PROGRAM

MQTT Publisher :

```
import paho.mqtt.client as mqtt
broker = "broker.hivemq.com"
topic = "test/topic"
client = mqtt.Client()
client.connect(broker, 1883, 60)
client.publish(topic, "Hello from MQTT Publisher!")
client.disconnect()
print("Message Published")
```

MQTT Subscriber :

```
import paho.mqtt.client as mqtt
broker = "broker.hivemq.com"
topic = "test/topic"
def on_message(client, userdata, message):
    print(f"Received message: {str(message.payload.decode('utf-8'))}")
client = mqtt.Client()
client.connect(broker)
client.subscribe(topic)
client.on_message = on_message
client.loop_forever()
```

Bluetooth Communication (Linux Terminal Commands)

bash

```
# On device A - scan and pair
bluetoothctl
scan on
pair <MAC_address_of_device_B>
trust <MAC_address_of_device_B>
```

```
connect <MAC_address_of_device_B>

# On device A - connect serial
sudo rfcomm connect /dev/rfcomm0 <MAC_address_of_device_B> 1

# Open terminal on /dev/rfcomm0 for communication
screen /dev/rfcomm0 9600
```

OUTPUT

Message Published

Received message: Hello from MQTT Publisher!

PROGRAM

```
#include <DHT.h>
#define DHTPIN 2 // Pin connected to DHT sensor
#define DHTTYPE DHT11  // DHT 11 sensor
#define LDR_PIN A0   // Light sensor connected to analog pin A0
#define IR_PIN 3     // Infrared sensor connected to digital pin 3
#define LED_PIN 13  // Built-in LED pin
#define BUZZER_PIN 8      // Buzzer connected to pin 8

 DHT dht(DHTPIN, DHTTYPE);
 void setup() {
   Serial.begin(9600);
   dht.begin();
   pinMode(LED_PIN, OUTPUT);
   pinMode(BUZZER_PIN, OUTPUT);
   pinMode(IR_PIN, INPUT);
 }

 void loop() {
   // Read temperature
   float temp = dht.readTemperature();

   // Read light sensor
   int lightValue = analogRead(LDR_PIN);

   // Read infrared sensor
   int irValue = digitalRead(IR_PIN);

   Serial.print("Temperature: ");
   Serial.print(temp);
```

```
Serial.print(" °C, Light: ");
Serial.print(lightValue);
Serial.print(", IR Sensor: ");
Serial.println(irValue);

// Control actuators based on sensor values
if(lightValue < 300) {
  digitalWrite(LED_PIN, HIGH);   // Turn ON LED if dark
} else {
  digitalWrite(LED_PIN, LOW);
}

if(irValue == HIGH) {
  digitalWrite(BUZZER_PIN, HIGH); // Turn ON buzzer if IR detected
} else {
  digitalWrite(BUZZER_PIN, LOW);
}

delay(2000); // Wait for 2 seconds
  }
```

OUTPUT

Temperature: 28.5 °C, Light: 250, IR Sensor: 0

Temperature: 28.6 °C, Light: 310, IR Sensor: 1

Temperature: 28.8 °C, Light: 280, IR Sensor: 0

Temperature: 29.0 °C, Light: 260, IR Sensor: 1

Temperature: 29.2 °C, Light: 340, IR Sensor: 0

PROGRAM

```
import RPi.GPIO as GPIO
import time
# Use BCM pin numbering
GPIO.setmode(GPIO.BCM)
LED_PIN = 17
GPIO.setup(LED_PIN, GPIO.OUT)

try:
    while True:
     GPIO.output(LED_PIN, GPIO.HIGH)  # LED ON
       time.sleep(1)   # Wait 1 second
            GPIO.output(LED_PIN, GPIO.LOW)   # LED OFF
                    time.sleep(1)    # Wait 1 second
            except KeyboardInterrupt:
                print("Program stopped")

            finally:
                GPIO.cleanup()  # Reset GPIO settings
```

OUTPUT

LED turns ON for 1 second

LED turns OFF for 1 second

This repeats continuously in a blinking pattern.

When press (Ctrl + C) to stop, the LED turns off completely.

Program stopped

PROGRAM

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

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

void setup() {
Serial.begin(115200);
WiFi.begin(ssid, password);
Serial.print("Connecting to WiFi");

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
Serial.println("\nWiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
    }
void loop() {
if (WiFi.status() == WL_CONNECTED) {
          HTTPClient http;

    // Example HTTP GET request to httpbin.org/get
    http.begin("http://httpbin.org/get?data=HelloArduino");
    int httpCode = http.GET();

    if (httpCode > 0) {
       String payload = http.getString();
```

```
      Serial.println("HTTP Response code: " + String(httpCode));
      Serial.println("Response: " + payload);
    } else {
      Serial.println("Error on HTTP request");
    }
    http.end();
  } else {
    Serial.println("WiFi Disconnected");
  }
  delay(10000); // Wait 10 seconds before next request
}
```

OUTPUT

Connecting to WiFi....

WiFi connected

IP address:

192.168.1.105

HTTP Response code: 200

Response: {


```
  "args": {
    "data": "HelloArduino"
  },
  "headers": {
    "Accept": "*/*",
    "Host": "httpbin.org",
    "User-Agent": "ESP8266HTTPClient",
    "X-Amzn-Trace-Id": "Root=1-671f6bcd-xxxxxxxxxxxxxxxxxxxxxxxx"
  },
  "origin": "122.176.xxx.xxx",
  "url": "http://httpbin.org/get?data=HelloArduino"
}
```

PROGRAM

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

#define DHTPIN 2   // GPIO pin where DHT sensor is connected
#define DHTTYPE DHT11

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

const char* server = "http://api.thingspeak.com/update";
const String apiKey = "YOUR_THINGSPEAK_API_KEY";

DHT dht(DHTPIN, DHTTYPE);

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

   WiFi.begin(ssid, password);
   Serial.print("Connecting to WiFi");

   while (WiFi.status() != WL_CONNECTED) {
     delay(500);
     Serial.print(".");
   }

  Serial.println("\nConnected to WiFi");
}
```

```
void loop() {
  float temperature = dht.readTemperature();

  if (isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temperature);
Serial.println(" °C");  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    String postData = String(server) + "?api_key=" + apiKey + "&field1=" +
String(temperature);
    http.begin(postData);
    int httpCode = http.GET();

    if (httpCode > 0) {
      Serial.println("Data posted successfully");
    } else {
      Serial.println("Error in posting data");
    }
    http.end();
  } else {
    Serial.println("WiFi Disconnected");
  }

  delay(20000); // Wait for 20 seconds before next reading
}
```

OUTPUT

Connecting to WiFi...

.....

Connected to WiFi

Temperature: 28.4 °C

Data posted successfully

Temperature: 28.6 °C

Data posted successfully

Temperature: 28.2 °C

Data posted successfully

Temperature: 29.0 °C

Data posted successfully

PROGRAM

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

// Wi-Fi Credentials

#define WIFI_SSID       "YOUR_SSID"
#define WIFI_PASSWORD   "YOUR_PASSWORD"

// Firebase Credentials
#define FIREBASE_HOST   "your-project-id.firebaseio.com"
#define FIREBASE_AUTH   "your-database-secret-or-auth-token"

// Pin Configuration
#define SOIL_MOISTURE_PIN A0

// Create Firebase Data object
FirebaseData firebaseData;

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

  // Connect to Wi-Fi
  Serial.print("Connecting to WiFi");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi");
```

```
  // Initialize Firebase
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
  Serial.println("Firebase connection initialized");
}

void loop() {
  // Read soil moisture value
  int soilMoistureValue = analogRead(SOIL_MOISTURE_PIN);
  Serial.print("Soil Moisture Value: ");
  Serial.println(soilMoistureValue);

  // Send data to Firebase
  if (Firebase.setInt(firebaseData, "/soilMoisture", soilMoistureValue)) {
    Serial.println("Firebase updated successfully");
  } else {
    Serial.print("Firebase update failed: ");
    Serial.println(firebaseData.errorReason());
  }

  // Wait 20 seconds before next update
  delay(20000);
}
```

OUTPUT

Connecting to WiFi...

.....

Connected to WiFi

Firebase connection initialized

Soil Moisture Value: 523

Firebase updated successfully

Soil Moisture Value: 518

Firebase updated successfully

Soil Moisture Value: 510

Firebase updated successfully

Soil Moisture Value: 495

Firebase updated successfully

PROGRAM

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>

// Ultrasonic Sensor Pins
#define TRIG_PIN 5
#define ECHO_PIN 4

// Wi-Fi Credentials
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

// Google Cloud IoT Core Credentials
const char* project_id   = "your-project-id";
const char* location     = "your-region";       // e.g., "us-central1"
const char* registry_id  = "your-registry-id";
const char* device_id    = "your-device-id";

const char* mqtt_server  = "mqtt.googleapis.com";
const int   mqtt_port    = 8883;

// Wi-Fi and MQTT Clients
WiFiClientSecure net;
PubSubClient client(net);

// Function Prototypes
long readUltrasonicDistance();
String createJwt();
void connectToCloudIoT();
```

```
// Setup Function
void setup() {
  Serial.begin(115200);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Connect to Wi-Fi
  Serial.print("Connecting to WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected successfully!");

  // Configure secure connection (add your root CA certificate)
  net.setCACert(your_root_ca_cert);  // <-- Replace with actual root CA certificate variable

  // Set MQTT server and port
  client.setServer(mqtt_server, mqtt_port);

  // Connect to Google Cloud IoT Core
  connectToCloudIoT();
}

// Connect to Google Cloud IoT Core
void connectToCloudIoT() {
  while (!client.connected()) {
    String jwt = createJwt();
```

```
    Serial.println("Connecting to MQTT...");

    // Username is unused; JWT is used as password
    if (client.connect(device_id, "unused", jwt.c_str())) {
      Serial.println("Connected to MQTT successfully!");
    } else {
      Serial.print("Failed MQTT connection, rc=");
      Serial.println(client.state());
      delay(5000);
    }
  }
}

// Main Loop
void loop() {
  if (!client.connected()) {
    connectToCloudIoT();
  }
  client.loop();

  // Measure distance using ultrasonic sensor
  long distance = readUltrasonicDistance();
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Publish distance data to Google Cloud IoT
  String topic = "/devices/" + String(device_id) + "/events";
  String payload = "{\"distance\":" + String(distance) + "}";
```

```
  if (client.publish(topic.c_str(), payload.c_str())) {
    Serial.println("Distance data published successfully!");
  } else {
    Serial.println("Failed to publish distance data!");
  }

  delay(20000); // Publish every 20 seconds
}

// Ultrasonic Distance Function
long readUltrasonicDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);
  long distanceCm = duration * 0.034 / 2;
  return distanceCm;
}

// JWT Creation Function (Placeholder)
String createJwt() {
  // Implement JWT creation using your private key and project info.
  // You can use an external JWT generation script or a library.
  return "YOUR_JWT_TOKEN";
}
```

OUTPUT

Connecting to WiFi...

.....

WiFi connected successfully!

Connecting to MQTT...

Connected to MQTT successfully!

Distance: 24 cm

Distance data published successfully!

Distance: 25 cm

Distance data published successfully!

Distance: 27 cm

Distance data published successfully!

Distance: 26 cm

Distance data published successfully!

PROGRAM

```
const int sensorPin = A0;  // Analog pin connected to LDR

void setup() {
  Serial.begin(9600);
  Serial.println("Light Sensor Reading:");
}

void loop() {
  int sensorValue = analogRead(sensorPin);
  Serial.print("LDR Value: ");
  Serial.println(sensorValue);
  delay(1000);  // Wait for 1 second
}
```

OUTPUT

Light Sensor Reading:

LDR Value: 820

LDR Value: 790

LDR Value: 760

LDR Value: 400

LDR Value: 250

LDR Value: 150

LDR Value: 820

LDR Value: 900

PROGRAM

// Install Django and Firebase Admin SDK via pip:

```
pip install django firebase-admin
```

//Start a new Django project:

```
django-admin startproject iotproject
cd iotproject
python manage.py startapp sensorapp
```

// Configure Firebase in Django

```
# sensorapp/firebase.py
import firebase_admin
from firebase_admin import credentials, firestore

# Load Firebase credentials from the JSON file
cred = credentials.Certificate('path/to/serviceAccountKey.json')

firebase_admin.initialize_app(cred)

# Create Firestore database client
db = firestore.client()
```

// Create Views to Fetch and Display Data

```
# sensorapp/views.py
from django.shortcuts import render
from .firebase import db

def sensor_data_view(request):
    # Fetch all documents from Firestore collection 'sensor_data'
    docs = db.collection('sensor_data').stream()
```

```
    data = []
    for doc in docs:
        data.append(doc.to_dict())

    return render(request, 'sensorapp/data.html', {'sensor_data': data})
```

//Define URL Routing

```
#iotproject/urls.py
from django.urls import path
from sensorapp.views import sensor_data_view

# Define URL routes
urlpatterns = [
    path('sensor-data/', sensor_data_view, name='sensor_data'),
]
```

// **Create HTML Template to Display Sensor Data**

```
#sensorapp/data.html
<!DOCTYPE html>
<html>
<head>
    <title>Sensor Data</title>
</head>
<body>
    <h1>Sensor Data from Firebase</h1>
    <ul>
        {% for item in sensor_data %}
            <li>{{ item.sensor_type }} : {{ item.value }} at {{ item.timestamp }}</li>
        {% empty %}
            <li>No sensor data found.</li>
```

```
        {% endfor %}
    </ul>
</body>
</html>
```

// Run Django Server

python manage.py runserver

Open browser and navigate to http://localhost:8000/sensor-data/ to view the sensor data fetched from Firebase.

OUTPUT

Sensor Data from Firebase

• Temperature : 28.4 °C at 2025-10-28 18:25:42

• Humidity : 61 % at 2025-10-28 18:25:42

• Light Intensity : 340 Lux at 2025-10-28 18:25:42

• CO_2 Level : 420 ppm at 2025-10-28 18:25:42

• Motion : Detected at 2025-10-28 18:25:42

PROGRAM

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

#define DHTPIN 4          // DHT11 sensor connected to pin 4
#define DHTTYPE DHT11
#define CURRENT_SENSOR A0 // ACS712 current sensor

DHT dht(DHTPIN, DHTTYPE);

const char* ssid = "Your_WiFi_Name";
const char* password = "Your_WiFi_Password";
const char* mqtt_server = "test.mosquitto.org";

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(9600);
  dht.begin();

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

```
  client.setServer(mqtt_server, 1883);
}

void loop() {
  if (!client.connected()) {
    client.connect("SmartHomeDevice");
  }

  float temp = dht.readTemperature();
  int sensorValue = analogRead(CURRENT_SENSOR);
  float voltage = (sensorValue / 1023.0) * 5.0;
  float current = voltage / 0.185;   // Convert to current
  float power = 230.0 * current;     // Power in watts

  String payload = String("{\"temperature\":") + temp + ",\"power\":" + power + "}";
  client.publish("home/sensors", payload.c_str());

  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.println(" °C");

  Serial.print("Power Consumption: ");
  Serial.print(power);
  Serial.println(" W");

  delay(5000); // Wait 5 seconds
}
```

OUTPUT

WiFi Connected!

Temperature: 28.6 °C

Power Consumption: 84.72 W

Temperature: 29.0 °C

Power Consumption: 89.33 W

Temperature: 30.2 °C

Power Consumption: 92.47 W