# INTERNET OF THINGS(IOT)

# SMART BILLING SYSTEM FOR WATER SUPPLIERS

WOWKI LINK: https://wokwi.com/projects/364909105770454017

CODE:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#define RELAY_PIN 18 // ESP32 pin GIOP18 connected to the IN pin of relay
#include "time.h"
float time1=0;
float motorbill;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "1z668o"//IBM ORGANITION ID
#define DEVICE_TYPE "Relay"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
String data3;
float h, t;
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 0;
const int   daylightOffset_sec = 3600;

//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd  REPRESENT command 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);
```

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin as an output.
  Serial.begin(115200);
  pinMode(RELAY_PIN, OUTPUT);
  delay(10);
  Serial.println();
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);









  wificonnect();
  mqttconnect();
}

// the loop function runs over and over again forever
void loop() {
 // digitalWrite(RELAY_PIN, HIGH);
  //delay(1000);
  //digitalWrite(RELAY_PIN, LOW);
  //delay(1000);
 // motorbill=random(60,200);
  //motorbill=motorbill*5;
  //delay(1000);
  //PublishData(motorbill);
  if (!client.loop()) {
    mqttconnect();
  }
}

void PublishData(float motorbill) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"motorbill\":";
  payload += motorbill;

  payload += "}";


  Serial.print("Sending payload: ");
  Serial.println(payload);


  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully 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 mqttconnect() {
  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 wificonnect() //function defination for wificonnect
{
  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(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

void callback(char* subscribetopic, 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=="on")
  {
Serial.println(data3);
digitalWrite(RELAY_PIN, HIGH);
PublishData(0);
Serial.println("The time at which the motor is switched on:");
printLocalTime();

time1+=1;

  }

  else if(data3=="off")
  {
  //Serial.print(time1);
Serial.println(data3);
digitalWrite(RELAY_PIN, LOW);
motorbill=random(60,200);
motorbill=motorbill*5;
delay(1000);
PublishData(motorbill);
Serial.println("The time at which the motor is switched off:");
printLocalTime();
time1=0;

  }

data3="";



}
void printLocalTime(){
  struct tm* timeinfo;
  time_t now;
  time(&now);

  timeinfo = localtime(&now);
  Serial.print(timeinfo,"%H:%M:%S");

  /* Serial.println("Hour: ");
  Serial.println(timeinfo, "%H");
  Serial.print("Hour (12 hour format): ");
```

```
  Serial.println(timeinfo, "%I");
  Serial.print("Minute: ");
  Serial.println(timeinfo, "%M");
  Serial.print("Second: ");
  Serial.println(timeinfo, "%S");*/


  Serial.println();
}
```