CODE

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15 // what pin we're connected to
##define DHTTYPE DHT22 // define type of sensor DHT 11
#define LED 2
DHT dht (DHTPIN, DHTTYPE);
void callback(char* subscribe topic, byte* payload, unsigned int
payloadLength);
//-------credentials of IBM Accounts------
#define ORG "yzs5sj"//IBM ORGANITION ID
#define DEVICE_TYPE "fire_IoT"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "17082001"//Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "1911089abcdefgh" //Token
String data3;
float t;
//-------- Customize the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of
event perform and format in which data to be send
char subscribe topic[] = "iot-2/cmd/command/fmt/String";// cmd
REPRESENTcommand type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
```

```
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
//-------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing parameter like server ip,port and wifi credential
void setup()// configuring the ESP32
{
 Serial.begin(115200);
dht.begin();
 pinMode(LED,OUTPUT);
 delay(10);
 Serial.println();
wifi connect();
mqtt connect();
}
void loop()// Recursive Function
{
t = dht.readTemperature();
 Serial.print("temperature:");
 Serial.println(t);
PublishData(t);
 delay(1000);
 if (!client.loop()) {
mqtt connect();
}
}
```

```
/*.....retrieving to
Cloud.....*/
void PublishData(float temp) {
mqtt connect();//function call for connecting to ibm
 /*
creating the String in in form JSon to update the data to ibm cloud
*/
 String payload = "{\"temperature\":";
payload += temp;
payload += "}";
 Serial.print("Sending payload: ");
 Serial.println(payload);
 if (client.publish(publishTopic, (char*) payload.c_str())) {
 Serial.println("Publish ok");// if it successfully upload data on the cloud
then it will print publish ok in Serial monitor or else it will print publish failed
} else {
 Serial.println("Publish failed");
}
}
void mqtt connect() {
 if (!client.connected()) {
 Serial.print("Reconnecting client to ");
 Serial.println(server);
 while (!!!client.connect(clientId, authMethod, token)) {
 Serial.print(".");
 delay(500);
}
```

```
initManagedDevice();
 Serial.println();
}
}
void wifi connect() //function definition for wifi connect
{
 Serial.println();
 Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish
the connection
 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
 Serial.print(".");
}
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
void initManagedDevice() {
 if (client.subscribe(subscribe topic)) {
 Serial.println((subscribe topic));
 Serial.println("subscribe to cmd OK");
} else {
 Serial.println("subscribe to cmd FAILED");
}
}
```

```
void callback(char* subscribe topic, byte* payload, unsigned int
payloadLength)
{
 Serial.print("callback invoked for topic: ");
 Serial.println(subscribetopic);
 for (int i = 0; i < payloadLength; i++) {
 //Serial.print((char)payload[i]);
data3 += (char)payload[i];
}
 Serial.println("data: "+ data3);
 if(data3=="lighton")
{
Serial.println(data3);
digitalWrite(LED,HIGH);
}
 else
{
Serial.println(data3);
digitalWrite(LED,LOW);
}
data3="";
}
```