

<b>Ex.No:01</b>	<b>To Study Various IoT Protocols-</b>
<b>Date:</b>	<b>6LowPAN,IPv4/IPv6,WiFi,Bluetooth,MQTT</b>

**Aim:**

To create various Iot protocol using 6LOWPAN,MQTT Arduino IDE developed in a sensor in a MQTT request

**Procedure:**

**Step 1:** start the arduino program

**Step 2:** create new project

**Step 3:**Write to the coding in Electronic MQTT RGB controller

**Step 4:**Check the MQTT Broker connectiong in URL

**Step 5:** Run to the Arduino

**Step 6:** Stop the program

## Program:

\* MQTT/Phant RGB Controller

\* by: Jim Lindblom, SparkFun Electronics

\* with lots of help from: Todd Treece (thanks Todd!)

\* Fun with MQTT, Arduino, and Phant. This example sketch uses

\* MQTT to subscribe to a Phant data stream. When values labeled

\* "red", "green", or "blue" change on the Phant stream, the

\* Arduino will be notified and update its LEDs accordingly.

```
#include <SPI.h> // Include SPI for the Ethernet library
```

```
#include <Ethernet.h>
```

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

```
// Enter a MAC address for your controller below.
```

```
// Newer Ethernet shields have a MAC address printed on a sticker on the shield
```

```
byte mac[] = { 0x42, 0x24, 0x4D, 0x3D, 0x2D, 0x00 };
```

```
// Destination server, our MQTT broker.
```

```
char server[] = "data.sparkfun.com";
```

```
EthernetClient ethClient;
```

```
// Create an mqttClient object. Give it the MQTT broker server,
```

```
// the MQTT port (default is 1883), a callback function, and
```

```
// an EthernetClient object to use:
```

```
PubSubClient mqttClient(server, 1883, mqttCallback, ethClient);
```

```
// To set up MQTT we need:
```

```
char arduinoID[] = "jimsArduino"; // An ID for the MQTT client
```

```
// ...and the topics we want to subscribe to. To subscribe to a
```

```
// Phant topic, the format is output/pub_key/field_name. Our
```

```
// stream's public key is roE6z8QZq5iyEdrMGwdG.
```

```
char redTopic[] = "output/roE6z8QZq5iyEdrMGwdG/red";
```

```
char greenTopic[] = "output/roE6z8QZq5iyEdrMGwdG/green";
```

```
char blueTopic[] = "output/roE6z8QZq5iyEdrMGwdG/blue";
```

```
// LED pins:
byte redLEDPin = 3;
byte greenLEDPin = 5;
byte blueLEDPin = 6;
void setup()
{
  // Initialize LED pins:
  pinMode(redLEDPin, OUTPUT);
  digitalWrite(redLEDPin, LOW);
  pinMode(greenLEDPin, OUTPUT);
  digitalWrite(greenLEDPin, LOW);
  pinMode(blueLEDPin, OUTPUT);
  digitalWrite(blueLEDPin, LOW);
  // Enable serial for debug messages:
  Serial.begin(9600);
  // Connect to Ethernet via DHCP:
  if (Ethernet.begin(mac) == 0)
  {
    Serial.println("Failed to connect. Looping :(.");
    while (1) ;
  }
  Serial.print("Connected! IP: ");
  Serial.println(Ethernet.localIP());
  // Set up MQTT connection:
  // For some reason, Phant only works if the user and ID fields
  // are present (it doesn't seem to care what they are).
  if (mqttClient.connect(arduinoID, "tempuser", "tempID"))
  {
    // Once MQTT is connected, subscribe to topics of interest:
```

```

    mqttClient.subscribe(redTopic);
    mqttClient.subscribe(greenTopic);
    mqttClient.subscribe(blueTopic);
    Serial.println("Subscribed! Here we go!");
}
else
{
    Serial.println("Failed to connect to MQTT. Looping :(.");
    while (1) ;
}
}

// loop just needs to call the MQTT loop, which keeps the
// connection alive, and checks for any new published data.
void loop()
{
    mqttClient.loop();
}

// mqttCallback function is entered whenever a subscribed value
// changes.
void mqttCallback(char* topic, byte* payload, unsigned int length)
{
    // Convert the topic char array to a string. And keep it within
    // the proper range of analog outputs. (0-255).
    int value =.toInt(payload, length);
    value = constrain(value, 0, 255);
    // Now check if the topic char array matches any of our red,
    // green, or blue LED topics. We'll use strstr to search a
    // string for another string:
    if (strstr(topic, redTopic))

```

```

{
    Serial.print("Setting Red LED to ");
    Serial.println(value);
    analogWrite(redLEDPin, value);
}
else if (strstr(topic, greenTopic))
{
    Serial.print("Setting Green LED to ");
    Serial.println(value);
    analogWrite(greenLEDPin, value);
}
else if (strstr(topic, blueTopic))
{
    Serial.print("Setting Blue LED to ");
    Serial.println(value);
    analogWrite(blueLEDPin, value);
}
}

// Helper function to convert a byte array to an equivalent
// integer value.
int toInt(byte* payload, int length)
{
    int i;
    char val[10];
    for(i = 0; i < length; i++)
        val[i] = payload[i];
    val[i] = '\0';
    return atoi(val);
}

```

**Output:**

Failed to connect. Looping : True

Connected! IP: <https://data.sparkfun.com/input/roE6z8QZq5iyEdrMGwdG?>

Ethernet.localIP()

Connected Succesfully

```
GET /output/roE6z8QZq5iyEdrMGwdG/latest.txt HTTP/1.1
```

```
Host: data.sparkfun.com
```

```
Connection: close
```

<b>Ex.No:02</b>	<b>Interfacing Sensor with Ardunio : Light Sensor</b>
<b>Date:</b>	

**AIM:**

To test LDR sensor with arduino UNO board and display in serial monitor of Arduino IDE.

**Apparatus requird:**

1. Ardunio UNO Board
2. LDR Sensor
3. Connecting wires

**Algorithm:**

**Step 1:** Declare the input pin

**Step 2:** Read the analog value through the ADC

**Step 3:** Convert to equitant decimal number

**Step 4:** Convert the value

**Step 5:** Display in the Serial Monitor

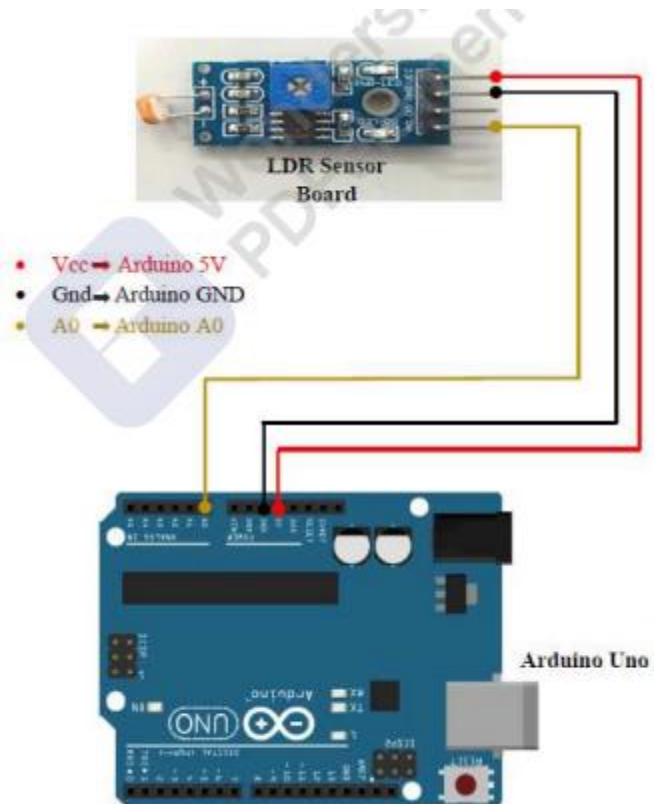
**Step 6:** Repeat the Step 1-5

**Program:**

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  unsigned int AnalogValue;
  AnalogValue = analogRead(A0);
  delay(1000);
  Serial.println(AnalogValue);
}
```





**Output:**

250 distance

<b>Ex.No:03</b>	<b>To study Raspberry Pi development board and to implement LED blinking</b>
<b>Date:</b>	

### **AIM:**

To implement a simple application using Raspberry Pi development board and to implement LED blinking

### **Algorithm:**

**Step 1:** Needed Components. For the build, you will need the following: ...

**Step 2:** Building the Circuit. Every LED has two sides - one negative and one positive. ...

**Step 3:** Setting Up the Raspberry. ...

**Step 4:** Writing the Program. ...

**Step 5:** Running the Program. ...

### **Step 1**

**need to component:**

**1 x Raspberry Pi**

**1 x USB cable**

**1 x LED**

**1 x Breadboard**

**1 x SD Card and adapter (4GB minimum)**

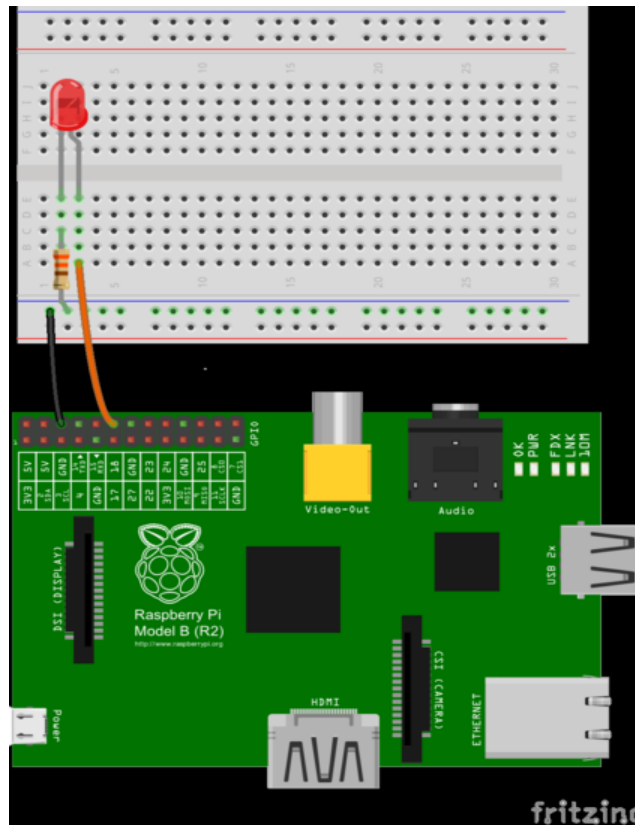
**1 x LAN cable**

**1 x 50-ohm resistor**

**2 x Jumper wires**

## Step2

build in circuit:



## Step 3

setting up:

```
pi@raspberrypi:~ $ sudo apt-get install python3
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.5.3-1).
python3 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ sudo apt-get install python
Reading package lists... Done
Building dependency tree
Reading state information... Done
python is already the newest version (2.7.13-2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

## **Step 4**

### **Program:**

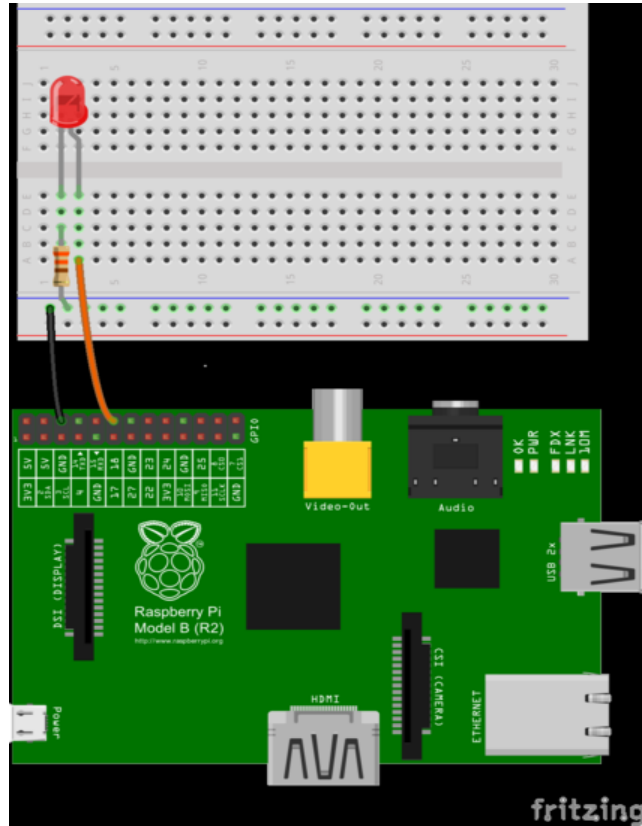
```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
print "LED on"
GPIO.output(18,GPIO.HIGH)
time.sleep(1)
print "LED off"
GPIO.output(18,GPIO.LOW)
```

## **Step 5**

### **Run:**

```
python file-name.py
```

**Output:**



<b>Ex.No:04</b>	<b>To develop an application to send and receive data with Arduino using HTTP request</b>
<b>Date:</b>	

**Aim:**

To implement a simple ESP32 Web Server with HTTP protocol responsible for communication between Client and Server

**Apparatus required:**

- |                       |    |
|-----------------------|----|
| 1. ESP32 Kit          | -1 |
| 2. Connecting Wire    | -1 |
| 3. Breadboard         | -1 |
| 4. LED                | -2 |
| 5. Resistance(220ohm) | -2 |

**Program:**

```
#include <WIFI.h>

#define gpio16LEDPin 16 /* One LED connected to GPIO16 - RX2 */
#define gpio17LEDPin 17 /* One LED connected to GPIO17 - TX2 */

const char* ssid = "ESP32-WiFi"; /* Add your router's SSID */
const char* password = "12345678"; /*Add the password */

int gpio16Value;
int gpio17Value;

WiFiServer espServer(80); /* Instance of WiFiServer with port number 80 */
/* 80 is the Port Number for HTTP Web Server */
/* A String to capture the incoming HTTP GET Request */
String request;

void setup()
{
  Serial.begin(115200); /* Begin Serial Communication with 115200 Baud Rate */
  /* Configure GPIO16 and GPIO17 Pins as OUTPUTs */
  pinMode(gpio16LEDPin, OUTPUT);
  pinMode(gpio17LEDPin, OUTPUT);
  /* Set the initial values of GPIO16 and GPIO17 as LOW*/
  /* Both the LEDs are initially OFF */
  digitalWrite(gpio16LEDPin, LOW);
  digitalWrite(gpio17LEDPin, LOW);
  Serial.print("\n");
  Serial.print("Connecting to: ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA); /* Configure ESP32 in STA Mode */

  WiFi.begin(ssid, password); /* Connect to Wi-Fi based on the above SSID and
  Password */

  while(WiFi.status() != WL_CONNECTED)
```

```
{  
  Serial.print("*");  
  delay(100);  
}  
Serial.print("\n");  
Serial.print("Connected to Wi-Fi: ");  
Serial.println(WiFi.SSID());  
delay(100);  
/* The next four lines of Code are used for assigning Static IP to ESP32 */  
/* Do this only if you know what you are doing */  
/* You have to check for free IP Addresses from your Router and */  
/* assign it to ESP32 */  
/* If you are comfortable with this step, */  
/* please un-comment the next four lines and make necessary changes */  
/* If not, leave it as it is and proceed */  
//IPAddress ip(192,168,1,6);  
//IPAddress gateway(192,168,1,1);  
//IPAddress subnet(255,255,255,0);  
//WiFi.config(ip, gateway, subnet);  
delay(2000);  
  Serial.print("\n");  
  Serial.println("Starting ESP32 Web Server...");  
  espServer.begin(); /* Start the HTTP web Server */  
  Serial.println("ESP32 Web Server Started");  
  Serial.print("\n");  
  Serial.print("The URL of ESP32 Web Server is: ");  
  Serial.print("http://");  
  Serial.println(WiFi.localIP());  
  Serial.print("\n");
```



```

Serial.println("Use the above URL in your Browser to access ESP32 Web
Server\n");
}
void loop()
{
WiFiClient client = espServer.available(); /* Check if a client is available */
if(!client)
{
return;
}
Serial.println("New Client!!!");
boolean currentLineIsBlank = true;
while (client.connected())
{
if (client.available())
{
char c = client.read();
request += c;
Serial.write(c);
/* if you've gotten to the end of the line (received a newline */
/* character) and the line is blank, the http request has ended, */
/* so you can send a reply */
if (c == '\n' && currentLineIsBlank)
{
/* Extract the URL of the request */
/* We have four URLs. If IP Address is 192.168.1.6 (for example),
* then URLs are:
* 192.168.1.6/GPIO16ON
* 192.168.1.6/GPIO16OFF

```

```
* 192.168.1.6/GPIO17ON
* 192.168.1.6/GPIO17OFF
*/
/* Based on the URL from the request, turn the LEDs ON or OFF */
if (request.indexOf("/GPIO16ON") != -1)
{
Serial.println("GPIO16 LED is ON");
digitalWrite(gpio16LEDPin, HIGH);
gpio16Value = HIGH;
}
if (request.indexOf("/GPIO16OFF") != -1)
{
Serial.println("GPIO16 LED is OFF");
digitalWrite(gpio16LEDPin, LOW);
gpio16Value = LOW; } if (request.indexOf("/GPIO17ON") != -1)
{
Serial.println("GPIO17 LED is ON");
digitalWrite(gpio17LEDPin, HIGH);
gpio17Value = HIGH;
}
if (request.indexOf("/GPIO17OFF") != -1)
{
Serial.println("GPIO17 LED is OFF");
digitalWrite(gpio17LEDPin, LOW);
gpio17Value = LOW;
}


/* HTTP Response in the form of HTML Web Page */
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
```

```
client.println("Connection: close");
client.println(); // IMPORTANT
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<head>");
client.println("<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1\">");
client.println("<link rel=\"icon\" href=\"data:;\">");
client.println("<style>");
client.println("html { font-family: Courier New; display: inline-block; margin: 0px
auto;
text-align: center;}");
client.println(".button {border: none; color: white; padding: 10px 20px; text-align:
center;}");
client.println("text-decoration: none; font-size: 25px; margin: 2px; cursor:
pointer;}");
client.println(".button1 { background-color: #FF0000;}");
client.println(".button2 { background-color: #00FF00;}");
client.println("</style>");
client.println("</head>");
client.println("<body>");
client.println("<h2>ESP32 Web Server</h2>");
if(gpio16Value == LOW)
{
client.println("<p>GPIO16 LED Status: OFF</p>");
client.print("<p><a href=\"/GPIO16ON\"><button class=\"button button1\">Click
to
turn ON</button></a></p>");
```

```
}
else
{
    client.println("<p>GPIO16 LED Status: ON</p>");
    client.print("<p><a href=\"/GPIO16OFF\"><button class=\"button  
button2\">Click to  
turn OFF</button></a></p>");
}
if(gpio17Value == LOW)
{
    client.println("<p>GPIO17 LED Status: OFF</p>");
    client.print("<p><a href=\"/GPIO17ON\"><button class=\"button button1\">Click  
to  
turn ON</button></a></p>");
}
else
{
    client.println("<p>GPIO17 LED Status: ON</p>");
    client.print("<p><a href=\"/GPIO17OFF\"><button class=\"button button2\">Click  
to  
turn OFF</button></a></p>");
}
client.println("</body>");
client.println("</html>");
break;
}
if(c == '\n')
{
    currentLineIsBlank = true;
```

```
}  
currentLineIsBlank = true;  
{  
currentLineIsBlank = false;  
}  
//client.print("\n");  
}  
}  
delay(1);  
request = "";  
//client.flush();  
client.stop();  
Serial.println("Client disconnected");  
Serial.print("\n");  
}
```

## Output:



```
COM4
Connecting to: TrailBlazer
Connected to Wi-Fi: TrailBlazer
Starting ESP32 Web Server...
ESP32 Web Server Started
The URL of ESP32 Web Server is: http://192.168.1.6
Use the above URL in your Browser to access ESP32 Web Server
```

☒ Autoscroll ☐ Show timestamp Both NL & CR 115200 baud Clear output

<b>Ex.No:05</b>	<b>Application to Measures the Room Temperature &amp; Posts the value on the Cloud platform</b>
<b>Date:</b>	

### **AIM:**

To develop an application to measures the room temperature & humidity using DHT11 sensor interface with ESP32 & Send the Sensor Readings to Google Firebase.

### **Apparatus required:**

1. ESP32
2. DHT11 Sensor
3. Connecting Wires
4. Breadboard

### **Software Required:**

1. Google Firebase
2. Arduino IDE
3. IDE Library files
  - a. Google Firebase ESP32 Library
  - b. DHT sensor library
  - c. Adafruit Unified Sensor library

### **Algorithm:**

**Step 1:** Declare the input pin

**Step 2:** Read the digital value through DHT11

**Step 3:** Convert to equitant decimal number

**Step 4:** Convert the value into Celsius using formula

**Step 5:** Display in the Serial Monitor

**Step 6:** Repeat the Step 1-5

**Program:**

```
#include <FirebaseESP32.h>
#include <WiFi.h>
#include "DHT.h"
#define FIREBASE_HOST "esp32-sensor-readings-app-default-
rtdb.firebaseio.com"
#define WIFI_SSID "Your_SSID" // Change the name of your WIFI #define
WIFI_PASSWORD "Your_Password" // Change the password of your WIFI
#define FIREBASE_Authorization_key "Replace_with_your_secret_key"
#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
FirebaseData firebaseData;
FirebaseJson json;
void setup() {
  Serial.begin(115200);
  dht.begin();
  WiFi.begin (WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting...");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  Firebase.begin(FIREBASE_HOST,FIREBASE_Authorization_key);
```



```
}  
void loop() {  
  float hum = dht.readHumidity();  
  float temp = dht.readTemperature();  
  if (isnan(hum) || isnan(temp) ){  
    Serial.println(F("Failed to read from DHT sensor!"));  
    return;  
  }  
  Serial.print("Temperature: ");  
  Serial.print(temp);  
  Serial.print("°C");  
  Serial.print(" Humidity: ");  
  Serial.print(hum);  
  Serial.print("%");  
  Serial.println();  
  Firebase.setFloat(firebaseData, "/ESP32_APP/TEMPERATURE", temp);  
  Firebase.setFloat(firebaseData, "/ESP32_APP/HUMIDITY", hum);  
  delay(200);  
}
```

### **Output:**

325°C

<b>Ex.No:06</b>	<b>Application to Measures the Moisture of Soil &amp; Posts the data over Google Firebase Cloud platform</b>
<b>Date:</b>	

### **AIM:**

To develop an application to measures the Soil Moisture level using Soil moisture sensor interface with ESP32 & Send the Sensor Readings to Google Firebase cloud platform.

### **Apparatus required:**

1. ESP32
2. Soil Moisture Sensor
3. Connecting Wires
4. Breadboard

### **Software Required:**

1. Google Firebase
2. Arduino IDE
3. IDE Library files
  - a. Google Firebase ESP32 Library

### **Algorithm:**

- Step 1:** Declare the input pin
- Step 2:** Read the Analog value through Soil moisture Sensor
- Step 3:** Convert to equitant decimal number
- Step 4:** Convert the value into Celsius using formula
- Step 5:** Display in the Serial Monitor
- Step 6:** Repeat the Step 1-5

**Program:**

```
#include <FirebaseESP32.h>
#include <WiFi.h>
#define FIREBASE_HOST "soilmois-40a82-default-rtdb.firebaseio.com"
#define WIFI_SSID "motog9" // Change the name of your WIFI
#define WIFI_PASSWORD "12345678" // Change the password of your
WIFI #define FIREBASE_Authorization_key
"Xx1Y3LnuKnxqvFYq56HBc7SbuJwoP6siCtR7kNRX"
FirebaseData firebaseData;
FirebaseJson json;
int _moisture,sensor_analog;
const int sensor_pin = A0; /* Soil moisture sensor O/P pin */
void setup(void){
  Serial.begin(115200); /* Set the baudrate to 115200*/
  WiFi.begin (WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting...");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  Firebase.begin(FIREBASE_HOST,FIREBASE_Authorization_key);
}
void loop(void){
  sensor_analog = analogRead(sensor_pin);
```

```
_moisture = ( 100 - ( (sensor_analog/4095.00) * 100 ) );  
Serial.print("Moisture = ");  
Serial.print(_moisture); /* Print Temperature on the serial window */  
Serial.println("%");  
Firebase.setFloat(firebaseData, "/ESP32_APP/SOIL MOISTURE", _moisture);  
delay(1000); /* Wait for 1000mS */  
}
```

**Output in Serial monitor:**

252

<b>Ex.No:07</b>	<b>Application for Measuring the Distance using Ultrasonic sensor and Post Distance value on Google Cloud IoT platform</b>
<b>Date:</b>	

### **AIM:**

To develop an application to measures the distance in cm & inch using Ultrasonic sensor interface with ESP32 & Send the Sensor Readings to Google cloud IoI platform.

### **Apparatus required:**

1. ESP32
2. Ultrasonic Sensor
3. Connecting Wires
4. Breadboard

### **Software Required:**

1. Google Firebase
2. Arduino IDE
3. IDE Library files
  - a. Google Firebase ESP32 Library

### **Algorithm:**

- Step 1:** Declare the input pin
- Step 2:** Read the Digital value through Ultrasonic Sensor
- Step 3:** Convert to equitant decimal number
- Step 4:** Convert the value into cm & inch using formula
- Step 5:** Display in the Serial Monitor
- Step 6:** Repeat the Step 1-5

**Program:**

```
#include <FirebaseESP32.h>
#include <WiFi.h>
#define FIREBASE_HOST "distus-cb6a2-default-rtdb.firebaseio.com"
#define WIFI_SSID "motog9" // Change the name of your WIFI
#define WIFI_PASSWORD "12345678" // Change the password of your WIFI
#define FIREBASE_Authorization_key
"h9C1lYYTitgWPSxmomeIjva7fZZNTfKhop32QXhi"
FirebaseData firebaseData;
FirebaseJson json;
const int trigPin = 5;
const int echoPin = 18;
//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701
long duration;
float distanceCm;
float distanceInch;
void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  WiFi.begin (WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting...");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
```

```
Serial.println();
Serial.print("IP Address: ");
Serial.println(WiFi.localIP());
Serial.println();
Firebase.begin(FIREBASE_HOST,FIREBASE_Authorization_key);
}
void loop()
{
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculate the distance
distanceCm = duration * SOUND_SPEED/2;
// Convert to inches
distanceInch = distanceCm * CM_TO_INCH;
// Prints the distance in the Serial Monitor
Serial.print("Distance (cm): ");
Serial.println(distanceCm);
Serial.print("Distance (inch): ");
Serial.println(distanceInch);
Firebase.setFloat(firebaseData, "/ESP32_APP/DISTANCE IN CM", distanceCm);
Firebase.setFloat(firebaseData, "/ESP32_APP/DIATANCE IN INCH",
distanceInch);
```

```
delay(1000);  
}
```

**Output:**

350

350

350

350

350

350





<b>Ex.No:08</b>	<b>To Develop an application based on ultrasonic sensor</b>
<b>Date:</b>	

**AIM:**

To test ultrasound sensor with arduino UNO board and display in serial monitor of Arduino IDE.

**Apparatus required:**

1. Arduinio UNO Board
2. Ultrasound Sensor
3. Connecting wires

**Algorithm:**

- Step 1:** Declare the input pin
- Step 2:** Read the analog value through the ADC
- Step 3:** Convert to equitant decimal number
- Step 4:** Convert the value
- Step 5:** Display in the Serial Monitor
- Step 6:** Repeat the Step 1-5

**Program:**

```
#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 3 //attach pin D3 Arduino to pin Trig of HC-SR04
// defines variables
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate
  speed
  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial
  Monitor
  Serial.println("with Arduino UNO R3");
}
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
```

```
Serial.print(distance);  
Serial.println(" cm");  
}
```

**Output:**

325

325

325

325

325

325

325

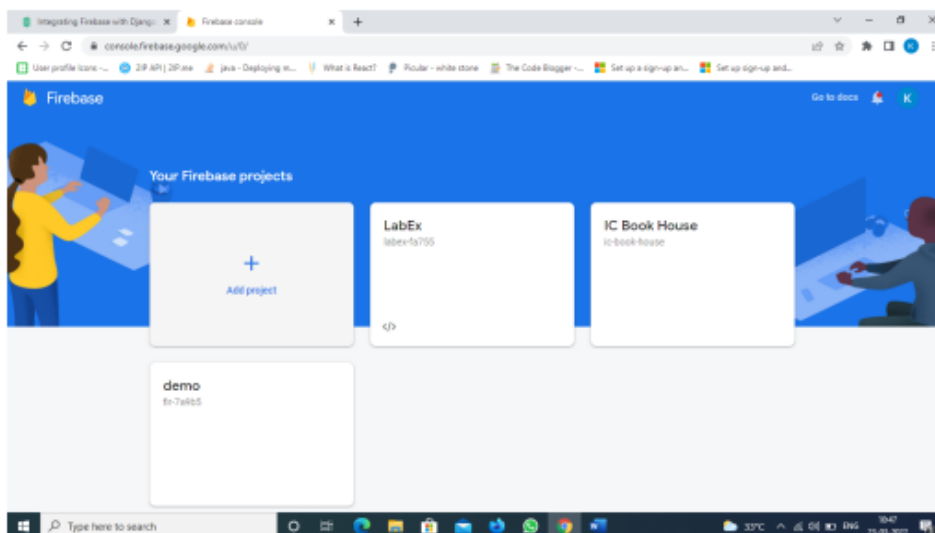
<b>Ex.No:09</b>	<b>To Develop IoT application using Django framework and Firebase/ Bluemix platform</b>
<b>Date:</b>	

### Aim:

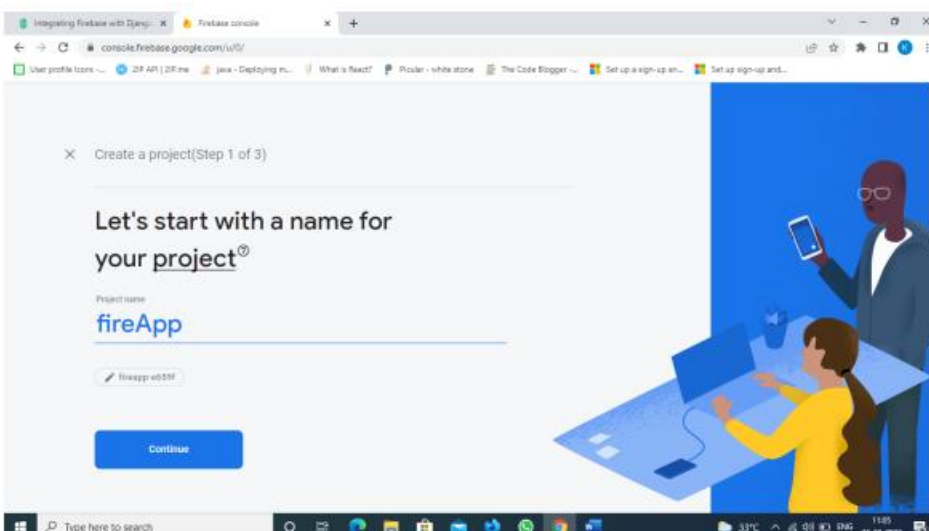
To create an application that reads the data from firebase using Django framework.

### Procedure:

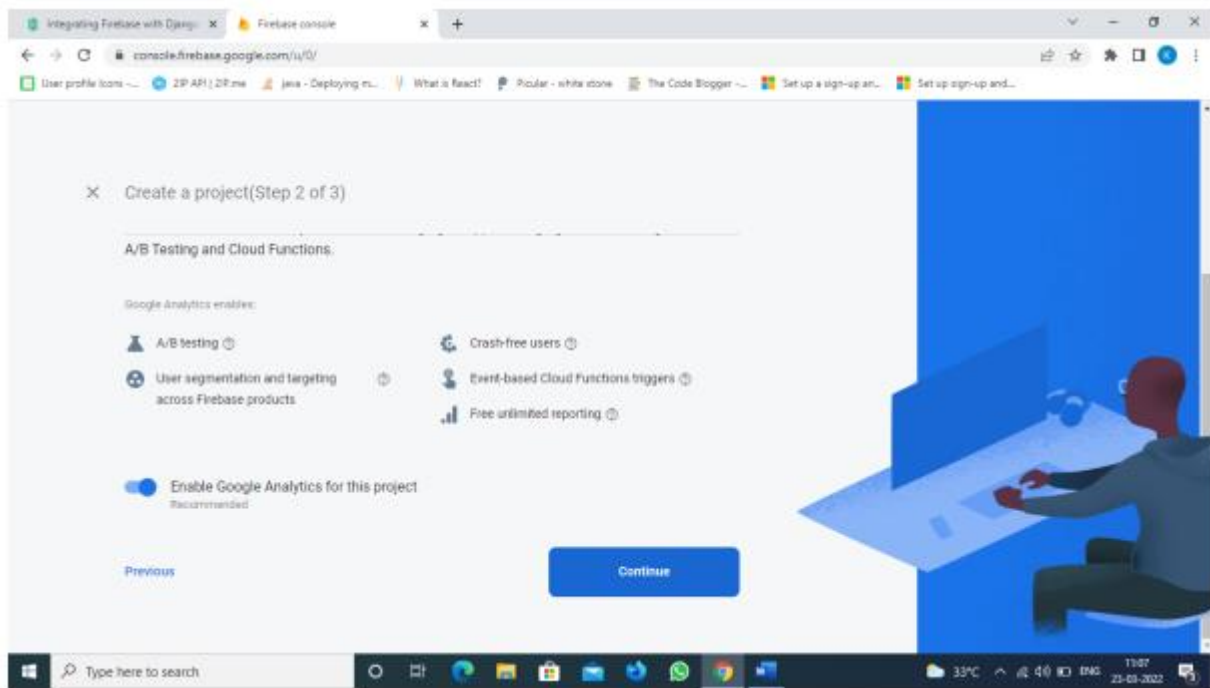
**Step 1:** Go to firebase console by logging into the google account and click on Add Project. <https://console.firebase.google.com/u/0/>



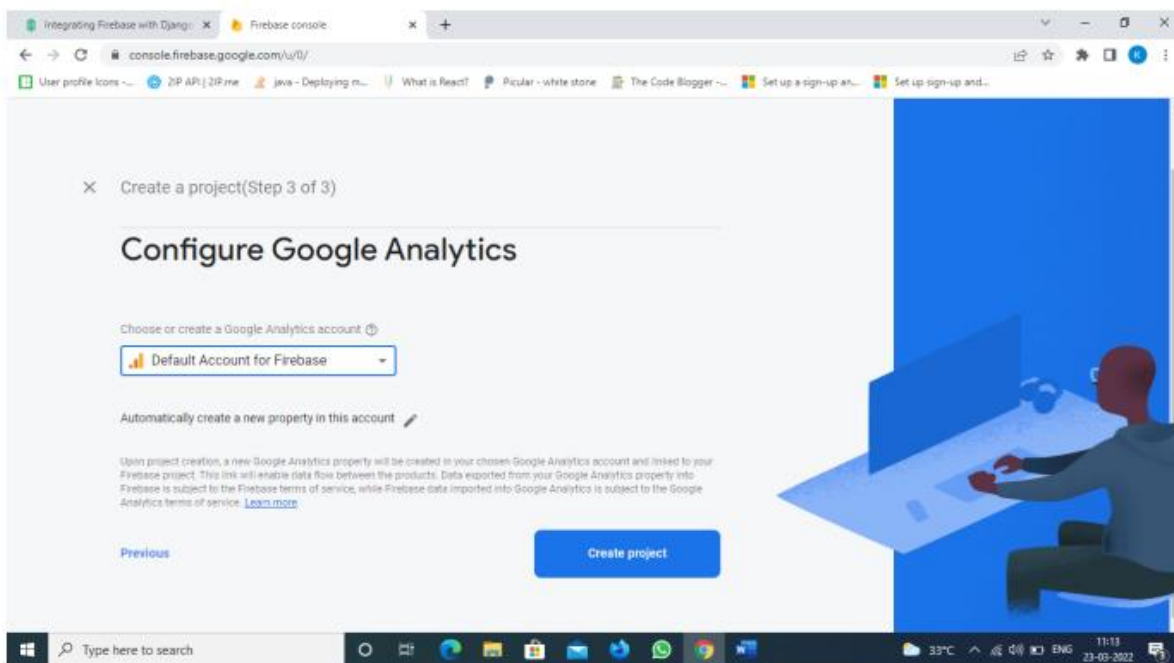
**Step 2.1:** Add the project name and click continue button



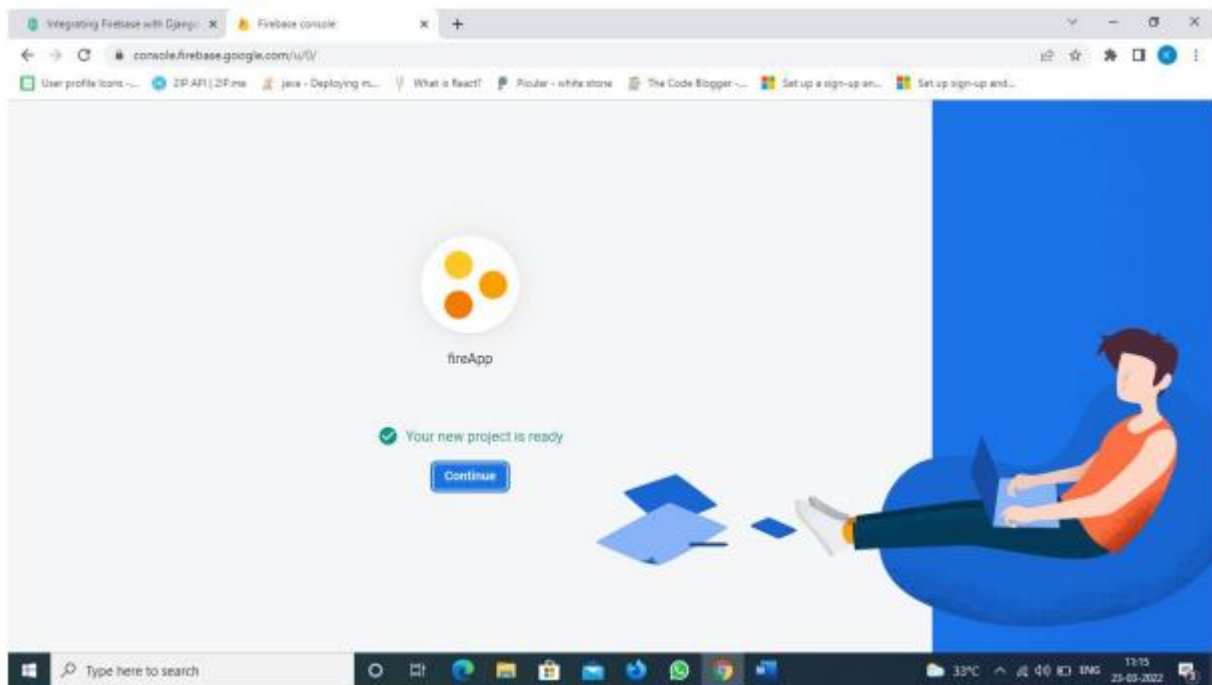
## Step 2.2: Again click on the continue button



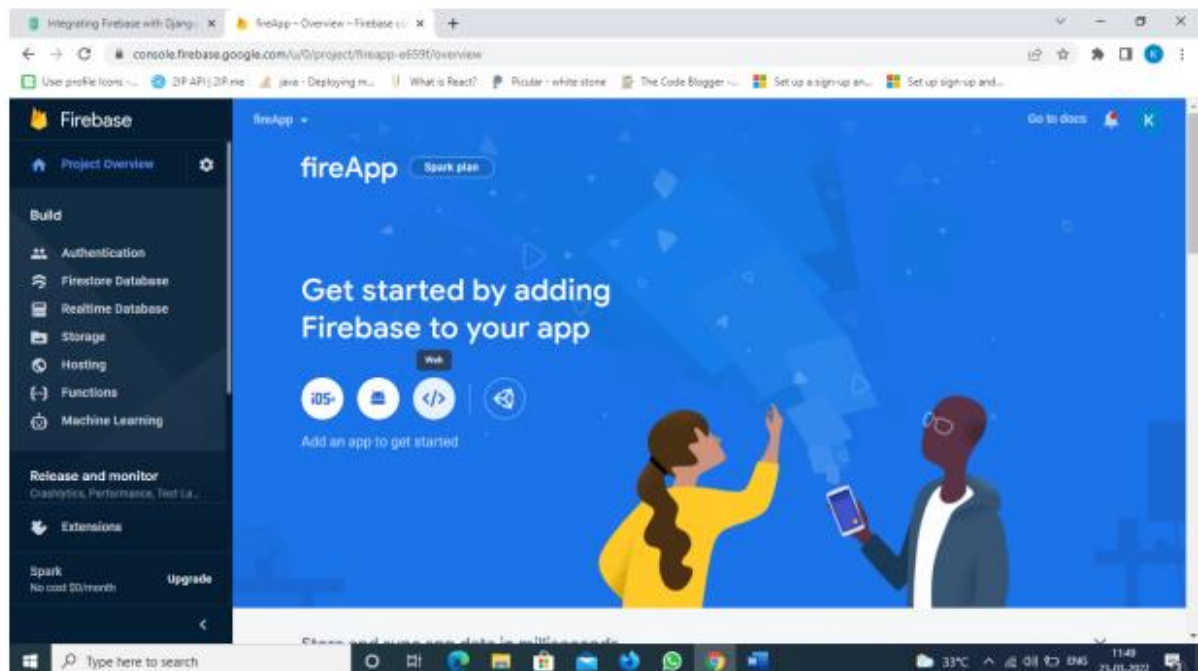
## Step 2.3: Select the Default Account for Firebase in the Select an account drop down list.



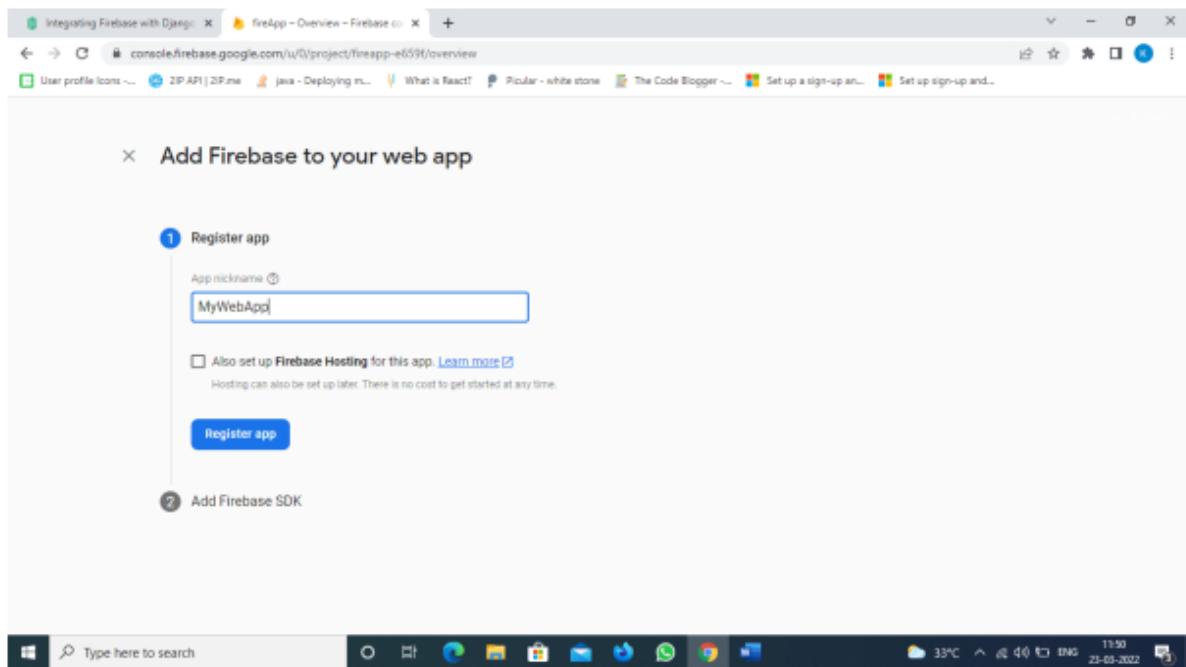
## Step 2.4: Click on the continue button



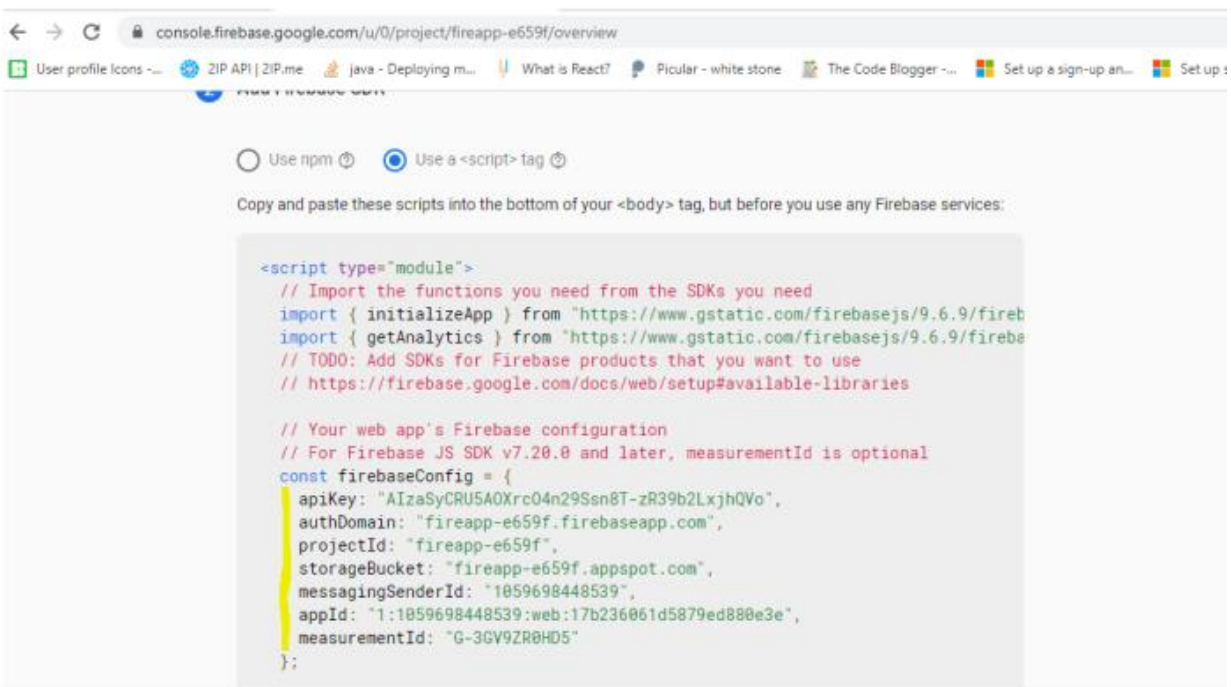
## Step 3.1: Register the web app by clicking on the web logo



### Step 3.2: Add the WebApp name and click on Register app

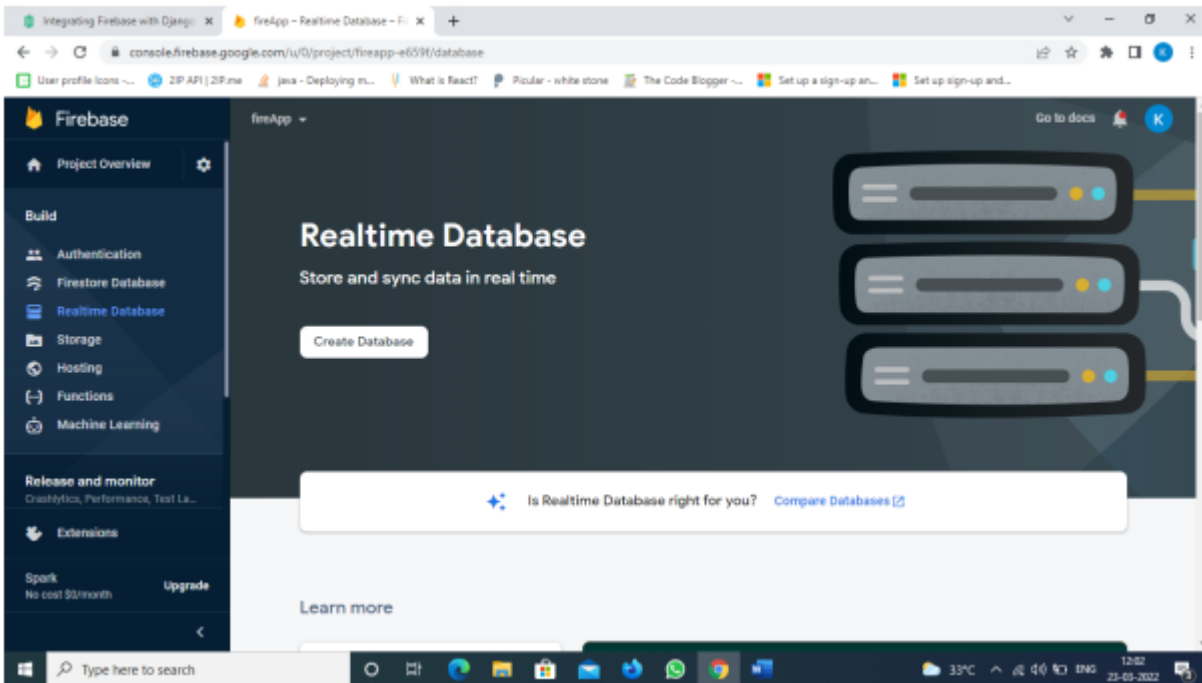


### Step 3.3: Select the Use a <script> tag radio button, copy the highlighted code and paste it in the notepad or text editor. Click on the Continue to the console button.

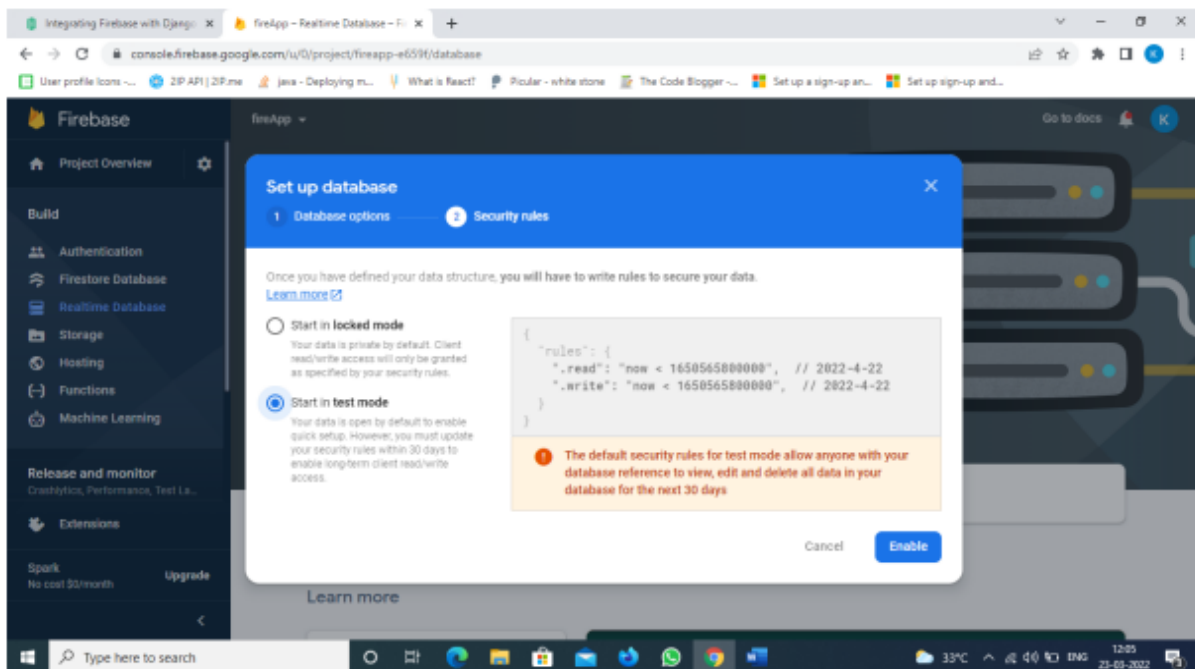




**Step 4:** Create a new real time database from the firebase console by clicking on the Create Database button and then click Next button.



**Step 4.2:** Select the test mode radio button and then click Enable button.



**Step 5:** Add a new child by clicking on the + button. Add the following data in the database.

The screenshot shows the Firebase Realtime Database web interface. At the top, the URL is `https://fireapp-e659f-default-rtdb.firebaseio.com/`. Below the URL, there is a tree view with a node labeled `fireapp-e659f-default-rtdb: null`. To the right of this node is a '+' button and an 'x' button. Below the tree view, there is a table with three rows of data. The first row has 'Name' as 'Name' and 'Value' as 'Kafeel'. The second row has 'Name' as 'Department' and 'Value' as 'MCA'. The third row has 'Name' as 'College' and 'Value' as 'PEC'. At the bottom of the table, there are 'Cancel' and 'Add' buttons.

Name	Data	
Name	Name	Value Kafeel
Name	Department	Value MCA
Name	College	Value PEC

**Step 6:** Open VS code and open a terminal from VS Code, Create a Django project by using the following command

**`django-admin startproject firebase_project`**

**Step 7:** Change the directory to `firebase_project` by using the following command:

**`cd firebase_project`**

**Step 8:** Create a Django App within the `firebase_project` folder by using the following command.

**`django-admin startproject fireapp`**

**Step 9:** Navigate to the `firebase_project` directory and edit the `settings.py` file, Add the created `fireapp` in the `INSTALLED_APPS` list.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'fireapp',  
]
```

**Step 10:** To connect Firebase and Django, we need to install a python package named pyrebase by using the following command.

**pip install pyrebase**

**Step 11:** Create a python file name it as view.py in the fireapp directory and add the following code

```
from django.shortcuts import render
```

```
import pyrebase
```

```
config = {
```

```
    "apiKey": "AIzaSyCRU5AOXrcO4n29Ssn8T-zR39b2LxjhQVo",
```

```
    "authDomain": "fireapp-e659f.firebaseio.com",
```

```
    "projectId": "fireapp-e659f",
```

```
    "storageBucket": "fireapp-e659f.appspot.com",
```

```
    "messagingSenderId": "1059698448539",
```

```
    "appId": "1:1059698448539:web:17b236061d5879ed880e3e",
```

```
    "measurementId": "G-3GV9ZR0HD5"
```

```
}
```

```
firebase=pyrebase.initialize_app(config)
```

```
authe = firebase.auth()
```

```
database=firebase.database()
```

```
def index(request):
```

```
    name = database.child('Data').child('Name').get().val()
```

```
    department = database.child('Data').child('Department').get().val()
```

```
    college = database.child('Data').child('College').get().val()
```

```
    context = {
```

```
        'name':name,
```

```
        'department':department,
```

```
        'college':college
```

```
    }
```

```
    return render(request, 'index.html', context)
```

**Step 12:** Create a new directory named as templates in the fireapp directory and create index.html file.

```
<!DOCTYPE html>

<html>

  <head>

    <title>Firebase Demo</title>

  </head>

  <body>

    <h1>Firebase Database</h1>

    <h5>Read the following data from Firebase</h5>

    <p>Name : {{ name }}</p>

    <p>Department : {{ department }}</p>

    <p>College : {{ college }}</p>

  </body>

</html>
```

**Step 13:** We need to update the DIRS to the path of the templates folder in our settings.py file. Don't forget to import the os librarar. Add import os on top of the code.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
```

**Step 14:** Next, we edit our `urls.py` by adding the following code

```
from Django.contrib import admin
from Django.URLs import path
from fireapp import view
urlpatterns = [
    path('admin/', admin.site.urls),
    path("", view.index, name='index'),
]
```

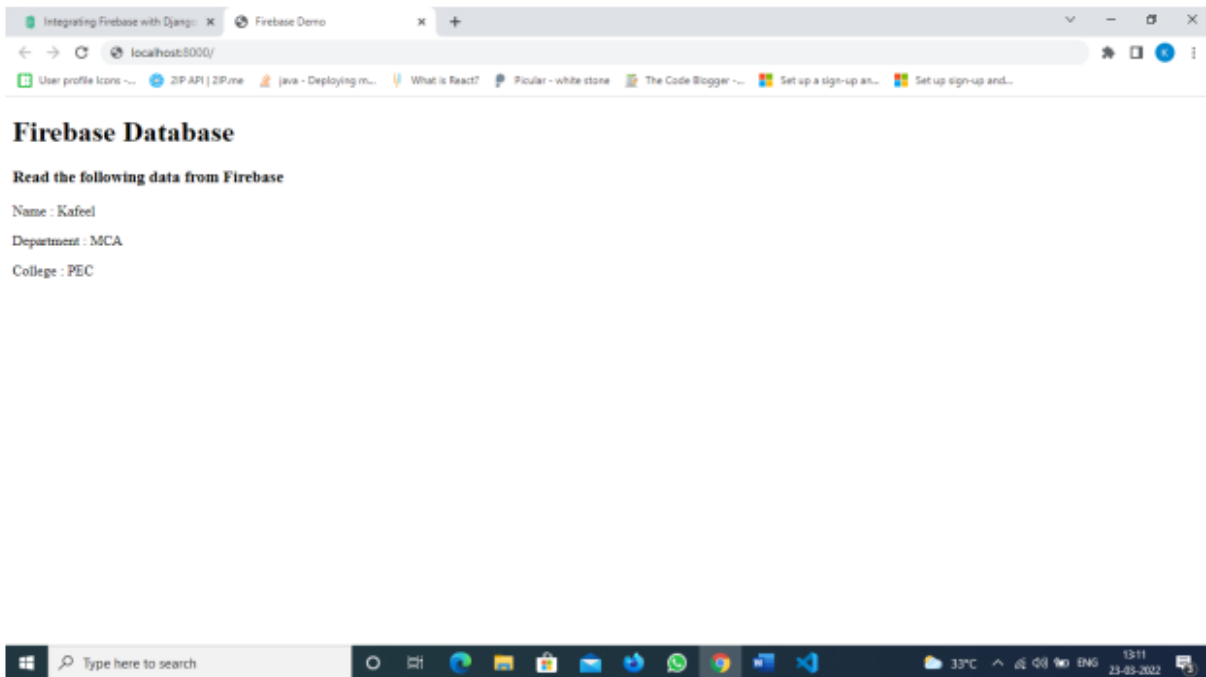
**Step 15:** Run the server by using the following command:

```
python manage.py runserver
```

**Step 16:** Open a web browser and type the URL in the address bar

<http://localhost:8000/>

## Output:



<b>Ex.No:10</b>	<b>To Develop a commercial IoT application using temperature</b>
<b>Date:</b>	

**AIM :**

To write Arduino sketch to sense the temperature using DHT 11 sensor and to monitor its output in mobile using IOT.

**COMPONENTS REQUIRED**

NODE MCU	1
DHT 11 SENSOR	1
USB cable	1

## PROGRAM :

```
#include <ESP8266WiFi.h>
#include<ESP8266WebServer.h>
#include "DHT.h"

// Uncomment one of the lines below for whatever DHT sensor type you're using!
//#define DHTTYPE DHT11 // DHT 11
//#define DHTTYPE DHT21 // DHT 21 (AM2301)
#define DHTTYPE DHT11 // DHT 22 (AM2302), AM2321

/*Put your SSID & Password*/

const char* ssid = "SAIHAAN"; // Enter SSID here/ / type the name of the hotspot
const char* password = "43214321..."; //Enter Password here // type the password
of the hotspot ESP8266WebServer server(80); // DHT Sensor uint8_t DHTPin
=D8;

// Initialize DHT sensor.
DHT dht(DHTPin, DHTTYPE);

float Temperature; float Humidity; void setup() {
Serial.begin(115200);

delay(100); pinMode(DHTPin, INPUT); dht.begin();

Serial.println("Connecting to "); Serial.println(ssid);

//connect to your local wi-fi network WiFi.begin(ssid, password);

//check wi-fi is connected to wi-fi network
while (WiFi.status() != WL_CONNECTED)
{ delay(1000);
Serial.print(".");
}

Serial.println("");
Serial.println("WiFiconnected..!");
Serial.print("Got IP: ");
Serial.println(WiFi.localIP());
```



```

server.on("/", handle_OnConnect);
server.onNotFound(handle_NotFound); server.begin();
Serial.println("HTTP server started");
}
void loop() { server.handleClient();
}
void handle_OnConnect() {
Temperature = dht.readTemperature(); // Gets the values of the temperature
Humidity = dht.readHumidity(); // Gets the values of the humidity
server.send(200, "text/html", SendHTML(Temperature,Humidity));
}
void handle_NotFound(){
server.send(404, "text/plain", "Notfound");
}
String SendHTML(float Temperaturestat,floatHumiditystat)
{
String ptr = "<!DOCTYPE html><html>\n";
ptr +="<head><meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0, user- scalable=no\">\n";
ptr +="<title>ESP8266 Weather Report</title>\n";
ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px
auto; text-
align: center;} \n";
ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px
auto30px;} \n"; ptr +="p {font-size: 24px;color: #444444;margin-
bottom:10px;} \n";
ptr +="</style>\n"; ptr +="</head>\n"; ptr +="<body>\n";
ptr +="<div id=\"webpage\">\n";
ptr +="</style>\n"; ptr +="</head>\n"; ptr +="<body>\n";

```

```
ptr +="<div id=\"webpage\">\n";  
ptr +="<h1>ESP8266 NodeMCU Weather Report</h1>\n";  
ptr +="<p>Temperature: "; ptr +=(int)Temperaturestat; ptr +="°C</p>";  
ptr +="<p>Humidity: "; ptr +=(int)Humiditystat; ptr +="%</p>";  
ptr +="</div>\n"; ptr +="</body>\n"; ptr +="</html>\n"; return ptr;  
}
```

## Output:

