing the output signals using a logic analyzer. The clock accurately updated its display at regular intervals, demonstrating the functionality of the implemented design.

EXPERIMENTS USING ARM

EX.NO:		INTERFACING ADC & DAC
DATE :		

Aim :

To develop a C-Language program for reading an on-chip ADC, convert into decimal and to display it in PC and to generate a square wave depending on this ADC reading. The ADC input is connected to any analog sensor/ on board potentiometer.

Apparatus & Software Required :

1. LPC2148 Development board.
2. KeilµVision5 software.
3. Flash Magic.
4. USB cable.
5. CRO.

Theory :

The LPC 2148 has 10-bit successive approximation analog to digital converter. Basic clocking

for the A/D converters is provided by the VPB clock. A programmable divider is included in each converter, to scale this clock to the 4.5 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 11 of these clocks. The ADC cell

can measure the voltage on any of the ADC input signals.

ARM Board has one potentiometer for working with A/D Converter. Potentiometer outputs are in the range of 0V to 3.3V. Switch select in right position for reading the Potentiometer value by ADC.

Procedure :

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using FlashMagic. Software.

ADC Program :

```
#include<lpc214x.h>
intk,z;
#define rs 30
#define en 29
void LCD(void);
void co(unsigned char );
void da(unsigned char );
```

```

void lcds(unsigned char* p);

int main()
{
    int in1; /*VARIABLE DECLARATION*/
    LCD(); /*INITIALIZES LCD*/

    while(1){
        PINSEL0|=1<<12|1<<13; /* SELECTING PINSEL FOR ADC1.0*/
        AD1CR=0x01200401; /*SETTING VALUE ON ADC CONTROL
REGISTER*/

        while((AD1STAT&0x01)==0); /*WAIT FOR ADC INTERRUPT*/
        in1=((AD1GDR&0x0000ffe0)>>6); /*GETTING ADC VALUE
IN VARIABLE*/
        /*-----DISPLAYING ADC VALUE ON LCD-----*/
        co(0x80);
        lcds(" ADC_VALUE :");
        da((in1/1000)+0x30);
        da((in1%1000)/100+0x30);
        da((in1%100)/10+0x30);
        da((in1%10)+0x30);
        for(k=0;k<1000;k++); /*DELAY*/
    }
}

//-----FUNCTION TO LOAD COMMAND TO LCD-----
-----/

void co(unsigned char x)
{
    IO0PIN=(IO0PIN&0xFFFFC3FF)|((x&0xF0)>>4)<<10;
    IO0CLR=1<<rs;
    IO0SET=1<<en;
    for(k=0;k<20000;k++);
    IO0CLR=1<<en;
    for(k=0;k<1000;k++);
    IO0PIN=(IO0PIN&0xFFFFC3FF)|((x&0x0F)<<10;
    IO0CLR=1<<rs;
    IO0SET=1<<en;
    for(k=0;k<20000;k++);
    IO0CLR=1<<en;

}

```


//-----FUNCTION TO DISPLAY DATA ON LCD-----/

```
void da(unsigned char y)
{
    IO0PIN=(IO0PIN&0xFFFFC3FF)|((y&0xF0)>>4)<<10;
    IO0SET=1<<rs;
    IO0SET=1<<en;
    for(k=0;k<20000;k++);
    IO0CLR=1<<en;
    for(k=0;k<1000;k++);
    IO0PIN=(IO0PIN&0xFFFFC3FF)|(y&0x0F)<<10;
    IO0SET=1<<rs;
    IO0SET=1<<en;
    for(k=0;k<20000;k++);
    IO0CLR=1<<en;

}
```

//-----LCD INITIALIZATION-----/

```
void LCD(void)
{
    unsigned char i[]={0x28,0x0f,0x01,0x06},j;
    IO0DIR|=0x60003c00;
    //for(k=0;k<1000;k++);
    for(j=0;j<4;j++)
    {
        co(i[j]);
        for(z=0;z<1000;z++);
    }
}
```

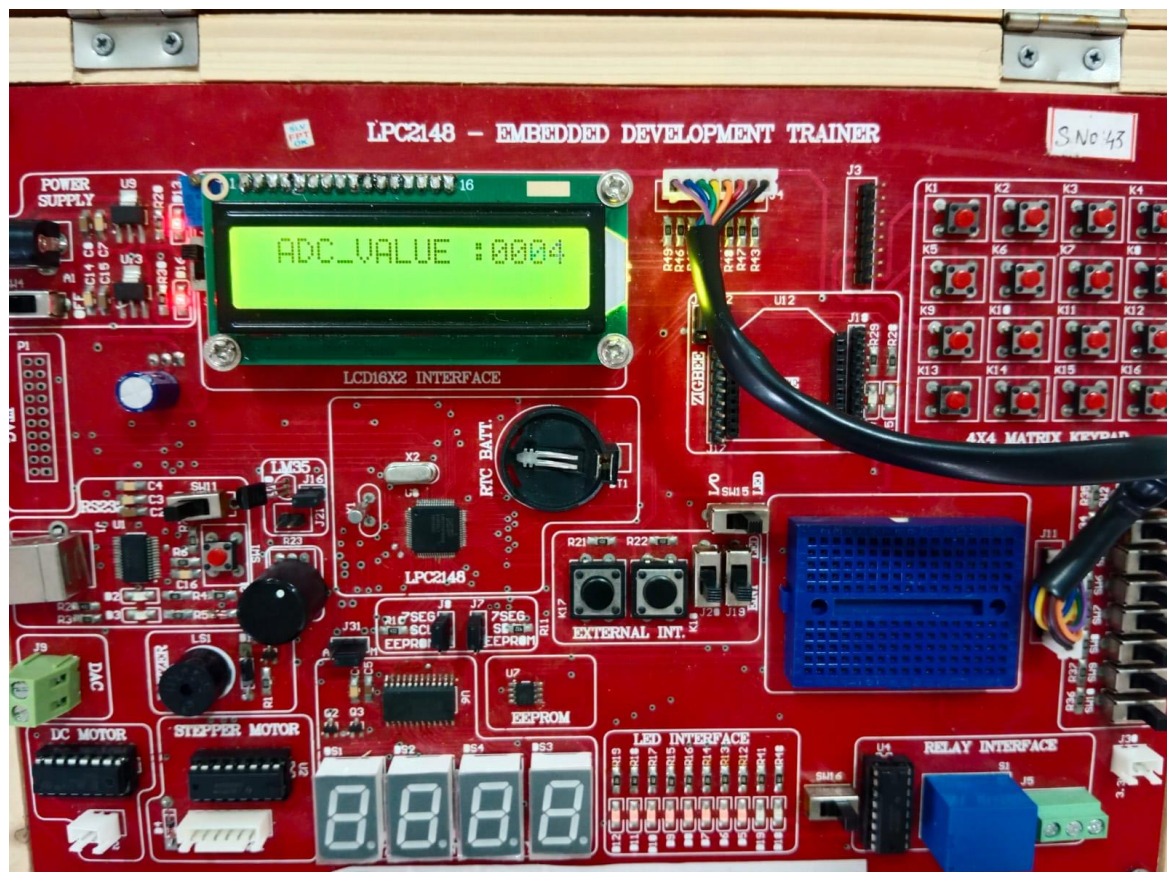
/*-----DISPLAY STRING ON LCD-----*/

```
void lcds(unsigned char* p)
{
    for(;*p!=0;p++)
    {
        da(*p);    }
}
```

ADC PROGRAM PORT DETAILS

ARM	DETAILS
P0.29	ADCO0.2
PO.10	RS LCD PIN
P1.11	CE LCD PIN

Output :



DAC Program :

Square Wave :

```
#include <LPC214x.H> /* LPC214x definitions */
////////// InitDAC ////////////
Init_DAC()
{
    // Convert Port pin 0.25 to function as DAC
    PINSEL1 = 0X00080000;
    DACR = 0;
}

////////// Write DAC ////////////
Write_DAC(unsigned int dacval)
{
    DACR = dacval << 6;
}

void delay(unsigned int count)
{

```

```
int j=0,i=0;
```

```
for(j=0;j<count;j++)
```

```
{
```

```
for(i=0;i<120;i++);
```

```
}
```

```
}
```

```
////////// MAIN //////////
```

```
int main (void)
```

```
{
```

```
Init_DAC();
```

```
while(1)
```

```
{
```

```
Write_DAC(00);
```

```
delay(100);
```

```
// change this value to change Frequency
```

```
Write_DAC(1023); // change this value to change Amplitude
```

```
delay(80);
```

```
// change this value to change Frequency
```

```
}
```

```
}
```

Sin Wave :

```
#include <LPC214x.H> /* LPC214x definitions */
```

```
const unsigned int tab[256] = { 514 , 526 , 538 , 551 , 563 , 576 , 588 , 600 , 613 , 625 , 637  
, 649 , 661 , 673 , 685 , 696 , 708 , 719 , 731 , 742 , 753 , 764 , 775 , 785 , 796 , 806 , 816 ,  
826,836 , 845 , 855 , 864 , 873 , 881 , 890 , 898 , 906 , 914 , 922 , 929 , 936 , 943 , 949 , 956 ,  
962, 967 , 973 , 978 , 983 , 987 , 992 , 996 , 1000 , 1003 , 1006 , 1009 ,1012 , 1014 , 1016 ,  
1018 , 1019 , 1020 , 1021 , 1021 , 1021 , 1021 , 1021 , 1020 , 1019 , 1018 , 1016 , 1014 ,  
1012,1009,1006 , 1003 , 1000 , 996 , 992 , 987 , 983 , 978 , 973 , 967 , 962 , 956 , 949 , 943  
, 936 , 929 , 922 , 914 , 906 , 898 , 890 , 881 , 873 , 864 , 855 , 845 , 836 , 826 , 816 , 806 ,  
796 , 785 , 775 , 764 , 753 , 742 , 731 , 719 , 708 , 696 , 685 , 673 , 661 , 649 , 637 , 625 , 613  
, 600 , 588 , 576 , 563 , 551 , 538 , 526 , 514 , 501 , 489 , 476 , 464 , 451 , 439 , 427 , 414 ,  
402 , 390 , 378, 366 , 354 , 342 , 331 , 319 , 308 , 296 , 285 , 274 , 263 , 252 , 242 , 231 , 221  
, 211 , 201 , 191 , 182 , 172 , 163 , 154 , 146 , 137 , 129 , 121 , 113 , 105 , 98 , 91 , 84 , 78 ,  
71 , 65 , 60 , 54 , 49 , 44 , 40 , 35 , 31 , 27 , 24 , 21 , 18 , 15 , 13 , 11 , 9 , 8 , 7 , 6 , 6 , 6 , 6 ,  
, 7 , 8 , 9 , 11 , 13 , 15 , 18 , 21 , 24 , 27 , 31 , 35 , 40 , 44 , 49 , 54 , 60 , 65 , 71 , 78 , 84 , 91 ,  
98 , 105 , 113 , 121 , 129, 137 , 146 , 154 , 163 , 172 , 182 , 191 , 201 , 211 , 221 , 231 , 242 ,  
252 , 263 , 274 , 285 , 296 , 308 , 319 , 331 , 342 , 354 , 366 , 378 , 390 , 402 , 414 , 427 , 439  
, 451 , 464 , 476, 489 , 501 };
```

```
////////// InitDAC //////////
```

```
Init_DAC()
```

```
{  
    // Convert Port pin 0.25 to function as DAC  
    PINSEL1 = 0X00080000;  
    DACR = 0;  
}
```

```
////////// Write DAC //////////
```

```
Write_DAC(unsigned intdacval)
```

```
{  
    DACR = dacval<< 6;  
}
```

```
////////// MAIN //////////
```

```
int main (void)
```

```
{  
    unsigned inti;
```

```
    Init_DAC();
```

```
    while(1)
```

```
    {  
        for(i=0;i<255;i++)  
            Write_DAC(tab[i]);  
    }  
}
```

TriangleWave :

```
#include <LPC214x.H> /* LPC214x definitions */
```

```
////////// InitDAC //////////
```

```
Init_DAC()
```

```
{  
    // Convert Port pin 0.25 to function as DAC  
    PINSEL1 = 0X00080000;  
    DACR = 0;  
}
```

```
////////// Write DAC //////////
```

```

Write_DAC(unsigned int dacval)
{
    DACR = dacval<< 6;
}

```

```

////////// MAIN //////////

```

```

int main (void)
{
    unsigned inti;

    Init_DAC();

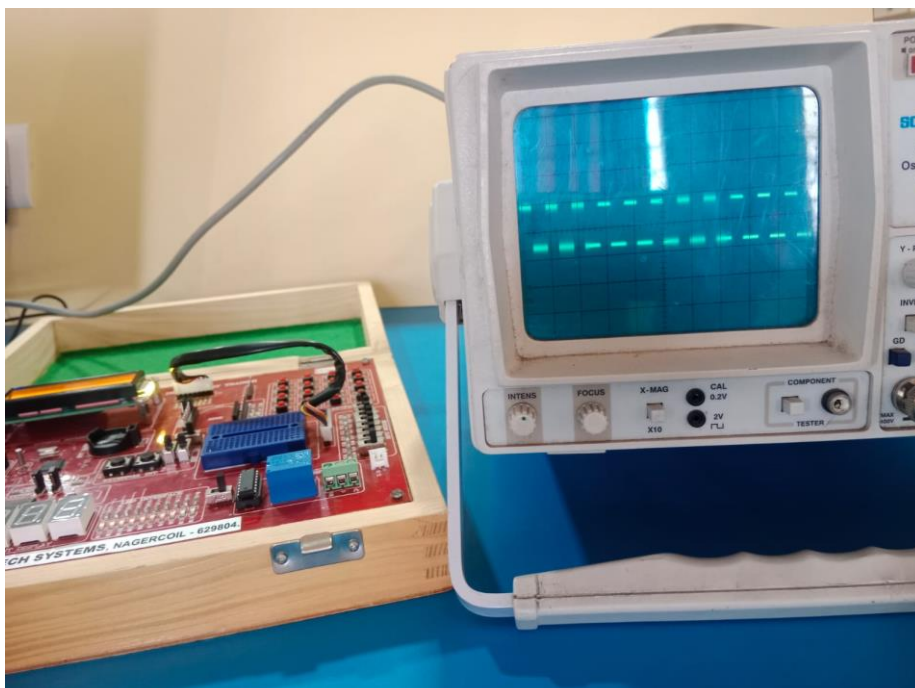
    while(1)
    {
        for(i=0;i<1024;i++)
            Write_DAC(i);
        for(i=1023;i>0;i--)
            Write_DAC(i);
    }
}

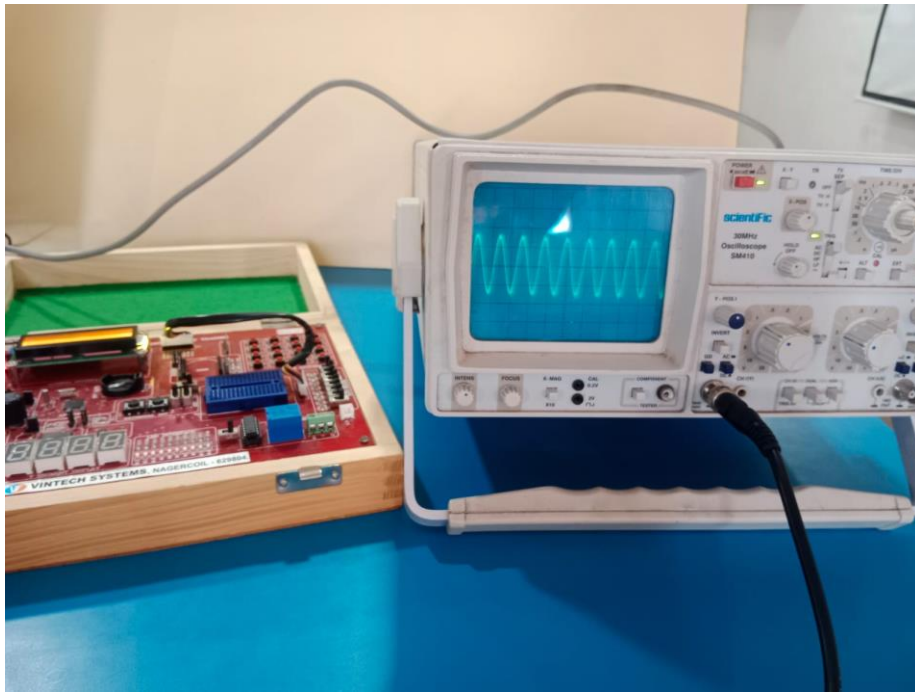
```

DAC PROGRAM PORT DETAILS

ARM	DETAILS
P0.25	DAC ENABLE PIN

Output :





Result:

The C-Language program for reading an on-chip ADC, convert into decimal and to display it in PC was written & output is verified with the ADC input is connected to on board potentiometer.

The DAC, convert digital data into analog signal& output is verified with the DAC input and the square wave has been generated to display it in CRO.

EX.NO:		BLINKING OF LEDS AND LCD
DATE :		

Aim :

To develop a 'C' program to make the LED blink(including delay routine) and LCD module. The LCD display should come in the desired line and column and Upon change in LED delay program the speed should vary.

Apparatus & Software Required :

1. LPC2148 Development board.
2. Keilp.Vision5 software.
3. Flash Magic.
4. USB cable.

Theory :

LEDs are based on the semiconductor diode. When the diode is forward biased (switched on), electrons are able to recombine with holes and energy is released in the form of light. This effect is called electroluminescence and the color of the light is determined by the energy gap of the semiconductor.

The LCD display should come in the desired line and column to print the letters of the program and verified the outputs.

Procedure :

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

Blink LED Program :

```
#include <lpc214x.h>
/* -----MAIN FUNCTION----- */
int main()
{
    inti;
    /*SETTING DIRECTION OF P1.16-P1.23 ,P0.15,P0.16 AS OUTPUT*/
    IODIR1|=0x00FF0000;
    IODIR0|=0x00018000;
```

```

/*-----INFINITE LOOP TO BLINK LED CONTINUOUSLY-----*/
while(1)
{
    /*SETTING P1.16-P1.23 ,P0.15,P0.16 AS HIGH TO POWER ON LED*/
    IOSET1 = 0x00FF0000;
    IOSET0 = 0x00018000;

    for(i=0;i<500000;i++); //DELAY FUNCTION

    /*CLEARING P1.16-P1.23 ,P0.15,P0.16 AS HIGH TO POWER OFF
LED*/
    IOCLR1 = 0x00FF0000;
    IOCLR0 = 0x00018000;

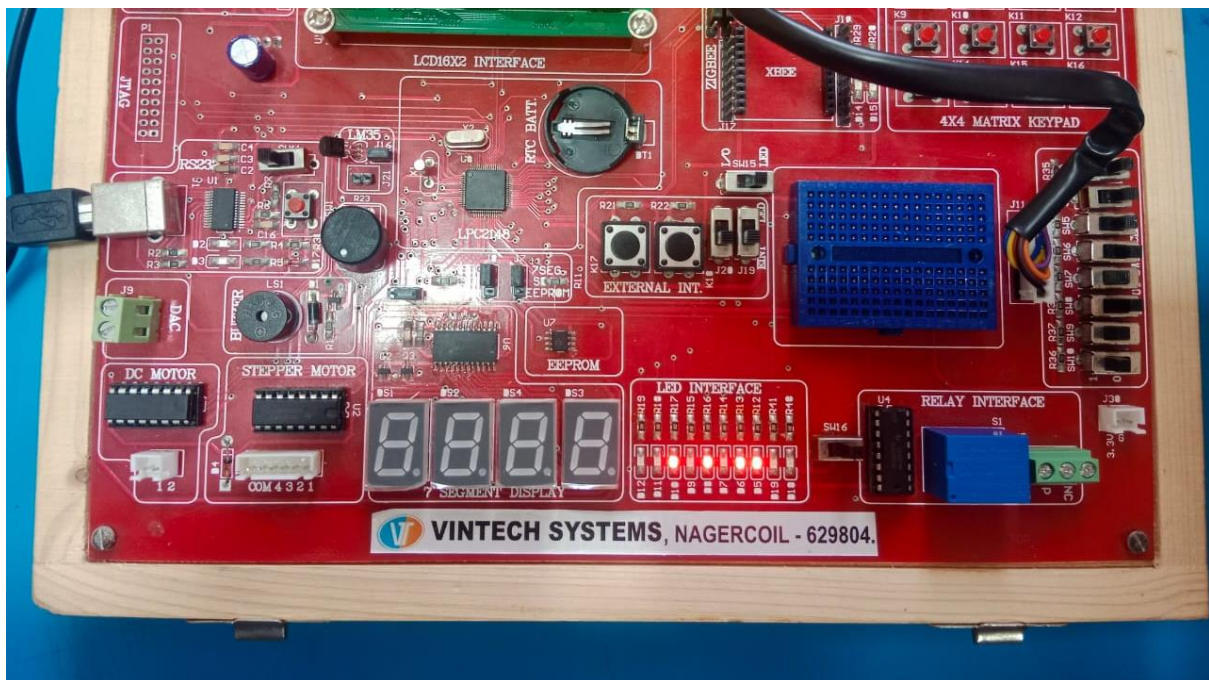
    for(i=0;i<500000;i++); //DELAY FUNCTION
}
}

```

LED PORT DETAILS

ARM	DETAILS
PI.16	S&L ENABLE PIN
PI.17	S&L ENABLE PIN
PI.18	S&L ENABLE PIN
PI.19	S&L ENABLE PIN
P1.20	S&L ENABLE PIN
P1.21	S&L ENABLE PIN
PI.22	S&L ENABLE PIN
P1.23	S&L ENABLE PIN
PI.24	S&L ENABLE PIN
P1.25	S&L ENABLE PIN
P1.26	S&L ENABLE PIN
P1.27	S&L ENABLE PIN
P1.28	S&L ENABLE PIN
P1.29	S&L ENABLE PIN
PI.30	S&L ENABLE PIN
PI.31	S&L ENABLE PIN

Output :



LCD Interfacing Program :

```
#include<lpc214x.h>
#define rs 30
#define en 29
    int k;
    void co (unsigned char);
    void da (unsigned char);
    void LCD (void);
    void lcds (unsigned char*);
intmain()
{

    LCD();
    co(0x80);
    lcds(" ROHINI COLLEGE ");
    while(1);
}

//-----COMMAND-----/
void co(unsigned char x)
{
    IO0PIN=(IO0PIN&0xFFFFC3FF)|((x&0xF0)>>4)<<10;
    IO0CLR=1<<rs;
    IO0SET=1<<en;
```

```

    for(k=0;k<100000;k++);
    IO0CLR=1<<en;
    IO0PIN=(IO0PIN&0xFFFFC3FF)|(x&0x0F)<<10;
    IO0CLR=1<<rs;
    IO0SET=1<<en;
    for(k=0;k<100000;k++);
    IO0CLR=1<<en;

}
//-----DATA-----/
void da(unsigned char y)
{
    IO0PIN=(IO0PIN&0xFFFFC3FF)|((y&0xF0)>>4)<<10;
    IO0SET=1<<rs;
    IO0SET=1<<en;
    for(k=0;k<100000;k++);
    IO0CLR=1<<en;
    IO0PIN=(IO0PIN&0xFFFFC3FF)|(y&0x0F)<<10;
    IO0SET=1<<rs;
    IO0SET=1<<en;
    for(k=0;k<100000;k++);
    IO0CLR=1<<en;

}
//-----LCD INITIALIZATION-----/
void LCD(void)
{
    unsigned char i[]={0x28,0x0f,0x01,0x06},j;
    IO0DIR|=0x60003c00;
    for(k=0;k<1000;k++);
    for(j=0;j<4;j++)
    {
        co(i[j]);
        for(k=0;k<200000;k++);
    }
}
//-----LCDSTRING-----/
void lcds(unsigned char* p)
{
    for(;*p!=0;p++)
    {
        da(*p);
        for(k=0;k<200000;k++);
    }
}

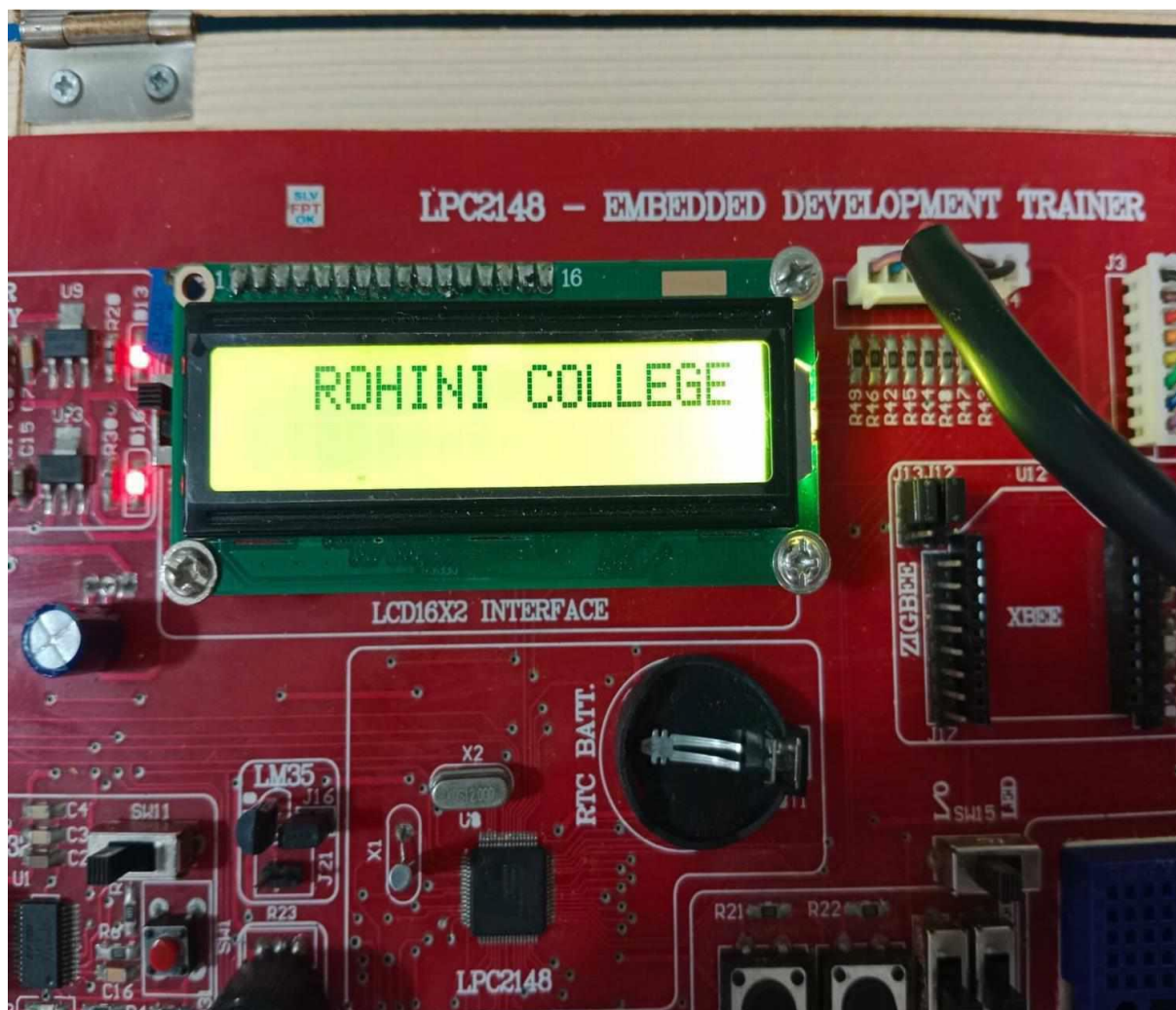
```

```
}  
}
```

LCD PROGRAM PORT DETAILS

ARM	DETAILS
PO.IO	RS LCD PIN
PI.11	CE LCD PIN

Output :



Result :

The C-Language program to make the LED blink was developed and output was verified and LCD module and the output was verified on the LCD on the desires line and column/address successfully.

EX.NO:		INTERFACING KEYBOARD AND STEPPER MOTOR
DATE :		

Aim :

To develop a C-Language program for Interfacing the Key pressed in the Keypad in the terminal and interfacing the running stepper motor either in clock- wise or counter-clock- wise and the direction of the rotation .

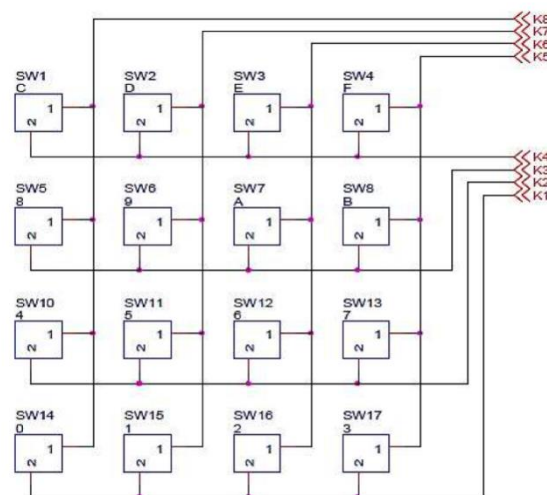
Apparatus & Software Required :

1. LPC2148 Development board.
2. Keilp.Vision5software.
3. Flash Magic.
4. USB cable.
5. Stepper Motor
6. Matrix Keyboard

Theory :

KEYBOARD

The Matrix keyboard is used to minimize the number of I/O lines. Normally it is possible to connect only one key or switch with an I/O line. If the number of keys in the system exceeds the more I/O lines are required. To reduce the number of I/O lines the keys are connected in the matrix circuit. Keyboards use a matrix with the rows and columns made up of wires. Each key acts like a switch. When a key is pressed a column wire makes contact with row wire and completes a circuit. For example 16 keys arranged in a matrix circuit uses only 8 I/O lines.



STEPPER MOTOR

Stepper motors, effectively have multiple "toothed" electromagnets arranged around a central metal gear. To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet.

So when the next electromagnet is turned on and the first will turn off, the gear rotates slightly to align with the next one and from there the process is repeated. Each of those slight rotations is called a "step." In that way, the motor can be turned to a precise angle. There are two basic arrangements for the electromagnetic coils: bipolar and unipolar.

Procedure :

1. Follow the steps to create a New project
2. Type the below code and save it with the name (anyname.c)
3. Follow the steps to create a New Project to compile and build the program
4. Follow the procedures in to download your Hex code to processor using Flash Magic Software.

KEYBOARD PROGRAM :

```
#include<lpc214x.h>
/*-----PROTOTYPE DEFINITION-----*/
void serini(void);
void tr(unsigned char);
unsigned char key_wa(void);
unsigned char keypad(void);
void delay(void);
/*-----MAIN FUNCTION-----*/
intmain()
{
    inti,k;
    unsigned char a[]="PRESS ANY KEY";
    serini();
    while(1)
    {
        /*-----ROUTINE TO DISPLAY PRESS ANY KEY STRING ON SERIAL-----*/
        tr(0x0d);
        tr(0x0A);
        for(i=0;a[i]!='\0';i++)
            tr(a[i]);

        tr(0x0d);
        tr(0x0A);
    }
}
```

```

/*-----SENDING PRESSED KEY ON SERIAL-----
-----*/
tr(key_wa());

    for(k=0;k<150000;k++); //DELAY FUNCTION
}
}
/*-----SERIAL INITIALIZATION-----*/
void serini(void)
{
    PINSEL0=0x00000005;
    U0LCR=0x83;
    U0DLL=0x5a;
    U0DLM=0x00;
    U0FDR=0xc1;
    U0LCR=0x03;
}

/*-----SERIAL TRANSMIT FUNCTION-----
-----*/
void tr(unsigned char a)
{
    U0THR=a;
    while((U0LSR&1<<5)==0);
}

/*-----KEYPAD SCAN FUNCTION-----
-----*/
unsigned char keypad(void)
{
    IO0DIR|=0x0F00<<8;

    IO0PIN =(IO0PIN&0xFFF0FFFF)|0x0E00<<8; // First Scan Line
    if(( IO0PIN & 0xF000<<8 )!= 0xF000<<8) //Check if any key is pressedin
    //4th row
    {
        switch(IO0PIN & 0xF000<<8) //Check which one of the key is pressed
        {
            case 0x0007000<<8 : delay();return 'c' ;
            case 0x000B000<<8 : delay();return 'd' ;
            case 0x000D000<<8 : delay();return 'e' ;

```



```

    case 0x000E000<<8 : delay();return 'f' ;
    }
    }
    IO0PIN =(IO0PIN&0xFFF0FFFF)|0x0D00<<8; //Move second data to scan line
    if(( IO0PIN & 0xF000<<8)!= 0xF000<<8) //Check if any key is pressed in 3rd
row.

    {
    switch(IO0PIN & 0xF000<<8) //check which one of the key is pressed
    {
        case 0x0007000<<8 : delay();return '9' ;
        case 0x000B000<<8 : delay();return '0' ;
        case 0x000D000<<8 : delay();return 'a' ;
        case 0x000E000<<8 : delay();return 'b' ;
        }
    }
    IO0PIN =(IO0PIN&0xFFF0FFFF)|0x0B00<<8; //Move 3rd scan data to port
line

    if(( IO0PIN & 0xF000<<8 )!= 0xF000<<8)//Scan any key is pressed in 2nd row
    {
    switch(IO0PIN & 0xF000<<8) //Check which one of the key is
//pressed in 2nd row
    {
        case 0x0007000<<8 : delay();return '5' ;
        case 0x000B000<<8 : delay();return '6' ;
        case 0x000D000<<8 : delay();return '7' ;
        case 0x000E000<<8 : delay();return '8' ;
        }
    }
    IO0PIN =(IO0PIN&0xFFF0FFFF)|0x0700<<8; //Move 4th scan data to port line
    if(( IO0PIN & 0xF000<<8 )!= 0xF000<<8) //Check any key is pressed in 1st
// row
    {
    switch(IO0PIN & 0xF000<<8) //Check which one of the key is
//pressed in 1st row
    {
        case 0x0007000<<8 : delay();return '1' ;
        case 0x000B000<<8 : delay();return '2' ;
        case 0x000D000<<8 : delay();return '3' ;
        case 0x000E000<<8 : delay();return '4' ;
        }
    }
    return 0xFF;
    }

```

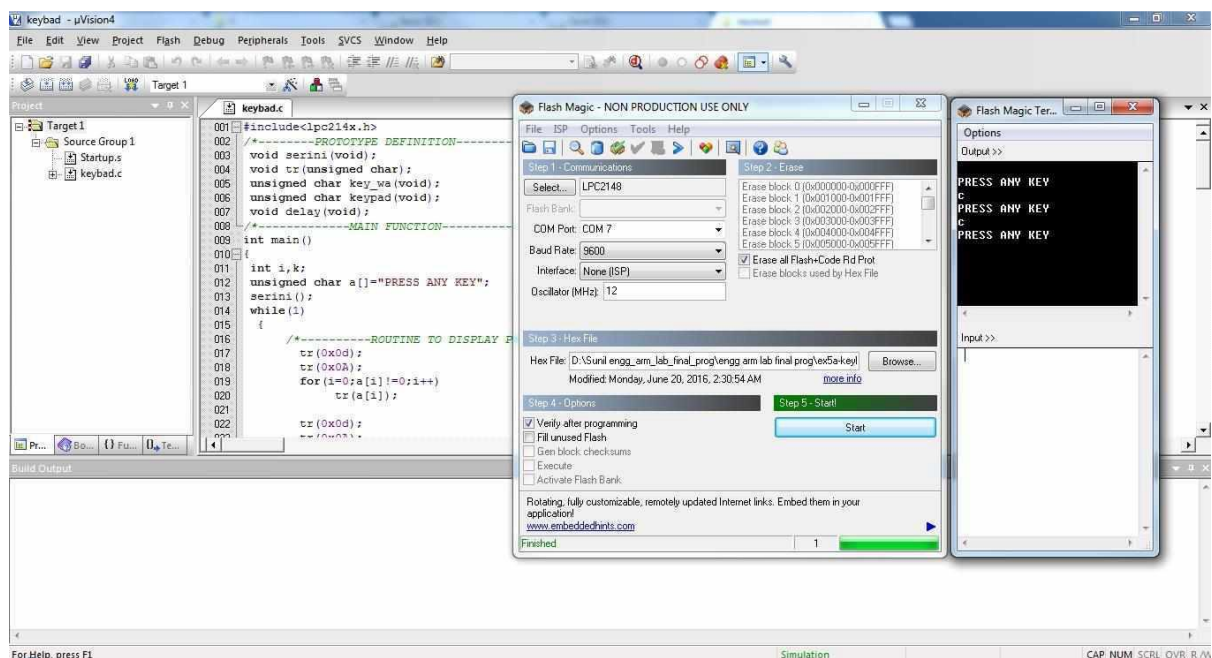
/*-----KEYPAD WAIT FUNCTION-----*/

```
unsigned char key_wa(void)
{
    unsigned char Read;
    while((Read=keypad())==0xFF);
    return Read;
}
```

/*-----DELAY FUNCTION-----*/

```
void delay()
{
    unsigned int i,j;
    for(i=0;i<0xff;i++)
    for(j=0;j<0xffff;j++);
}
```

OUTPUT :



Stepper Motor Program :

counter clockwise

```
#include<lpc214x.h>
```

/*-----MAIN FUNCTION-----*/

```
int main(void)
```



```

{
inti;
IO1DIR|=0x3c<<24;
while(1)
{
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x1<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x3<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x2<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x6<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x4<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0xC<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x8<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x9<<26;
for(i=0;i<25000;i++);
} }

```

Anti clockwise

```
#include<lpc214x.h>
```

```
/*-----MAIN FUNCTION-----*/
```

```

int main(void)
{
inti;
IO1DIR|=0x3c<<24;
while(1)
{
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x1<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x3<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x2<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x6<<26;
for(i=0;i<25000;i++);
IO1PIN=(IO1PIN&0xc3FFFFFF)|0x4<<26;

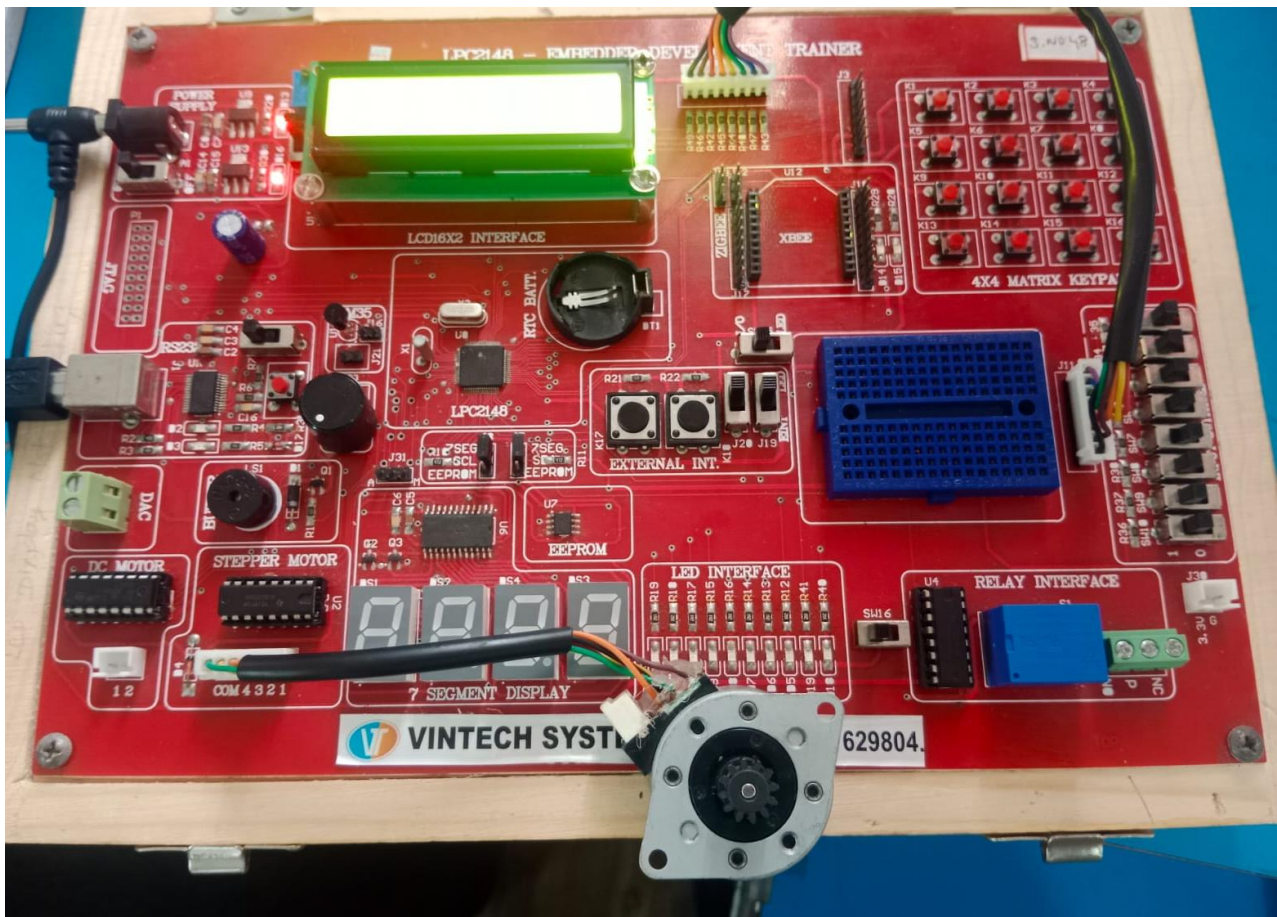
```

```

    for(i=0;i<25000;i++);
    IO1PIN=(IO1PIN&0xc3FFFFFF)|0xC<<26;
    for(i=0;i<25000;i++);
    IO1PIN=(IO1PIN&0xc3FFFFFF)|0x8<<26;
    for(i=0;i<25000;i++);
    IO1PIN=(IO1PIN&0xc3FFFFFF)|0x9<<26;
    for(i=0;i<25000;i++);
}
}

```

OUTPUT :



Result :

The C-Language program for running stepper motor either in clock-wise or counter-clock-wise direction.

MINIPROJECTS FOR IOT

EX.NO:		GARBAGE SEGREGATOR AND BIN LEVEL INDICATOR
DATE :		

Aim :

To design the simulation and circuit for a Garbage Segregator and Bin Level Indicator system and verify the output.

Apparatus & Software Required :

1. PC
2. Wokwi Online Simulation Platform

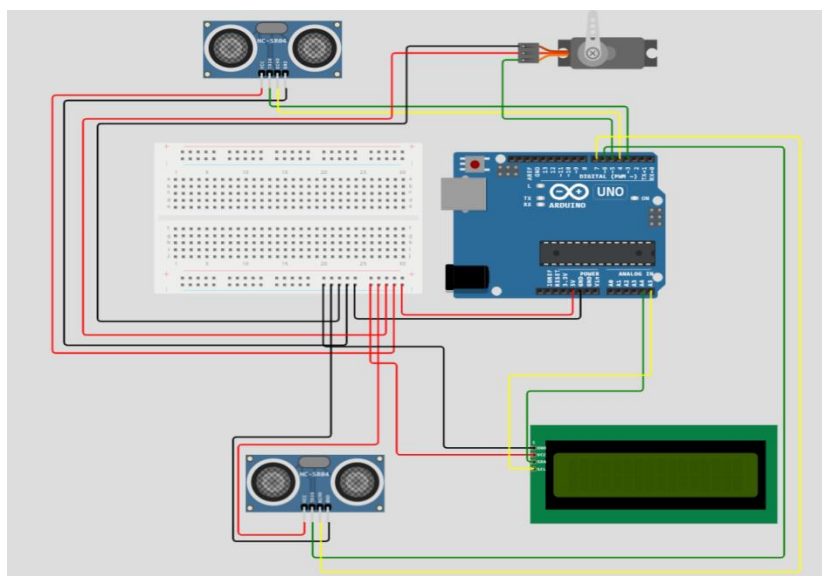
Procedure :

1. Open the Google Chrome and Open the <https://wokwi.com/> Online Simulation platform.
2. To select the Components to start the designs per circuit diagram.
3. To write and upload the coding from the simulation platform .
4. Finally Verified the output of the simulation.

Theory :

The Garbage Segregator and Bin Level Indicator system aims to efficiently manage waste disposal by automatically segregating different types of garbage and providing real-time information about the level of waste in the bins. This system utilizes various sensors and actuators along with microcontroller-based control to achieve its functionality.

Circuit Diagram :



Pin Details:

LCD DISPLAY	MIRCOCONTROLLER
VCC	5V
GND	GND
SDA	A4
SCL	A5
ULTRASONIC SENSOR 1	PIN
VCC	5V
GND	GND
TRIG	6
ECHO	7
ULTRASONIC SENSOR 2	PIN
VCC	5V
GND	GND
TRIG	3
ECHO	4
SERVO	PIN
VCC	5V
GND	GND
PWM	5

Program :

```
#include <Servo.h> //servo library
```

```

#include <LiquidCrystal_I2C.h> // if you don't have I2C version of the display, use
LiquidCrystal.h library instead

LiquidCrystal_I2C lcd(0x27,16,2);

Servo servo;
int trigPin = 3;
int echoPin = 4;
int servoPin = 5;
int trigPin2 = 6;
int echoPin2 = 7;
int active = 0;
long duration, dist, average;
long aver[3]; //array for average
long duration2, dist2, average2;
long aver2[3];
int percent;

byte gauge_left[8]
= { B11111, B10000, B10000, B10000, B10000, B10000, B10000, B11111}; // left
part of empty gauge [
byte gauge_center[8] =
{ B11111, B00000, B00000, B00000, B00000, B00000, B00000, B11111}; // center
part of empty gauge _
byte gauge_right[8]
= { B11111, B00001, B00001, B00001, B00001, B00001, B00001, B11111}; // right
part of empty gauge ]
byte gauge_fill[8]
= { B11111, B11111, B11111, B11111, B11111, B11111, B11111, B11111}; // filled
gauge
char buffer[10];

void setup() {
  Serial.begin(9600);
  lcd.init(); // initialize the 16x2 lcd module
  lcd.createChar(0, gauge_left); // create special character on position 0, gauge left [
  lcd.createChar(1, gauge_center); // create special character on position 1, gauge center _
  lcd.createChar(2, gauge_right); // create special character on position 2, gauge right ]
  lcd.createChar(3, gauge_fill); // create special character on position 3, gauge fill ■
  servo.attach(servoPin);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(echoPin2, INPUT);
  pinMode(trigPin2, OUTPUT);
}

```

```
servo.write(0); //close cap on power on
lcd.backlight(); // enable backlight for the LCD module
lcd.setCursor(0,0);
lcd.print("INITIALIZATION");
delay(100);
servo.detach();
}
```

```
void measure() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(30);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);
  dist = (duration / 2) / 29.1; //obtain distance
}
```

```
void measure2() {
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(30);
  digitalWrite(trigPin2, LOW);
  pinMode(echoPin2, INPUT);
  duration2 = pulseIn(echoPin2, HIGH);
  dist2 = (duration2 / 2) / 29.1; //obtain distance
}
```



```
void loop() {
  for (int i = 0; i <= 2; i++) { //average distance
    measure();
    aver[i] = dist;
    delay(10); //delay between measurements
  }
  dist = (aver[0] + aver[1] + aver[2]) / 3;
```

```
if(active == 0){
  for (int i = 0; i <= 2; i++) { //average distance
    measure2();
    aver2[i] = dist2;
    delay(10); //delay between measurements
  }
```

```

dist2 = (aver2[0] + aver2[1] + aver2[2]) / 3;
percent = ((float)dist2/50.0) * 100.0;
if (percent > 100)
{
    percent = 100;
}
Serial.print(percent);
Serial.print("Distance: ");
Serial.println(dist2);

lcd.setCursor(0,0);           // move cursor to top left
sprintf(buffer, "LVL:%3d%%", percent); // set a string as CPU: XX%, with the number
always taking at least 3 character
lcd.print(buffer);           // print the string on the display

// 8 characters to draw the gauge --- 
float cpu_gauge_step = 100.0/8.0; // 100% is the maximum number, gauge is 8
characters wide, calculate one step
for (int i=0; i<8; i++) {
    if (percent <= cpu_gauge_step*i) { // value is smaller than step*i, draw "empty"
character
        if (i==0) {lcd.write(0);} // [ first cell, opening bracket
        else if (i==7) {lcd.write(2);} // ] last cell, closing bracket
        else {lcd.write(1);} // _ any other cell, lines top and bottom
    }
    else { // value is bigger than step*i, draw filled character - 
        lcd.write(3);
    }
}

if (dist < 20) {
//Change distance as per your need
    lcd.setCursor(0,1);
    lcd.print("TRASH BIN OPEN");
    servo.attach(servoPin);
    delay(1);
    servo.writeMicroseconds(1000);
    delay(1000);
    servo.detach();
    delay(2000);
    active++;
}
else if (percent >= 90)
{

```



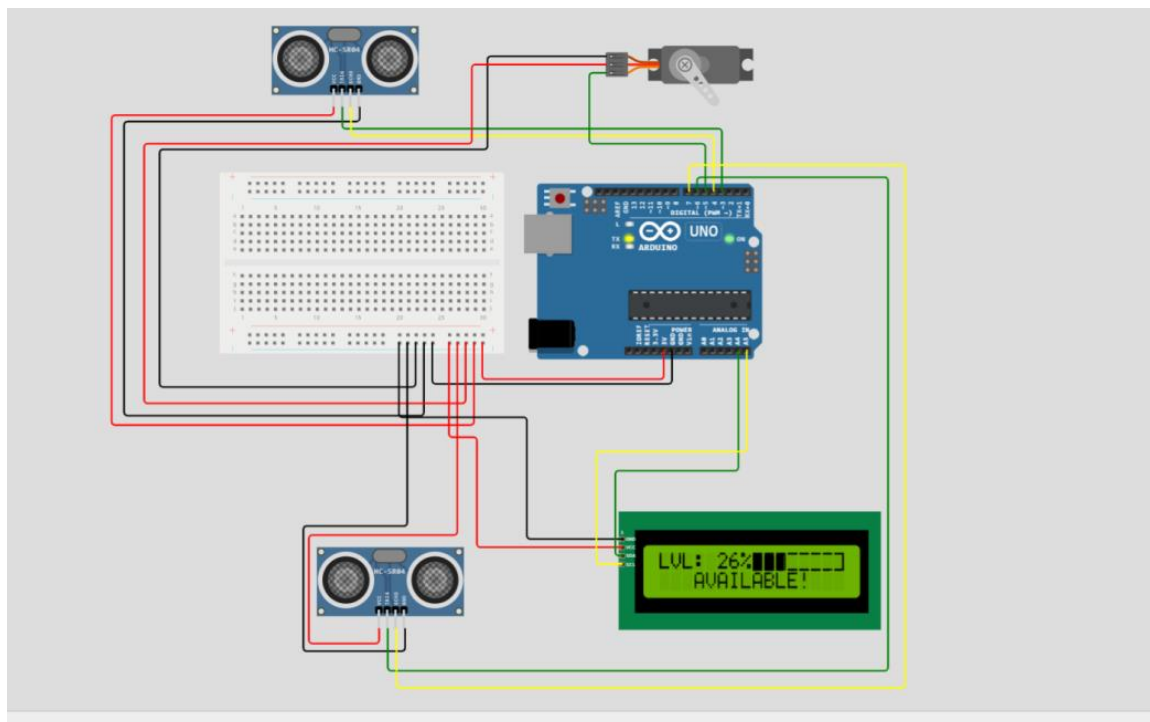
```

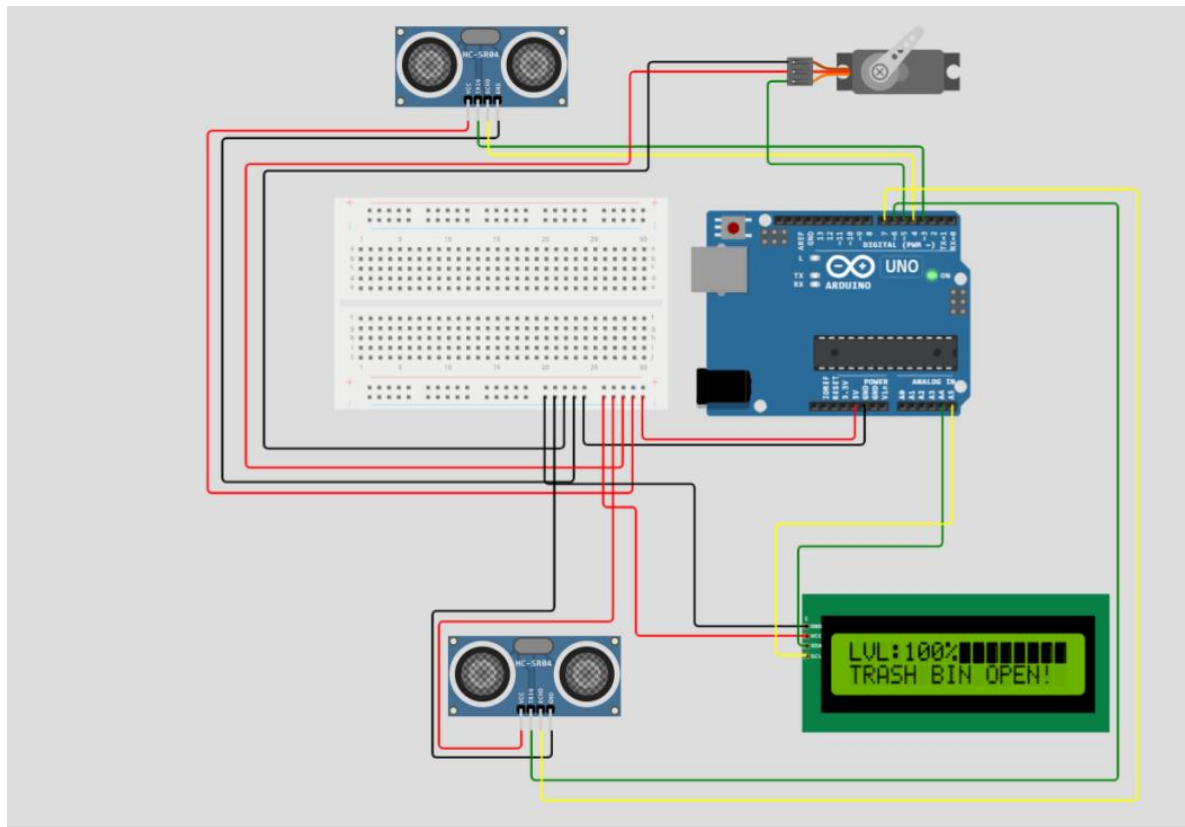
    lcd.setCursor(0,1);
    lcd.print("TRASH BIN FULL!");
  }
  else
  {
    lcd.setCursor(0,1);
    lcd.print("  AVAILABLE! ");
  }
}

else if (active > 0){
  if (dist >= 20){
    lcd.setCursor(0,1);
    lcd.print("TRASH BIN CIOSE");
    servo.attach(servoPin);
    delay(1);
    servo.writeMicroseconds(2000);
    delay(1000);
    servo.detach();
    active = 0;
  }
}
}

```

Output :





Result :

The simulation and circuit design of the Garbage Segregator and Bin Level Indicator system were successfully implemented using Wokwi Online Platform.

EX.NO:		IMAGE PROCESSING BASED FIRE DETECTION
DATE :		

Aim:

The aim of the experiment is to develop a Python program using OpenCV that detects the of fire in an image.

Apparatus & Software Required :

1. PC
2. Python software

Procedure :

1. To open the python Software.
2. Click the New File and Write the program.
3. Save the file in the name.py format.
4. To run the python File .
5. To upload the Image to the path.
6. Observe the Output in Terminal.

Theory :

Fire detection is a crucial aspect of safety and security systems in various environments such as buildings, forests, and industrial sites. Image processing offers a non-intrusive method for detecting fires in visual data. In this experiment, we utilize OpenCV, a powerful library for computer vision tasks in Python, to analyze video or image frames for signs of fire. The system identifies the characteristic features of flames, such as color, shape, and motion, to distinguish them from the background. By implementing algorithms for fire detection, we aim to create a reliable and efficient solution for early fire detection and alerting.

Program :

```
import cv2

import numpy as np

import tkinter as tk

from tkinter import filedialog
```

```

def detect_fire(frame):

    blur = cv2.GaussianBlur(frame, (21, 21), 0)

    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)

    lower = np.array([0, 50, 50]) # Lower bound for red color in HSV
    upper = np.array([10, 255, 255]) # Upper bound for red color in HSV

    mask = cv2.inRange(hsv, lower, upper)

    output = cv2.bitwise_and(frame, frame, mask=mask)

    # Count the number of non-zero pixels in the mask
    no_red = cv2.countNonZero(mask)

    # If the number of non-zero pixels exceeds a threshold, consider it as fire detected
    if no_red > 15000:

        return True

    else:

        return False

def upload_image():

    root = tk.Tk()

    root.withdraw() # Hide the main window

    file_path = filedialog.askopenfilename() # Show the file dialog and return the
selected file path

    return file_path

def main():

    image_path = upload_image() # Obtain the image path from the file dialog

    if image_path: # Check if the user selected an image

```

```
# Load the image

frame = cv2.imread(image_path)

if frame is not None: # Check if the image is successfully loaded

    frame = cv2.resize(frame, (960, 540))

    cv2.imshow("Original", frame)

    fire_detected = detect_fire(frame)

    if fire_detected:

        print("Fire detected!")

        # You can add further actions here such as sounding an alarm or sending
        notifications.

    else:

        print("No fire detected.")

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

    print("Error: Failed to load the image.")

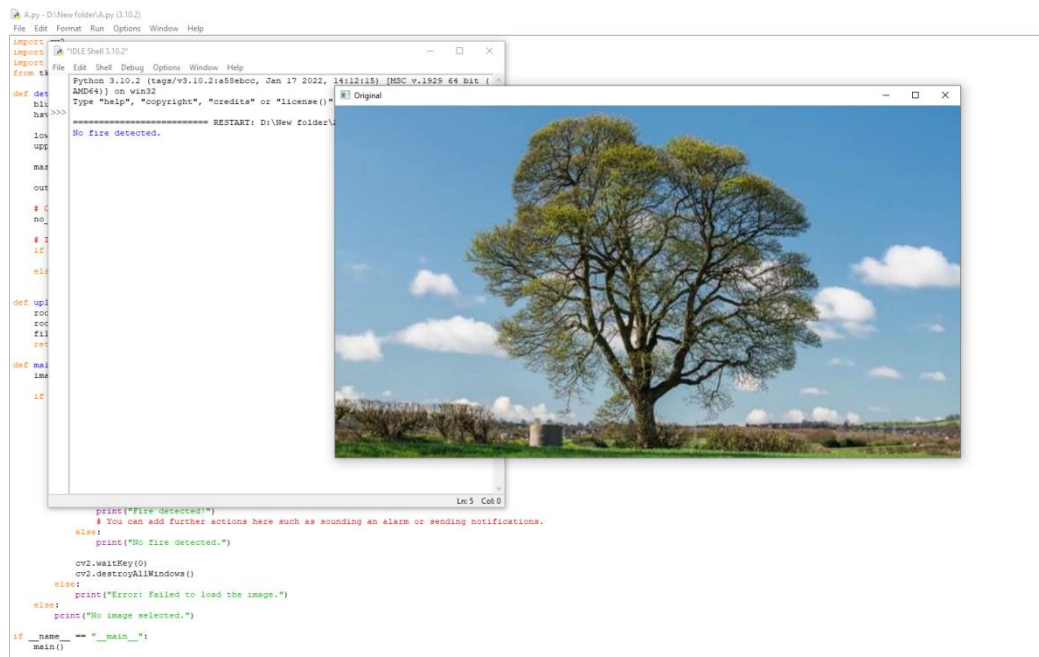
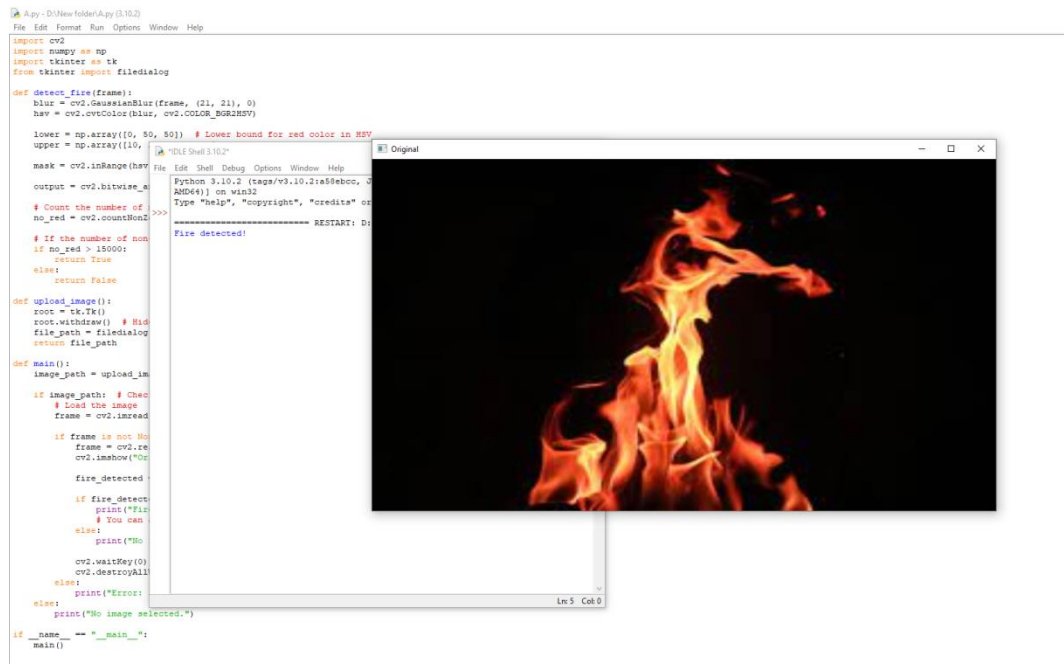
else:

    print("No image selected.")

if __name__ == "__main__":

    main()
```

Output :



Result :

The developed fire detection system demonstrates promising results in identifying fire occurrences in images and videos. The system successfully detects flames based on their characteristic features and distinguishes them from non-fire elements in various environments.

EX.NO:		VEHICLE NUMBER PLATE DETECTION
DATE :		

Aim:

The aim of this experiment is to develop a system for detecting vehicle number plates using OpenCV and Python.

Apparatus & Software Required :

1. PC
2. Python software

Procedure :

1. To open the python Software.
2. Click the New File and Write the program.
3. Save the file in the name.py format.
4. To run the python File .
5. To upload the Image to the path.
6. Observe the Output in Terminal.

Theory :

Vehicle number plate detection is a crucial task in various applications such as traffic monitoring, automatic toll collection, and law enforcement. In this experiment, we utilize OpenCV, an open-source computer vision library, along with Python programming language to implement a number plate detection system. The process involves several steps including image preprocessing, edge detection, contour detection, and character recognition. By leveraging these techniques, we aim to accurately locate and extract number plates from input images or video streams.

Program :

```
import cv2

import numpy as np

from tkinter import filedialog

import tkinter as tk
```

```
plat_detector = cv2.CascadeClassifier(cv2.data.harcascades +  
"haarcascade_russian_plate_number.xml")
```

```
# Create a Tkinter window to select file
```

```
root = tk.Tk()
```

```
root.withdraw() # Hide the root window
```

```
# Ask user to select the image file
```

```
file_path = filedialog.askopenfilename()
```

```
if not file_path: # Check if user canceled the selection
```

```
    print("No file selected. Exiting...")
```

```
    exit()
```

```
# Read the selected image
```

```
img = cv2.imread(file_path)
```

```
if img is None: # Check if image loading was successful
```

```
    print("Error: Unable to read the image.")
```

```
    exit()
```

```
# Detect license plates in the image
```

```
plates = plat_detector.detectMultiScale(img, scaleFactor=1.2, minNeighbors=5,  
minSize=(25, 25))
```


for (x, y, w, h) in plates:

```
cv2.putText(img, text='License Plate', org=(x - 3, y - 3),  
fontFace=cv2.FONT_HERSHEY_COMPLEX, color=(0, 0, 255), thickness=1,  
fontScale=0.6)
```

```
img[y:y+h, x:x+w] = cv2.blur(img[y:y+h, x:x+w], ksize=(1, 1))
```

```
cv2.rectangle(img, (x, y), (x+w, y+h), (100, 200, 150), 2)
```

Resize the output image including detected plates

resize_factor = 0.5 # Adjust the resize factor as needed

```
resized_img = cv2.resize(img, (int(img.shape[1] * resize_factor), int(img.shape[0] *  
resize_factor)))
```

Display the resized image with detected plates

```
cv2.imshow('Plates', resized_img)
```

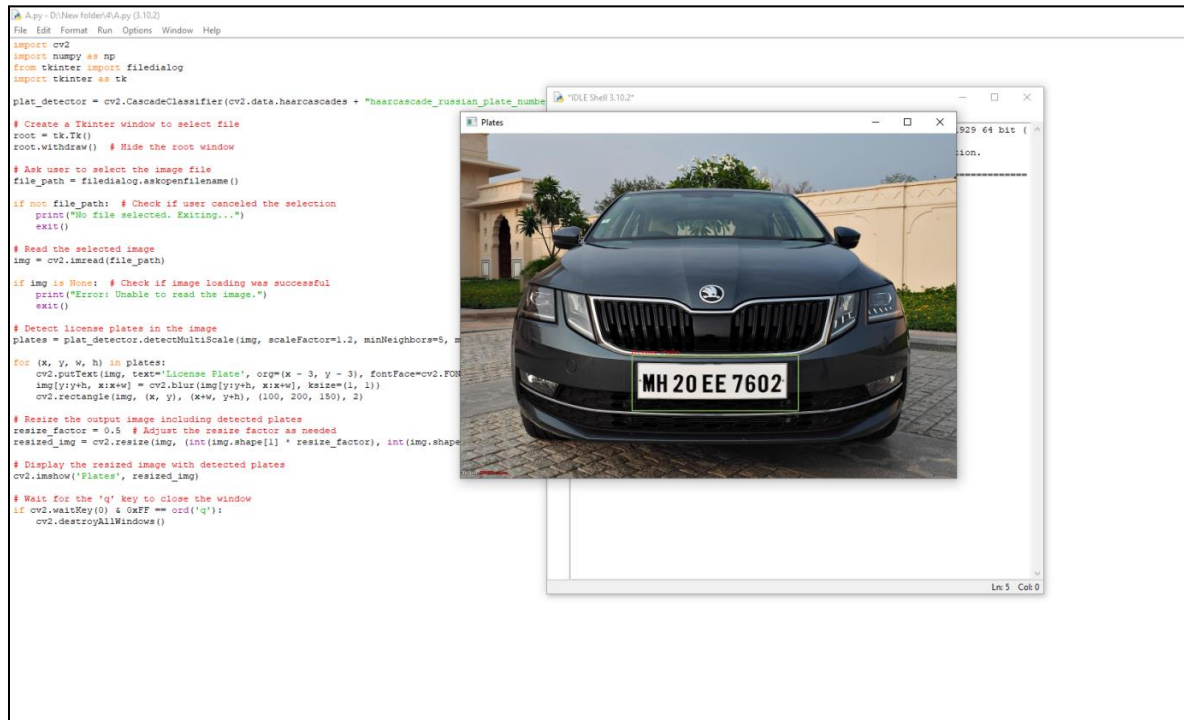
Wait for the 'q' key to close the window

```
if cv2.waitKey(0) & 0xFF == ord('q'):
```

```
cv2.destroyAllWindows()
```

Output :





Result :

The vehicle number plate detection system successfully detects and recognizes number plates from the input images, demonstrating the effectiveness of the implemented approach in real-world scenarios.

EX.NO:		SMART LOCK SYSTEM
DATE :		

Aim:

The aim of this experiment is to design and simulate a Smart Lock System and verify its output.

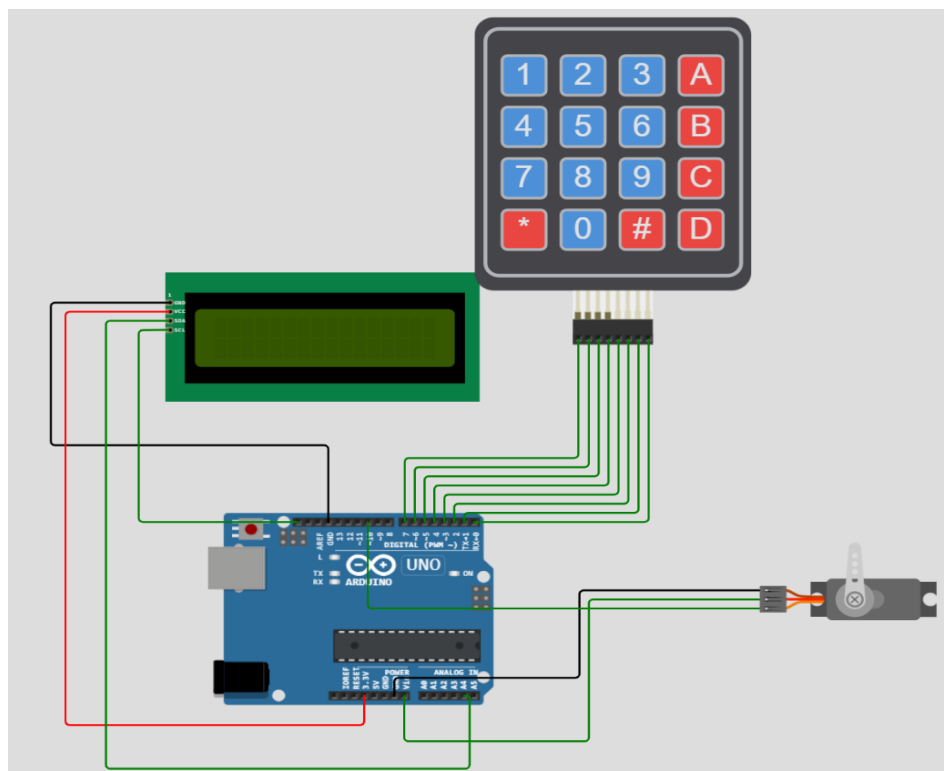
Apparatus & Software Required :

- PC
- Wokwi Online Simulation Platform

Theory :

The Smart Lock System is an advanced security mechanism that allows access to a door or a lock using electronic means rather than traditional physical keys. It employs various technologies such as RFID (Radio Frequency Identification), Bluetooth, or biometric recognition to authenticate and authorize users. The system typically consists of a controller unit, a locking mechanism, and sensors. When a user attempts to access the locked area, the system verifies their identity based on the provided credentials. If the authentication is successful, the locking mechanism is triggered to unlock the door, granting access to the authorized user.

Circuit Diagram :



Pin Details:

LCD DISPLAY	MIRCOCONTROLLER
VCC	3.3V
GND	GND
SDA	SDA
SCL	A4
SERVO	PIN
VCC	V _{in}
GND	GND
PWM	10
KEYBOARD	PIN
R1	7
R2	6
R3	5
R4	4
C1	3
C2	2
C3	1
C4	0

Procedure :

1. Open the Google Chrome and Open the <https://wokwi.com/> Online Simulation platform.
2. To select the Components to start the designs per circuit diagram.
3. To write and upload the coding from the simulation platform .
4. Finally Verified the output of the simulation.

Program :

```
#include <Keypad.h>    // the library for the 4x4 keypad
#include <LiquidCrystal_I2C.h> // the library for the i2c 1602 lcd
#include <Servo.h>    // the library to control the servo motor
LiquidCrystal_I2C lcd(0x27,16,2); // gets the lcd
Servo servo;

#define Password_Length 5 // the length of the password, if the password is 4 digits long set
this to 5
int Position = 0; // position of the servo
char Particular[Password_Length]; // the password length
char Specific[Password_Length] = "1234"; // the password which is called specific in the
code, change this to anything you want with the numbers 0-9 and the letters A-D
byte Particular_Count = 0, Specific_Count = 0; // counts the amount of digits and and checks
to see if the password is correct
char Key;
const byte ROWS = 4; // the amount of rows on the keypad
const byte COLS = 4; // the amount of columns on the keypad
char keys[ROWS][COLS] = { // sets the rows and columns
    // sets the keypad digits
    {'1','2','3','A'},

    {'4','5','6','B'},

    {'7','8','9','C'},

    {'*','0','#','D'}
};
bool SmartDoor = true; // the servo
// the pins to plug the keypad into
byte rowPins[ROWS] = {7, 6, 5, 4};
byte colPins[COLS] = {3, 2, 1, 0};
Keypad myKeypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS); // gets the data
from the keypad

// locked charcater
byte Locked[8] = {
    B01110,
    B10001,
    B10001,
    B11111,
    B11011,
```

```

    B11011,
    B11011,
    B11111
};
// open character
byte Opened[8] = {
    B01110,
    B00001,
    B00001,
    B11111,
    B11011,
    B11011,
    B11011,
    B11111
};
void setup()
{
    servo.attach(10); // attaches the servo to pin 10
    ServoClose();    // closes the servo when you say this function
    lcd.init();       // initializes the lcd
    lcd.backlight();  // turns on the backlight
    lcd.setCursor(0,0); // sets the cursor on the lcd
    lcd.print("Vector X"); // prints the text/charater
    lcd.setCursor(0,1); // sets the cursor on the lcd
    lcd.print("Arduino Lock!!!"); // prints text
    delay(4000);      // waits 4 seconds
    lcd.clear();       // clears the lcd diplay
}

void loop()
{
    if (SmartDoor == 0) // opens the smart door
    {
        Key = myKeypad.getKey(); // the word key = myKeypad which gets the value

        if (Key == '#') // when the '#' key is pressed

        {
            lcd.clear( ); // clears the lcd diplay
            ServoClose( ); // closes the servo motor
            lcd.setCursor(2,0); // sets the cursor on the lcd
            lcd.print("Door Closed"); // prints the text to the lcd
            lcd.createChar(0, Locked); // prints the locked character
        }
    }
}

```

```

    lcd.setCursor(14,0); // sets the cursor on the lcd
    lcd.write(0); // prints the first character when you are on the door closed page
    delay(3000); // waits 3 seconds
    SmartDoor = 1; // closes the door
  }
}

else Open(); // keeps the door open
}

void clearData( ) // clears the data
{
  while (Particular_Count != 0) // counts the digits pressed
  {
    Particular[Particular_Count--] = 0; // counts how many digits
  }
  return; // returns the data
}

void ServoOpen( ) // opens the servo
{
  for (Position = 180; Position >= 0; Position -= 5) { // moves from 0 to 180 degrees
    servo.write(Position); // moves to the position
    delay(15); // waits 15 milliseconds
  }
}

void ServoClose() // closes the servo
{
  for (Position = 0; Position <= 180; Position += 5) { // moves from position 0 to 180 degrees
    servo.write(Position); // moves to the position
    delay(15); // waits 15 milliseconds
  }
}

void Open() // function declarations
{
  lcd.setCursor(1,0); // sets the cursor on the lcd
  lcd.print("Enter Password:"); // prints the text

  Key = myKeypad.getKey(); // gets the keys you press from the keypad
  if (Key)
  {
    Particular[Particular_Count] = Key;
  }
}

```

```

    lcd.setCursor(Particular_Count, 1); // sets the cursor on the lcd
    lcd.print("*"); // prints '*' instead of the password
    Particular_Count++; // counts the length of the password
}

if (Particular_Count == Password_Length - 1) // gets the length of the password
{
    if (!strcmp(Particular, Specific)) // counts the length and checks to see if the password is
correct
    {
        lcd.clear();
        ServoOpen(); // moves the servo 180 degrees
        lcd.setCursor(2,0); // sets the cursor on the lcd
        lcd.print("Door Opened");
        lcd.createChar(1, Opened);
        lcd.setCursor(14,0); // sets the cursor on the lcd
        lcd.write(1);
        lcd.setCursor(0,1); // sets the cursor on the lcd
        lcd.print("Press # to Close");
        SmartDoor = 0;
    }
    else
    {
        lcd.clear();
        lcd.setCursor(0,0); // sets the cursor on the lcd
        lcd.print("Wrong Password"); // prints the text/character
        lcd.setCursor(0,1);
        lcd.print("Try Again In");
        lcd.setCursor(13,1);
        lcd.print("10");
        delay(1000);
        lcd.setCursor(13,1);
        lcd.print("09");
        delay(1000);
        lcd.setCursor(13,1);
        lcd.print("08");
        delay(1000);
        lcd.setCursor(13,1);
        lcd.print("07");
        delay(1000);
        lcd.setCursor(13,1);
        lcd.print("06");
        delay(1000);
        lcd.setCursor(13,1);
    }
}

```

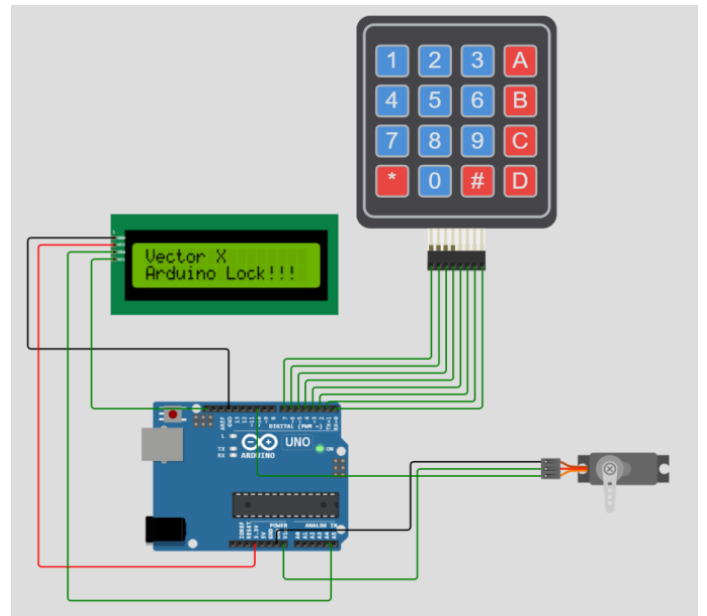
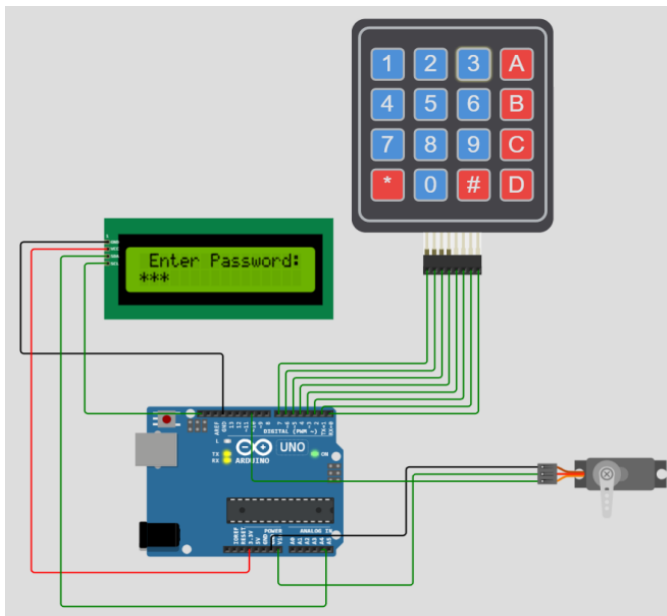


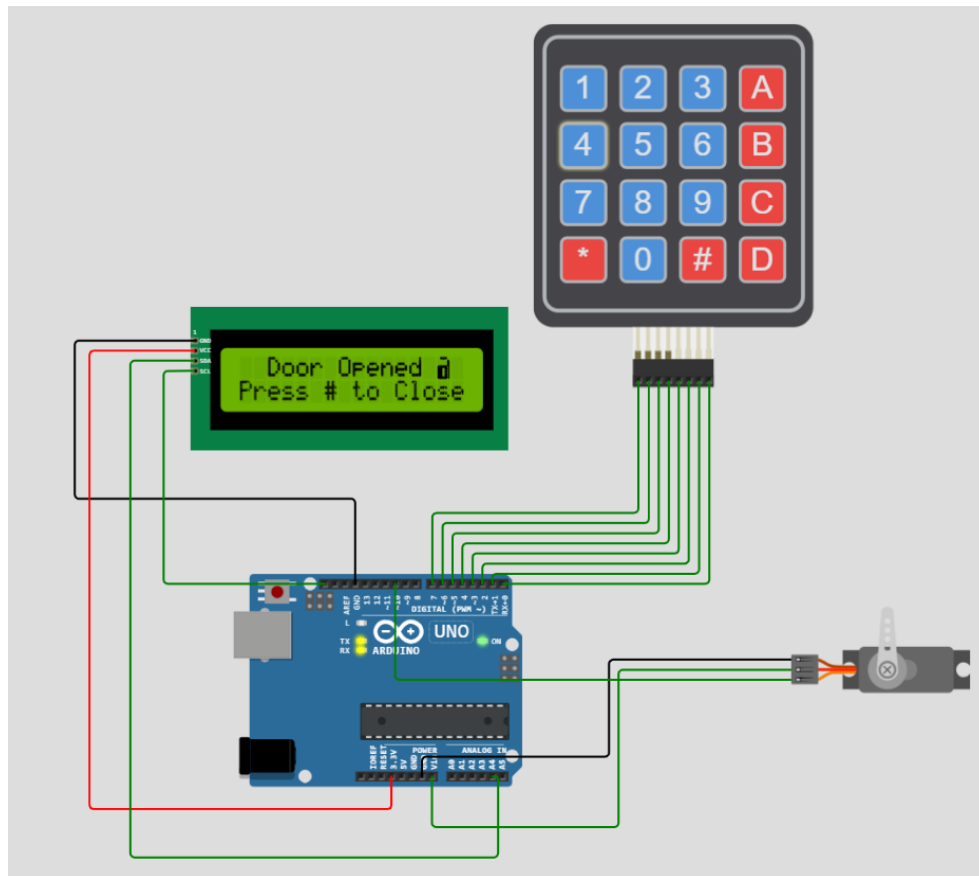
```

lcd.print("05");
delay(1000);
lcd.setCursor(13,1);
lcd.print("04");
delay(1000);
lcd.setCursor(13,1);
lcd.print("03");
delay(1000);
lcd.setCursor(13,1);
lcd.print("02");
delay(1000);
lcd.setCursor(13,1);
lcd.print("01");
delay(1000);
lcd.setCursor(13,1);
lcd.print("00");
delay(1000);
lcd.clear();
SmartDoor = 1; // closes the smart door
}
clearData(); // clears the data
}
}

```

Output :





Result :

The simulation of the Smart Lock System demonstrates its ability to authenticate users and grant access accordingly. The system successfully unlocks the door when presented with valid credentials, thus verifying its functionality and effectiveness as a security mechanism.