

## NAME : JANANII.P

**REG. NO. : 211521106065**

## DEGREE : B.E.

**BRANCH : ELECTRONICS AND COMMUNICATION ENGINEERING**

## YEAR/SEM : III YEAR/ VI SEM

**LAB CODE : ET3491**

## LAB NAME : EMBEDDED SYSTEMS AND IOT

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.**  **NO.** | **DATE** | **NAME OF THE EXPERIMENTS** | **MARK** | **SIGN** |
| **EXPERIMENTS USING 8051** | | | | |
| 1. |  | Programming Arithmetic and Logical  Operations in 8051 |  |  |
| 2. |  | Generation of Square waveform/Triangular  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** | | | | |
| 6. |  | 1. Interfacing ADC 2. Interfacing DAC |  |  |
| 7. |  | 1. Blinking of LEDs 2. Blinking of LCD |  |  |
| 8. |  | 1. Interfacing keyboard Matrix 2. Interfacing Stepper Motor. |  |  |
| **IOT MINI-PROJECTS** | | | | |
| 9. |  | Password Door Locking System |  |  |



**REGISTER NO .**

**Certified that this is the bonafide record of work done by**

**in the III year B.E. ELECTRONICS AND COMMUNICATION ENGINEERING in the**

### ET3491 - EMBEDDED SYSTEMS AND IOT during the academic year 2023-2024.

**Signature of the HOD Signature of the Staff-In-Charge Date:**

### Submitted for the Anna University Practical Examination held on

**at Panimalar Institute of Technology, Chennai-600 123.**

### INTERNAL EXAMINER EXTERNAL EXAMINER

**ARITHMETIC AND LOGICAL OPERATION USING 8051**

**EXP NO: 1 DATE :**

##### AIM:

To perform Arithmetic and logical operation experiment using simulator

##### SOFTWARE USED:

Keil uvision5

##### PROGRAM:

//ADDITION MOV A,#27H MOV B,#11H ADD A,B MOV 40H,A

// SUBTRACTION MOV A,#27H SUBB A,B

MOV 41H,A

//MULTIPLICATION MOV A,#27H

MOV B,#11H MUL AB MOV 42H,A MOV 43H,B

//DIVISION MOV A,#27H MOV B,#11H DIV AB MOV 44H,A MOV 45H,B

// AND OPERATION MOV A,#27H

MOV B,#11H ANL A,B MOV 46H,A

// OR OPERATION MOV A,#27H MOV B,#11H

ORL A,B MOV 47H,A

// EXOR OPERATION MOV A,#27H

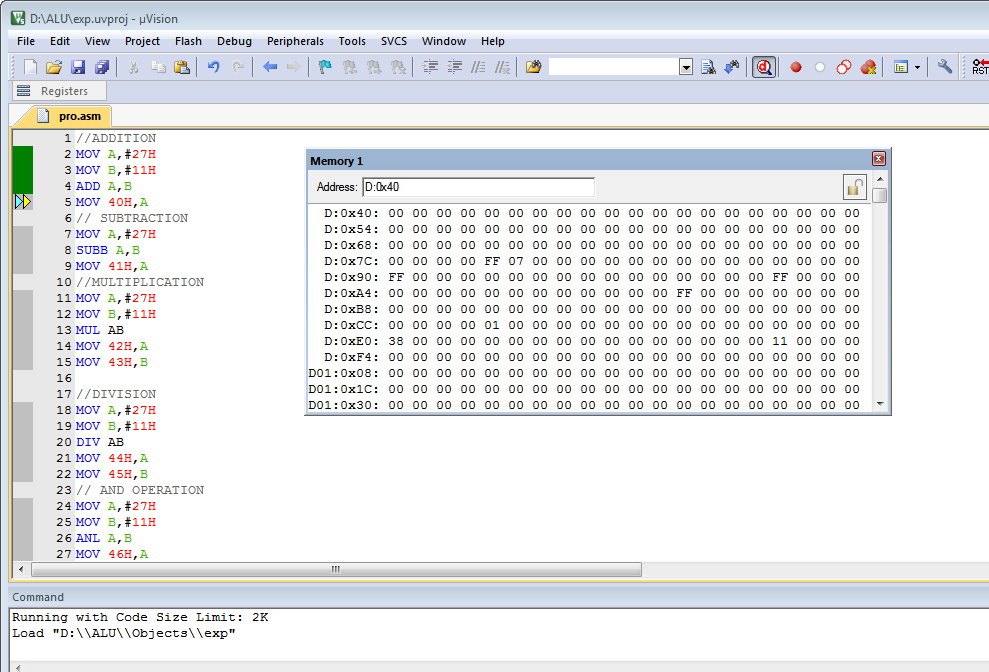
MOV B,#11H XRL A,B MOV 48H,A

// NOT OPERATION MOV A,#27H

CPL A MOV 49H,A

LCALL 0003H END

##### OUTPUT:



**RESULT:**

Thus the Arithmetic and logical operation was performed and got output successfully

**GENERATION OF SQUARE /TRIANGULAR WAVEFORM USING 8051**

**EXP NO: 2 DATE :**

##### AIM:

9

To perform Generation of square waveform/Triangular Waveform experiment using simulator

##### SOFTWARE USED:

Keil uvision5

##### PROGRAM: (Square wave)

ORG 00H SJMP START ORG 30H START:

MOV P0,#00H; AGAIN:

MOV P0,#0FFH;

ACALL DELAY;

MOV P0,#00H;

ACALL DELAY;

SJMP AGAIN;

DELAY:

MOV R0,#0FFH;

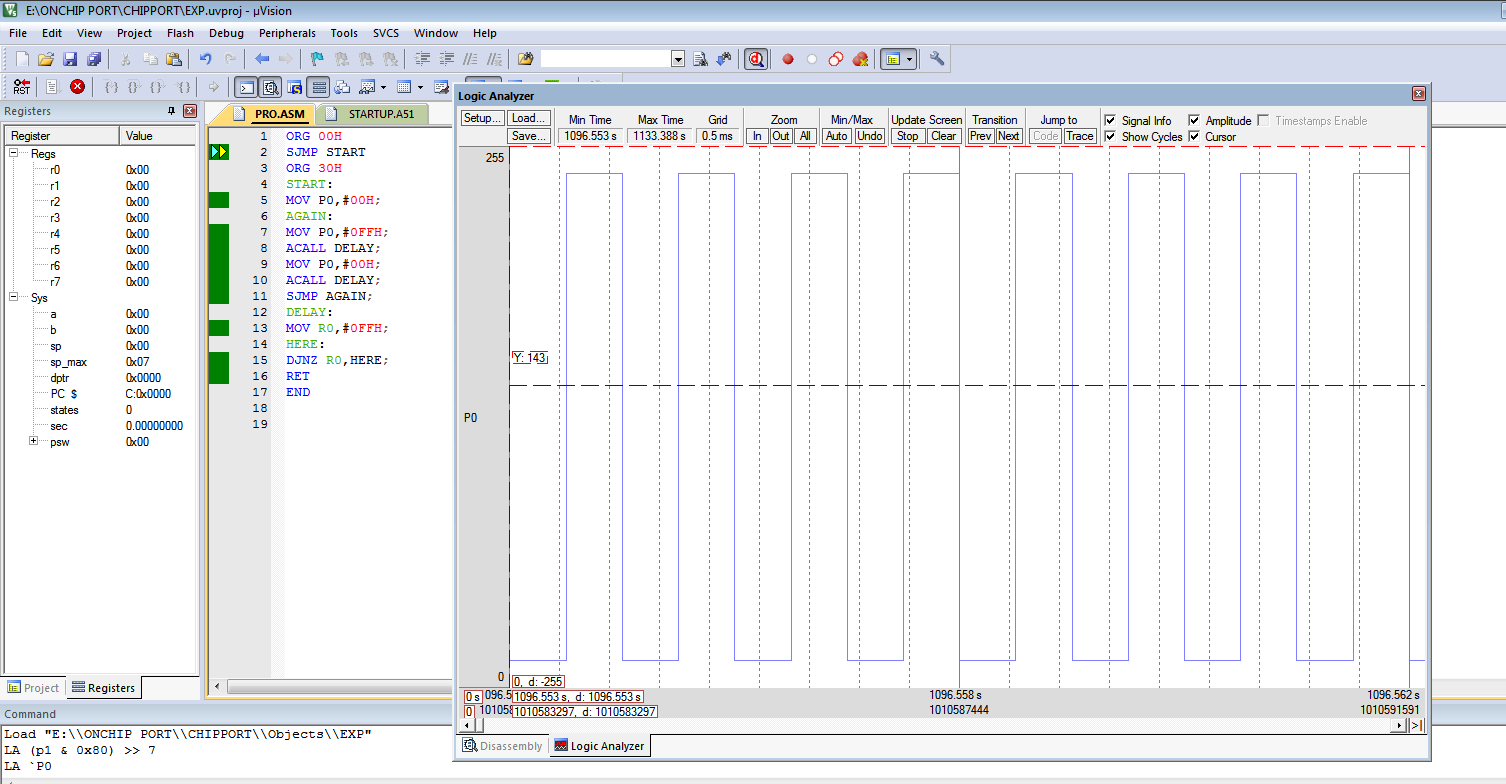
HERE:

DJNZ R0,HERE;

RET

END

##### OUTPUT IN LOGIC ANALYZER:



**PROGRAM: (Triangular wave)**

ORG 00H SJMP START ORG 30H START:

MOV P0,#00H; AGAIN:

MOV A,#00H LOOP:

MOV P0,A; ACALL DELAY; INC A;

jNC LOOP SJMP AGAIN; DELAY:

MOV R0,#0FFH;

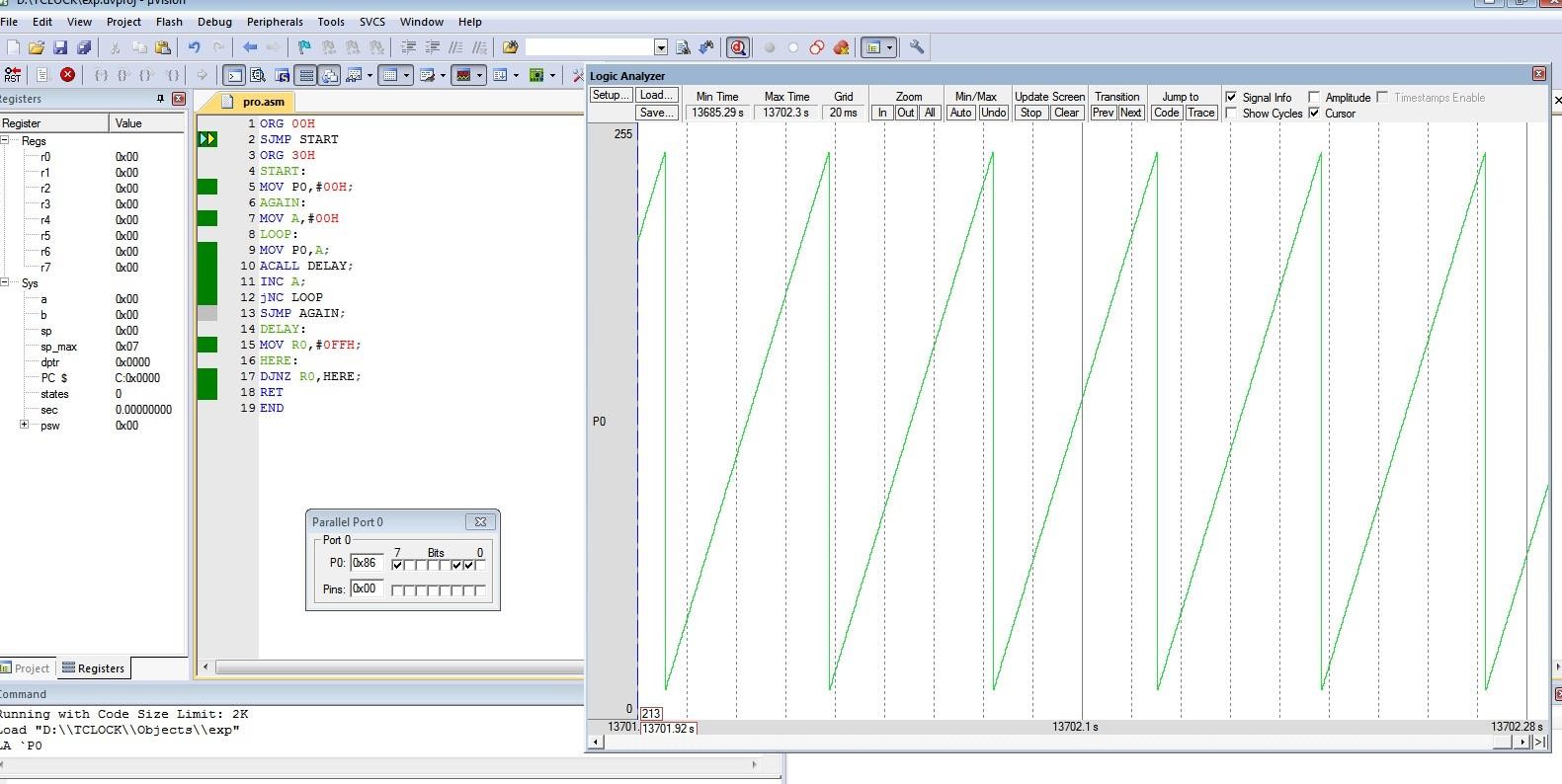
HERE:

DJNZ R0,HERE;

RET

END

##### OUTPUT IN LOGIC ANALYZER:



**RESULT:**

Thus the Generation of square waveform program was performed and got output successfully.

**PROGRAMMING USING ON-CHIP PORTS IN 8051**

**EXP NO: 3 DATE :**

##### AIM:

To perform programming using on-chip ports experiment using simulator

##### SOFTWARE USED:

Keil uvision5

##### PROGRAM:

#include <reg51.h>

void MSDelay (unsigned int); sbit Dsensor=P1^1;

sbit Buzzer=P1^7; void main(void)

{

Dsensor=1; //make p1.1 an input while (1)

{

while (Dsensor==1) //while it opens

{

Buzzer=0;

MSDelay(250); Buzzer=1; MSDelay(250);

}

}

}

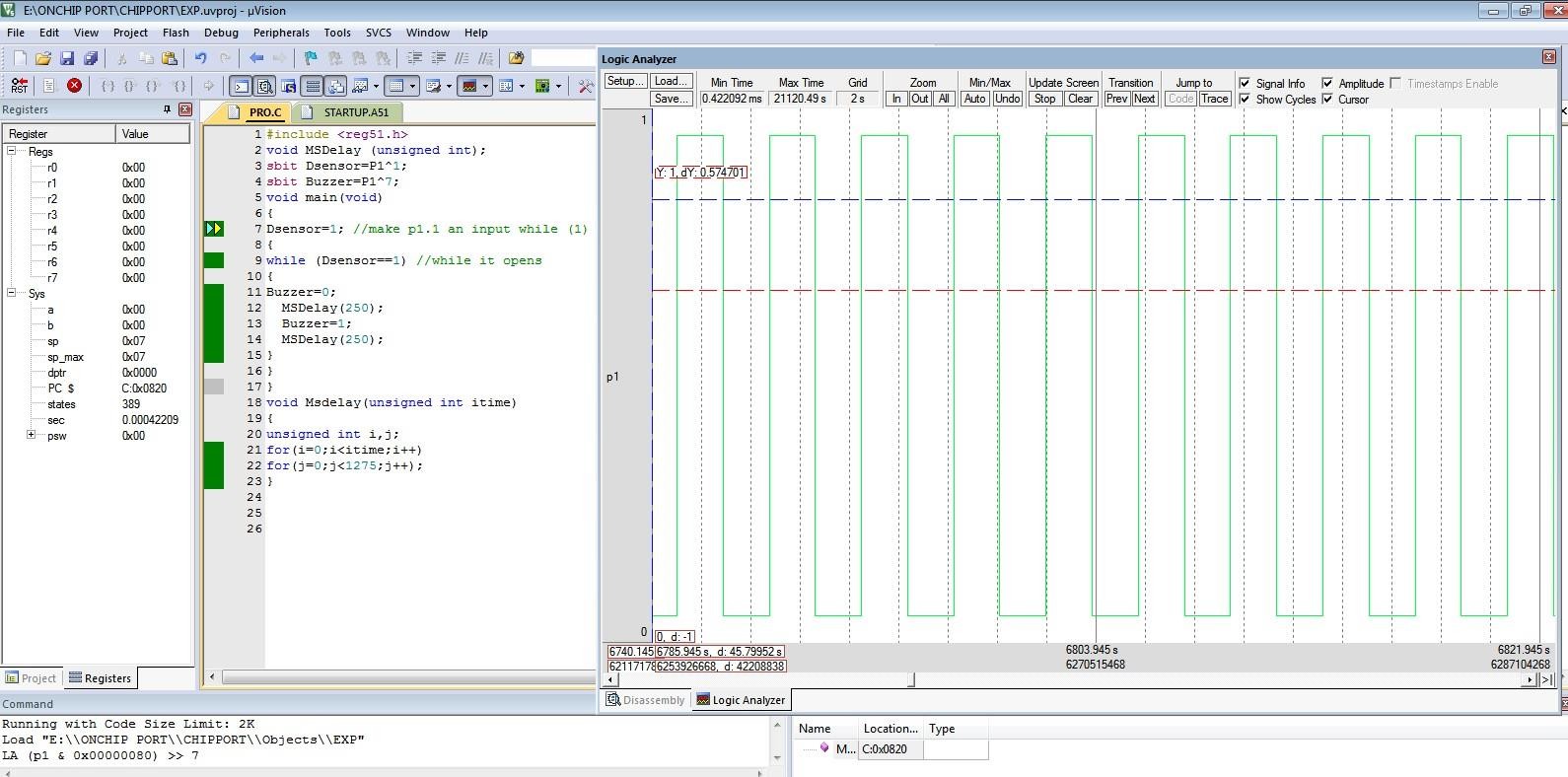
void Msdelay(unsigned int itime)

{

unsigned int i,j; for(i=0;i<itime;i++) for(j=0;j<1275;j++);

}

##### OUTPUT IN LOGIC ANALYZER:



**RESULT:**

Thus the programming using on-chip ports program was performed and got output successfully

##### AIM:

**PROGRAMMING USING SERIAL PORTS IN 8051**

**EXP NO: 4 DATE :**

To perform programming using serial ports experiment using simulator

##### SOFTWARE USED:

Keil uvision5

##### PROGRAM:

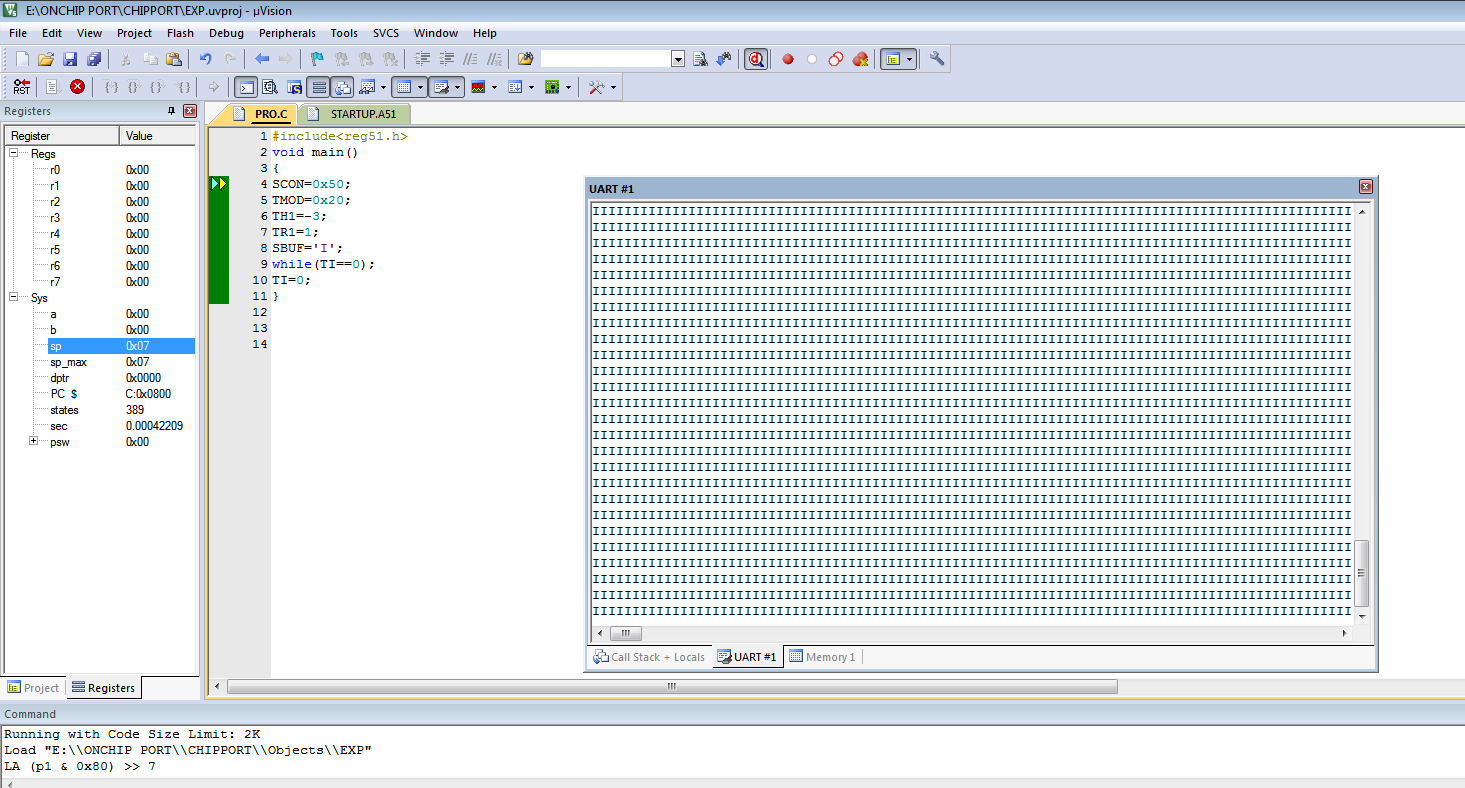
#include<reg51.h> void main()

{ SCON=0x50; TMOD=0x20; TH1=-3; TR1=1; SBUF='I';

while(TI==0); TI=0;

}

##### OUTPUT IN SERIAL WINDOW:



**RESULT:**

Thus the programming using serial ports program was performed and got output successfully.

17

##### AIM:

**DESIGN OF A DIGITAL CLOCK USING TIMERS/COUNTERS IN 8051**

**EXP NO: 5 DATE :**

To perform design of a digital clock using timers/counters experiment using simulator

##### SOFTWARE USED:

Keil uvision5

##### PROGRAM:

#include<reg51.h> void DELAY(void); void main(void)

{

int v1,v2; v1=0;

while(1)

{

P1=v1; // P1 =count from 00 to FF DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

v1=v1+1;

P1 =v1; // DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

DELAY();

}

}

void DELAY(void)

{

TMOD=0x01; // Timer 0 in mode 1 TH0=0x4B;

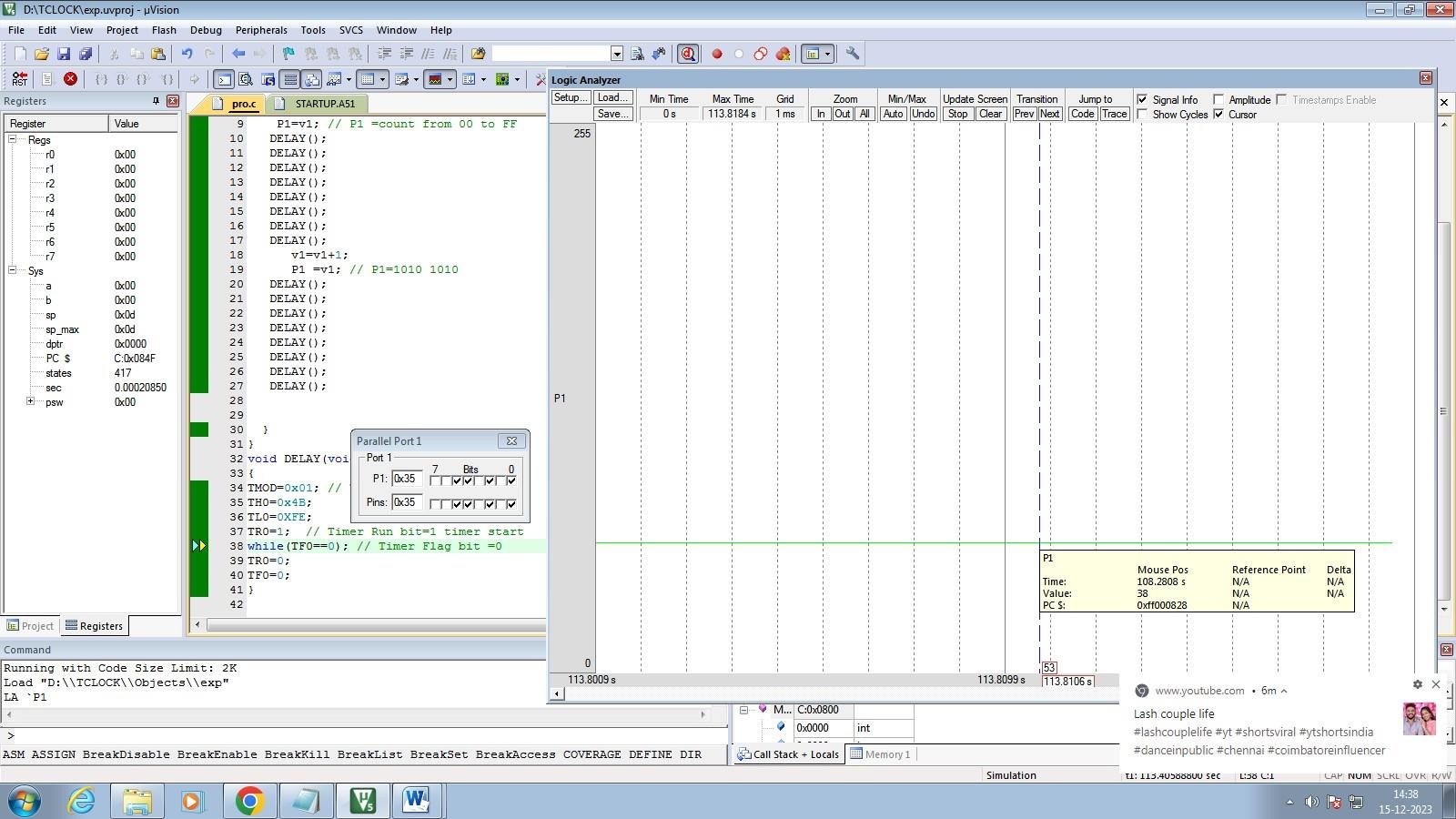
TL0=0XFE;

TR0=1; // Timer Run bit=1 timer start while(TF0==0); // Timer Flag bit =0 TR0=0;

TF0=0;

}

##### OUTPUT IN PORT (P1) WINDOW:



**COUNTER EXPERIMENT SEVEN SEGMENT DISPLAY**

#include<reg51.h> sbit com1=P1^0; sbit com2=P1^1; void main()

{

unsigned char disp[10]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90}; unsigned char v1,v3,v4;

unsigned int v2; com1=0; com2=0; P2=0x00; TMOD=0X60; TL1=0X00; TR1=1;

while(1)

{ v1=TL1;

v3=v1/10; //quotient v4=v1%10; //reminder com1=1;

com2=0;

P2=disp[v3];

for(v2=0;v2<500;v2++);

com1=0;com2=1; P2=disp[v4]; for(v2=0;v2<500;v2++);

com2=0;

}

}

##### RESULT:

Thus the design of a digital clock using timers/counters program was performed and got output successfully.

**INTERFACING ADC AND DAC**

###### Ex. No. 6(i) INTERFACING ADC WITH LPC 1768

**Date:**

###### Aim :

To interface ADC with LPC 1768.

###### Algorithm :

Step 1: Start the program

Step 2: Declare the variable to hold ADC converted value Step 3: Function is declared and defined for ADC initialization Step 4: Configure PO.23 as ADC input using PINSELI

Step 5: Declare the variable ADC data

Step 6: Select clock for ADC to start of conversion Step 7: Check end of conversion and read result Step 8: Return the 12 bit result to main function

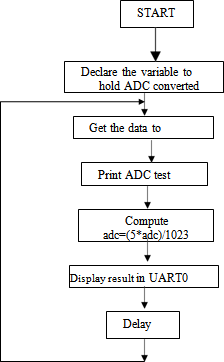
Step 9: Call the functions TargetResetInit(), ADC\_Init(), InitUART0() Step 10: Display ADC test

Step 11: Read the ADC value to convert

Step 12: Convert to volts using adc=(5\*adc)/1023 Step 13: Display the result on UART

Step 14: Stop

###### Flowchart :



**Program:**

#include "LPC17xx.h" #include "stdio.h" #include "UART0.h" #include "target.h"

#define PCADC 0x00001000 char adcreading[16] ;

void ADC\_Init (void)

{

LPC\_SC->PCONP |= PCADC ;

LPC\_PINCON->PINSEL1 = (LPC\_PINCON->PINSEL1 | 0x00004000); // P0.23 is configured as ADC Input

}

unsigned int ADC\_GetAdcReading ()

{

unsigned int adcdata;

LPC\_ADC->ADCR =0x01200301 ;

while(!((adcdata = LPC\_ADC->ADGDR) & 0x80000000))

{

}

return((adcdata >> 4) & 0x3ff) ; // Return 12 bit result

}

int main (void)

{

unsigned int delay; float adc;

TargetResetInit(); ADC\_Init() ; InitUart0(); printf("ADC Test");

while(1)

{

}

}

adc = ADC\_GetAdcReading(); adc=(5\*adc)/1023;

sprintf(adcreading,"\n ADC0 CH1= %.2f V",adc); UART0Puts(adcreading) ; for(delay=0;delay<60000;delay++);

###### Peripheral Interface:

To give analog input from Potentiometer R15 present in Analog Input region on ADTV1.1. Analog input range is from 0 to 1023.

###### Output:

You can see output on Hyper Terminal.

Therefore Open Hyper Terminal. Go to Start->All Programs->Accessories->Communications->

Hyper Terminal->Assign the Respective port-> Settings. Do proper settings (Baud Rate: 19200, Data Bits: 8, Stop Bits: 1, Echo: Off, Parity: None, Com Port: Com 1 (if other choose it)). Click on OK. Go to Port -> Open. If required Reset the ADT board. Now vary the R15 POT and hence see the change in voltage on Hyper Terminal.

###### Result:

Thus interfacing of ADC with LPC1768 is executed and verified successfully.

###### Ex. No. 6(ii) INTERFACING DAC WITH LPC 1768 DATE:

**Aim :**

To interface DAC with LPC1768.

###### Algorithm :

Step 1: Start the program

Step 2: Declare the variable to store output of DAC Step 3: Call Target Reset Init() function

Step 4: Clock selection for DAC is done Step 5: Select the DAC output pin

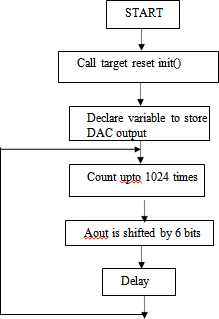
Step 6: Call delay for sometime

Step 7: Aout value is shifted right by 6bits for the loop to count from 0 to 1024 Step 8: Call Delay

Step 9: Again the loop count for 1024 times

Step 10: Aout value is again shifted right by 6times Step 11: Stop

**FLOW CHART:**



###### Program:

#include "LPC17xx.h" #include <stdio.h> #include "target.h"

unsigned int delay,AoutValue = 0 ;

int main (void)

{

TargetResetInit();

LPC\_PINCON->PINSEL1 |= 0x00200000 ;

while(1)

{

for(AoutValue = 0 ; AoutValue < 1024 ; AoutValue ++)

{

for(delay=0;delay<1000;delay++); LPC\_DAC->DACR = AoutValue << 8 ;

}

for(AoutValue = 1024 ; AoutValue > 0 ; AoutValue --)

{

for(delay=0;delay<1000;delay++); LPC\_DAC->DACR = AoutValue << 8 ;

}

}

}

###### Output:

You can see ramp wave on oscilloscope.

**Note**: Keep S11.2 switch in OFF position, after execution of program

###### Result:

Thus interfacing of DAC with LPC1768 is executed and verified successfully.

###### Exercise:

1. Develop an embedded C code to generate sine wave form using DAC.

#### BLINKING OF LEDS AND LCD

###### Ex. No. 7(i) Interfacing LEDS with LPC1768 Date:

**Aim :**

To interface the LEDs with LPC1768

###### Algorithm :

Step 1: Start the program

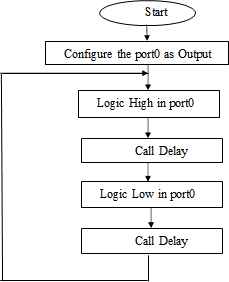
Step 2: Configure Port0 as GPI0 Step 3: Interfacing Port0 as Output Step 4: Logic High on Port0

Step 5: Call Delay

Step 6: Logic Low on Port0 Step 7: Call Delay

Step 8: Go to step 4

###### Flow Chart:



**Program:**

#include "LPC17xx.h" volatile unsigned int delay; int main (void)

{

LPC\_PINCON->PINSEL0 = 0x00000000; LPC\_GPIO0->FIODIR = 0x00000FF0;

while(1)

{

LPC\_GPIO0->FIOCLR =0x00000FF0;

for(delay=0; delay<500000; delay++); LPC\_GPIO0->FIOSET =0x00000FF0;

for(delay=0; delay<500000; delay++);

}

}

**Peripheral Interface:** Keep SW19.2 switch in ON position. 8 LEDs (D11 to D18) present on ADT are connected to P0.4,P0.5,P0.6,P0.7,P0.8,P0.9,P0.10 and P0.11 respectively by Common Anode method.

###### Output:

You can see blinking of LEDs.

**Note**: Keep SW19.2 switch in OFF position to save power, after execution of program.

###### Result:

Thus interfacing of leds with LPC1768 is executed and verified successfully.

###### Exercise:

1. Develop an embedded C code to blink LED with different delays (5 secs ,10 secs ) using the software delay.
2. Develop an embedded C code to blink LED with different sequence.

###### Ex. No. 7(ii) INTERFACING OF LCD DATE:

**Aim :**

To interface LCD with LPC1768

###### Algorithm :

Step 1: Start the program

Step 2: Write delay function for small delay and large delay Step 3: Send command on the data lines

Step 4: Clear Reset (RS) and enable (EN) Step 5: Set the enable PIN and call delay Step 6: Clear the enable (EN) and call delay Step 7: Send the data on data lines (D0 to D7)

Step 8: After sending the data Reset (RS) is set and call delay Step 9: Clear the pin EN and call delay

Step 10: Set the pin EN and Call delay Step 11: Clear the pin EN and call delay

Step 12: Configure the pins PO.15 (D4), PO.17 (D5), PO.22 (D6) and O.30 (D7) as output Step 13: Clear all the pins and configure the pins as GPIO and clear all pins

Step 14: Call the function LCD cmd and call delay

Step 15: Delay function is written which has pass pointer to character string Step 16: Display the character using display data

Step 17: Stop

###### Program:

#include "LPC17xx.h" #include <stdio.h> #include "LCD.h" #include "target.h"

int main ()

{

volatile int l; TargetResetInit(); LcdInit(); while(1)

{

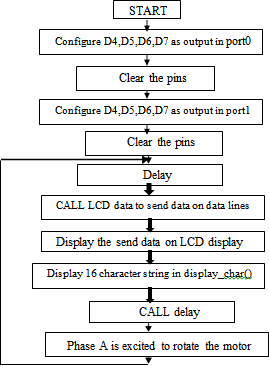
DisplayRow (1," SPJ EMBEDDED ");

DisplayRow (2," Technologies ");

}

}

###### Flow Chart:



**Peripheral Interface:**

No switches are there to turn ON/OFF I2C EEPROM

###### Output:

In this program after pressing any key from SW2 to SW17, its code will be displayed on 16x2 Text LCD.

###### Result:

Thus interfacing of LCD with LPC1768 is executed and verified successfully

###### Exercise:

Develop an embedded C code to scroll the text from left to right in LCD display.

#### INTERFACING KEYBOARD AND STEPPER MOTOR

###### Ex. No. 8(i) INTERFACING MATRIX KEYBOARD DATE:

**Aim :**

To interface Matrix Keyboard with LPC1768.

###### Algorithm :

Step 1: Start the program

Step 2: Define the ports P1.21, P1.20, P1.19, P1.18 for RET pins Step 3: For scanning assign pins P2.25, p1.26, P1.23, P1.22

Step 4: Assign crystal frequency 12KHz and PLL multiplier as 1 Step 5: Direct the Rs, Rw of LCD as GPIO

Step 6: Call the delay

Step 7: Check whether the key is down or up Step 8: If the key is down return 1

Step 9: If key is not pressed check for other key

Step 10: if already the key is pressed wait until it is pressed

Step 11: Return the key has been released and it is enable 10-1 also return 0 Step 12: Stop

###### Program:

#include "LPC17xx.h" #include <stdio.h> #include <string.h> #include "target.h" #include "UART0.h" #include "KBD.h"

int i8ch ;

char szTemp[16] ; int main (void)

{

TargetResetInit(); InitUart0();

UART0Puts("Keypad Test \n"); while(1)

{

i8ch = KBD\_rdkbd() ; // Read Keyboard

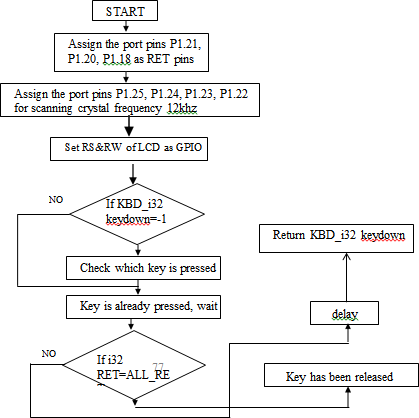
sprintf(szTemp,"\nKeyCode = %02X",i8ch); // Convert keycode into ASCII to display it on LCD

UART0Puts(szTemp) ; // Display keycode on 2nd line of LCD

}

}

###### Flow chart:



**Result:**

Thus interfacing of matrix keyboard is executed and verified successfully.

###### Ex. No. 8(ii) INTERFACING STEPPER MOTOR DATE:

**Aim :**

To interface Stepper Motor with LPC1768.

###### Algorithm :

Step 1: Start the program

Step 2: Assign the values for the phase A,B,C,D

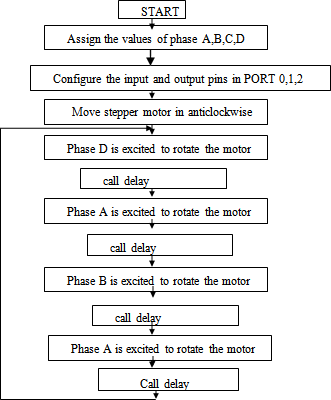
Step 3: Configure the pins as GPIO and assign the pin as output in Port0 and set the LED as OFF Step 4: Configure the pins P0.28 to P0.31 as GPIO and also as output

Step 5: Set the LED OFF in Port1

Step 6: Configure the pins P0.28 to p0.31 as GPIO and as output in Port2 Step 7: Set the LEDs OFF

Step 8: Move the Stepper Motor in clockwise direction Step 9: Phase D is excited to rotate in clockwise direction Step 10: Phase C is excited to rotate motor and call delay Step 11: Phase B is excited to rotate the motor and call delay Step 12: Phase A is excited to rotate motor and call delay Step 13: Stop

###### Flow chart:



**Program:**

#include "LPC17xx.h"

#define PHASEA 0x00000009 // Phase A value 1001 #define PHASEB 0x00000002 // Phase B value 0010 #define PHASEC 0x00000003 // Phase C value 0011 #define PHASED 0x0000000A // Phase D value 1010

unsigned int delay ; // Delay function definition int main(void)

{

LPC\_PINCON->PINSEL0 = 0x00000000;

LPC\_GPIO0->FIODIR = LPC\_GPIO0->FIODIR | 0x00000001;

LPC\_PINCON->PINSEL0 = 0x00000000;

LPC\_GPIO0->FIODIR = LPC\_GPIO0->FIODIR | 0x00000002;

LPC\_PINCON->PINSEL0 = 0x0000000;

LPC\_GPIO0->FIODIR = LPC\_GPIO0->FIODIR | 0x00010000;

LPC\_PINCON->PINSEL2 = 0x0000000;

LPC\_GPIO2->FIODIR = LPC\_GPIO2->FIODIR | 0x00000008;

while(1)

{

// Move stepper motor in anti-clockwise direction

LPC\_GPIO0->FIOCLR = LPC\_GPIO0->FIOCLR | 0x00000001; LPC\_GPIO0->FIOSET = LPC\_GPIO0->FIOSET | 0x00000002; LPC\_GPIO0->FIOCLR = LPC\_GPIO0->FIOCLR | 0x00010000;

LPC\_GPIO2->FIOSET = LPC\_GPIO2->FIOSET | 0x00000008; // Phase D is excited to rotate the motor for(delay=0; delay<80000; delay++) ; // Small time Delay

LPC\_GPIO0->FIOSET = LPC\_GPIO0->FIOSET | 0x00000001; LPC\_GPIO0->FIOCLR = LPC\_GPIO0->FIOCLR | 0x00000002; LPC\_GPIO0->FIOCLR = LPC\_GPIO0->FIOCLR | 0x00010000;

LPC\_GPIO2->FIOSET = LPC\_GPIO2->FIOSET | 0x00000008; // Phase A is excited to rotate the motor for(delay=0; delay<80000; delay++) ; // Small time Delay

LPC\_GPIO0->FIOSET = LPC\_GPIO0->FIOSET | 0x00000001; LPC\_GPIO0->FIOCLR = LPC\_GPIO0->FIOCLR | 0x00000002; LPC\_GPIO0->FIOSET = LPC\_GPIO0->FIOSET | 0x00010000;

LPC\_GPIO2->FIOCLR = LPC\_GPIO2->FIOCLR | 0x00000008; // Phase B is excited to rotate the motor for(delay=0; delay<80000; delay++); // Small time Delay

LPC\_GPIO0->FIOCLR = LPC\_GPIO0->FIOCLR | 0x00000001 ; LPC\_GPIO0->FIOSET = LPC\_GPIO0->FIOSET | 0x00000002 ; LPC\_GPIO0->FIOSET = LPC\_GPIO0->FIOSET | 0x00010000 ;

LPC\_GPIO2->FIOCLR = LPC\_GPIO2->FIOCLR | 0x00000008 ; // Phase C is excited to rotate the motor for(delay=0; delay<80000; delay++) ; // Small time Delay

}

}

###### Peripheral Interface:

No Switches are there to Stepper Motor is connected to P0.28, P0.29, P0.30 and P0.31 through LEDs D7 to D10.Connect the Stepper motor at connector X3.

###### Output:

You can see stepper motor moving in a particular direction and corresponding phase changes you can observe on LEDs D7 to D10.

###### Result:

Thus interfacing of stepper motor with LPC1768 is executed and verified successfully

###### Exercise:

1. Develop an embedded C code to move the stepper motor in different direction

# IOT MINI PROJECT

### PROJECT TITLE:-

PASSWORD DOOR LOCKING SYSTEM USING ARDUINO

### TEAM MEMBERS:-

* JANANII.P - 211521106065
* DHARANI SRI.T.G- 211521106036

**Ex. No. 9** PASSWORD DOOR LOCKING SYSTEM USING ARDUINO

**DATE:**

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
| **1** | **1(i)ABSTRACT** |  |
| **1(ii) INTRODUCTION** |  |
| **1(iii) OBJECTIVE** |  |
| **2** | **2(i) EXISITING SYSTEM** |  |
| **2(ii) PROPOSED SYSTEM** |  |
| **3** | **3(i)BLOCK DIAGRAM** |  |
| **3(ii) CIRCUIT DIAGRAM** |  |
| **3(iii) WORKING MODEL** |  |
| **4** | **4(i) CODE** |  |
| **4(ii) OUTPUT** |  |
| **5** | **5(i)ADVANTAGES** |  |
| **5(ii) APPLICATIONS** |  |
| **6** | **6(i)FUTURE ENHANCEMENT OR**  **SCOPE** |  |
| **6(ii) CONCULUSION** |  |
| **7** | **7(i)REFERRENCES** |  |

# PASSWORD DOOR LOCKING SYSTEM

## ABSTRACT:-

This project presents the development of a secure door locking system based on Arduino microcontroller technology. The system integrates a keypad for password entry, a servo motor for physical door locking/unlocking, and an Arduino board for control and processing. The primary objective is to design a cost-effective, reliable, and user-friendly alternative to traditional key-based locks. The system allows users to set and change passwords easily, providing flexibility and security. Additionally, the system incorporates features such as password encryption and anti-tampering mechanisms to enhance security. The implementation of the system demonstrates its effectiveness in providing secure access control to various applications, including residential and commercial environments. Overall, this project showcases the feasibility and practicality of utilizing Arduino-based solutions for modern security challenges.

**INTRODUCTION:-**

Arduino, an open-source electronics platform, facilitates the creation of interactive electronic objects with its easy-to-use hardware and software. Utilizing a microcontroller and various input/output pins, Arduino enables the development of innovative projects, including the implementation of security systems like the Password Door Lock System. One of the key advantages of using Arduino for security system development is its scalability. Whether you're designing a simple door lock for a residential property or a complex access control system for a commercial facility, Arduino provides a scalable platform that can accommodate varying levels of complexity. This scalability allows developers to start small and expand their systems as needed, making Arduino an ideal choice for projects with evolving requirements.

**OBJECTIVE:-**

The objective of this project is to design and implement a secure and user-friendly password door locking system using Arduino, a keypad, and a servo motor, with the goal of providing reliable access control while integrating innovative features such as speech-to-text functionality for enhanced usability

### EXISTING SYSTEM:

The traditional lock and key systems have been widely used for securing doors. However, they lack flexibility and convenience. Modern security systems incorporate electronic components for enhanced security and user experience. The Password Door Lock System replaces traditional keys with a password-based authentication mechanism for door access.

**1.Key-Based Locks:** Key-based locks are the most common traditional locking mechanism, where physical keys are used to unlock doors. The key fits into the lock's cylinder, and turning it aligns the internal components to release the lock mechanism, allowing the door to be opened.

**2.Combination Locks:** Combination locks use a predefined numeric or alphanumeric code to unlock doors. Users input the correct combination using a dial or keypad, and if the code matches the stored combination, the lock disengages, allowing access. Combination locks are often used in safes and padlocks for added security.

### PROPOSED SYSTEM:-

The proposed Password Door Lock System utilizes Arduino Uno R3 as the core microcontroller for managing the authentication process and controlling the locking mechanism. In addition to the Arduino Uno board, the system incorporates the following components:

1. Keypad: An input device for entering the password.

2. Servo Motor: Controls the door lock mechanism.

3. LCD Display: Provides feedback and prompts during the authentication process.

4. Connecting Wires: Facilitate electrical connections between components.

5. Power Supply: Provides the necessary electrical power to the system.

6. 4x4 Keypad**:-** The keypad provides a user-friendly interface for entering passwords and interacting with the system

### How the System Works :-

* Upon receiving input from the keypad, the Arduino validates the entered password against predefined values stored in its memory.
* If the entered password matches the predefined value, the Arduino sends a signal to the servo motor to rotate and unlock the door.
* After a specified duration, the servo motor returns to its original position, locking the door once again.
* If the entered password is incorrect, the Arduino denies access and prompts the user to retry or seek assistance. The proposed smart door locking system offers a secure, user-friendly, and customizable solution for residential security, leveraging Arduino and keypad technology to enhance access control and protect homes against unauthorized entry.

## BLOCK DIAGRAM:-

+ +

| Matrix Keypad |

+ +

| v

+ +

| Arduino |

| Microcontroller |

+ +

| v

+ +

| Servo Motor |

+ +

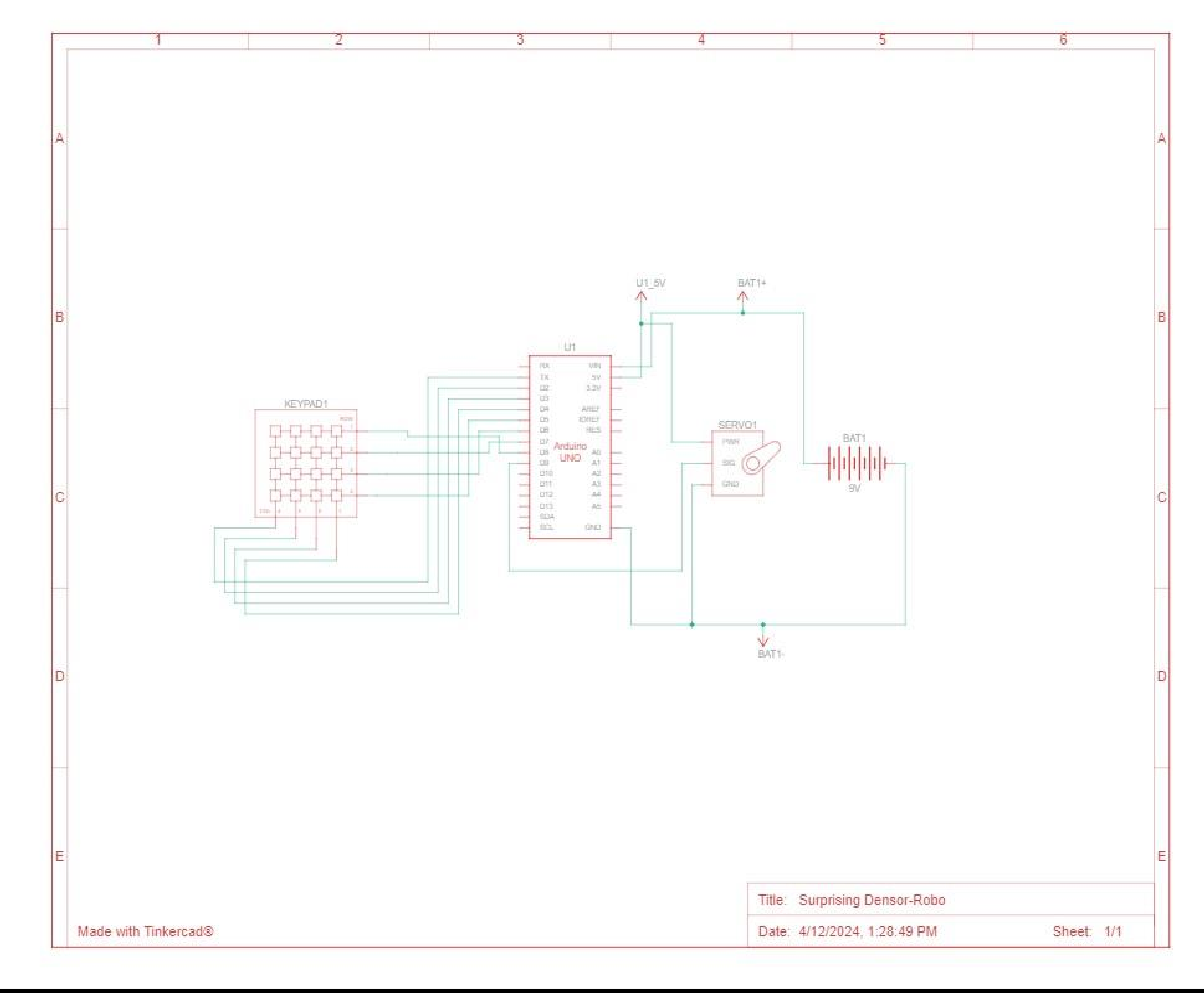
| v

+ +

| 9V Battery |

+ +

**CIRCUIT DIAGRAM:-**



**WORKING MODEL:-**

#### Initialization:

**1.**

* + The system starts with the Arduino and servo motor powered up from the 9V battery.
  + The Arduino initializes the servo motor and waits for user input.

#### User Input:

When a user approaches the door, they interact with the matrix keypad to input the predefined password.

#### Password Verification:

* + The Arduino receives the keypad input and compares it with the stored password.
  + If the entered password matches the stored password, the Arduino proceeds to unlock the door; otherwise, it denies access.

#### Feedback and Status Indication:

* + The system provides feedback to the user indicating whether access is granted or denied. This feedback can be visual (LED indicator) or audible (buzzer).
  + Additionally, the system may include status indicators to show the current state of the lock (locked or unlocked).

#### Power Management:

The system optimizes power usage to conserve battery life, implementing sleep modes or power-saving strategies when the system is idle.

#### Security Considerations:

* + The project implements security best practices such as using strong, complex passwords, avoiding hard-coding passwords in the code, and securing the physical components from tampering.

#### Testing and Calibration:

* + Before deployment, the system undergoes rigorous testing to ensure reliable operation and accuracy in locking/unlocking the door.
  + Calibration of the servo motor may be required to fine-tune its movement for smooth locking and unlocking actions.

#### Feedback and Status Indication:

* + The system provides feedback to the user indicating whether access is granted or denied. This feedback can be visual (LED indicator) or audible (buzzer).
  + Additionally, the system may include status indicators to show the current state of the lock (locked or unlocked).

#### Power Management:

The system optimizes power usage to conserve battery life, implementing sleep modes or power-saving strategies when the system is idle.

#### Security Considerations:

* + The project implements security best practices such as using strong, complex passwords, avoiding hard-coding passwords in the code, and securing the physical components from tampering.

#### Testing and Calibration:

* + Before deployment, the system undergoes rigorous testing to ensure reliable operation and accuracy in locking/unlocking the door.
  + Calibration of the servo motor may be required to fine-tune its movement for smooth locking and unlocking actions.

### CODE:-

### #include <Servo.h>

### #include <Keypad.h>

### Servo ServoMotor;

### char\* password = "953"; // change the password here, pick any 3 numbers

### int position = 0;

### const byte ROWS = 4;

### const byte COLS = 4;

### char keys[ROWS][COLS] = {

### {'1','2','3','A'},

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

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

### {'\*','0','#','D'}

### };

### byte rowPins[ROWS] = { 8, 7, 6, 9 };

### byte colPins[COLS] = { 5, 4, 3, 2 };

### Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

### int RedpinLock = 12;

### int GreenpinUnlock = 13;

### void setup()

### {

### ServoMotor.attach(11);

### LockedPosition(true);

### }

### void loop()

### {

### char key = keypad.getKey();

### if (key == '\*' || key == '#')

### {

### position = 0;

### LockedPosition(true);

### }

### if (key == password[position])

### {

### position ++;

### }

### if (position == 3)

### {

### LockedPosition(false);

### }

### delay(100);

### }

### void LockedPosition(int locked)

### {

### if (locked)

### {

### digitalWrite(RedpinLock, HIGH);

### digitalWrite(GreenpinUnlock, LOW);

### ServoMotor.write(11);

### }

### else

### {

### digitalWrite(RedpinLock, LOW);

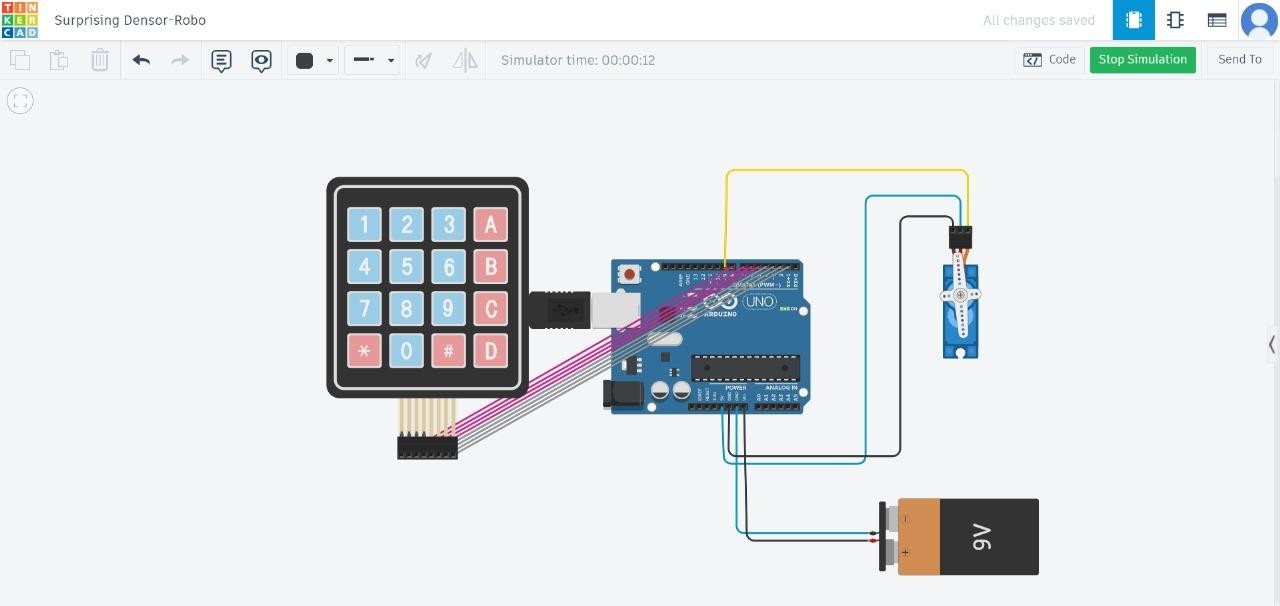
### digitalWrite(GreenpinUnlock, HIGH);

### ServoMotor.write(90);

### }

### }

**OUTPUT:**



**Advantages:**

1. Enhanced Security: The password-based authentication mechanism provides a higher level of security compared to traditional lock and key systems.

2. User Convenience: Users can easily change or update the password as needed without the need for physical key replacements.

3. Flexibility: The system can be customized to support multiple users and access levels, offering greater flexibility in managing door access.

4. Real-time Monitoring: Integration with additional components such as WiFi module enables real-time monitoring and remote access to the door lock system.

### FUTURE ENHANCEMENT OR SCOPE:-

1. **Biometric Integration**: Incorporating biometric authentication methods such as fingerprint or facial recognition can enhance security and user convenience.
2. **Mobile App Integration**: Develop a mobile application that allows users to control the door lock remotely, receive notifications of access attempts, and manage access permissions.
3. **Cloud Connectivity**: Implement cloud connectivity to store access logs, receive software updates, and enable remote management and monitoring of the locking system.
4. **Multi-factor Authentication (MFA):** Add an extra layer of security by integrating MFA methods such as combining a password with a one-time code sent to the user's mobile device.
5. **Voice Recognition:** Enhance usability by integrating voice recognition technology, allowing users to unlock the door using voice commands after successful authentication.
6. **Integration with Home Automation Systems**: Make the door locking system part of a larger home automation ecosystem, enabling seamless integration with other smart devices and automation routines.
7. **Enhanced User Interface**: Design a more intuitive and interactive user interface for the keypad or mobile app, with features like visual feedback, customizable settings, and user-friendly instructions.
8. **Advanced Security Features**: Explore advanced security features such as tamper detection, intrusion alerts, and automatic lockdown in case of suspicious activity or multiple failed access attempts.
9. **Energy Efficiency:** Implement power-saving features such as automatic locking after a period of inactivity, low-power modes, and energy-efficient components to prolong battery life or reduce electricity consumption.
10. **Scalability:** Design the system to be easily scalable, allowing for the addition of more doors, users, and access control features as needed without significant redesign or reconfiguration.

**Application:**

1. Residential Security: The Password Door Lock System can be installed in homes, apartments, and condominiums to enhance residential security and access control.

2. Commercial Buildings: Offices, shops, and other commercial establishments can deploy the system to secure entry points and restrict unauthorized access.

3. Educational Institutions: Schools, colleges, and universities can implement the system to control access to classrooms, laboratories, and administrative offices.

4. Industrial Facilities: Factories, warehouses, and manufacturing plants can utilize the system to secure restricted areas and protect valuable assets.

### CONCLUSION:-

The "Door Locking System Using Password and Arduino" project presents a secure and user-friendly solution for access control, integrating Arduino technology with a matrix keypad, servo motor, and 9V battery. This system enhances security, offers customizable access, and serves as an educational platform for Arduino programming and security principles.

**References:**

1. K. H. Park, J. W. Yoo, and Y. S. Yoon, "Password-based smart door lock system using Arduino UNO," 2016 International Conference on Information and Communication Technology Convergence (ICTC), 2016, pp. 506–508.

2. S. M. K. A. Rezvi and S. S. Kabir, "Design and implementation of a smart door lock system using password and fingerprint," 2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2), 2018, pp. 1–5.

3. H. Zhang, J. Ye, X. Liu, and X. Liu, "Design and implementation of a password-based smart door lock system," 2019 IEEE 2nd International Conference on Electronic Information and Communication Technology (ICEICT), 2019, pp. 144–147.

4. Y. H. Kim, H. K. Moon, and J. M. Jang, "A design of smart door lock system based on password and IoT environment," 2019 2nd International Conference on Electronics and Electrical Engineering Technology (EEET), 2019, pp. 154–157.

5. Y. Pan, S. Wei, Y. Zou, and Z. Huang, "A password-based smart door lock system design and implementation," 2020 International Conference on Smart Internet of Things (SmartIoT), 2020, pp. 73–76.

This comprehensive overview outlines the architecture, components, advantages, and applications of the Password Door Lock System, demonstrating its effectiveness in enhancing security and access control in various settings.