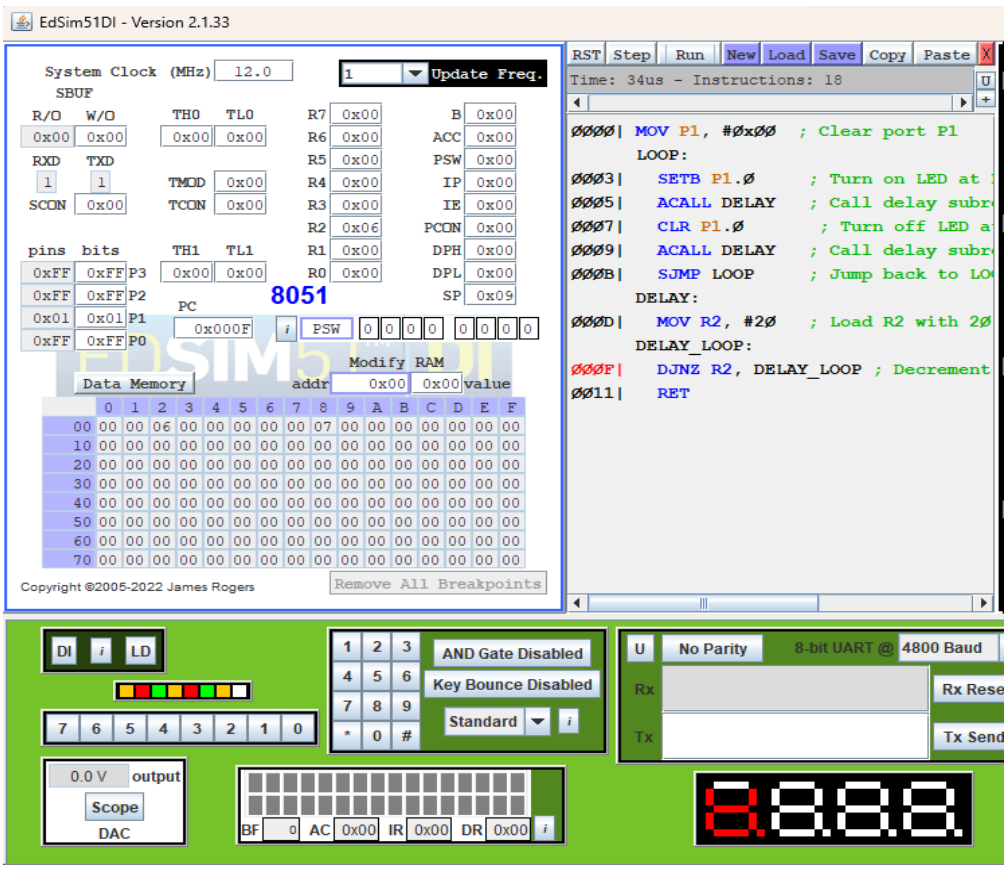
Blinking LED:

```
MOV P1, #0x00  ; Clear port P1
LOOP:
  SETB P1.0     ; Turn on LED at P1.0
  ACALL DELAY   ; Call delay subroutine
  CLR P1.0      ; Turn off LED at P1.0
  ACALL DELAY   ; Call delay subroutine
  SJMP LOOP     ; Jump back to LOOP
DELAY:
  MOV R2, #20   ; Load R2 with 20
DELAY_LOOP:
  DJNZ R2, DELAY_LOOP ; Decrement R2 and repeat if not zero
  RET
```

Output:



Ex.No : 01	Write 8051 Assembly language experiments using Edsim
Date :	

Aim:

Write an 8051 Assembly language program to achieve a specific task or demonstrate a concept.

Apparatus Required:

- Computer with EdSim simulator installed.
- 8051 microcontroller simulation environment (provided by EdSim).
- Text editor for writing Assembly language code.

Procedure:

Launch EdSim:

Open the EdSim simulator on your computer.

Create a New Project:

Start a new project or open an existing project if applicable.

Open the 8051 Microcontroller Environment:

Access the 8051 microcontroller simulation environment within EdSim.

Write Assembly Language Code:

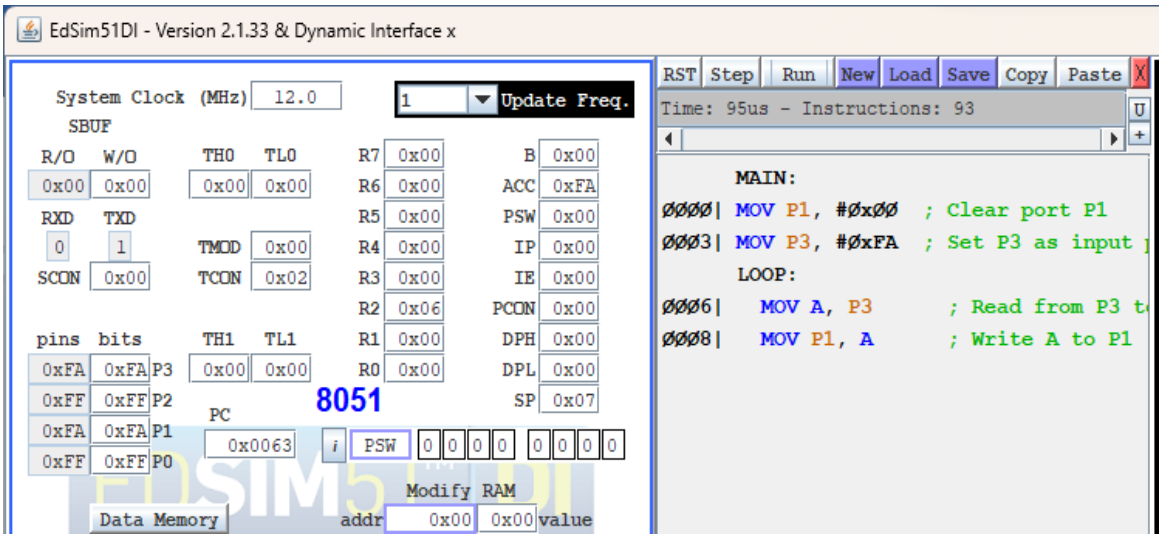
Use a text editor to write the 8051 Assembly language code for your experiment. The code should include:

- Instructions to achieve the experiment's aim.
- Proper comments to explain the code.
- Labels for different sections of the code.
- Load the Code into the Simulator:
- Load your Assembly language code into the 8051 microcontroller simulation environment. This is usually done through the simulator's interface.

Digital Input and Output:

```
MAIN:
MOV P1, #0x00  ; Clear port P1
MOV P3, #0xFA  ; Set P3 as input port
LOOP:
  MOV A, P3      ; Read from P3 to A
  MOV P1, A      ; Write A to P1
```

Output:



Serial Communication

```
ORG 0x00
MOV TMOD, #0x20   ; Set Timer 1 in Mode 2 (8-bit auto-reload)
MOV TH1, #0xFD    ; Load TH1 with FD (9600 Baud Rate)
MOV SCON, #0x50   ; Set Serial Mode and Enable Receiver
SETB TR1          ; Start Timer 1
MAIN:
  JNB TI, MAIN    ; Wait for Transmit Interrupt Flag to set
  MOV SBUF, 'A'   ; Send 'A' character
  CLR TI          ; Clear Transmit Interrupt Flag
  SJMP MAIN
```

Configure the Simulation:

Set any necessary simulation parameters, such as clock speed, input values, or initial conditions.

Run the Simulation:

Execute the simulation to observe the behavior of the 8051 microcontroller based on your Assembly language program.

Debug and Modify Code (If Necessary):

If the simulation does not behave as expected, use the debugging features of EdSim to identify and fix issues in your code.

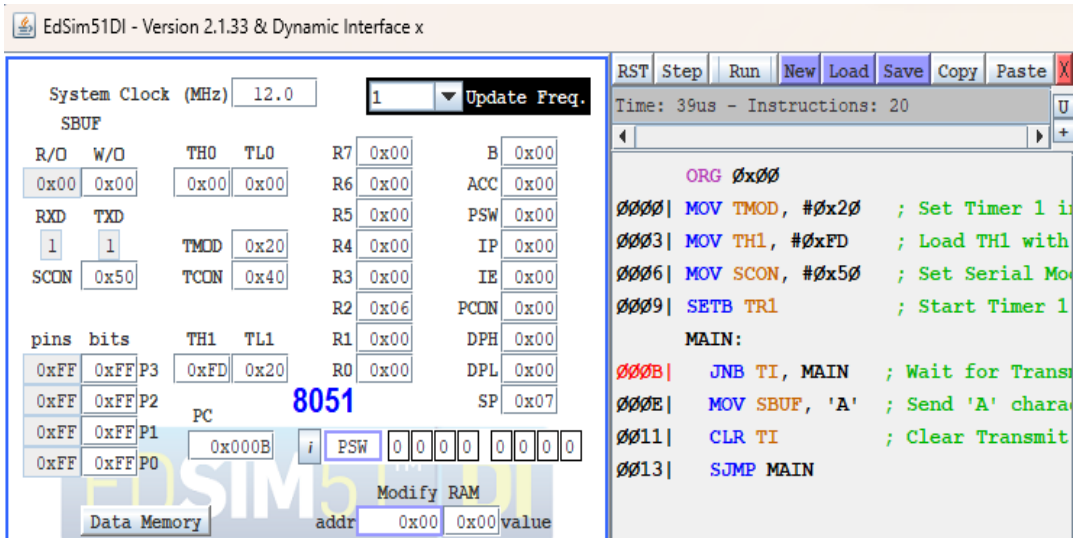
Record and Analyze Results:

Record the results of your experiment and analyze the behavior of the 8051 microcontroller during the simulation.

Save Project:

Save your project to preserve the code, settings, and results for future reference.

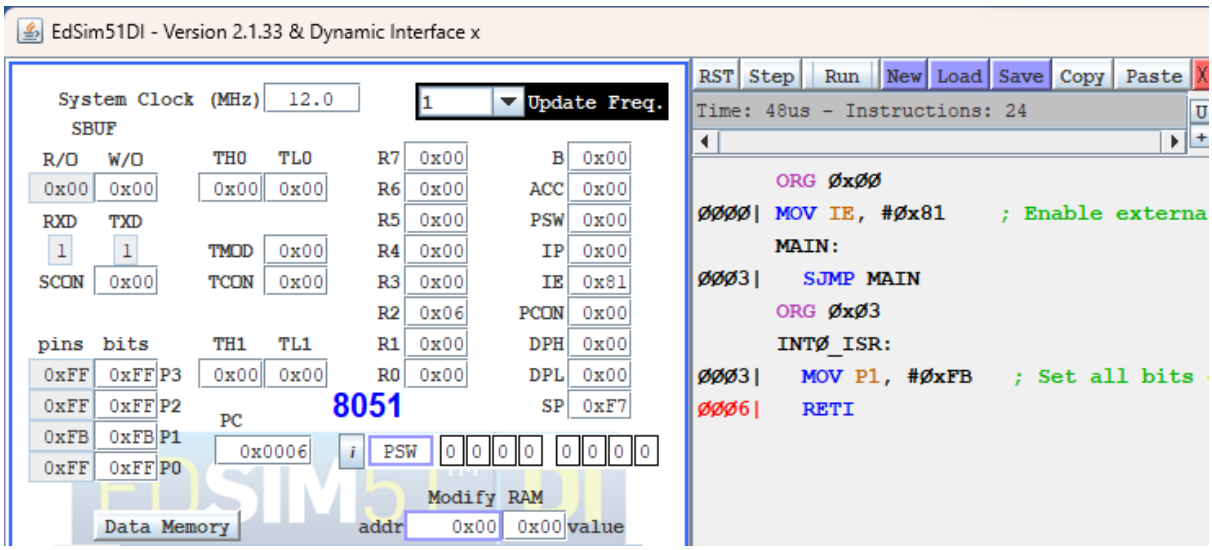
Output:



External Interrupt

```
ORG 0x00
MOV IE, #0x81    ; Enable external interrupt 0 (INT0) and the global interrupt
MAIN:
 SJMP MAIN
ORG 0x03
INT0_ISR:
 MOV P1, #0xFF   ; Set all bits of P1 to high
 RETI
```

Output :



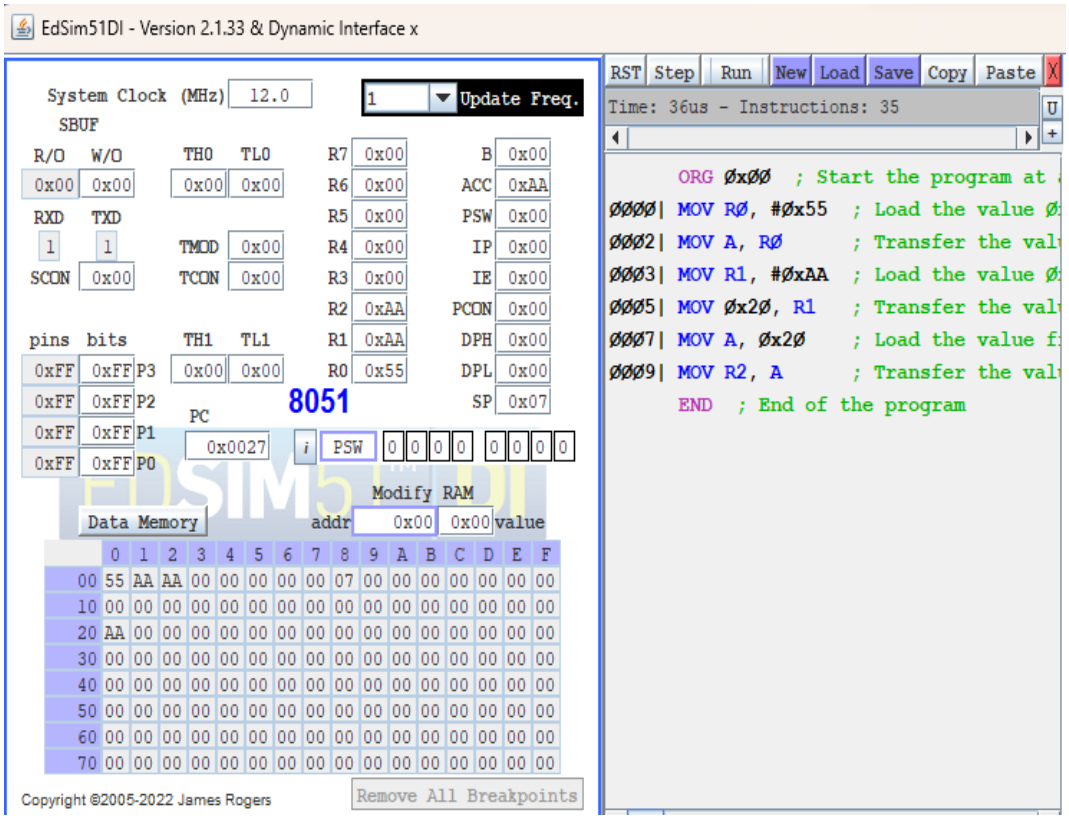
Inference:

Result:

Program:

```
ORG 0x00  ; Start the program at address 0
MOV R0, #0x55  ; Load the value 0x55 into register R0
MOV A, R0      ; Transfer the value from R0 to the accumulator A
MOV R1, #0xAA  ; Load the value 0xAA into register R1
MOV 0x20, R1   ; Transfer the value from R1 to memory location 0x20
MOV A, 0x20    ; Load the value from memory location 0x20 into A
MOV R2, A      ; Transfer the value from A to register R2
END  ; End of the program
```

Output:



Ex.No : 02	Test data transfer between register and memory using Edsim
Date :	

Aim:

Write an 8051 Assembly language program to test data transfer between register and memory using Edsim

Apparatus Required:

- Computer with EdSim simulator installed.
- 8051 microcontroller simulation environment (provided by EdSim).
- Text editor for writing Assembly language code.

Procedure:

Launch EdSim:

Open the EdSim simulator on your computer.

Create a New Project:

Start a new project or open an existing project if applicable.

Open the 8051 Microcontroller Environment:

Access the 8051 microcontroller simulation environment within EdSim.

Write Assembly Language Code:

Use a text editor to write the 8051 Assembly language code for your experiment. The code should include:

- Instructions to achieve the experiment's aim.
- Proper comments to explain the code.
- Labels for different sections of the code.
- Load the Code into the Simulator:
- Load your Assembly language code into the 8051 microcontroller simulation environment. This is usually done through the simulator's interface.

Configure the Simulation:

Set any necessary simulation parameters, such as clock speed, input values, or initial conditions.

Run the Simulation:

Execute the simulation to observe the behavior of the 8051 microcontroller based on your Assembly language program.

Debug and Modify Code (If Necessary):

If the simulation does not behave as expected, use the debugging features of EdSim to identify and fix issues in your code.

Record and Analyze Results:

Record the results of your experiment and analyze the behavior of the 8051 microcontroller during the simulation.

Save Project:

Save your project to preserve the code, settings, and results for future reference.

Inference:

Result:

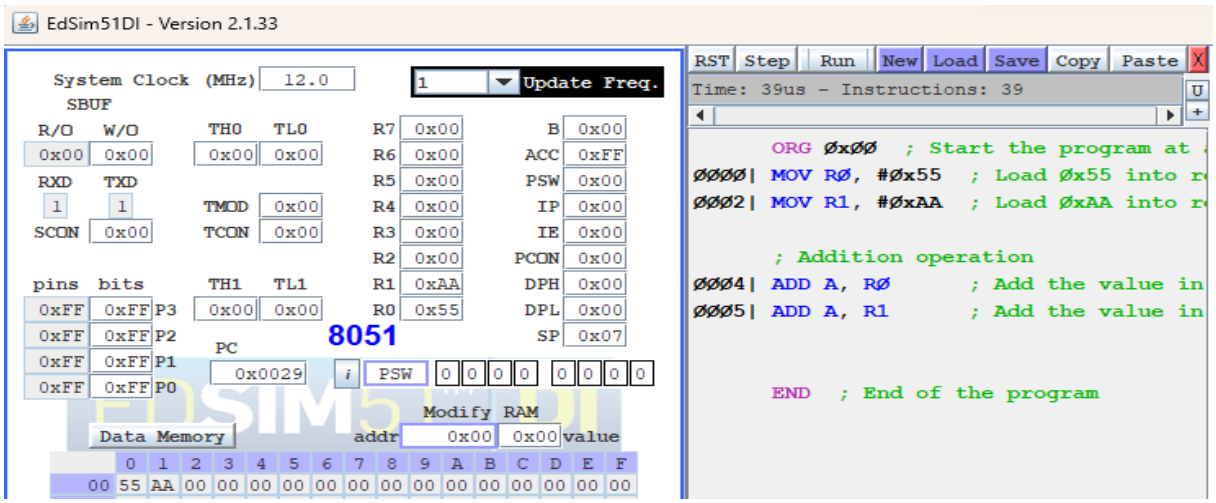
Program:

```
ORG 0x00  ; Start the program at address 0
MOV R0, #0x55  ; Load 0x55 into register R0
MOV R1, #0xAA  ; Load 0xAA into register R1
```

Addition operation

```
ADD A, R0      ; Add the value in R0 to accumulator A
ADD A, R1      ; Add the value in R1 to accumulator A
```

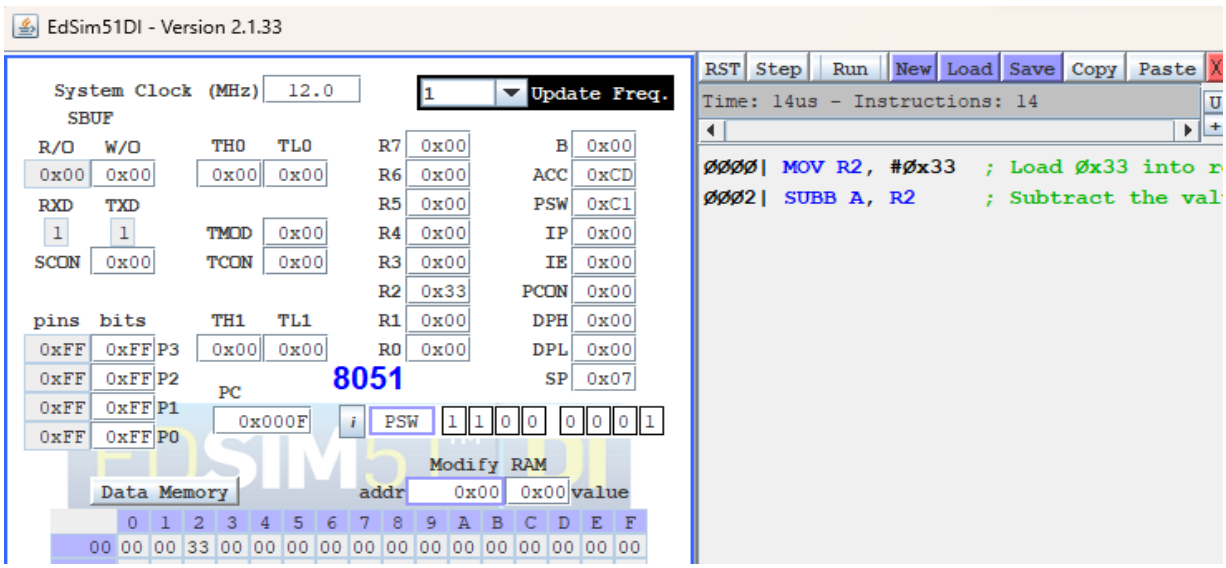
Output:



Subtraction operation

```
MOV R2, #0x33  ; Load 0x33 into register R2
SUBB A, R2     ; Subtract the value in R2 from accumulator A
```

Output:



Ex.No : 03	Perform ALU operations of 8051 using Edsim
Date :	

Aim:

Write an 8051 Assembly language program to perform ALU operations using Edsim

Apparatus Required:

- Computer with EdSim simulator installed.
- 8051 microcontroller simulation environment (provided by EdSim).
- Text editor for writing Assembly language code.

Procedure:

Launch EdSim:

Open the EdSim simulator on your computer.

Create a New Project:

Start a new project or open an existing project if applicable.

Open the 8051 Microcontroller Environment:

Access the 8051 microcontroller simulation environment within EdSim.

Write Assembly Language Code:

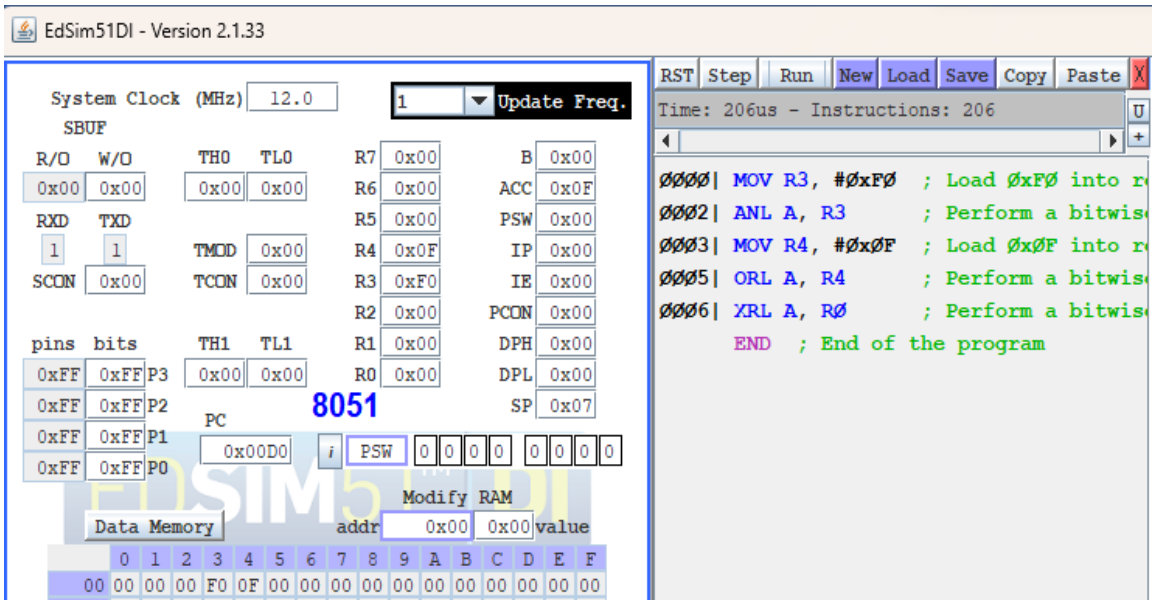
Use a text editor to write the 8051 Assembly language code for your experiment. The code should include:

- Instructions to achieve the experiment's aim.
- Proper comments to explain the code.
- Labels for different sections of the code.
- Load the Code into the Simulator:
- Load your Assembly language code into the 8051 microcontroller simulation environment. This is usually done through the simulator's interface.

Logic operations (AND, OR, XOR)

```
MOV R3, #0xF0  ; Load 0xF0 into register R3
ANL A, R3      ; Perform a bitwise AND between A and R3
MOV R4, #0x0F  ; Load 0x0F into register R4
ORL A, R4      ; Perform a bitwise OR between A and R4
XRL A, R0      ; Perform a bitwise XOR between A and R0
END  ; End of the program
```

Output:



Configure the Simulation:

Set any necessary simulation parameters, such as clock speed, input values, or initial conditions.

Run the Simulation:

Execute the simulation to observe the behavior of the 8051 microcontroller based on your Assembly language program.

Debug and Modify Code (If Necessary):

If the simulation does not behave as expected, use the debugging features of EdSim to identify and fix issues in your code.

Record and Analyze Results:

Record the results of your experiment and analyze the behavior of the 8051 microcontroller during the simulation.

Save Project:

Save your project to preserve the code, settings, and results for future reference.

Inference:

Result:

Basic Program: LED Control

```
#include <8051.h>
void delay(unsigned int time) {
   while(time--) {
      for(int i = 0; i < 120; i++); // Adjust this loop for the desired delay
   }
}
void main() {
   P1 = 0x00; // Initialize Port 1
   while(1) {
      P1 = 0xFF; // Turn on all LEDs connected to Port 1
      delay(1000); // Delay for some time
      P1 = 0x00; // Turn off all LEDs
      delay(1000); // Delay again
   }
}
```

Arithmetic Program: Addition of Two Numbers

```
#include <8051.h>
void main() {
   unsigned char num1 = 10; // First number
   unsigned char num2 = 5;  // Second number
   unsigned char result;
   // Perform the addition
   result = num1 + num2;
   // Store the result in memory or display it as needed
   // For example, you can use P1 to display the result:
   P1 = result;
   while(1) {
      // Your program continues here
   }
}
```

Ex.No : 04	Write Basic and arithmetic programs using Embedded C
Date :	

Aim:

To write basic and arithmetic programs using Embedded C

Apparatus Required:

- Embedded system or microcontroller development board
- Integrated Development Environment (IDE) for embedded programming (e.g., Keil, MPLAB X, etc.)
- USB cable for programming and debugging (if applicable)

Procedure:

- Open your IDE and create a new C project.
- Write the code
- Save the file with a .c extension (e.g., hello.c).
- Compile the code in your IDE.
- Load the compiled code onto your embedded system or microcontroller.
- Run the program and observe the output on the console.

Inference:

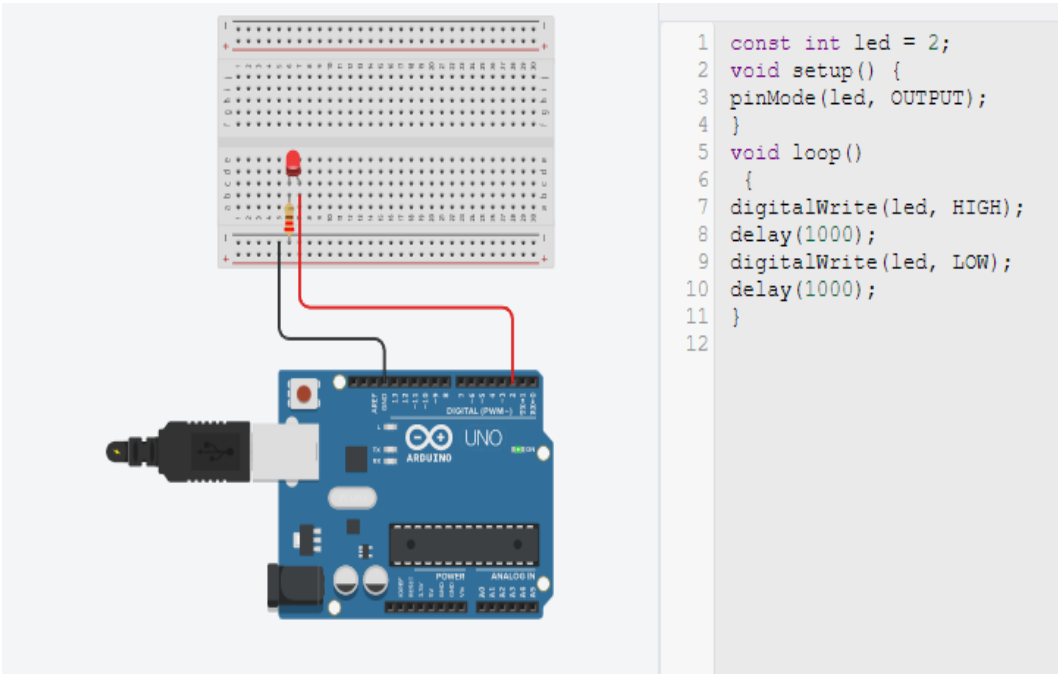
Result:

Control LED Using an Arduino Uno board

Program:

```
const int led = 2;
void setup() {
pinMode(led, OUTPUT);
}
void loop()
 {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

Simulation Output:



Ex.No : 05	Introduction to Arduino platform and programming

Date :	

Aim:

To write a program and execute various sensors using Arduino UNO Board.

Apparatus required:

- Sensors (Ultrasonic Sensor, Gas Sensor, Temperature and Humidity sensors)
- Computer
- Arduino UNO board
- USB Cable
- Jumper Wires and Breadboard
- LED
- Resistors

Software Procedure: (Common to all)

1) Click on Arduino IDE
2) Click on file
3) Click on New
4) Write a Program as per circuit Pin connections
5) Click on Save
6) Click on Verify
7) Click on Upload the code into Arduino Uno by using USB cable.

Hardware Procedure:

(i) To control LED Using an Arduino Uno board

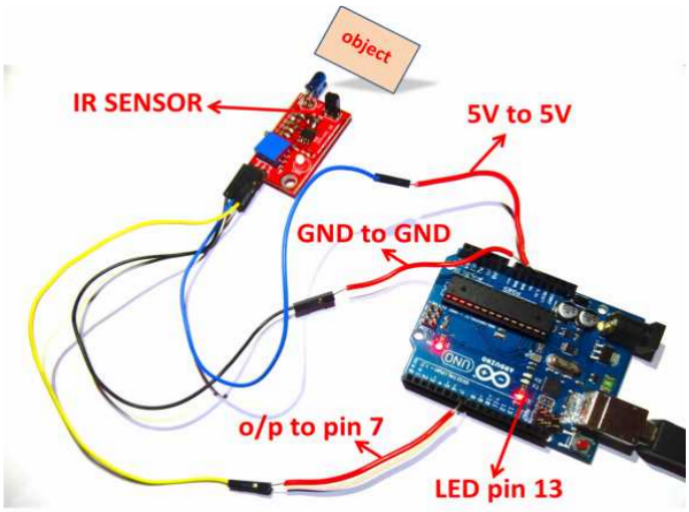
1) LED pin is Connected to Arduino Uno pin of 2.
2) Power jack is connected to the Arduino Uno.
3) A USB connector is connected to Arduino Uno to monitor.
4) Connect the 12V power supply to a development board.
5) Check the output from the development board.

Automatically turn ON and OFF a LED using an Arduino UNO with IR Sensor

Program:

```
void setup() {
pinMode(7, INPUT); // initialize the IR sensor pin as an input:
pinMode(2, OUTPUT); // initialize pin 13 led as output
Serial.begin(9600); //baud rate
}
void loop() {
if(digitalRead(7) == LOW) // if object detected IR sensor sends 0 to pin 7
{
Serial.println("OBJECT detected"); //"object detected" message will be displayed in serial monitor
digitalWrite(2, HIGH); //led pin 13 will be turned on
}
else {
Serial.println("OBJECT not detected"); //"object not detected" message will be displayed in serial
digitalWrite(2, LOW); //led pin 13 will be turned off
}
delay(1000); //delay of one second
}
```

Experimental Setup:



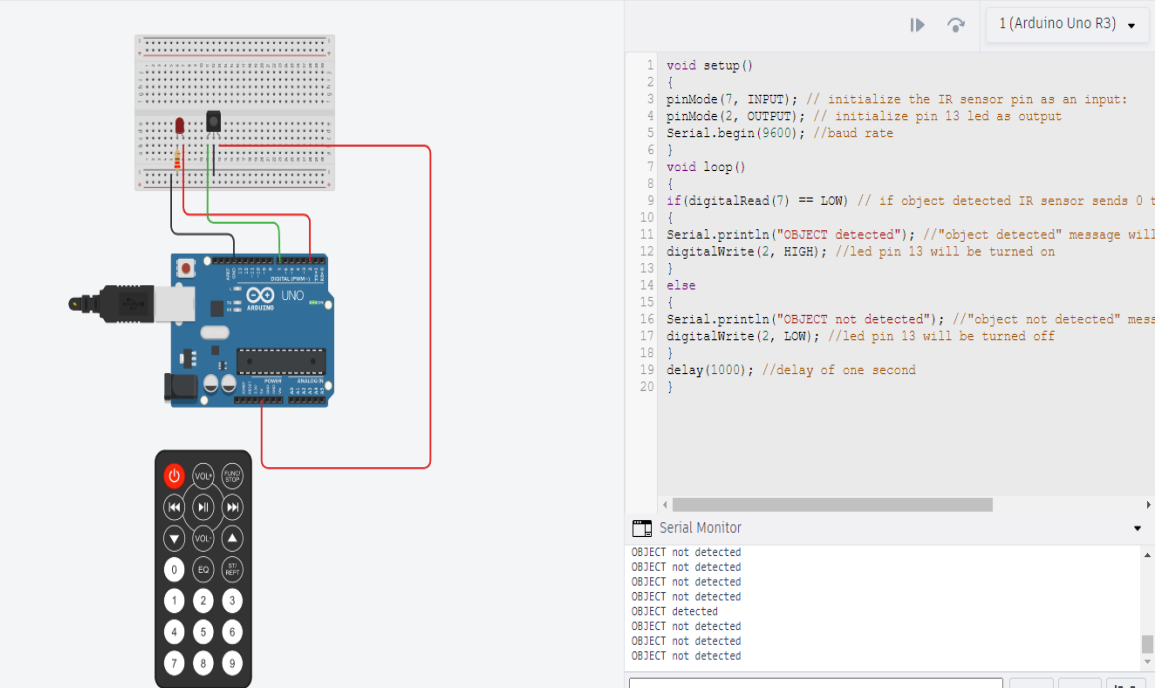
(ii) To write a program and execute automatically turn ON and OFF a LED using an Arduino UNO with IR Sensor

1) LED pin is Connected to Arduino Uno pin of 2.
2) The IR sensor power pin is connected to the Arduino Uno pin of 5V,GND pin is connected to the Arduino Uno pin GND
3) The IR Data pin is connected to the Arduino Uno pin of 7
4) Power jack is connected to the Arduino Uno.
5) A USB connector is connected to Arduino Uno to monitor.
6) Connect the 12V power supply to a development board.
7) Check the output from the development board.

(iii) To write a program and execute automatically turn ON and OFF an LED and Buzzer using an Arduino UNO with Gas Sensor and LCD Display

1) LED pin is Connected to Arduino Uno pin of 5.
2) The buzzer pin is connected to the Arduino Uno pin of 6.
3) The LCD pin is connected to the Arduino Uno pin of 8,9,10,11,12,13.
4) The GAS sensor power pin is connected to the Arduino Uno pin of A0
5) Power jack is connected to the Arduino Uno.
6) A USB connector is connected to Arduino Uno to monitor.
7) Connect the 12V power supply to a development board.
8) Check the output from the development board and Serial monitor

Simulation Output:



Automatically turn ON and OFF an LED and Buzzer using an Arduino UNO with Gas Sensor and LCD Display

Program:

```
#include<LiquidCrystal.h>
int gas;
LiquidCrystal lcd(13,12,11,10,9,8);
void setup()
```

(iv) To write a program and measure Humidity and Temperature values using an Arduino UNO with Temperature and Humidity Sensor

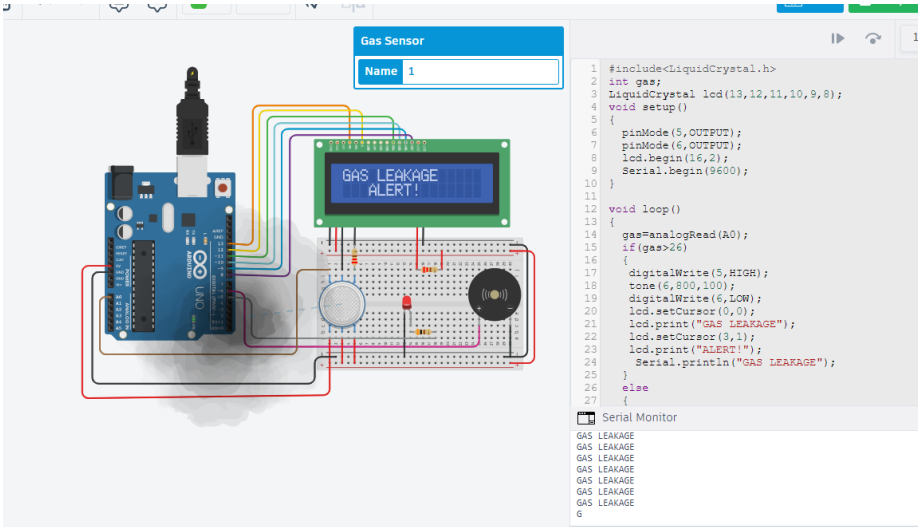
1) The DHT22 sensor Vcc pin is connected to the Arduino Uno pin of Vcc
2) The DHT22 sensor SDA pin is connected to the Arduino Uno pin of 7
3) The DHT22 sensor GND pin is connected to the Arduino Uno pin of GND
4) Power jack is connected to the Arduino Uno.
5) A USB connector is connected to Arduino Uno to monitor.
6) Connect the 12V power supply to a development board.
7) Check the output from the development board and Serial monitor

(v) To write a program and execute turn on an LED when it is dark and turn it off when it is light using an Arduino UNO with an LDR Sensor

1) LED pin is Connected to Arduino Uno pin of 8.
2) LDR sensor pin terminal 1 is connected to Arduino Uno Analog pin A0.
3) Power jack is connected to the Arduino Uno.
4) A USB connector is connected to Arduino Uno to monitor.
5) Connect the 12V power supply to a development board.
6) Check the output from the development board and Serial monitor

```
{
 pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  lcd.begin(16,2);
  Serial.begin(9600);
}
void loop()
{
  gas=analogRead(A0);
  if(gas>26)
   {
    digitalWrite(5,HIGH);
    tone(6,800,100);
    digitalWrite(6,LOW);
    lcd.setCursor(0,0);
    lcd.print("GAS LEAKAGE");
    lcd.setCursor(3,1);
    lcd.print("ALERT!");
     Serial.println("GAS LEAKAGE");
   }
  else
   {
    digitalWrite(5,LOW);
    digitalWrite(6,LOW);
    lcd.setCursor(0,0);
    lcd.clear();
     Serial.println("GAS NOT LEAKAGE");
   }
}
```

Simulation Output



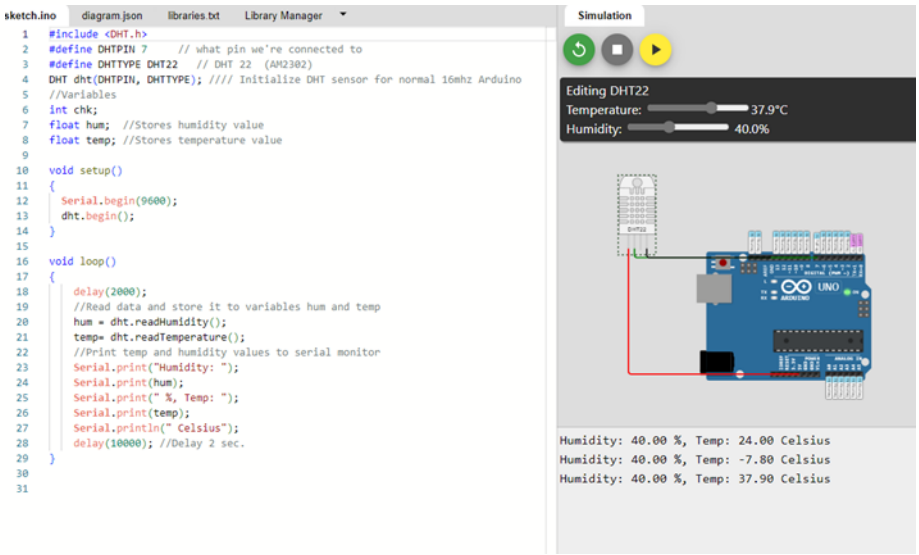
Measure Humidity and Temperature values using an Arduino UNO with Temperature and Humidity Sensor

Program 1:

```
#include <DHT.h>
#define DHTPIN 7     // what pin we're connected to
#define DHTTYPE DHT22   // DHT 22  (AM2302)
DHT dht(DHTPIN, DHTTYPE); //// Initialize DHT sensor for normal 16mhz
Arduino
//Variables
int chk;
float hum;  //Stores humidity value
float temp; //Stores temperature value
void setup()
{
 Serial.begin(9600);
 dht.begin();
}
void loop()
{
   delay(2000);
   //Read data and store it to variables hum and temp
```

```
  hum = dht.readHumidity();
  temp= dht.readTemperature();
  //Print temp and humidity values to serial monitor
  Serial.print("Humidity: ");
  Serial.print(hum);
  Serial.print(" %, Temp: ");
  Serial.print(temp);
  Serial.println(" Celsius");
  delay(10000); //Delay 2 sec.
}
```

Simulation Output:



ii) Program 2:

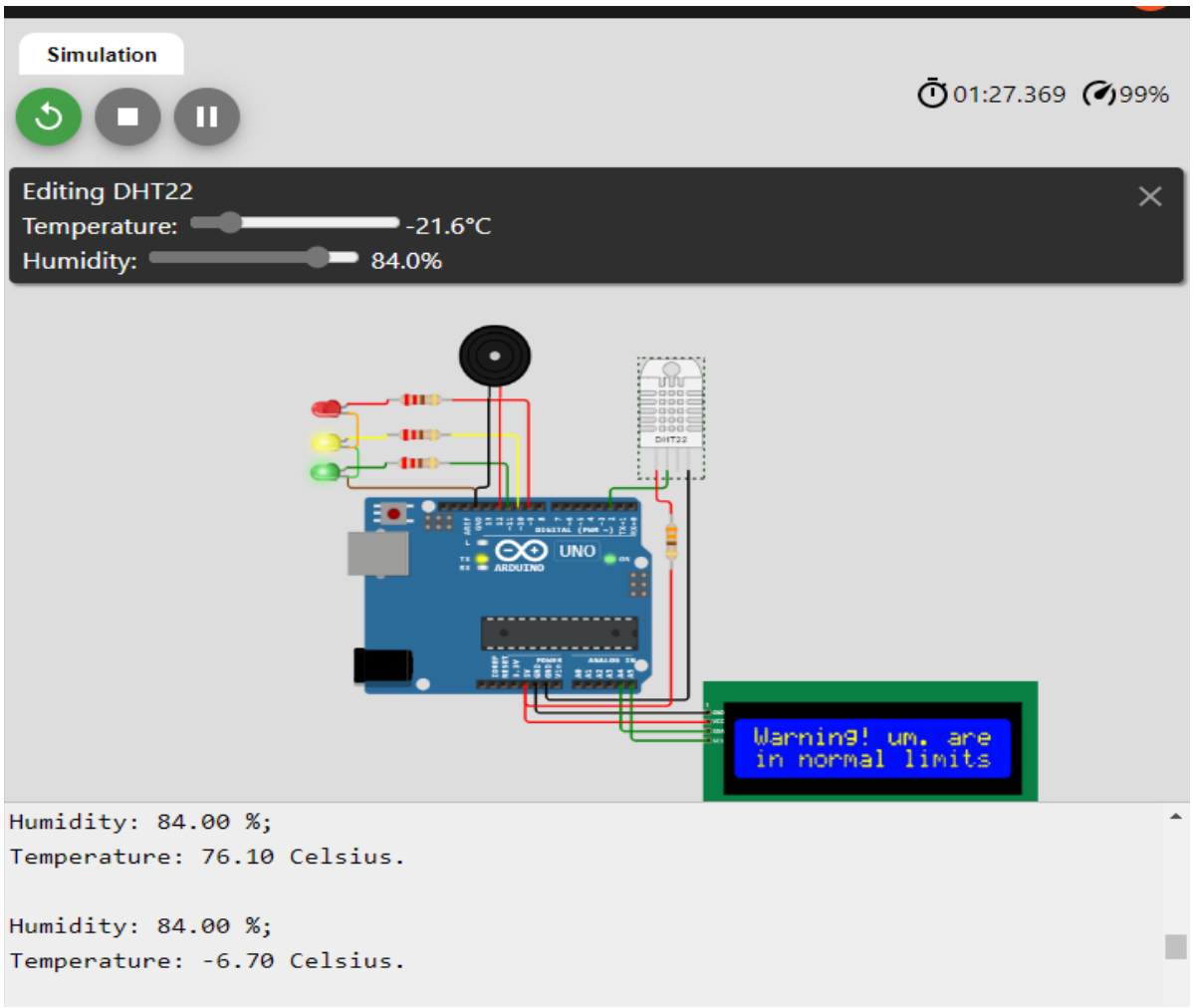
```
//LCD I2C library:
#include <LiquidCrystal_I2C.h>
//DHT22 sensor library:
#include <DHT.h>;
//LCD I2C address 0x27, 16 column and 2 rows!
LiquidCrystal_I2C lcd(0x27, 16, 2);
//Constants:
#define DHTPIN 2          //what pin we're connected to
#define DHTTYPE DHT22     //DHT 22  (AM2302)
```

```
DHT dht(DHTPIN, DHTTYPE); //Initialize DHT sensor for normal 16mhz
Arduino
//Variables:
int chk;
float H; //Humidity value
float T; //Temperature value
int buzzer = 12;
void setup(){
 //Initialize LCD, DHT22 sensor and buzzer:
 lcd.init(); lcd.backlight();
 //Serial Communication is starting with 9600 of baudrate speed
 Serial.begin(115200);
 dht.begin();
 pinMode(13, OUTPUT); pinMode(buzzer, OUTPUT);
 //Print some text in Serial Monitor
 Serial.println("DHT22 sensor with Arduino Uno R3!");
 pinMode(9, OUTPUT); pinMode(10, OUTPUT); pinMode(11, OUTPUT);
}
void loop(){
 delay(2000);
 //Read data and store it to variables hum and temp
 H = dht.readHumidity();
 T = dht.readTemperature();
 //Print temp and humidity values to serial monitor
 Serial.print("Humidity: ");
 Serial.print(H);
 Serial.println(" %; ");
 Serial.print("Temperature: ");
 Serial.print(T);
 Serial.println(" Celsius.\n");
 /*If humidity is higher than 70% &
 temperature is higher than 30 degrees Celsius
 then it will show on LCD „Too warm! Cool down!”*/
 if(H >= 70.00 && T >= 30.00){
```

```
 digitalWrite(9, HIGH);
 digitalWrite(10, LOW);
 digitalWrite(11, LOW);
 lcd.println("   Too warm!    ");
 lcd.setCursor(0, 1);
 lcd.println("   Cool down!   ");
 lcd.setCursor(0, 0);
 digitalWrite(buzzer, 1); tone(buzzer, 900, 100);
 delay(400);
 digitalWrite(buzzer, 0); tone(buzzer, 900, 100);
 delay(400);
 digitalWrite(buzzer, 1); tone(buzzer, 900, 100);
 delay(400);
 digitalWrite(buzzer, 0); tone(buzzer, 900, 100);
 delay(400);
}else{
/*If humidity is lower than 70% &
temperature is lower than 30 degrees Celsius
then it will show on LCD „Temp. & hum. are in normal limits”*/
 digitalWrite(9, LOW);
 digitalWrite(10, LOW);
 digitalWrite(11, HIGH);
 lcd.println("Temp. & hum. are");
 lcd.setCursor(0, 1);
 lcd.println("in normal limits");
 lcd.setCursor(0, 0);
 digitalWrite(buzzer, 0);
}
/*If either humidity is lower than 70%, but
temperature is higher than 30 degrees Celsius,
then it will show on LCD „Be ware! Temp. too high” or
humidity is higher than 70%, but
temperature is lower than 30 degrees Celsius, then
it will show on LCD „Be ware! Hum. too high”*/
```

```
 if(H < 70.00 && T >= 30.00){
  digitalWrite(9, LOW);
  digitalWrite(10, HIGH);
  digitalWrite(11, LOW);
  lcd.println("Warning!        ");
  lcd.setCursor(0, 1);
  lcd.println("Temp. too high! ");
  lcd.setCursor(0, 0);
  digitalWrite(buzzer, 1); tone(buzzer, 400, 400);
  delay(400);
  digitalWrite(buzzer, 0); tone(buzzer, 400, 400);
  delay(400);
 }
 if(H >= 70.00 && T < 30.00){
  digitalWrite(9, LOW);
  digitalWrite(10, HIGH);
  digitalWrite(11, LOW);
  lcd.println("Warning!        ");
  lcd.setCursor(0, 1);
  lcd.println("Hum. too high!  ");
  lcd.setCursor(0, 0);
  digitalWrite(buzzer, 1); tone(buzzer, 400, 400);
  delay(400);
  digitalWrite(buzzer, 0); tone(buzzer, 400, 400);
  delay(400);
 }
}
```

Simulation Output:



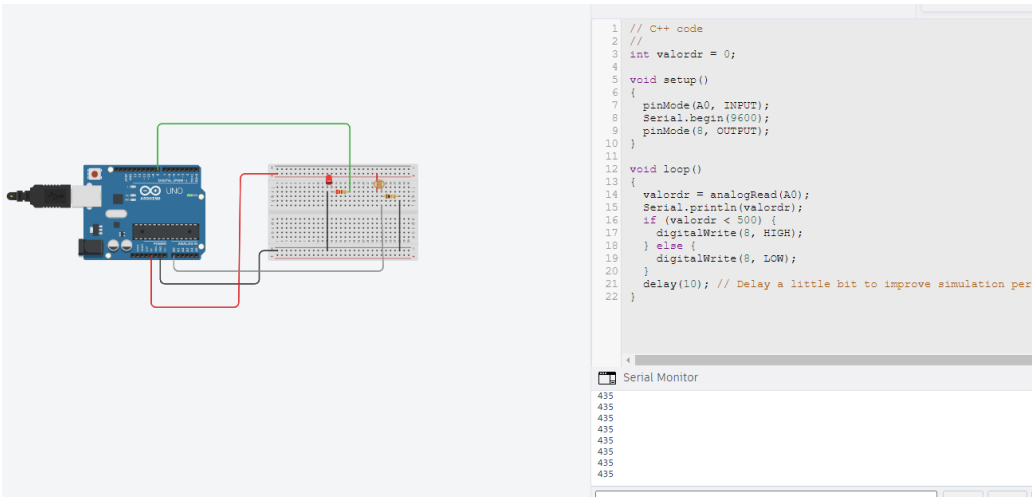
Turn on an LED when it is dark and turn it off when it is light using an Arduino UNO with an LDR Sensor

Program:

```
int valordr = 0;
void setup()
{
 pinMode(A0, INPUT);
 Serial.begin(9600);
 pinMode(8, OUTPUT);
}
void loop()
{
```

```
  valordr = analogRead(A0);
  Serial.println(valordr);
  if (valordr < 500) {
    digitalWrite(8, HIGH);
  } else {
    digitalWrite(8, LOW);
  }
  delay(10); // Delay a little bit to improve simulation performance
}
```

Simulation Output:



Inference:

Result :

Program:

```
#include <SoftwareSerial.h>
SoftwareSerial BTserial(D2, D3); // RX, TX pins of HC-05 module
#define LED_PIN D4 // Digital pin for LED control
void setup() {
  Serial.begin(115200);
  BTserial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
  if (BTserial.available()) {
    char data = BTserial.read();
    if (data == '1') {
      digitalWrite(LED_PIN, HIGH);
      Serial.println("LED: ON");
    } else if (data == '0') {
      digitalWrite(LED_PIN, LOW);
      Serial.println("LED: OFF");
    }
  }
}
```

Ex.No : 06	Write a program to interface Bluetooth and control a LED using an ESP8266
Date :	

Aim:

To write a program to interface Bluetooth and control a LED using an ESP8266.

Hardware Required:

- Computer
- ESP8266 (NodeMCU)
- USB Cable
- Bluetooth Module HC-05
- Power Supply
- Jumper Wires
- Breadboard

Software Required:

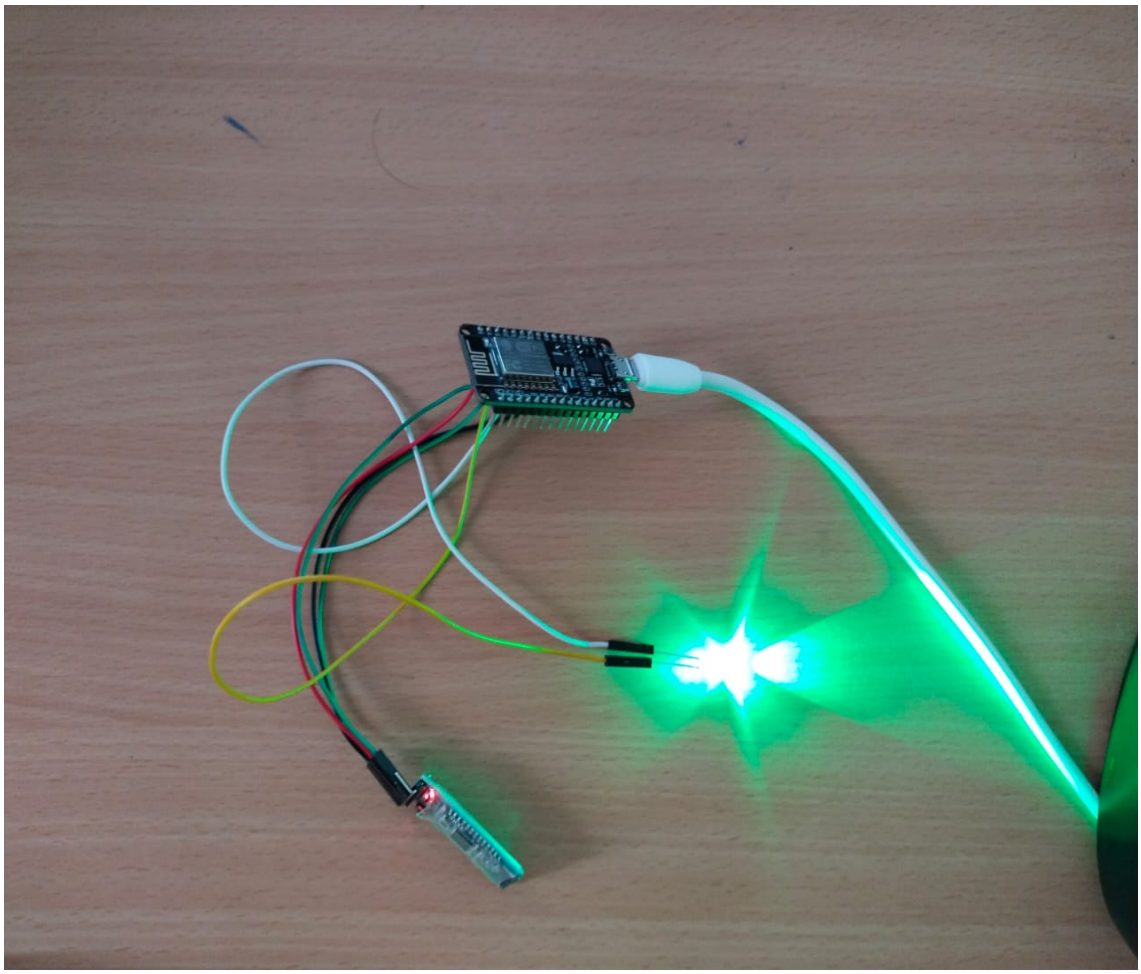
- Arduino IDE
- Bluetooth Terminal App (for testing)

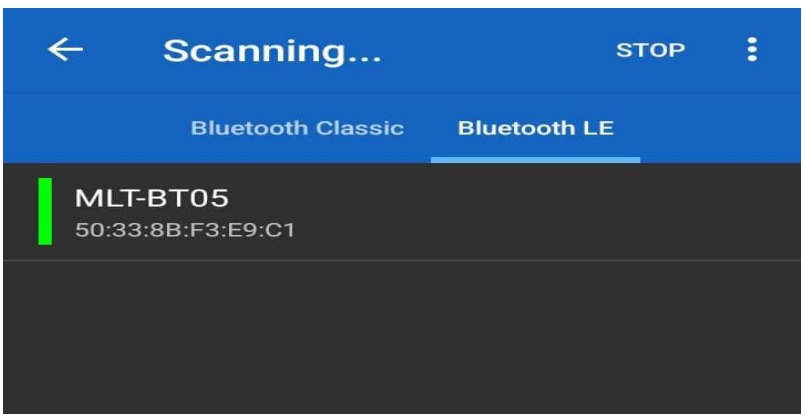
Procedure:

Step 1:. Arduino IDE:

- Open Arduino IDE.
- Click on File.
- Click on Preferences and paste the Additional Boards Manager URLs.
- Click on Tools > Board > Boards Manager > Search ESP8266 and click Install.
- Go to Boards, click on ESP8266, and select Nodemcu 1.0 (ESP-12 E module).
- Write a program as per the circuit pin connections.
- Click on Save.
- Click on Verify.
- Upload the code into ESP8266 using a USB cable.

OUTPUT:




Scanning...
STOP
Bluetooth Classic
Bluetooth LE
MLT-BT05
50:33:8B:F3:E9:C1

- Open the serial monitor and check whether the Wi-Fi is connected.

Step 2: Install SoftwareSerial Library

The SoftwareSerial library allows you to create additional software serial ports on the ESP8266. Go to Sketch -> Include Library -> Manage Libraries, search for "SoftwareSerial", and install the library.

Step 3: Connect the Hardware

- Connect your ESP8266 to your computer and set up the hardware connections:
- Connect VCC of the Bluetooth module to the 3.3V pin on the ESP8266.
- Connect GND of the Bluetooth module to the GND pin on the ESP8266.
- Connect TX of the Bluetooth module to pin D2 on the ESP8266.
- Connect RX of the Bluetooth module to pin D3 on the ESP8266.
- Connect the anode of the LED to pin D4 on the ESP8266.
- Connect the cathode of the LED to a current-limiting resistor (e.g., 220 Ohms).
- Connect the other end of the resistor to the GND pin on the ESP8266.

Step 4: Upload the Code

Copy and paste the provided Arduino code (from the previous messages) into the Arduino IDE.

Step 5: Set the Board and Port

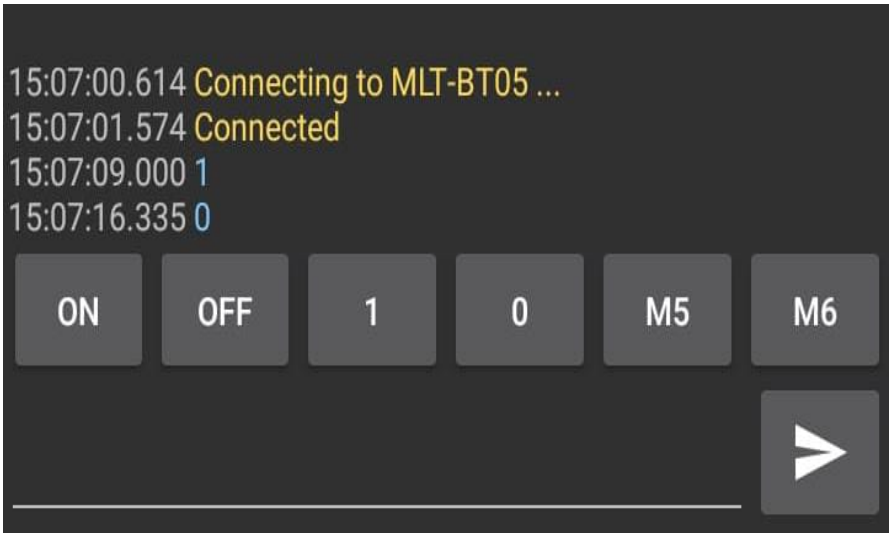
Go to Tools -> Board and select your ESP8266 board (e.g., NodeMCU 1.0). Also, go to Tools -> Port and select the port your ESP8266 is connected to.

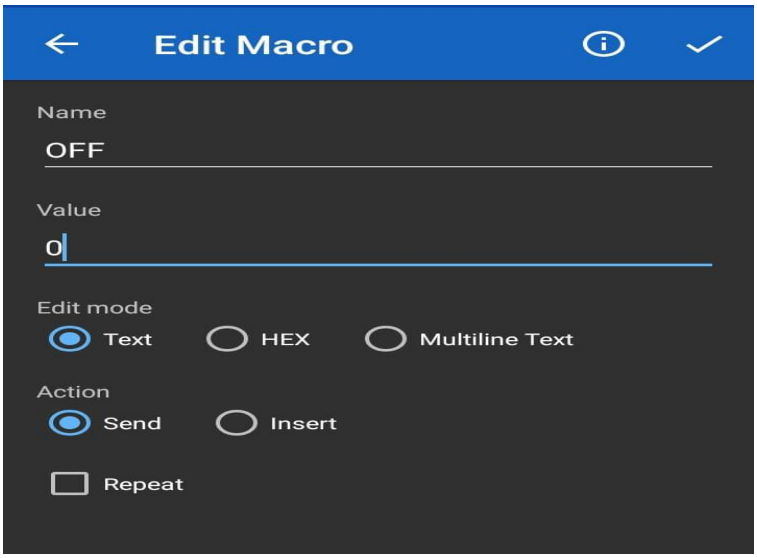
Step 6 : Upload the Code

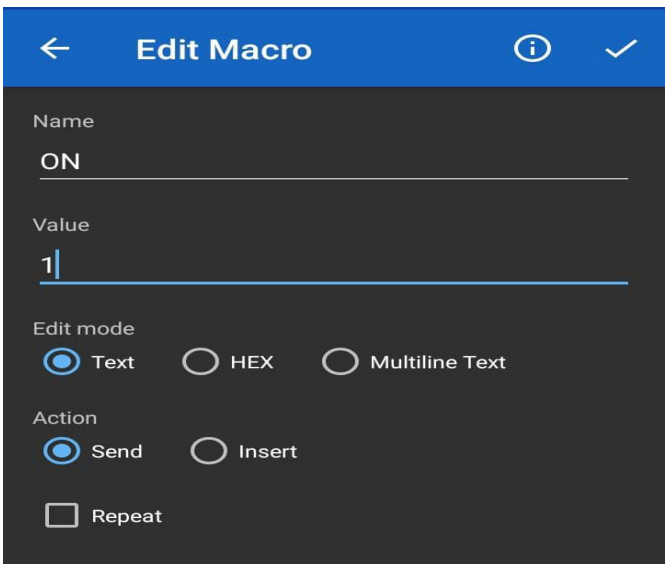
Click on the Upload button (arrow icon) to compile and upload the code to your ESP8266.

Step 7: Open the Serial Monitor

After the upload is complete, open the Serial Monitor (Tools -> Serial Monitor) to view the debug messages. This helps you troubleshoot and see if your ESP8266 is communicating with the Bluetooth module correctly.


15:07:00.614 Connecting to MLT-BT05 ...
15:07:01.574 Connected
15:07:09.000 1
15:07:16.335 0
ON
OFF
1
0
M5
M6


Edit Macro
Name
OFF
Value
0
Edit mode
Text
HEX
Multiline Text
Action
Send
Insert
Repeat


Edit Macro
Name
ON
Value
1
Edit mode
Text
HEX
Multiline Text
Action
Send
Insert
Repeat

Step 8: Connect Bluetooth

Pair your Bluetooth module with a mobile device. Use a Bluetooth terminal app on your mobile device to connect to the Bluetooth module.

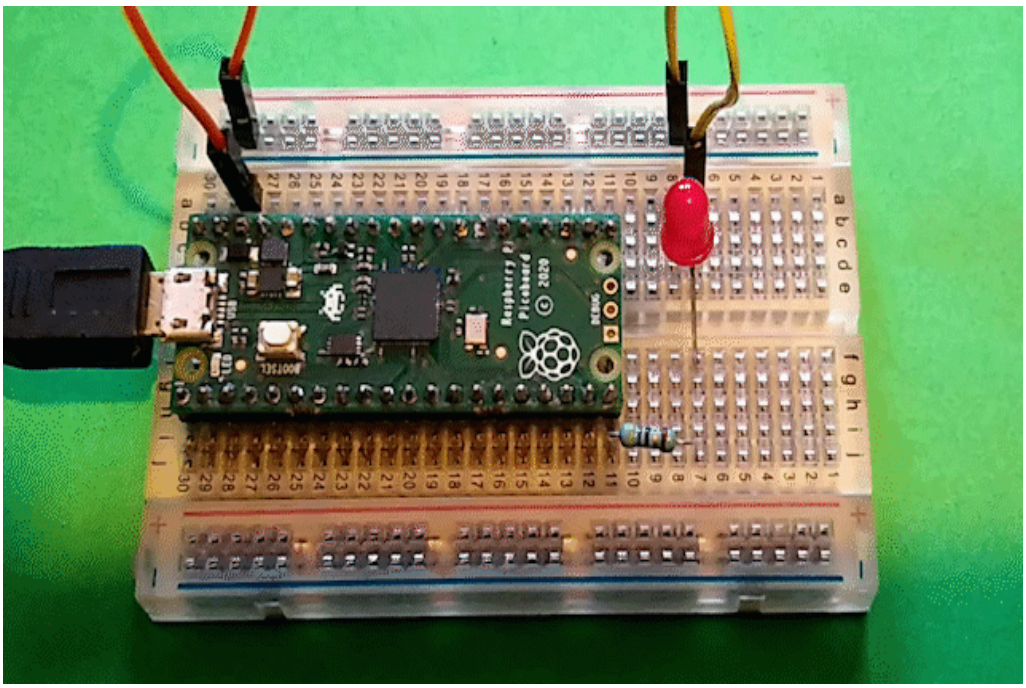
Step 9: Control the LED

Send '1' to turn on the LED and '0' to turn it off via the Bluetooth terminal app. The Arduino code reads the Bluetooth input and controls the LED accordingly.

Inference:

Result:

Raspberry Pi Picco:



Download and Install Raspberry Pi Imager:

Downloads

Raspberry Pi OS (previously called Raspbian) is our official operating system for **all** models of the Raspberry Pi.
Use **Raspberry Pi Imager** for an easy way to install Raspberry Pi OS and other operating systems to an SD card ready to use with your Raspberry Pi:

- Raspberry Pi Imager for Windows
- Raspberry Pi Imager for macOS
- Raspberry Pi Imager for Ubuntu

Ex.No : 07	Introduction to Raspberry PI platform and python programming
Date :	

Aim:

To study about the Raspberry Pi platform and to demonstrate basic GPIO interactions using Python programming

Apparatus Required:

- Raspberry Pi board (any model with GPIO pins)
- Breadboard and jumper wires
- LED (Light Emitting Diode)
- Resistors (e.g., 220Ω)
- Tactile switch/button

Procedure:

1. Setting up the Raspberry Pi:
   - Connect the Raspberry Pi to a monitor, keyboard, and mouse.
   - Power up the Raspberry Pi using a micro USB power supply.
   - Install the Raspbian operating system (or any preferred distribution) if not already installed.
2. Connecting Hardware:
   - Connect an LED to GPIO pin 17 (you can choose a different pin if desired).
   - Connect a resistor (e.g., 220Ω) in series with the LED.
   - Connect the other end of the resistor to the ground (GND) pin on the Raspberry Pi.
   - Connect a tactile switch/button to GPIO pin 18 (you can choose a different pin if desired).
   - Connect one side of the switch to the GPIO pin and the other side to the ground.
3. Python Programming:
   - Open a Python IDE on the Raspberry Pi (IDLE or Thonny, for example).
   - Write a Python script to control the LED based on the state of the switch.

```
# Import necessary libraries
import RPi.GPIO as GPIO
import time
# Set GPIO mode and setup pins
```

```
GPIO.setmode(GPIO.BCM)
led_pin = 17
switch_pin = 18
GPIO.setup(led_pin, GPIO.OUT)
GPIO.setup(switch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
try:
    while True:
        # Check the state of the switch
        if GPIO.input(switch_pin) == GPIO.LOW:
            print("Button pressed! Turning on LED.")
            GPIO.output(led_pin, GPIO.HIGH)
        else:
            print("Button not pressed. Turning off LED.")
            GPIO.output(led_pin, GPIO.LOW)
        # Add a short delay to avoid button debouncing
        time.sleep(0.2)
except KeyboardInterrupt:
    print("Experiment terminated by user.")
finally:
    # Cleanup GPIO settings on exit
    GPIO.cleanup()
```

4. Run the Experiment:

- Save the Python script on your Raspberry Pi.
- Open a terminal and navigate to the directory where the script is saved.
- Run the script using the command python script_name.py (replace script_name.py with the actual name of your script).

5. Observations:

- Observe the LED behavior when the button is pressed and released.
- Understand how GPIO pins are used for input and output.
- Learn the basics of controlling hardware using Python on the Raspberry Pi.

Inference:

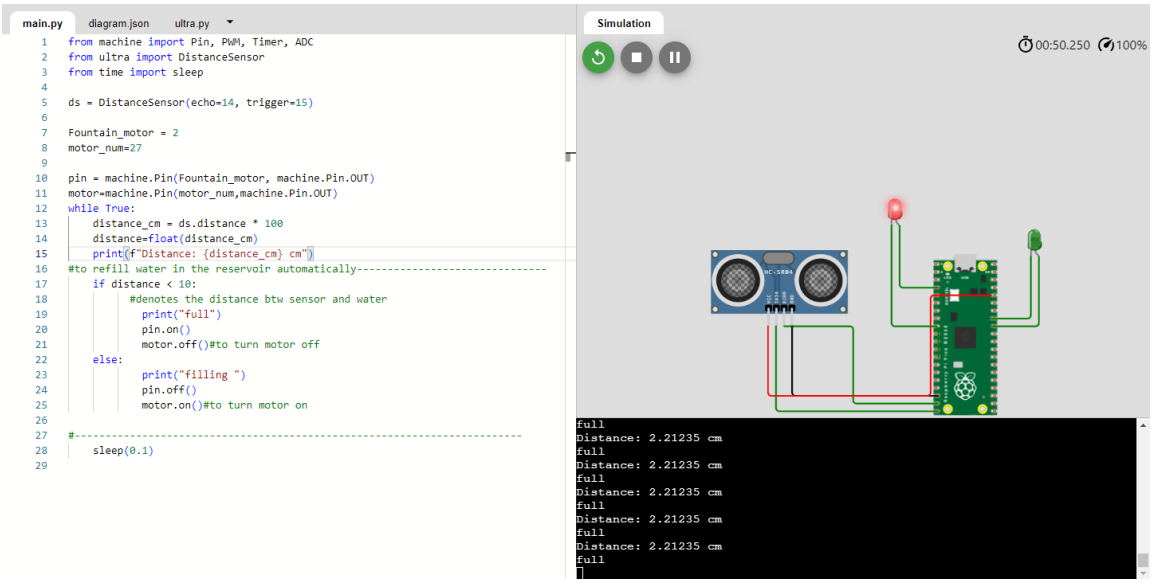
Result:

Interfacing Ultrasonic Sensor with Raspberry PI

Program:

```
from machine import Pin, PWM, Timer, ADC
from ultra import DistanceSensor
from time import sleep
ds = DistanceSensor(echo=14, trigger=15)
Fountain_motor = 2
motor_num=27
pin = machine.Pin(Fountain_motor, machine.Pin.OUT)
motor=machine.Pin(motor_num,machine.Pin.OUT)
while True:
   distance_cm = ds.distance * 100
   distance=float(distance_cm)
   print(f"Distance: {distance_cm} cm")
#to refill water in the reservoir automatically------------------------------
   if distance < 10:
        #denotes the distance btw sensor and water
         print("full")
         pin.on()
         motor.off()#to turn motor off
   else:
         print("filling ")
         pin.off()
         motor.on()#to turn motor on
#------------------------------------------------------------------------
   sleep(0.1)
```

Simulation Output:



Ex.No : 08	Interfacing Sensors with Raspberry PI
Date :	

Aim:

To interface various sensors with Raspberry PI

Apparatus Required:

- Sensors (Ultrasonic Sensor, PIR Sensor & Temperature and Humidity sensors)
- Computer
- Raspberry pi board(picco)
- USB Cable
- Jumper Wires and Breadboard
- LED
- Wowki simulation environment

Software Procedure: (Common to all)

1) Click on raspberry pi platform
2) Click on file
3) Click on New
4) Write a Program as per circuit Pin connections
5) Click on Save
6) Click on Verify

7) Click on Upload the code into Raspberry pi by using USB cable.

Hardware Procedure:

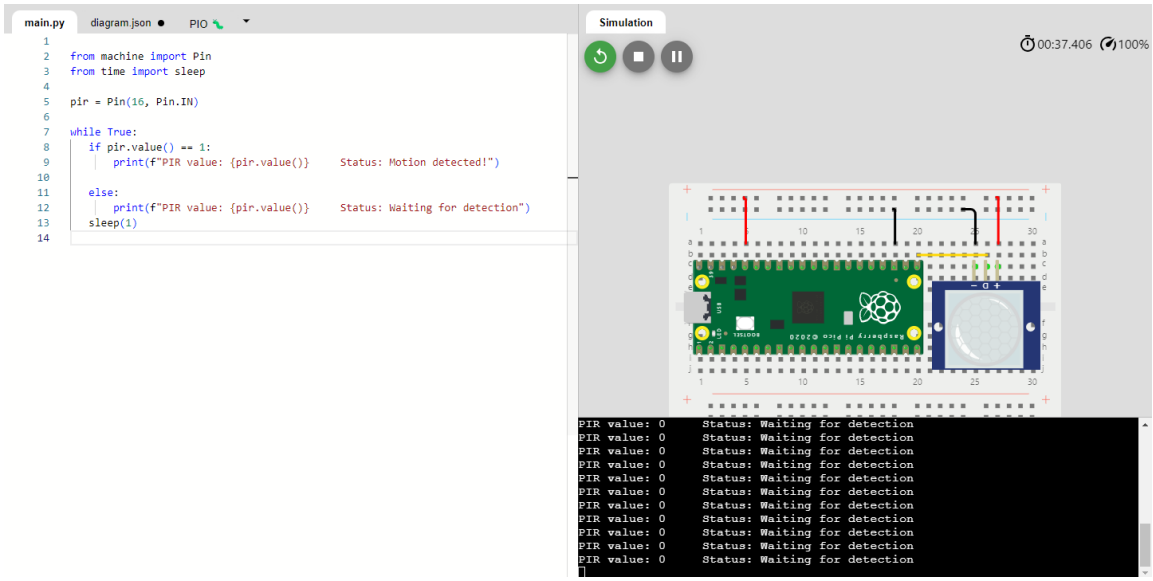
(i) Interfacing Ultrasonic Sensor with Raspberry PI

6) LED pin and Ultrasonic pin is Connected to Raspberry pi pin .
7) Power jack is connected to the Raspberry pi.
8) A USB connector is connected to Raspberry pi to monitor.
9) Connect the 12V power supply to a development board.
10) Check the output from the development board.

Interfacing PIR Sensor with Raspberry PI
Program:

```
from machine import Pin
from time import sleep
pir = Pin(16, Pin.IN)
while True:
  if pir.value() == 1:
     print(f"PIR value: {pir.value()}     Status: Motion detected!")
  else:
     print(f"PIR value: {pir.value()}     Status: Waiting for detection")
  sleep(1)
```

Simulation Output:



Interfacing DHT22 Sensor with Raspberry PI

Program:

```
from machine import Pin
from utime import sleep
from dht import DHT22
sensor = DHT22(Pin(16))
red=Pin(11,Pin.OUT)
blue=Pin(3,Pin.OUT)
green=Pin(8,Pin.OUT)
while True:
    sleep(1)
```

(ii) Interfacing PIR Sensor with Raspberry PI

1) PIR sensor pin is Connected to Raspberry pi pin .
2) Power jack is connected to the Raspberry pi.
3) A USB connector is connected to Raspberry pi to monitor.
4) Connect the 12V power supply to a development board.
5) Check the output from the development board.

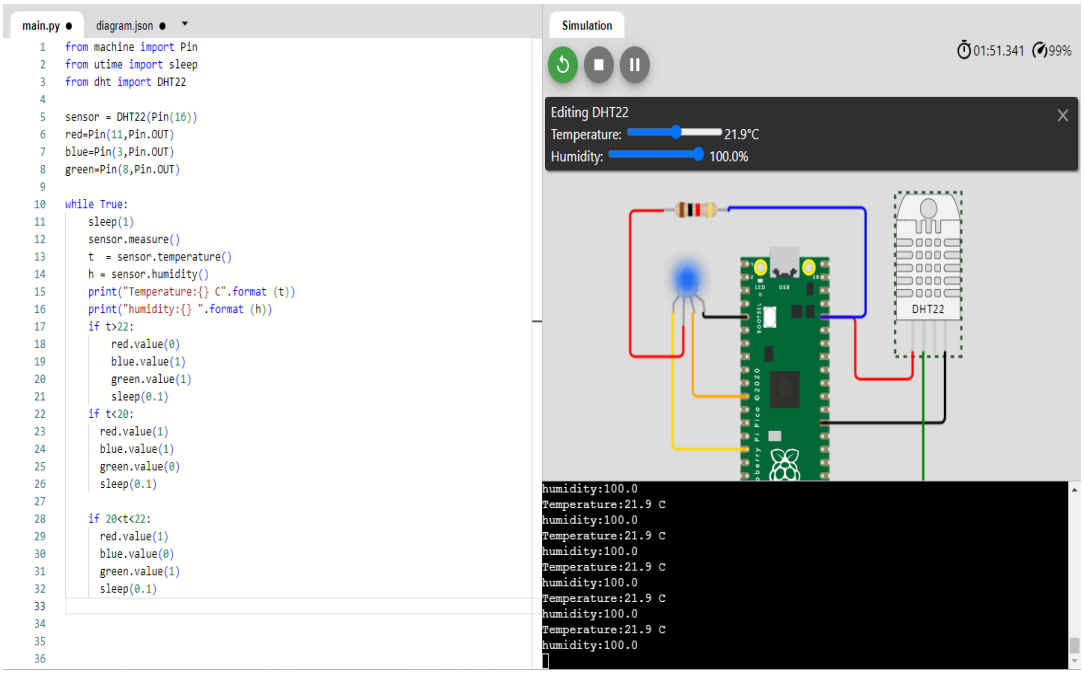
(iii) Interfacing DHT22 Sensor with Raspberry PI

1) DTH22 sensor pin is Connected to Raspberry pi pin .
2) Power jack is connected to the Raspberry pi.
3) A USB connector is connected to Raspberry pi to monitor.
4) Connect the 12V power supply to a development board.
5) Check the output from the development board.

```
sensor.measure()
t  = sensor.temperature()
h = sensor.humidity()
print("Temperature:{} C".format (t))
print("humidity:{} ".format (h))
if t>22:
    red.value(0)
    blue.value(1)
    green.value(1)
    sleep(0.1)
if t<20:
  red.value(1)
  blue.value(1)
  green.value(0)
  sleep(0.1)
if 20<t<22:
  red.value(1)
  blue.value(0)
  green.value(1)
```

sleep(0.1)

Simulation Output



Inference:

Result:

Arduino Code:

```
#include <SoftwareSerial.h>
SoftwareSerial bluetoothSerial(2, 3); // RX, TX pins on Arduino (connect to
TX, RX pins on HC-05)
void setup() {
  Serial.begin(9600);       // Serial monitor for debugging
  bluetoothSerial.begin(9600);  // Bluetooth module baud rate
}
void loop() {
  if (bluetoothSerial.available()) {
    char data = bluetoothSerial.read();
    Serial.print("Received data from Raspberry Pi: ");
    Serial.println(data);
    // You can add your logic here based on the received data
    // Example: Sending a response back to Raspberry Pi
```

```
            bluetoothSerial.print("Hello from Arduino!");
          }
        }
```

Python Code:

```
import serial
import time
# Replace '/dev/rfcomm0' with the address of your Bluetooth serial port
bluetooth_port = '/dev/rfcomm0'
# Open the serial port
ser = serial.Serial(bluetooth_port, 9600, timeout=1)
try:
    while True:
        # Send data to Arduino
        data_to_send = input("Enter data to send to Arduino: ")
        ser.write(data_to_send.encode())
        # Read data from Arduino
        received_data = ser.readline().decode().strip()
        print(f"Received data from Arduino: {received_data}")
        time.sleep(1)
except KeyboardInterrupt:
    print("Exiting program")
finally:
    ser.close()
```

Ex.No : 09	Communicate between Arduino and Raspberry PI using any wireless medium
Date :	

Aim:

To establish a communication between Arduino and Raspberry PI using Bluetooth.

Components Required:

For Arduino:

- Arduino board (e.g., Arduino Uno)
- HC-05 or HC-06 Bluetooth module
- Jumper wires
- Power source for Arduino (USB cable or battery)

For Raspberry Pi:

- Raspberry Pi board (any model with Bluetooth support)

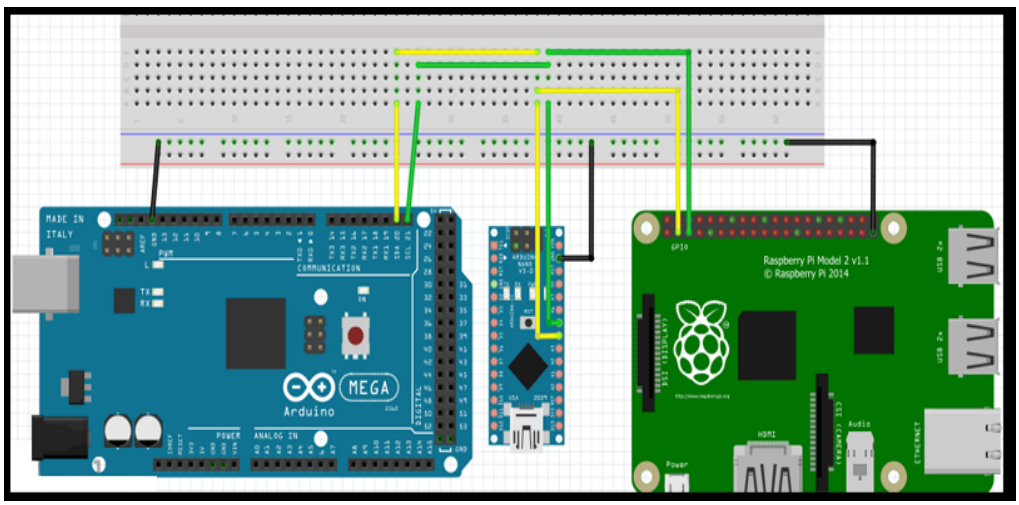
- HC-05 or HC-06 Bluetooth module
- Jumper wires
- Power source for Raspberry Pi (USB cable and power supply)

Procedure:

1. Connect the Bluetooth Modules:

- For Arduino (HC-05 or HC-06):
- Connect the VCC pin to 5V on Arduino.
- Connect the GND pin to GND on Arduino.
- Connect the TX pin to an RX pin on Arduino (e.g., pin 2 using SoftwareSerial).
- Connect the RX pin to a TX pin on Arduino (e.g., pin 3 using SoftwareSerial).
- For Raspberry Pi (HC-05 or HC-06):
- Connect the VCC pin to 3.3V on Raspberry Pi.
- Connect the GND pin to GND on Raspberry Pi.
- Connect the TX pin to the RX pin on Raspberry Pi (e.g., GPIO14).
- Connect the RX pin to the TX pin on Raspberry Pi (e.g., GPIO15).

Connection Diagram:



2. Power Up the Boards:

- Power up both the Arduino and Raspberry Pi.

3. Pair the Bluetooth Modules:

- Make sure that the Bluetooth modules are in discoverable/pairing mode.
- On the Raspberry Pi, use Bluetooth tools to scan and pair with the Arduino's Bluetooth module. You can use the bluetoothctl tool for this.

4. Upload Arduino Code:

- Upload the Arduino code to your Arduino board using the Arduino IDE.

5. Run Raspberry Pi Code:

- Run the Python code on the Raspberry Pi. This code will establish a serial connection with the Arduino over Bluetooth.

6. Test Communication:

- Enter data on the Raspberry Pi, and the Arduino should receive and process it. You can also send data from the Arduino to the Raspberry Pi.

Inference:

Result:

Program:

```
#include <ThingSpeak.h>
#include <ESP8266WiFi.h>
#define TRIG_PIN D1 // Change to the appropriate GPIO pin
#define ECHO_PIN D2 // Change to the appropriate GPIO pin
#define WIFI_SSID " " // WiFi Name
#define WIFI_PASS " " // Password for WiFi
#define THINGSPEAK_CHANNEL_ID 123456 // Replace with your ThingSpeak channel ID
#define THINGSPEAK_API_KEY "YOUR_API_KEY" // Replace with your ThingSpeak API Key
WiFiClient client;
void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
```

```
  ThingSpeak.begin(client);
  connectToWiFi();
}
void loop() {
  long duration, distance_cm;
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance_cm = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.print(distance_cm);
  Serial.println(" cm");
  ThingSpeak.setField(1, distance_cm);
                                    ThingSpeak.writeFields(THINGSPEAK_CHANNEL_ID,
THINGSPEAK_API_KEY);
```

Ex.No : 10	Write a program to measure distances using an ultrasonic sensor and send real-time distance data to ThingSpeak
Date :	

Aim:

To write a program to measure distances using an ultrasonic sensor and send real-time distance data to ThingSpeak

Hardware Required:

- Computer
- ESP8266 (NodeMCU)
- USB Cable
- Ultrasonic Sensor (HC-SR04)
- Resistor (around 1kΩ)
- Power Supply
- Jumper Wires
- Breadboard

Software Required:

- Arduino IDE
- ThingSpeak Account

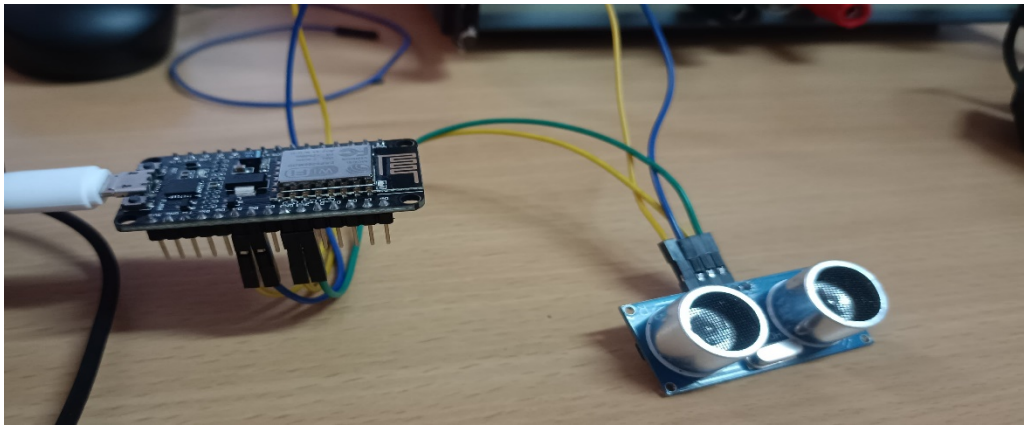
Procedure:

1. Arduino IDE:

- Open Arduino IDE.
- Click on File.
- Click on Preferences and paste the Additional Boards Manager URLs.
- Click on Tools > Board > Boards Manager > Search ESP8266 and click Install.
- Go to Boards, click on ESP8266, and select Nodemcu 1.0 (ESP-12 E module).
- Install the ThingSpeak library: Sketch > Include Library > Manage Libraries > Search "ThingSpeak" and click Install.
- Write a program as per the circuit pin connections.
- Click on Save.
- Click on Verify.
- Upload the code into ESP8266 using a USB cable.
- Open the serial monitor and check whether the Wi-Fi is connected

```
  delay(15000); // Send data every 15 seconds (ThingSpeak limit)
}
void connectToWiFi() {
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("Connected to WiFi");
}
```

Connection:



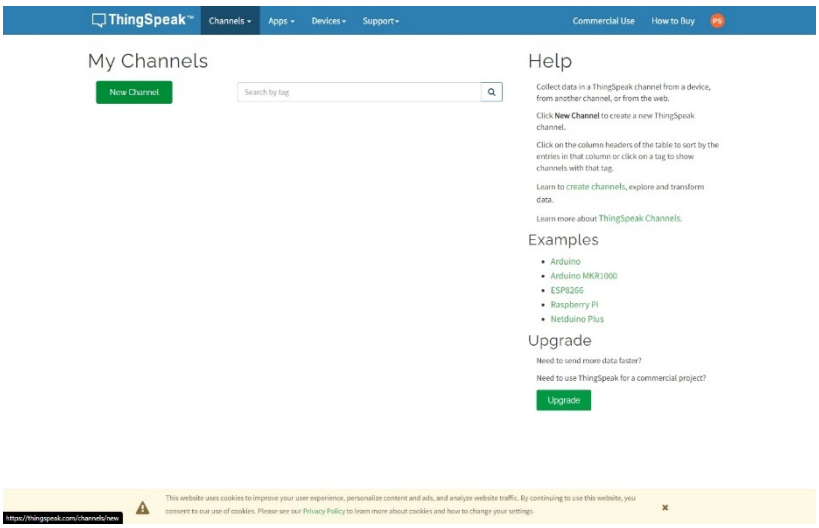
2.ThingSpeak:

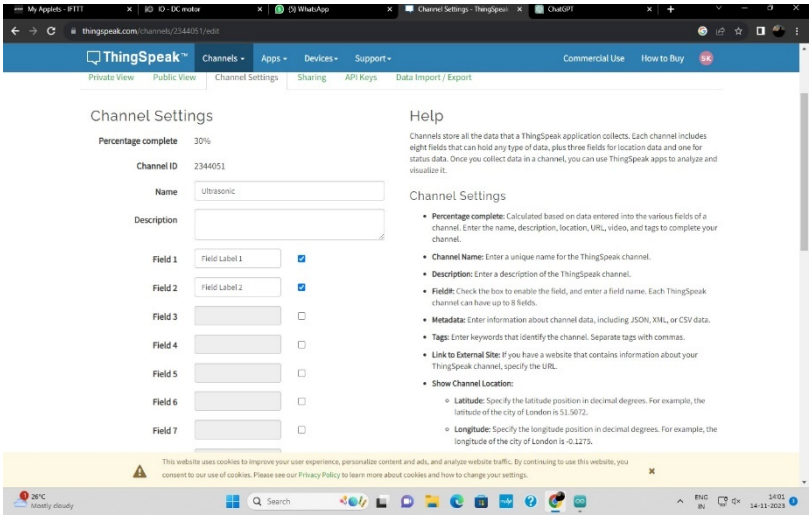
- Create a ThingSpeak account and sign in.
- Create a new channel with two fields: one for distance and one for timestamp.

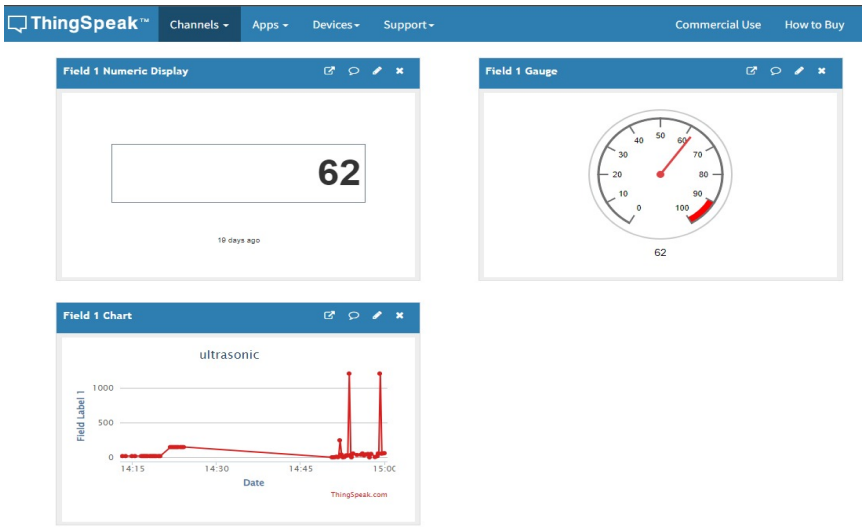
3.Circuit Wiring:

- Connect the trig pin of the ultrasonic sensor to a GPIO pin on the ESP8266 (e.g., D1).
- Connect the echo pin of the ultrasonic sensor to a GPIO pin on the ESP8266 (e.g., D2).
- Connect the VCC and GND of the ultrasonic sensor to the 5V and GND of the ESP8266, respectively.
- Connect a resistor (around 1kΩ) between the echo pin and GND to create a voltage divider.

Simulation Output:




ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Channel Settings
Percentage complete 30%
Channel ID 2344051
Name Ultrasonic
Description
Field 1 Field Label 1
Field 2 Field Label 2
Field 3
Field 4
Field 5
Field 6
Field 7
Help
Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.
Channel Settings
Percentage complete: Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
Channel Name: Enter a unique name for the ThingSpeak channel.
Description: Enter a description of the ThingSpeak channel.
FieldN: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
Metadata: Enter information about channel data, including JSON, XML, or CSV data.
Tags: Enter keywords that identify the channel. Separate tags with commas.
Link to External Site: If you have a website that contains information about your ThingSpeak channel, specify the URL.
Show Channel Location:
Latitude: Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
Longitude: Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.


ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Field 1 Numeric Display
62
19 days ago
Field 1 Gauge
62
Field 1 Chart
ultrasonic
Field Label 1
1000
500
0
14:15
14:30
14:45
15:00
Date
ThingSpeak.com

Inference:

Result:

CONTENT BEYOND SYLLABUS

PROGRAM:

```
#include <ESP8266WiFi.h>
#include <SinricPro.h>
#include <SinricProSwitch.h>
#define WIFI_SSID       "your-ssid"
#define WIFI_PASS       "your-password"
#define APP_KEY         "your-app-key"
#define APP_SECRET      "your-app-secret"
#define DEVICE_ID       "your-device-id"
#define LED_PIN         D1  // Assuming the LED is connected to pin D1 on ESP8266
bool onPowerState(const String& deviceId, bool &state) {
  // Control the LED based on the 'state' parameter
  digitalWrite(LED_PIN, state ? HIGH : LOW);
  return true;
```

```
}
void setup() {
  Serial.begin(115200);
  // Initialize WiFi
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi");
  // Initialize SinricPro
  SinricPro.begin(APP_KEY, APP_SECRET);
  // Add a switch to the device
  SinricProSwitch& mySwitch = SinricPro[DEVICE_ID];
  mySwitch.onPowerState(onPowerState);
  // Set up LED pin
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW); // Turn off the LED initially
}
void loop() {
  SinricPro.handle();
}
```

Ex.No : 11	Program to control LEDs using Alexa Echodot
Date :	

Aim:

To write a program to control LEDs using Alexa Echodot

Hardware Required:

- Computer
- ESP8266 (NodeMCU)
- USB Cable
- LED

- Power Supply
- Jumper Wires
- Breadboard

Software Required:

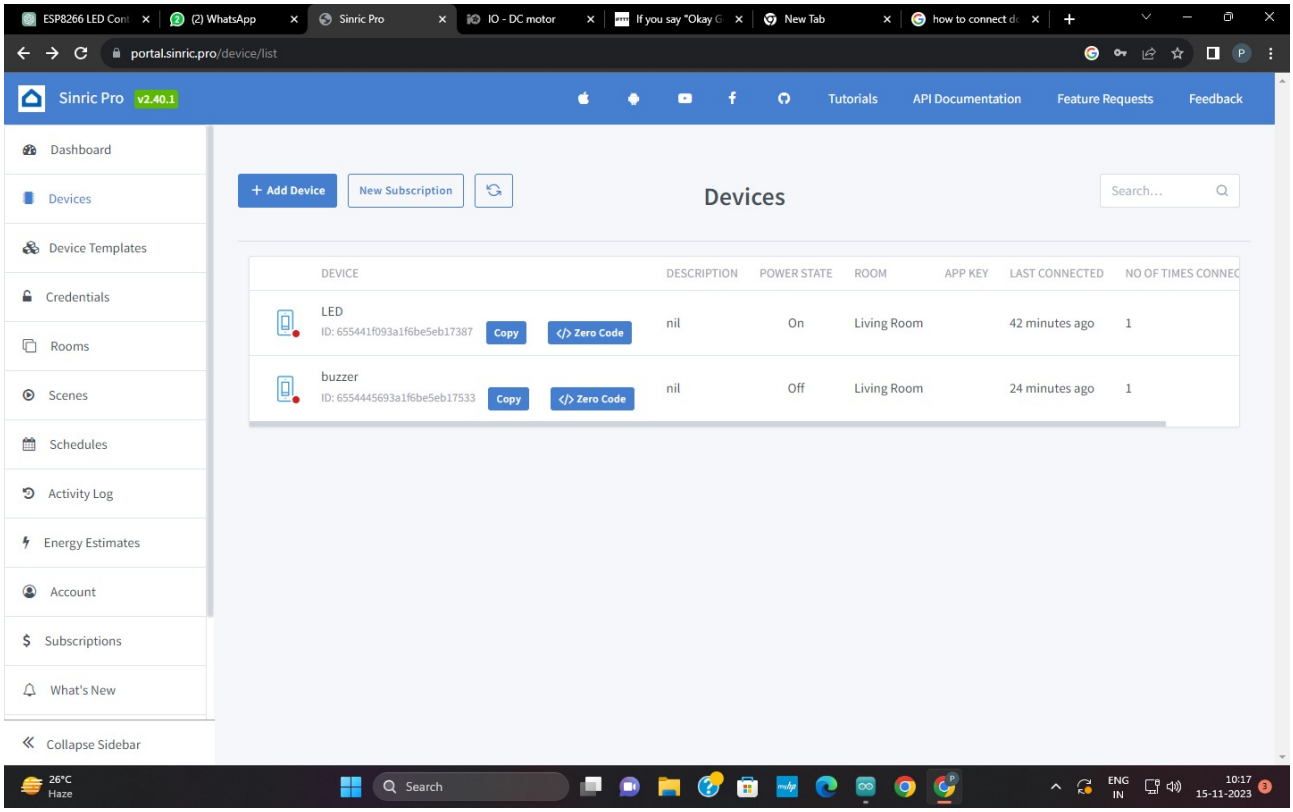
- Sinripro
- Arduino ide
- Amazon alexa

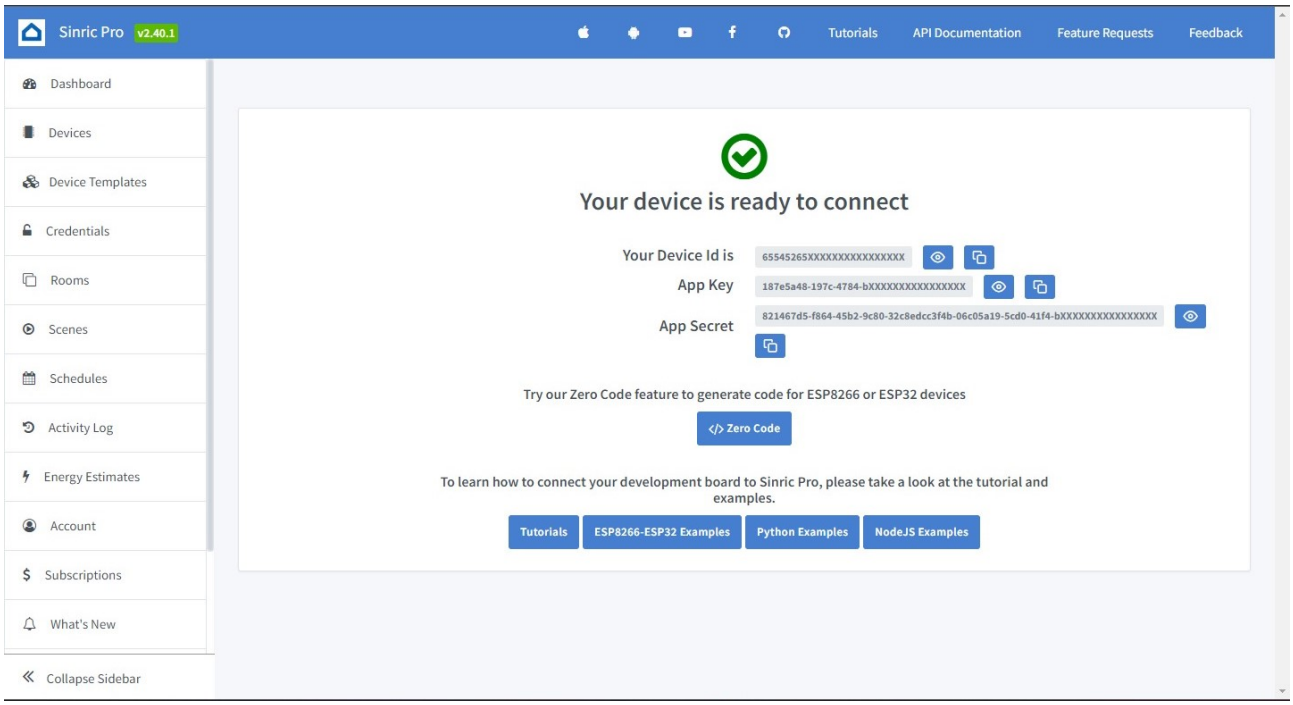
Procedure:

Arduino IDE :

1. Click on Arduino IDE
2. Click on file
3. Click on Preferences and paste on Additional boards manager URLs
4. Click on Tools > Click on Board >Click on Boards Manager >Search ESP8266 and Click Install
5. And Then Go to boards click on the esp8266 and click on the Nodemcu 1.0(ESP –12 E module)
6. Write a Program as per circuit Pin connections
7. Click on Save
8. Click on Verify
9. Click on Upload the code into ESP8266 by using USB cable.
10. Open serial monitor and check whether the Wi-Fi is connected

Output:



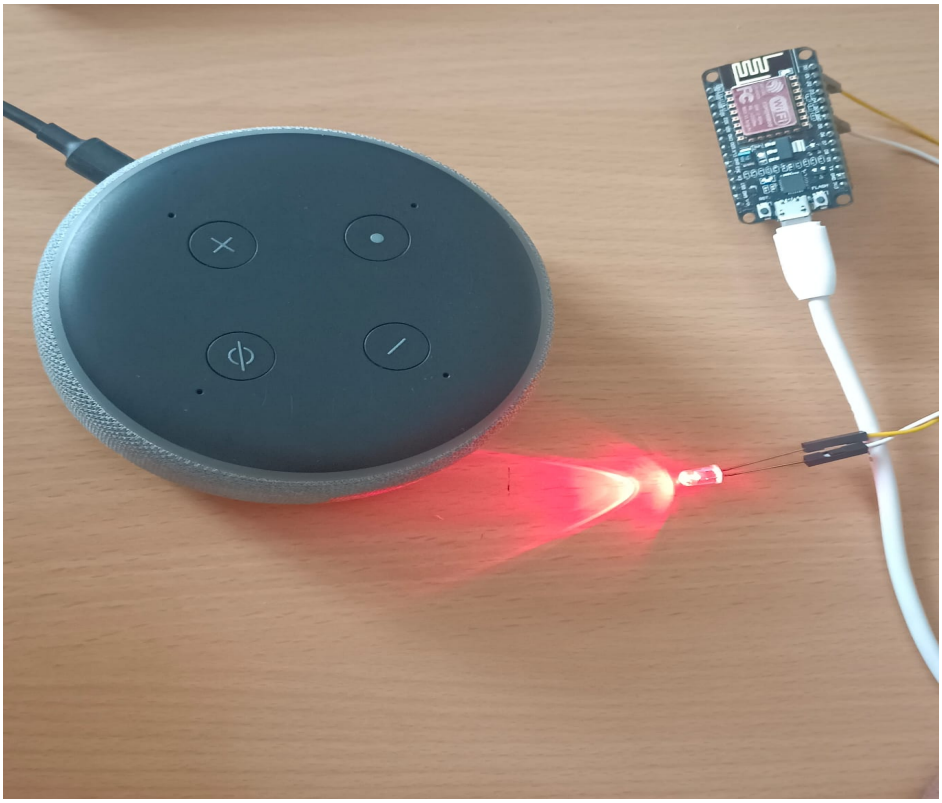


Sign Up on Sinric Pro:

- Go to [Sinric Pro](https://portal.sinric.pro/) and sign up for an account.

- Create a New Smart Home Device:
    - After logging in, click on "Add Device."
    - Choose the device type (e.g., Switch) and provide a name for your device.
    - Click on "Save."
- Note Down Device ID and Device Key:
    - After saving the device, note down the Device ID and Device Key. You will use these in your Arduino sketch.
- Add Sinric Pro Library to Arduino IDE:
    - Open the Arduino IDE.
    - Go to "Sketch" -> "Include Library" -> "Manage Libraries..."
    - Search for "SinricPro" and install the library.
- Write the Arduino Sketch:
    - Use the Arduino code provided in the previous response, replacing placeholder values with your actual Sinric Pro credentials and WiFi details.
- Upload Code to ESP8266:
    - Connect your ESP8266 to your computer.
    - Select the correct board and port in the Arduino IDE.
    - Upload the code to your ESP8266.
- Discover Devices on Alexa:
    - Open the Alexa app on your mobile device.
    - Go to "Devices" in the bottom right corner.
    - Tap on the "+" icon to add a new device.
    - Select "Add Device" and choose "Other."
    - Alexa will discover your device; make sure to name it appropriately.
- Control the Device with Alexa:
    - Now, you should be able to control the LED using voice commands with Alexa. For example:
    - "Alexa, turn on [Device Name]."
    - "Alexa, turn off [Device Name]."

Connection:



- Troubleshooting:
    - If Alexa doesn't recognize your device, try saying, "Alexa, discover devices" to refresh the device list.

Inference:

Result:

Program:

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#define Buzzer D1
#define WLAN_SSID       ""
#define WLAN_PASS       ""
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    ""
#define AIO_KEY         ""
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);
Adafruit_MQTT_Subscribe Light1 = Adafruit_MQTT_Subscribe(&mqtt,
AIO_USERNAME"/feeds/BUZZER 01");
void MQTT_connect();
void setup() {
 Serial.begin(115200);
 pinMode(Buzzer, OUTPUT);
 Serial.println(); Serial.println();
 Serial.print("Connecting to ");
 Serial.println(WLAN_SSID);
 WiFi.begin(WLAN_SSID, WLAN_PASS);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.println();
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
 mqtt.subscribe(&Light1);
}
void loop() {
 MQTT_connect();
 Adafruit_MQTT_Subscribe *subscription;
 while ((subscription = mqtt.readSubscription(20000))) {
  if (subscription == &Light1) {
   Serial.print(F("Got: "));
   Serial.println((char *)Light1.lastread);
   int Light1_State = atoi((char *)Light1.lastread);
   digitalWrite(Buzzer, Light1_State);
  }
```

Ex.No : 12	Program to control a buzzer using Google Assistant
Date :	

Aim:

To write a program to control a buzzer using Google Assistant.

Hardware Required:

- Computer
- ESP8266 (NodeMCU)
- USB Cable
- Buzzer
- Power Supply
- Jumper Wires
- Breadboard

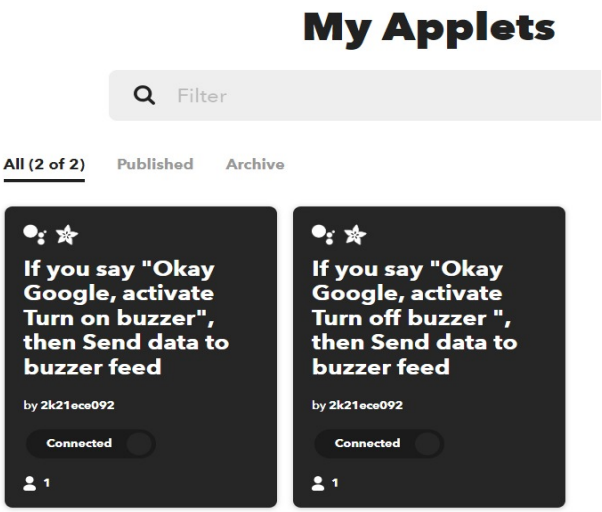
Software Required:

- Arduino IDE
- Adafruit IO
- IFTTT
- Google Home

Software Procedure:

1. **Adafruit:**
    1. Open Adafruit(adafruit.com) and Sign in
    2. Click on dashboards and Create new dashboards and give name for your experiment and click on create button
    3. Then click on your experiment and click on setting and click on Create New Block and create a block you wants,Then enter new feed name
    4. After enter feed name,Move the cursor to right side to visual Create button and click on create
    5. After create feed and click on your feed name and click on Next step
    6. And change the ON to 1 and OFF to 0,Then Click on Create block
    7. Click on new device
    8. Then now copy the API key and the User name and paste in the Program
2. **IFTTT:**
    1. Open IFTTT(ifttt.com) and Sigin In

```
  }
 }
 void MQTT_connect() {
  int8_t ret;
  if (mqtt.connected()) {
   return;
  }
  Serial.print("Connecting to MQTT... ");
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) {
   Serial.println(mqtt.connectErrorString(ret));
   Serial.println("Retrying MQTT connection in 5 seconds...");
   mqtt.disconnect();
   delay(5000);
   retries--;
   if (retries == 0) {
    while (1);
   }
  }
  Serial.println("MQTT Connected!");
 }
```

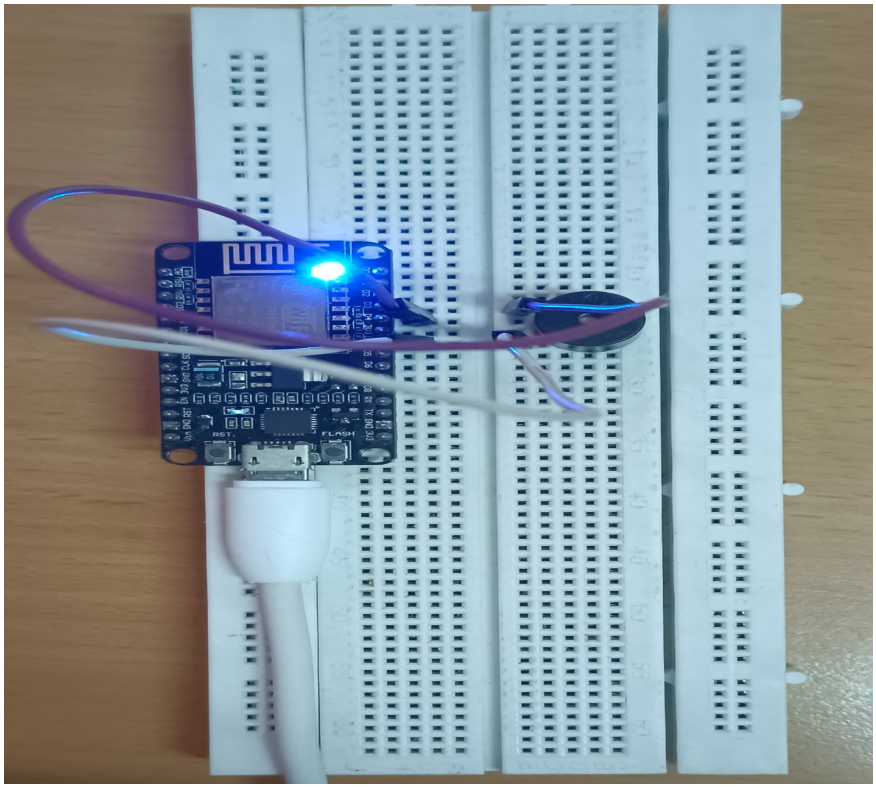


2. Click on My Applets and Scroll down to see Create your Own and click on it
3. And Click on Add And choose a service and Search Google Assistant and click on it
4. Create a activate scene and create a scene name (Eg:Turn ON DC Motor) and click on create trigger
5. Then,click on Add And choose a service and Search for Adafurit and click on it
6. Click on send data to Adafruit IO and click on Connect and Link the Adafruit to IFTTT
7. After Enter the feed name you create in Adafurit and Data to save (which means 1-ON and 0-OFF) And Click on Create Action
   Then click on Continue and click on Finis

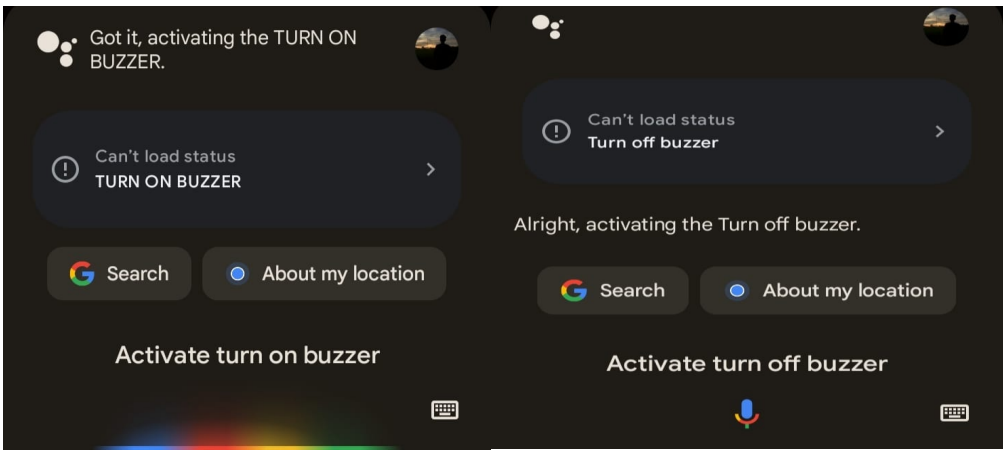
Pin Configuration:

- Connect the positive terminal of the buzzer to a GPIO pin on the ESP8266 (e.g., GPIO2).
- Connect the negative terminal of the buzzer to the ground (GND) of the ESP8266.

Connection:



SIMULATION OUTPUT:



Inference:

Result: