ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

NH-46, Chennai-Bengaluru National Highways, Arappakkam,
Ranipet-632517, Tamil Nadu, India
Telephone: 04172-292925 Fax: 04172-292926
Email:amcet.rtet@gmail.com/info@amcet.in Web: www.amcet.in

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

CS3691 – EMBEDDED SYSTEMS AND IOT LABORATORY

Name : ……………………………………..

Register Number : ……………………………………..

Year & Branch : ……………………………………..

Semester : ……………………………………

Academic Year : …………………………..................

ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

NH-46, Chennai-Bengaluru National Highways, Arappakkam,

Ranipet-632517, TamilNadu, India

Telephone: 04172-292925 Fax: 04172-292926

CERTIFICATE

This is to Certify that the Bonafide record of the practical work done by………………………………… Register Number..of III year B.Tech (Artificial Intelligence and Data Science Engineering) submitted for the B.Tech -Degree practical examination (VI Semester) in **CS3691 EMBEDDED SYSTEMS AND IOT LABORATORY** during the academic year **20 – 20**

Staff in –Charge **Head of the Department**

Submitted for the practical examination held on ------------------

Internal Examiner **External Examiner**

TABLE OF CONTENT

S.NO	DATE	NAME OF THE EXPERIMENTS	PAGE NO	SIGN

S.NO	DATE	NAME OF THE EXPERIMENTS	PAGE NO	SIGN

EXPT NO : 1 DATE :	8051 ASSEMBLY LANGUAGE EXPERIMENTS USING SIMULATOR

AIM:

To write 8051 Assembly Language experiments using Simulator.

PROCEDURE:

STEP 1 : Initialize Port 1 as an output port.
STEP 2 : Enter an infinite loop labeled as "LOOP."
STEP 3 : The delay subroutine is called "DELAY."

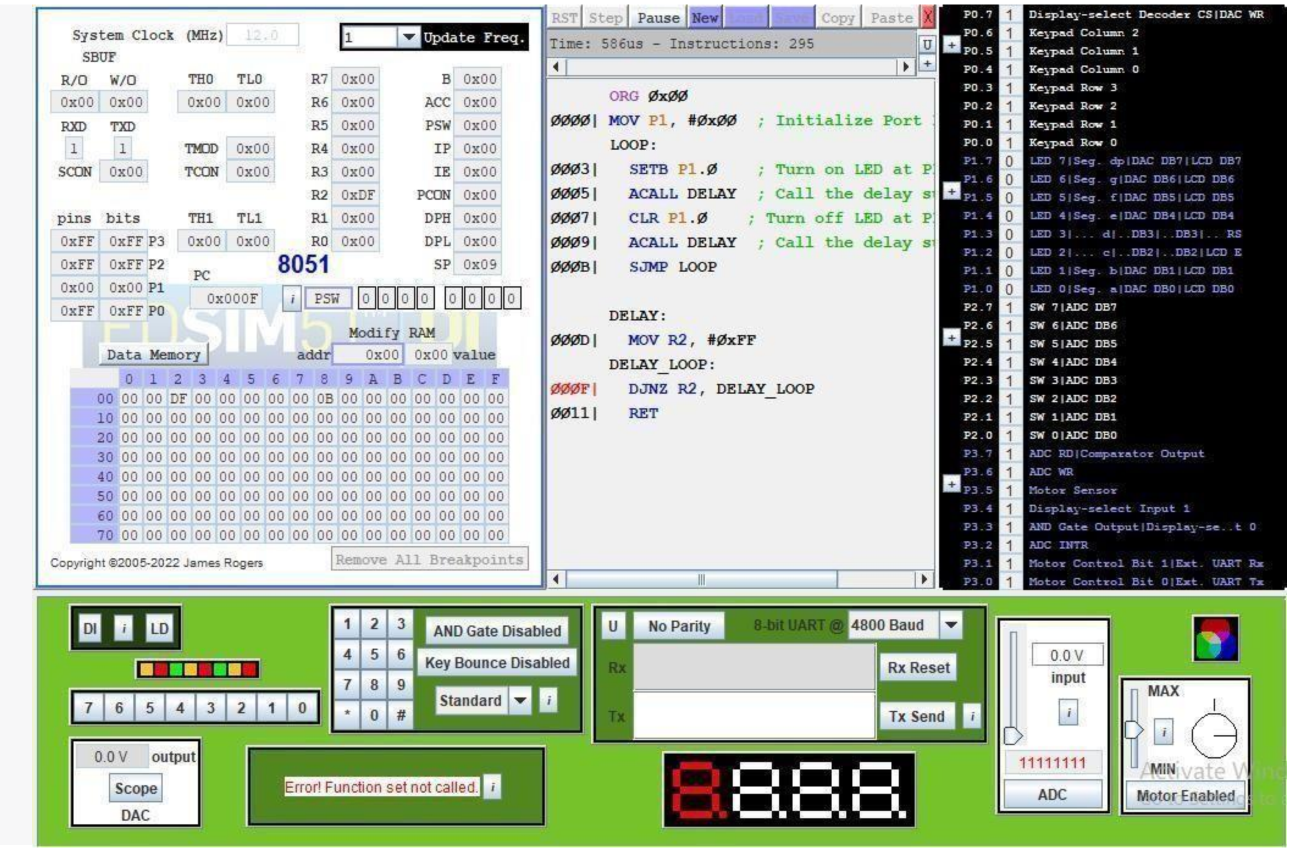
- Initialize register R2 with the value 0xFF (255 in decimal).
- Enter a loop labeled as "DELAY_LOOP."

STEP 4 : The program continues to loop indefinitely, creating a blinking LED effect on P1.0.

PROGRAM 1 :

```
ORG 0x00
MOV P1, #0x00 ; Initialize Port 1 as output LOOP:
  SETB P1.0 ; Turn on LED at P1.0

ACALL DELAY ; Call the delay subroutine
  CLR P1.0  ; Turn off LED at P1.0
  ACALL DELAY ; Call the delay subroutine
  SJMP LOOP

DELAY:
  MOV R2, #0xFF DELAY_LOOP:
  DJNZ R2, DELAY_LOOP
  RET
```

OUTPUT:

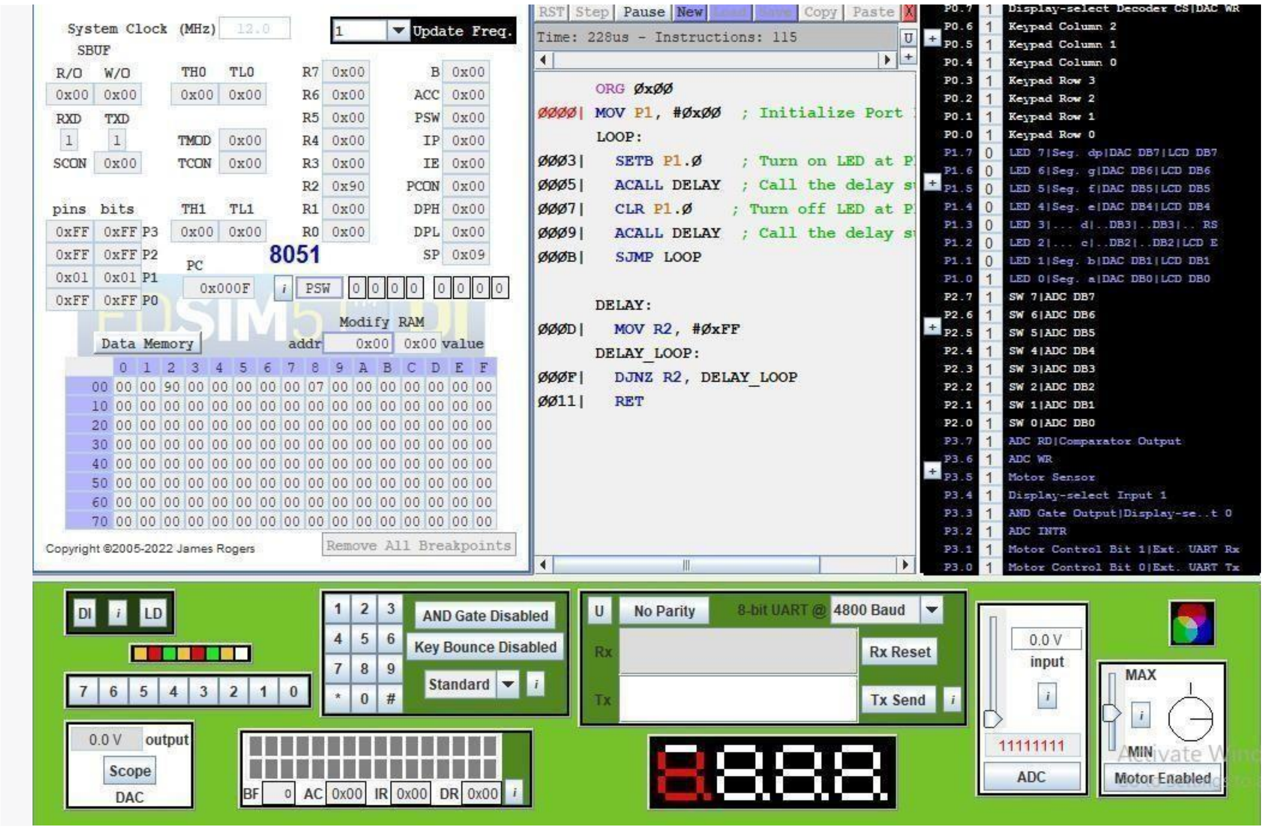


ORG Ø×ØØ
ØØØØ| MOV P1, #ØxØØ ; Initialize Port
LOOP:
ØØØ3| SETB P1.Ø ; Turn on LED at P
ØØØ5| ACALL DELAY ; Call the delay s
ØØØ7| CLR P1.Ø ; Turn off LED at P
ØØØ9| ACALL DELAY ; Call the delay s
ØØØB| SJMP LOOP
DELAY:
ØØØD| MOV R2, #ØxFF
DELAY_LOOP:
ØØØF| DJNZ R2, DELAY_LOOP
Ø011| RET
Time: 586us - Instructions: 295
8051
Error! Function set not called.
Copyright ©2005-2022 James Rogers

PROGRAM :

```
ORG 0x00
MOV P1, #0x00 ; Initialize Port 1 as output LOOP:
  SETB P1.0  ; Turn on LED at P1.0
  ACALL DELAY ; Call the delay subroutine
  CLR P1.0     ; Turn off LED at P1.0
  ACALL DELAY ; Call the delay subroutine
  SJMP LOOP
DELAY;


 MOV R2, #0xFF DELAY_LOOP:
 DJNZ R2, DELAY_LOOP
 RET
```

OUTPUT:



ORG 0x00
0000| MOV P1, #0x00 ; Initialize Port
LOOP:
0003| SETB P1.0 ; Turn on LED at P
0005| ACALL DELAY ; Call the delay s
0007| CLR P1.0 ; Turn off LED at P
0009| ACALL DELAY ; Call the delay s
000B| SJMP LOOP
DELAY:
000D| MOV R2, #0xFF
DELAY_LOOP:
000F| DJNZ R2, DELAY_LOOP
0011| RET
Time: 228us - Instructions: 115
8051
Copyright ©2005-2022 James Rogers

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 2 DATE :	DATA TRANSFER BETWEEN REGISTER AND MEMORY

AIM:

To Write a Data Transfer Between Register and Memory

PROCEDURE:

Transfer a value from a register to memory and then from memory back to another register. We'll use the '**MOV**' (move) instruction for this purpose.

STEP 1: Text Section:

We declare the program's entry point using the global main directive.

STEP 2: Main Function:

The main function begins.

STEP 3: Load a Value into a Register:

STEP 4: Store the Value in Memory

STEP 5: Load the Value from Memory

STEP 6: Program Exit

PROGRAM:

8051 Data Transfer Between Register and Memory Example

ORG 0x0000 ; Start address

MAIN:

; Move a value (e.g., 0xAA) from register R0 to a memory location (e.g., 0x30)

MOV A, #0xAA ; Load the accumulator with the value 0xAA

MOV R0, A ; Move the accumulator content to Register R0

MOV 0x30, R0 ; Move the content of Register R0 to memory location 0x30

Move a value from a memory location (e.g., 0x30) to register R1 MOV A, 0x30 ; Load the accumulator with the content of memory

location 0x30

MOV R1, A ; Move the accumulator content to Register R1

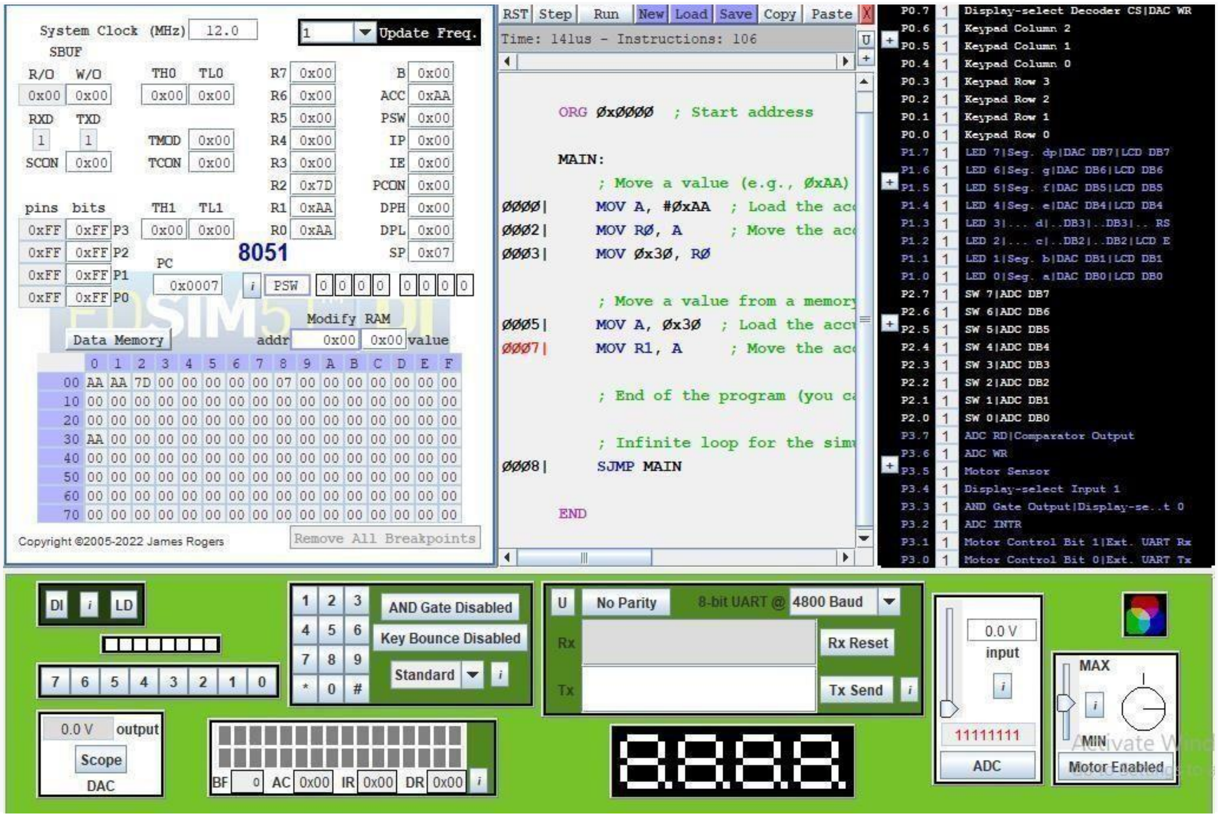
; End of the program (you can add more instructions as needed)

; Infinite loop for the simulator SJMP MAIN

END

OUTPUT:

The output screen of the above program



RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 3 DATE :	PERFORM ALU OPERARTIONS

AIM:

To Write a Perform ALU Operations

PROCEDURE:

Practical Exercise: ALU Operations using EdSim51

STEP 1: Download and install EdSim51 from the official website (http://www.edsim51.com/).

STEP 2: Open EdSim51 and create a new file.

STEP 3: Initialization:

- Load the value 10 into R0 (First operand).
- Load the value 5 into R1 (Second operand). **STEP4:**

Addition,Subtraction,Multiplication,Division**STEP5:** Infinite Loop:

- Enter an infinite loop using the HERE label and the SJMP HERE instruction. This loop keeps the program running indefinitely.

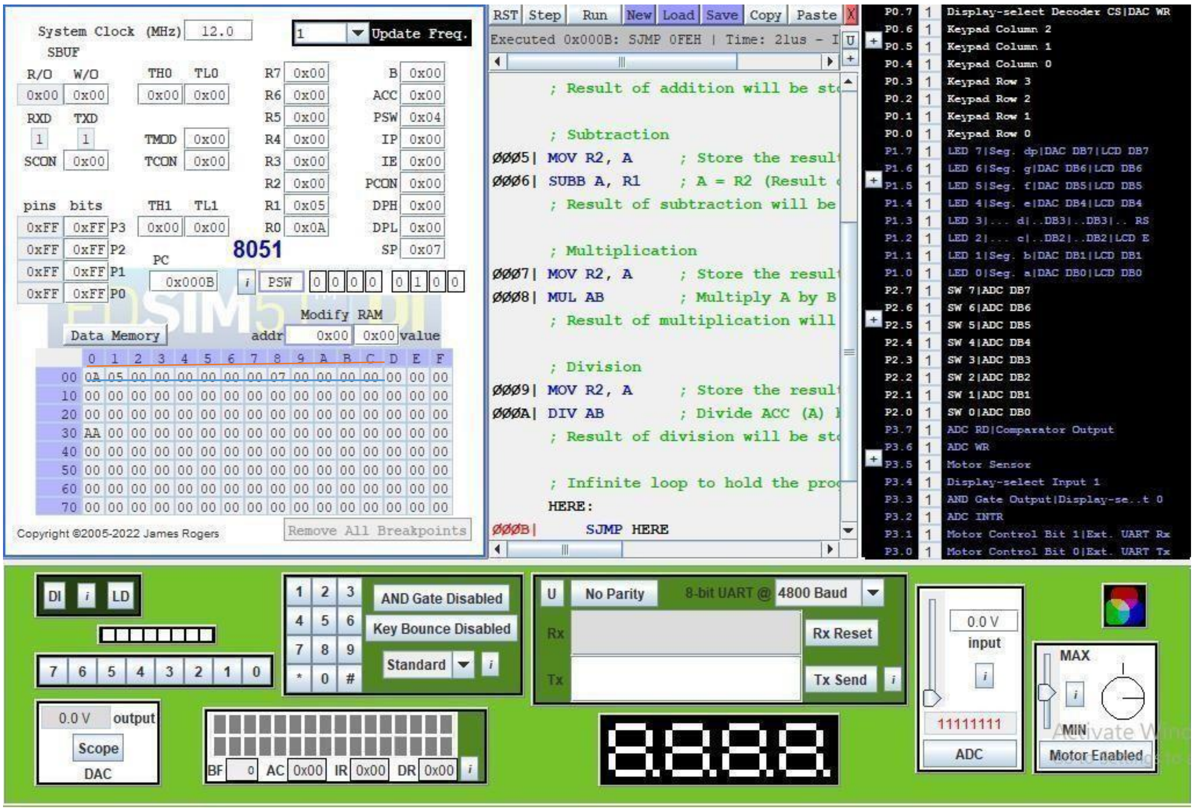
PROGRAM:

```
; ALU Operations in 8051 Assembly for EdSim51ORG

0x0000

; Initialize data in RAM
MOV R0, #10       ; First operand (e.g., 10)
MOV R1, #5        ; Second operand (e.g., 5)

; Addition
ADD A, R0         ; A = A + R0
; Result of addition will be stored in the Accumulator (A)

; Subtraction
MOV R2, A         ; Store the result of addition in R2
SUBB A, R1        ; A = R2 (Result of addition) - R1
; Result of subtraction will be stored in the Accumulator (A)

; Multiplication
MOV R2, A         ; Store the result of subtraction in R2
MUL AB            ; Multiply A by B (R2), Result will be in ACC (A,lower
byte) and B (higher byte)
; Result of multiplication will be stored in the Accumulator (A)

; Division
MOV R2, A         ; Store the result of multiplication in R2
DIV AB            ; Divide ACC (A) by B (R2), Quotient will be in ACC(A),
Remainder in B
; Result of division will be stored in the Accumulator (A) and B
(Remainder)

; Infinite loop to hold the program
HERE:
    SJMP HERE
```

END

OUTPUT:

The output screen of the above program



RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 4 DATE :	WRITE BASIC AND ARITHMETIC PROGRAM USING EMBEDDED C.

AIM:

To write basic and arithmetic Program using embedded c.

BASIC PROGRAM:

Blinking LED

OBJECTIVE:

To blink an LED connected to a micro controller pin.

ALGORITHM:

STEP 1 : Include necessary header files

STEP 2 : Define the LED pin

STEP 3 : Start the main function

STEP 4 : Configure the LED pin as an output **STEP 5 :** Enter an infinite loop with while(1) **STEP 6 :** Create a delay

STEP 7 : Repeat the LED toggle and delay

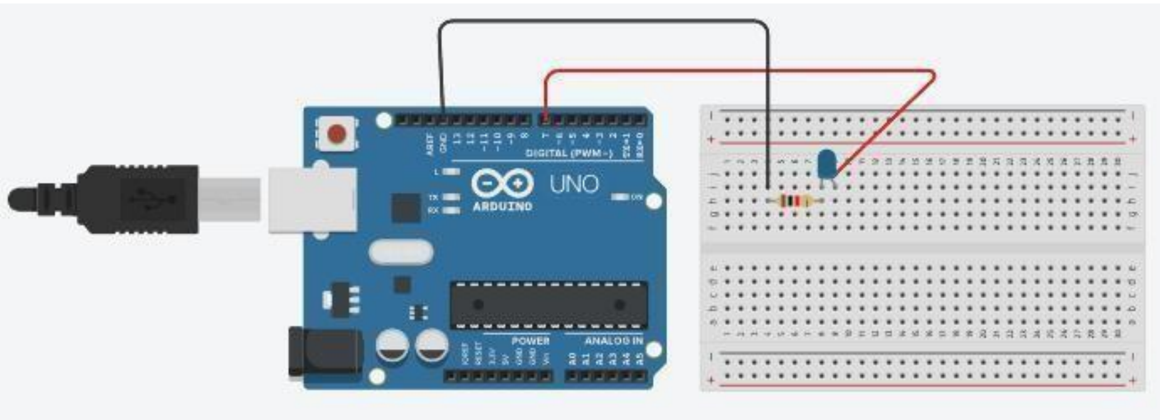
STEP 8 : Exit the main function

PROGRAM IN EMBEDDED C:

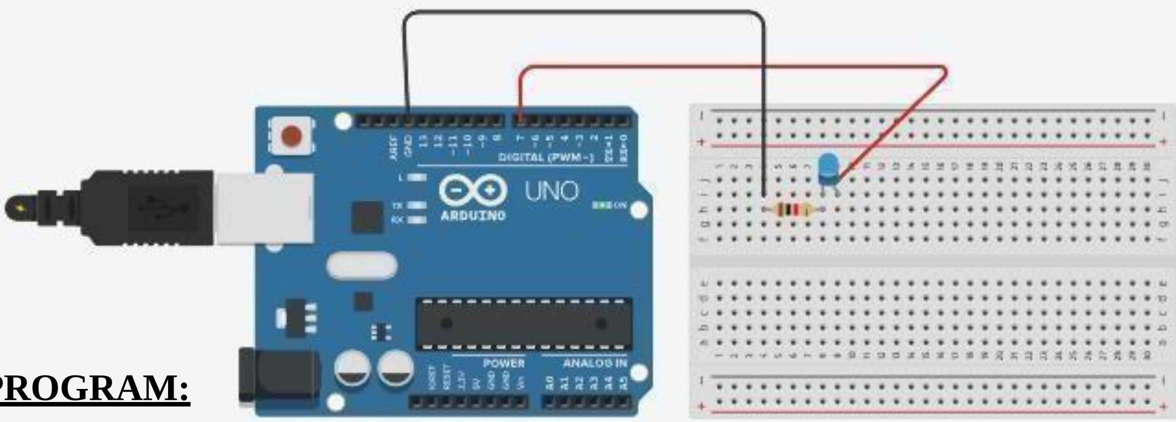
```
#include <avr/io.h>
#include <util/delay.h>

#define LED_PIN PB0int

main(void) {
    // Set the LED pin as outputDDRB |=
    (1 << LED_PIN);

    while (1) {
        // Toggle the LED pin PORTB
        ^= (1 << LED_PIN);

        // Delay for a period of time
        _delay_ms(500);
    }

    return 0;
}
```

OUTPUT:

INITIALLY LIGHT IS OFF..



LIGHT IS GLOWING AFTER THE EXECUTING OF THE PROGRAM.



PROGRAM:

ARITHMETIC PROGRAM:

Addition of Two Numbers

Objective: To add two numbers and display the result.

ALGORITHM :

STEP 1: Include the necessary header file

STEP 2: Start the main function

STEP 3: Declare variables

STEP 4: Read the first number from the user **STEP**

5: Read the second number from the user

STEP 6: Perform the addition

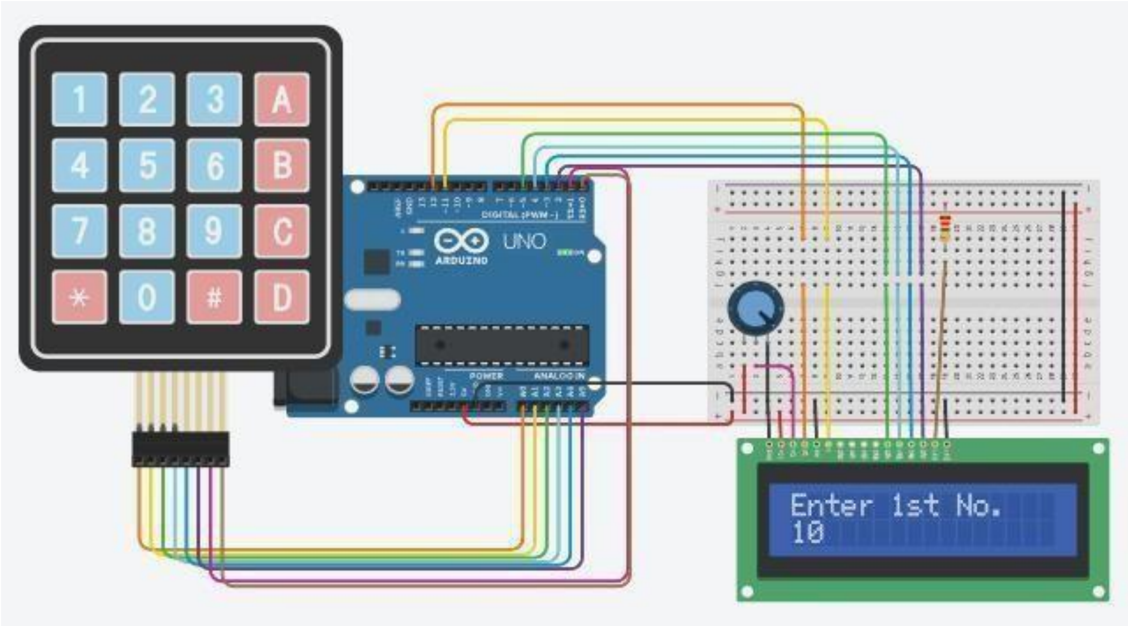
STEP 7: Display the result

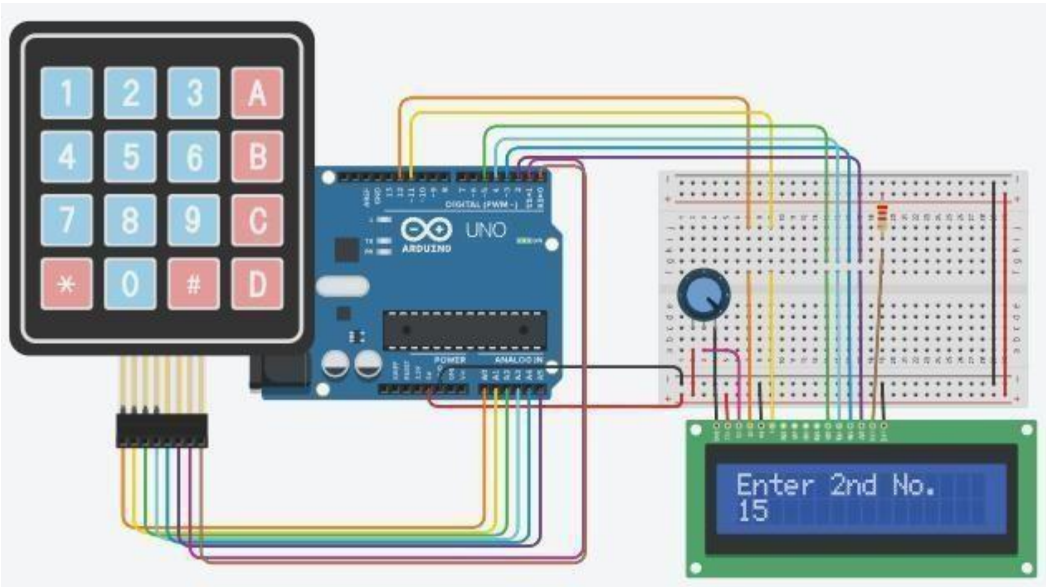
STEP 8: Exit the main function

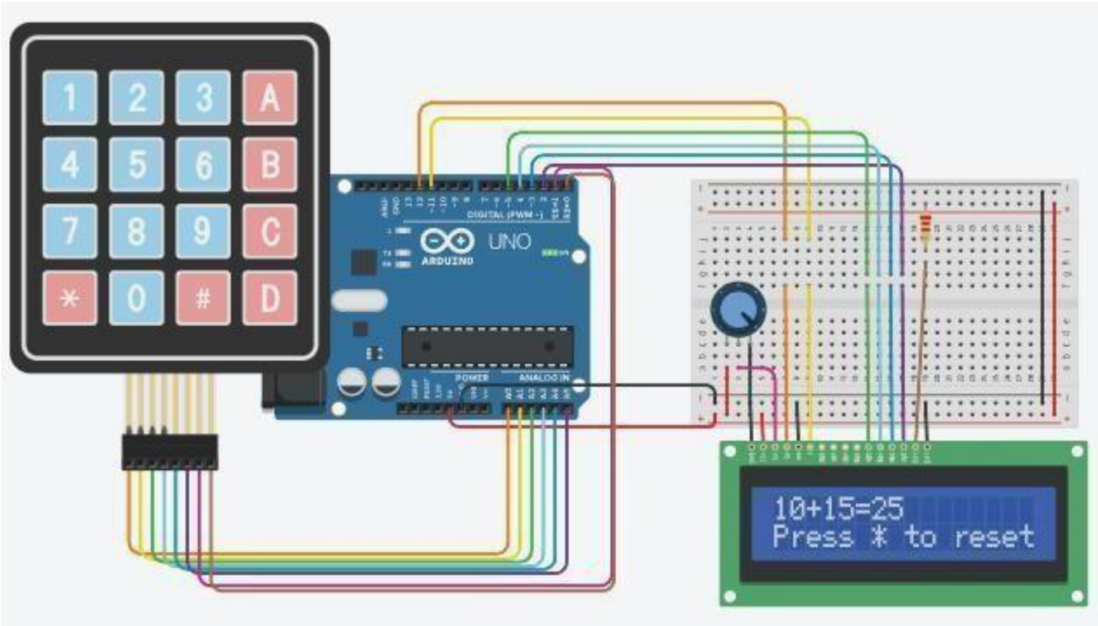
PROGRAM:

```
#include <stdio.h>

int main() {
    int num1, num2, sum;

    // Read two numbers from the user
    printf("Enter the 1st no: ");
    scanf("%d", &num1);

    printf("Enter the 2nd no: ");
    scanf("%d", &num2);

    // Perform the addition
    sum = num1 + num2;

    // Display the result
     printf(num1+"+"+num2+"=")
     printf(“press * to reset”)
);

    return 0;
}
```

OUTPUT:







RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 5 DATE :	INTRODUCTION TO ARDUINO PLATFORM AND PROGRAMMING

AIM:

A study of introduction to Arduino Platform and programming

INTRODUCTION:

Arduino is an open-source electronics platform that has gained immense popularity among beginners, students, hobbyists, and professionals for its simplicity and versatility. This introduction provides anoverview of the Arduino platform and its programming aspects.

What is Arduino?

Arduino is a versatile microcontroller-based hardware and softwareplatform that enables users to create interactive and programmable electronic projects. It consists of two main components:

Hardware: Arduino boards are the physical computing devices at the coreof the platform. They come in various shapes and sizes but share common elements, including a microcontroller, digital and analog input/output pins,power supply, and communication interfaces.

Software: The Arduino Integrated Development Environment (IDE) is a user-friendly programming environment for writing, compiling, and uploading code to Arduino boards. It uses a simplified version of the C andC++ programming languages.

Why Use Arduino?

Here are some key reasons why Arduino has become so popular:

Accessibility: Arduino is designed to be easy for beginners to start working with electronics and programming. It lowers the entry barrier for those newto the field.

Open Source: Arduino's hardware and software are open source, meaningthe designs and source code are freely available. This encourages a vibrantcommunity of users and developers who share their knowledge and contribute to its growth.

Versatility: Arduino is not limited to any specific application. It can be used for a wide range of projects, including robotics, home automation, artinstallations, and scientific experiments.

Abundance of Resources: There is a wealth of online resources, tutorials,and libraries available to help users get started and solve problems they encounter.

Expansion Capability: Arduino can be easily extended by adding "shields" – additional boards that provide extra functionality. These shieldscan be stacked on top of the Arduino board.

The Arduino Programming Environment

Arduino programming is done in the Arduino IDE, which provides asimple and straightforward way to write code for your projects. Here are some key aspects of the Arduino programming environment:

Sketch: In Arduino, a program is called a "sketch." A sketch typically consists of two essential functions: setup() (for initialization) and loop()(for continuous execution).

Libraries: Arduino libraries are pre-written code packages that simplify working with external components like sensors and displays. Many librariesare available for various purposes.

Upload: Once you've written your code, you can upload it to the Arduino board via a USB connection. The Arduino IDE handles the compilation anduploading process for you.

ARDUINO PROGRAM:

```
const int LED=2;
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Conclusion

Arduino is a powerful and accessible platform for learning about electronics and programming. It allows you to turn your creative ideas intotangible projects, whether you're a student learning the basics or an experienced engineer building advanced systems. This introduction sets thestage for exploring Arduino further and getting hands-on experience with this remarkable platform.

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 6 DATE :	EXPLORE DIFFERENT COMMUNICATION METHODS WITH IOT DEVICES (ZIGBEE, GSM, BLUETOOTH)

AIM:

To explore different communication methods with iot devices (zigbee, GSM, bluetooth)

ALGORITHM:

ZIGBEE:

Home Automation and Control:

- Zigbee is an excellent choice for home automation projects where theaim is to connect and control smart devices within a home network. This includes smart lighting, thermostats, door locks, and more.
- Zigbee's mesh networking capability allows devices to form a self- organizing network, enhancing coverage and reliability. It's ideal forcreating a seamless and interconnected smart home ecosystem.
- Mesh networking also means that Zigbee devices can relay messages,making the network more robust.

PROGRAM: (Receiver code)

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>

LiquidCrystal_PCF8574 lcd(0x27);  // set the LCD address to 0x27 for a 16 chars and
2 line display

int show = -1;

// 2 custom characters

byte dotOff[] = { 0b00000, 0b01110, 0b10001, 0b10001,
                  0b10001, 0b01110, 0b00000, 0b00000 };
byte dotOn[] = { 0b00000, 0b01110, 0b11111, 0b11111,
```

```
                 0b11111, 0b01110, 0b00000, 0b00000 };

#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti WiFiMulti;

const char* ssid = "ESP8266-Access-Point";
const char* password = "123456789";

//Your IP address or domain name with URL path
const char* serverNameTemp = "http://192.168.4.1/temperature";
const char* serverNameHumi = "http://192.168.4.1/humidity";
const char* serverNamePres = "http://192.168.4.1/pressure";


String temperature;
String humidity;
String pressure;

unsigned long previousMillis = 0;
const long interval = 5000;

void setup() {
  Serial.begin(115200);
  Serial.println();

  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("Connected to WiFi");
   int error;
  Serial.println("LCD...");

  // wait on Serial to be available on Leonardo
  while (!Serial)
    ;

  Serial.println("Probing for PCF8574 on address 0x27...");

  // See http://playground.arduino.cc/Main/I2cScanner how to test for a I2C device.
  Wire.begin();
  Wire.beginTransmission(0x27);
  error = Wire.endTransmission();
  Serial.print("Error: ");
  Serial.print(error);

  if (error == 0) {
    Serial.println(": LCD found.");
```

```
    show = 0;
    lcd.begin(16, 2);  // initialize the lcd

    lcd.createChar(1, dotOff);
    lcd.createChar(2, dotOn);

  } else {
    Serial.println(": LCD not found.");
  }  // if
   lcd.setBacklight(255);
    lcd.home();
    lcd.clear();
    lcd.print("EXP NO 9");
    delay(1000);
}

void loop() {
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= interval) {
     // Check WiFi connection status
    if ((WiFiMulti.run() == WL_CONNECTED)) {
      temperature = httpGETRequest(serverNameTemp);
      humidity = httpGETRequest(serverNameHumi);
      pressure = httpGETRequest(serverNamePres);
      Serial.println("D1=" + temperature + "D2=" + humidity + "D3=" + pressure
+"..");


   lcd.clear();
    lcd.setCursor(0, 0);
   // lcd.print("");
   lcd.print(temperature);
//   lcd.print(" D2=");
//   lcd.print(humidity);
//    lcd.setCursor(0, 1);
//  lcd.print("D3=");
//   lcd.print(pressure);
    delay(1000);
      // save the last HTTP GET Request
      previousMillis = currentMillis;
    }
    else {
      Serial.println("WiFi Disconnected");
    }
  }
}

String httpGETRequest(const char* serverName) {
  WiFiClient client;
  HTTPClient http;

  // Your IP address with path or Domain name with URL path
```

```
  http.begin(client, serverName);

  // Send HTTP POST request
  int httpResponseCode = http.GET();

  String payload = "--";

  if (httpResponseCode>0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    payload = http.getString();
  }
  else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  // Free resources
  http.end();

  return payload;

}
```

PROGRAM (Transmitter code)

```
String input1;
 String input2;
 String input3;

// Import required libraries
#include <ESP8266WiFi.h>
#include "ESPAsyncWebServer.h"


// Set your access point network credentials
const char* ssid = "ESP8266-Access-Point";
const char* password = "123456789";
String inputString = "",data="";          // a String to hold incoming data
bool stringComplete = false;  // whether the string is complete
// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

String readTemp() {

//  if (Serial.available() > 0)
//  {
//    // read the incoming byte:
   input1= data;

    // say what you got:
   // Serial.print("I received: ");
    //Serial.println(input);
```

```
     return String(input1);

}

String readHumi() {
//   if (Serial.available() > 0)
//  {
//     // read the incoming byte:
     input2= data;
    // say what you got:
   // Serial.print("I received: ");
    //Serial.println(input);
     return String(input2);

}

String readPres() {
//  if (Serial.available() > 0)
//  {
//     // read the incoming byte:
     input3= data;

    // say what you got:
   // Serial.print("I received: ");
    //Serial.println(input);
     return String(input3);

}
void setup(){
  // Serial port for debugging purposes
  Serial.begin(9600);
  Serial.println();
  inputString.reserve(200);
  // Setting the ESP as an access point
  Serial.print("Setting AP (Access Point)…");
  // Remove the password parameter, if you want the AP (Access Point) to be open
  WiFi.softAP(ssid, password);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
 server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readTemp().c_str());
  });
  server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readHumi().c_str());
  });
  server.on("/pressure", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readPres().c_str());
  });

  bool status;
```

```
  // default settings
  // (you can also pass in a Wire library object like &Wire2)
 // status = bme.begin(0x76);
//  if (!status) {
//    Serial.println("Could not find a valid BME280 sensor, check wiring!");
//    while (1);
//  }

  // Start server
  server.begin();
}

void loop(){

  if (stringComplete) {
    Serial.println(inputString);
    // clear the string:
    data=inputString;
    inputString = "";
    stringComplete = false;
  }
}



void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag so the main loop can
    // do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```

In environmental monitoring projects, Zigbee sensor nodes can be strategically deployed throughout an area. These nodes can continuously collect data and irelessly transmit it to a central hub forreal-time analysis and control.

- The low-power characteristics of Zigbee devices are advantageous for applications that require extended battery life.

Industrial Automation:

- Zigbee is used in industrial IoT (IIoT) projects where the goal is tomonitor and control machinery and equipment.
- In industrial settings, Zigbee provides reliable and low-power communicatio.It's suitable for applications that demand long-term,energy-efficient operation.
- The ability to create a mesh network is valuable in large industrialfacilities where coverage and redundancy are important.
-

GSM (GLOBAL SYSTEM FOR MOBILE COMMUNICATIONS):

Asset Tracking and Monitoring:

- GSM is an excellent choice for asset tracking and monitoring projects, particularly when wide-area coverage and real-time location data are crucial.

- It allows you to track the location of vehicles, shipping containers, or equipment using GPS and GSM connectivity. This is invaluable in logistics, transportation, and fleet management.
- The ability to transmit location data over wide areas ensures assetscan be tracked even in remote or mobile scenarios.

PROGRAM:

```
void setup() {

  Serial.begin(9600);     // Setting the baud rate of Serial Monitor (Arduino)
}

void loop() {

  Serial.println("AT+CMGF=1");     //Sets the GSM Module in Text Mode
  delay(1000);  // Delay of 1000 milli seconds or 1 second
  Serial.println("AT+CMGS=\"9944740541\"\r"); //Replace x with mobile number
  delay(1000);
  Serial.println("THIS GOKUL FROM CALIBER EMBEDDED");// The SMS text you want to
send
  delay(1000);
  Serial.println((char)26);// ASCII code of CTRL+Z
  delay(8000);
}
```

Remote Monitoring and Control:

- GSM is highly effective in projects requiring remote monitoring andcontrol of devices and systems.
- In scenarios where devices are distributed over a wide geographicarea, such as agricultural applications or remote infrastructure monitoring, GSM enables real-time data collection and remote control.
- GSM can also be utilized in scenarios where local communicationnetworks may not be available.

Security Systems:

- GSM can be integrated into security systems, enhancing their reliability and reach.
- In security projects, GSM can be used to send alerts and notifications in case of intrusions or other security breaches. Thisensures that data is transmitted even when local communication networks fail.
- The ubiquity of GSM networks makes it a dependable choice forcritical security applications.

BLUETOOTH:

Wearable and Health Devices:

- Bluetooth, particularly Bluetooth Low Energy (BLE), is extensivelyused in wearable health devices and fitness trackers.
- These devices collect health-related data, such as heart rate, stepstaken, and sleep patterns, and transmit this information to smartphones or cloud platforms for analysis and tracking.
- The low energy consumption of BLE ensures that wearables canoperate for extended periods without frequent recharging.

PROGRAM (Receiver Code)

```
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>

LiquidCrystal_PCF8574 lcd(0x27);  // set the LCD address to 0x27 for a 16 chars and
2 line display

int show = -1;

// 2 custom characters

byte dotOff[] = { 0b00000, 0b01110, 0b10001, 0b10001,
                  0b10001, 0b01110, 0b00000, 0b00000 };
byte dotOn[] = { 0b00000, 0b01110, 0b11111, 0b11111,
                 0b11111, 0b01110, 0b00000, 0b00000 };
String inputString = "";       // a String to hold incoming data
bool stringComplete = false;  // whether the string is complete

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // reserve 200 bytes for the inputString:
  inputString.reserve(200);
   int error;


  Serial.println("LCD...");

  // wait on Serial to be available on Leonardo
  while (!Serial)
    ;

  Serial.println("Probing for PCF8574 on address 0x27...");

  // See http://playground.arduino.cc/Main/I2cScanner how to test for a I2C device.
  Wire.begin();
  Wire.beginTransmission(0x27);
  error = Wire.endTransmission();
  Serial.print("Error: ");
  Serial.print(error);

  if (error == 0) {
    Serial.println(": LCD found.");
    show = 0;
    lcd.begin(16, 2);  // initialize the lcd

    lcd.createChar(1, dotOff);
    lcd.createChar(2, dotOn);

  } else {
```

```
    Serial.println(": LCD not found.");
  }  // if
  lcd.setBacklight(255);
    lcd.home();
    lcd.clear();
    lcd.print("Hello LCD");
    delay(1000);

}

void loop() {
  // print the string when a newline arrives:
  if (stringComplete) {
    Serial.println(inputString);
    // clear the string:

    stringComplete = false;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("data:");
    lcd.setCursor(0, 1);
    lcd.print(inputString);
    inputString = "";
  }
}


void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag so the main loop can
    // do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```

PROGRAM (Transmitter Code)

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {

  Serial.println("iot lab");
  delay(100);

  Serial.println("iot lab");
  delay(100);

  Serial.println("iot lab");
  delay(100);  // delay in between reads for stability
}
```

Proximity-Based Services:

- Bluetooth beacons are employed in projects that provide proximity-based services and notifications.
- This technology is invaluable for location-based marketing, guidingvisitors in a museum or retail environment, and creating interactive exhibits. Visitors receive contextually relevant information and notifications on their smartphones when in proximity to Bluetooth beacons.

Smartphone Integration:

- Bluetooth is a preferred choice for projects that necessitate interactionwith smartphones.
- It allows IoT devices to connect to mobile apps for configuration, control, and data exchange. This is especially useful in consumer IoTprojects, where user-friendliness and smartphone integration are paramount.
- When planning an IoT project, it's vital to carefully consider the specific needs and objectives, as well as factors like power consumption, range, and data throughput. Additionally, security and data privacy should be a top priority, particularly in applications where sensitive information is involved. The choice of communicationmethod plays a pivotal role in the project's success, and selecting the most appropriate technology will ensure that the project meets its goalsefficiently and effectively.

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 7 DATE :	INTRODUCTION TO RASPBERRY PI PLATFORM AND PYTHON PROGRAMMING

AIM:

To write introduction to raspberry pi platform and python programming

ALGORITHM:

Introduction to Raspberry Pi and Python Programming in IoT:

Raspberry Pi is a small, affordable single-board computer that canbe used for a wide range of applications, including Internet of Things (IoT)projects. Python is a popular programming language for IoT development on the Raspberry Pi due to its simplicity and extensive libraries. In this program, we'll provide a basic introduction to the Raspberry Pi platform and demonstrate how to write a simple Python program for an IoT project.

Prerequisites:

- Raspberry Pi (any model)
- MicroSD card with Raspbian OS installed
- Power supply for Raspberry Pi
- Internet connection (Wi-Fi or Ethernet)
- LED and resistor (220-330Ω)
- Jumper wires
- Breadboard (optional)

Program Outline:

Setting Up Raspberry Pi:

- Insert the microSD card with Raspbian OS into the Raspberry Pi.
- Connect the power supply and boot up the Raspberry Pi.
- Connect to the Raspberry Pi via SSH (if headless) or use a monitorand keyboard for direct access.

Basic Python Installation:

By default, Raspberry Pi comes with Python pre-installed.

Verify the version using the *python --version* command.

GPIO (General Purpose Input/Output) Pins:

- Raspberry Pi has GPIO pins that can be used to control externaldevices.
- Identify and understand the GPIO pins on your Raspberry Pi model.

Python GPIO Library:

- Python has libraries like RPi.GPIO or gpiozero to work with GPIOpins.
- Install the library if not already installed (*pip install RPi.GPIO*).

Blinking an LED:

- Connect an LED to a GPIO pin and a ground pin via a resistor.

- Write a Python program to blink the LED on and off.

IoT Integration:

To make it an IoT project, you can:

- Add sensors *(e.g., temperature, humidity*) to collect data.
- Send data to a cloud service or server.
- Create a web interface to monitor and control your IoT device remotely.

Further Learning:

a) Explore more advanced IoT projects and sensors.
b) Learn about MQTT for IoT communication.
c) Dive deeper into Python libraries and frameworks for IoT development.
d) This program provides a basic introduction to Raspberry Pi andPython programming for IoT.
e) Depending on your interests and project goals, you can expand your knowledge and create more complex IoT applications.

Further Learning:

```
from machine import Pin, Timer
led = Pin(2, Pin.OUT)
timer = Timer()

def blink(timer):
   led.toggle()

timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)
```

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 8 DATE :	INTERFACING SENSORS WITH RASPBERRY PI

AIM:

To write a interfacing sensors with Raspberry pi

DEFINITION:

Interfacing sensors with a Raspberry Pi using embedded C programming for IoT applications involves several steps. Here's a high-level overview of the process:

Selecting the Sensor:

Choose a sensor suitable for your IoT project. Common sensors include temperature sensors (*e.g.*, *DHT11/DHT22*), humidity sensors, light sensors (*e.g.*, *LDR*), motion sensors (e.g., PIR), and more. Ensure the sensoryou choose has good documentation and, if possible, code examples for Raspberry Pi.

Wiring:

Connect the sensor to the Raspberry Pi. You'll typically need to connect the sensor's power, ground, and data pins to the appropriate GPIOpins on the Raspberry Pi. Refer to the sensor's datasheet or documentationfor the pinout information.

Raspberry Pi Setup:

Make sure your Raspberry Pi is set up with the necessary software. This includes installing an operating system (*e.g.*, *Raspberry Pi OS*), enabling SSH if necessary, and ensuring you have access to GPIO librariesfor C programming.

C Programming Environment:

Set up your C programming environment on the Raspberry Pi. You can use a text editor like Nano or Vim and compile your C code usingGCC.

Writing the C Code:

Write C code to read data from the sensor. This code typicallyinvolves configuring GPIO pins, initializing the sensor, and reading data from it. Here's a simplified example using the WiringPi library for GPIOaccess:

PROGRAM:

```
# Complete project details at https://RandomNerdTutorials.com/raspberry-pi-pico-dht11-dht22-micropython/

from machine import Pin
from time import sleep
import dht

#sensor = dht.DHT22(Pin(2))
sensor = dht.DHT11(Pin(2))

while True:
  try:
    sleep(2)
    sensor.measure()
    temp = sensor.temperature()
    hum = sensor.humidity()
    temp_f = temp * (9/5) + 32.0
    print('Temperature: %3.1f C' %temp)
    print('Temperature: %3.1f F' %temp_f)
    print('Humidity: %3.1f %%' %hum)
  except OSError as e:
    print('Failed to read sensor.')
```

Data Processing and IoT Integration:

Once you can read data from the sensor, you can process it and integrate it into your IoT project. This might involve sending the data to a cloud platform (e.g., AWS IoT, Google Cloud IoT, or Azure IoT Hub) or alocal IoT gateway for further processing and storage.

Error Handling and Debugging:

Implement error handling in your C code and use debugging techniques to troubleshoot any issues that may arise during development.

Power Management:

Consider how the sensor and Raspberry Pi will be powered in your IoT application. Depending on your project, you may need to addresspower consumption and battery management.

Security:

Implement security best practices for your IoT project, especially ifit involves transmitting sensitive data over the internet.

Testing and Deployment:

- Thoroughly test your IoT system with the sensor and Raspberry Pi in different scenarios. Once you're confident in its functionality, deploy it in your target environment.
- Remember that this is a simplified overview, and the specific details will vary depending on the sensor and IoT platform you're using.

Additionally, consider documenting your project and maintaining clean, modular code to make future enhancements and maintenance easier.

OUTPUT:

```
Sensor Value: 0
Sensor Value: 1
Sensor Value: 0
Sensor Value: 1
...
```

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 9 DATE :	COMMUNICATE BETWEEN ARDUINO AND RASPBERRY PI USING ANY WIRELESS MEDIUM

AIM:

To write a communicate between arduino and raspberry pi using any wireless medium

DEFINITION:

To communicate between an Arduino and a Raspberry Pi using a wireless medium in an IoT (Internet of Things) project, you can choose from several wireless communication protocols such as Wi-Fi, Bluetooth, Zigbee, LoRa, or MQTT, depending on your specific project requirements. Here, I'll provide an example of using Wi-Fi *(ESP8266*) for communicationbetween Arduino and Raspberry Pi with embedded C programs. Please notethat for this example, you'll need an Arduino board with an ESP8266 Wi-Fimodule and a Raspberry Pi with Wi-Fi capabilities.

ALGORITHM:

STEP 1: Hardware Setup:Connect a Bluetooth module to both the Arduino and Raspberry Pi. For Arduino, commonly used modules include HC-05 or HC-06

STEP 2: Install Required Libraries

STEP 3: Arduino Sketch:Initializes the Bluetooth module,communication parameters (baud rate, etc.).

STEP 4: Ensure the Raspberry Pi has Bluetooth support and is discoverable.

STEP 5: Python Script on Raspberry Pi:Discover nearby Bluetoothdevices, including the Arduino module.

STEP 6: Define a protocol for data exchange between the twodevices

STEP 7: Implement error handling in both the Arduino sketch andthe Raspberry Pi script.

STEP 8: Test the communication between the Arduino and Raspberry Pi.

STEP 9: Integrate the wireless communication into your project asneeded and document for future reference.

Arduino with ESP8266 (Wi-Fi) - Embedded C Program

- Set up the Arduino IDE to program your Arduino with ESP8266.
- Install the necessary libraries for ESP8266, including the Wi-Filibrary.
- Write an Arduino sketch *(embedded C program*) that connects to your Wi-Fi network and sends data to the Raspberry Pi over Wi-Fi.

Below is a simplified example:

PROGRAM:

```
#include <ESP8266WiFi.h>

const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";
const char* serverAddress = "RaspberryPiIPAddress";const
int serverPort = 8080;

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void loop() {
  // Your data collection and processing code here

  // Send data to Raspberry Pi
  sendDataToRaspberryPi("Hello, Raspberry Pi!");

  delay(10000); // Send data every 10 seconds
}

void sendDataToRaspberryPi(String data) {
  WiFiClient client;

  if (client.connect(serverAddress, serverPort)) {
    client.print(data);
    client.stop();
    Serial.println("Data sent to Raspberry Pi: " + data);
  } else {
```

```
    Serial.println("Failed to connect to server");
  }
}
```

OUPPUT:

```
Connecting to WiFi...
Connected to WiFi
Data sent to Raspberry Pi: Hello, Raspberry Pi! Data
sent to Raspberry Pi: Hello, Raspberry Pi! Data sent to
Raspberry Pi: Hello, Raspberry Pi!
...
```

Raspberry Pi - Embedded C Program

On your Raspberry Pi, you can write a C program to receive data from the Arduino over Wi-Fi. You can use a socket-based approach for this.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080int

main() {
    int serverSocket, newSocket;
    struct sockaddr_in serverAddr, newAddr;
    socklen_t addrSize;
    char buffer[1024];

    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
```

```
if (serverSocket < 0) {
        perror("Error in socket");
        exit(1);
    }
    printf("Server socket created...\n");

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = PORT;
    serverAddr.sin_addr.s_addr = INADDR_ANY;

    if (bind(serverSocket, (struct sockaddr*)&serverAddr,
sizeof(serverAddr)) < 0) {
        perror("Error in binding");
        exit(1);
    }
    printf("Binding success...\n");

    if (listen(serverSocket, 10) == 0) {
        printf("Listening...\n");
    } else {
        printf("Error in listening\n");
        exit(1);
    }

    addrSize = sizeof(newAddr);
    newSocket = accept(serverSocket, (struct sockaddr*)&newAddr,&addrSize);

    while (1) {
        recv(newSocket, buffer, 1024, 0); printf("Data
        from Arduino: %s\n", buffer);
    }

    close(newSocket);
    close(serverSocket);
```

```
    return 0;
}
```

OUTPUT:

```
Server socket created...
Binding success...
Listening...
```

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 10 DATE :	SETUP A CLOUD PLATFORM TO LOG THE DATA

AIM:

To write setup a cloud platform to log The data

DEFINITION:

Setting up a cloud platform to log data from an IoT device using embedded C programming involves several steps. In this example, I'll provide a high-level overview of the process, but please note that specific implementation details will depend on the cloud platform and IoT hardwareyou're using.

For this example,I'll use Amazon Web Services (AWS) as the cloudplatform and an ESP8266-based IoT device.

ALGORITHM:

STEP 1: An AWS account : Python installed on your local machine.

STEP 2: Create an S3 Bucket for Data Storage:You can use an Amazon S3 bucket to store your log data.

STEP 3: Set Up AWS Access Credentials.

STEP 4: Install Boto3:Boto3 is the AWS SDK for Python. You caninstall it using pip.

STEP 5: simple Python program to log data to an S3 bucket.

Replace 'YOUR_BUCKET_NAME' with your S3 bucket name.

STEP 6: Run the c program to log data to your S3 bucket.

C PROGRAM:

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
const int LM35 = A0;
//---------- Enter you Wi-Fi Details--------//
char ssid[] = "project"; //SSID
char pass[] = "12345678"; // Password
//-----------------------------------------//

WiFiClient  client;

unsigned long myChannelNumber = 000000; // Channel ID here
const int FieldNumber = 1;
const char * myWriteAPIKey = "GIKQJE5UMJ4H3VJI"; // Your Write API Key here

void setup()
{
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
}
void loop()
{
  if (WiFi.status() != WL_CONNECTED)
  {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    while (WiFi.status() != WL_CONNECTED)
    {
      WiFi.begin(ssid, pass);
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }
  int ADC;
  ADC = analogRead(A0);  /* Read Temperature */
  /* LM35 gives output of 10mv/°C */
  Serial.print("gas val = ");
  Serial.print(ADC);
  Serial.println("");
  delay(1000);
  ThingSpeak.writeField(myChannelNumber, FieldNumber,ADC, myWriteAPIKey);
  delay(1000);
}
```

Set Up AWS IoT Thing Shadow (Optional):

If you want to implement device shadow functionality for remotecontrol and status reporting, configure the AWS IoT Thing Shadow accordingly in your embedded C code.

Set Up AWS IoT Rule and Topic:

Create an IoT Rule on AWS IoT Core to route incoming messagesfrom your device to the desired storage service *(e.g., AWS IoT Analytics, AWS Lambda, or AWS S3)*. Define the IoT topic where your device publishes data.

Configure Data Storage:

Set up data storage on AWS. For example, you can store data in an Amazon S3 bucket, a DynamoDB table, or an RDS database, dependingon your requirements.

Testing:

Test your embedded C program on the IoT device to ensure that itcan connect to AWS IoT Core and publish data successfully.

Monitoring and Scaling:

Implement monitoring and scaling solutions as needed to ensurethe reliability and scalability of your IoT data logging system.

Security:

Ensure that you follow best practices for securing your IoT device and the communication with the cloud platform. This includes using securecertificates, implementing access control, and regularly updating device firmware.

Remember that this is a simplified overview, and actual implementation can vary depending on your specific requirements and theIoT platform you choose. Always refer to the documentation and best practices provided by your chosen cloud platform and IoT hardware.

EXPECTED OUTPUT:

The code should connect to the AWS IoT Core endpoint, publish thesensor data to the defined topic, and disconnect without any errors. You won't see any specific output messages other than any potential error messages. The actual sensor data being published will depend on the specific data you're collecting from your device.

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 11 DATE :	LOG DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM

AIM:

To write a log data using raspberry pi and Upload to the cloud platform

DEFINITION:

Logging data with a Raspberry Pi and uploading it to a cloud platform typically involves several steps, including setting up your Raspberry Pi, writing an embedded C program to collect data, and using acloud service for storage and analysis. Here's a high-level overview of theprocess:

1. Raspberry Pi Setup:

- Get a Raspberry Pi board *(e.g., Raspberry Pi 3 or 4)* and necessary accessories (microSD card, power supply, keyboard, mouse, monitor).
- Install a Linux-based operating system on the Raspberry Pi, such asRaspbian (now Raspberry Pi OS), and ensure it's up to date.

2. Hardware Setup:

- Connect any sensors or devices you want to collect data from to the Raspberry Pi GPIO pins or USB ports.
- Install necessary drivers or libraries for the connected sensors/devices.

3. Embedded C Programming:

- Write an embedded C program to read data from your sensors or devices. You can use libraries like WiringPi to interface with GPIOpins.
- Implement data processing and formatting logic within your Cprogram.
- Create a data structure or protocol for sending data to the cloudplatform.

4. Cloud Platform Setup:

- Choose a cloud platform for data storage and analysis *(e.g., AWS, Azure, Google Cloud, IBM Cloud, or a dedicated IoT platform likeAWS IoT, AzureIoT, etc.).*
- Set up an account on the chosen cloud platform.
- Create an IoT device or service on the cloud platform to receive andstore data.

5. Data Transmission:

- In your C program, implement a mechanism to transmit data to the cloud platform. Common protocols for IoT data transmission includeMQTT, HTTP, or WebSocket.
- Securely configure your Raspberry Pi to connect to the internet. Youmay needto set up Wi-Fi or Ethernet.

6. Data Upload:

- Use the appropriate SDK or library in C to connect to your chosen cloud platform's IoT service. For example, if you're using AWS IoT,you can use the AWS IoT Device SDK for C.

- Send data to the cloud platform using the established communicationprotocol.

7. Data Storage and Analysis:

- On the cloud platform, configure data storage solutions such as databasesor object storage to receive and store the data sent fromyour Raspberry Pi.

- Set up data processing and analysis pipelines if needed.

8. Security Considerations:

- Ensure data encryption during transmission (*e.g., TLS/SSL for MQTTorHTTPS)*.

- Implement authentication and authorization mechanisms to secureyour IoTdevice's access to the cloud.

9. Monitoring and Management:

Implement monitoring and management features in your C program to handle errors, reconnect to the cloud in case of disconnection,and provide status updates.

10. Testing and Deployment:

- Thoroughly test your C program on the Raspberry Pi in a controlledenvironment.

- Deploy the Raspberry Pi in your target location and monitor itsperformance remotely.

- Remember that the exact implementation details will vary based on your specific sensors, cloud platform, and requirements. Additionally,consider power management and fault tolerance strategies to ensure your IoT system operates reliably.

PROGRAM:

```
import machine
import urequests
from machine import Pin
import network, time
from dht import DHT11,InvalidChecksum

HTTP_HEADERS = {'Content-Type': 'application/json'}
THINGSPEAK_WRITE_API_KEY = 'GIKQJE5UMJ4H3VJI'

ssid = 'project'
password = '12345678'

# Configure Pico W as Station
sta_if=network.WLAN(network.STA_IF)
sta_if.active(True)

if not sta_if.isconnected():
    print('connecting to network...')
    sta_if.connect(ssid, password)
    while not sta_if.isconnected():
     pass
print('network config:', sta_if.ifconfig())
```

```
while True:
    time.sleep(5)
    pin = Pin(2)
    sensor = DHT11(pin)
    t  = (sensor.temperature)
    h = (sensor.humidity)
    print("Temperature: {}".format(sensor.temperature))
    print("Humidity: {}".format(sensor.humidity))

    dht_readings = {'field1':t, 'field2':h}
    request = urequests.post( 'http://api.thingspeak.com/update?api_key=' +
      THINGSPEAK_WRITE_API_KEY, json = dht_readings, headers =
      HTTP_HEADERS )
    request.close()
    print(dht_readings)
```

RESULT:

This program was successfully executed and output was obtained.

EXPT NO : 12 DATE :	DESIGN AN IOT BASED SYSTEM WITH EMBEDDED C PROGRAM IN IOT

AIM:

To Design an IOT based system with Embedded c program in IOT.

DEFINITION:

Designing an IoT (Internet of Things) system with an embedded C program involves several components and steps. In this example, I'll outlinea basic IoT system for monitoring environmental conditions (temperature and humidity) and controlling a connected device (a smart light) using an embedded C program. Please note that this is a simplified example, and
real-world IoT projects can be much more complex.

COMPONENTS:

✧ **SENSORS:**

1. Temperature and humidity sensor (*e.g., DHT22*)
2. Microcontroller *(e.g., ESP8266 or ESP32*) to interface with thesensor and connect to Wi-Fi

✧ **ACTUATORS:**

Smart light bulb (*e.g., a smart bulb with Wi-Fi connectivity*)

✧ **COMMUNICATION:**

1. Wi-Fi for connecting the microcontroller to the Internet

2. MQTT (Message Queuing Telemetry Transport) protocol for communication between the IoT devices and a cloud-based broker

✧ **CLOUD SERVER:**

A cloud server for receiving sensor data, controlling the smart light,and storing historical data (e.g., AWS IoT, Google Cloud IoT, or a custom server)

Embedded C Program:

The embedded C program will run on the microcontroller andperform the following tasks:

- Read data from the temperature and humidity sensor.
- Connect to the Wi-Fi network.
- Establish an MQTT connection to the cloud broker.
- Publish sensor data to a specific MQTT topic.
- Subscribe to an MQTT topic to receive control commands for thesmart light.
- Control the smart light based on incoming MQTT messages.

Embedded C Program Flow:

1. Initialize the microcontroller and sensor.
2. Connect to the Wi-Fi network using Wi-Fi credentials.
3. Establish an MQTT connection to the cloud broker using MQTT credentials (username, password, and broker address).
4. Create a loop to periodically read sensor data (temperature and humidity).
5. Publish the sensor data to an MQTT topic (e.g., "sensor/environment").
6. Subscribe to an MQTT topic (e.g., "light/control") to receive control commands for the smart light.

7. When a control command is received, interpret it (e.g., "turn on,""turn off"), and control the smart light accordingly.

8. Repeat the loop to continue monitoring and controlling as needed.

9. Handle error conditions and re-establish connections if necessary.

Cloud Server:

1) Set up an MQTT broker (e.g., Mosquitto) on your cloud server.

2) Create an MQTT topic for receiving sensor data (*e.g., "sensor/environment"*) and another for sending control commands tothe smart light (e.g., "light/control").

3) Develop a backend application on the cloud server to:

- Receive and store incoming sensor data.
- Forward control commands to the smart light.
- Implement security measures to ensure data privacy and authentication.

User Interface:

To interact with your IoT system, you can create a web or mobile app that connects to the cloud server. This app can display real-time sensor data, allow users to control the smart light remotely, and provide historical data analysis.

Remember that this is a high-level overview, and implementing an IoT system involves detailed hardware integration, security considerations,and scalability planning. Additionally, you may choose different microcontrollers, sensors, and cloud platforms based on your specific project requirements.

PROGRAM:

```
#define BLYNK_TEMPLATE_ID "TMPL3nUMslvyF"
#define BLYNK_TEMPLATE_NAME "exp 12 iot system"
#define BLYNK_AUTH_TOKEN "_BVCliF1E9c2_sBNqXFMsSky1hb_fUD7"

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
/// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "_BVCliF1E9c2_sBNqXFMsSky1hb_fUD7";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "project";
char pass[] = "12345678";
void setup()
{
  // Debug console
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass,"Blynk.cloud",80);

}
void loop()
{
  Blynk.run();
}
```

RESULT:

This program was successfully executed and output was obtained.