ASSIGNMENT-2

Code:

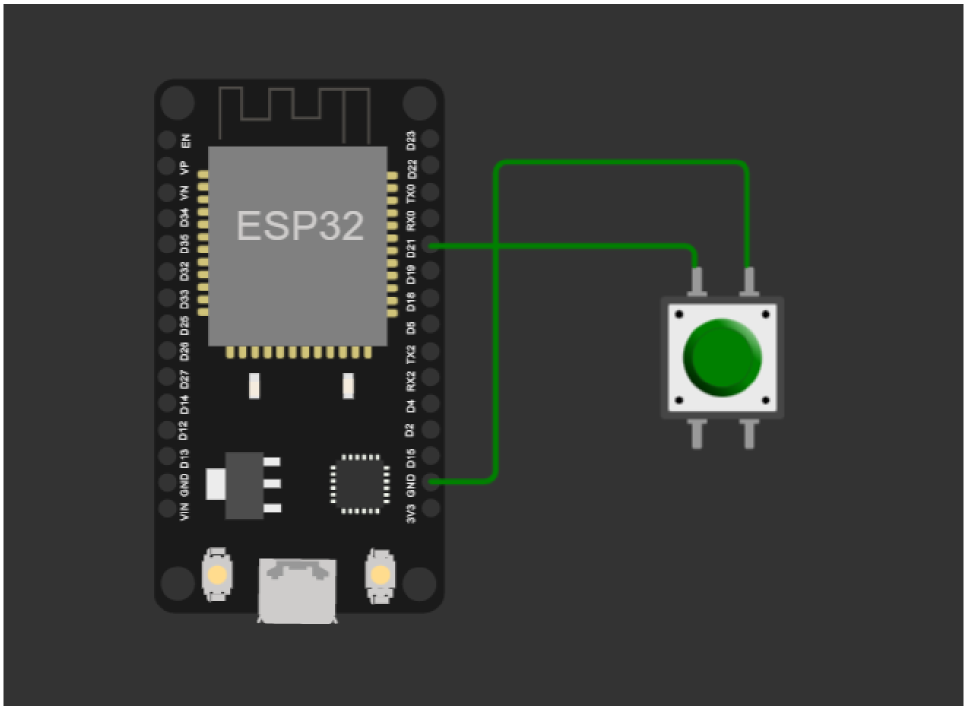
```
#include <WiFi.h>//library for wifi

#include <PubSubClient.h>//library for MQtt

#define BUTTON_PIN 21

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

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

#define ORG "9u4b2o"//IBM ORGANITION ID

#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform

#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "12345678"     //Token

String data3;

int lastState = HIGH;

int currentState;

//-------- 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/command/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); //calling the predefined client id by passing parameter like server id,portand wificredential
```

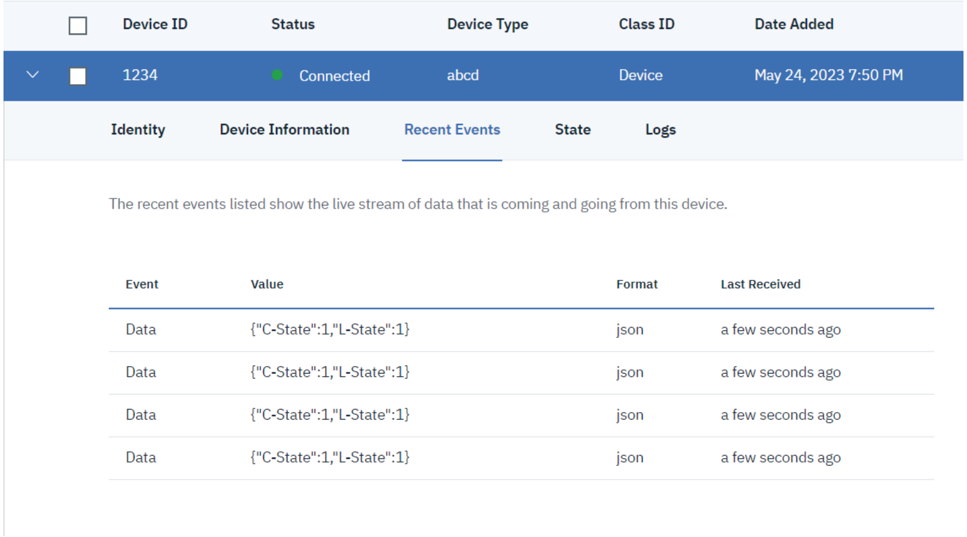
```
void setup()// configureing the ESP32
{
  Serial.begin(9600);
  // initialize the pushbutton pin as an pull-up input
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  wificonnect();
  mqttconnect();
}
void loop()// Recursive Function
{
   currentState = digitalRead(BUTTON_PIN);
  if(lastState == LOW && currentState == HIGH)
    Serial.println("The state changed from LOW to HIGH");
  // save the last state
  lastState = currentState;
  PublishData(currentState,lastState);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}
/*....................................retrieving to Cloud...............................*/

void PublishData(int currentState, int lastState) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"C-State\":";
  payload += currentState;
  payload += "," "\"L-State\":";
  payload += lastState;
  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);

}
```

SCHEMATIC + WORKING SCHEMATIC

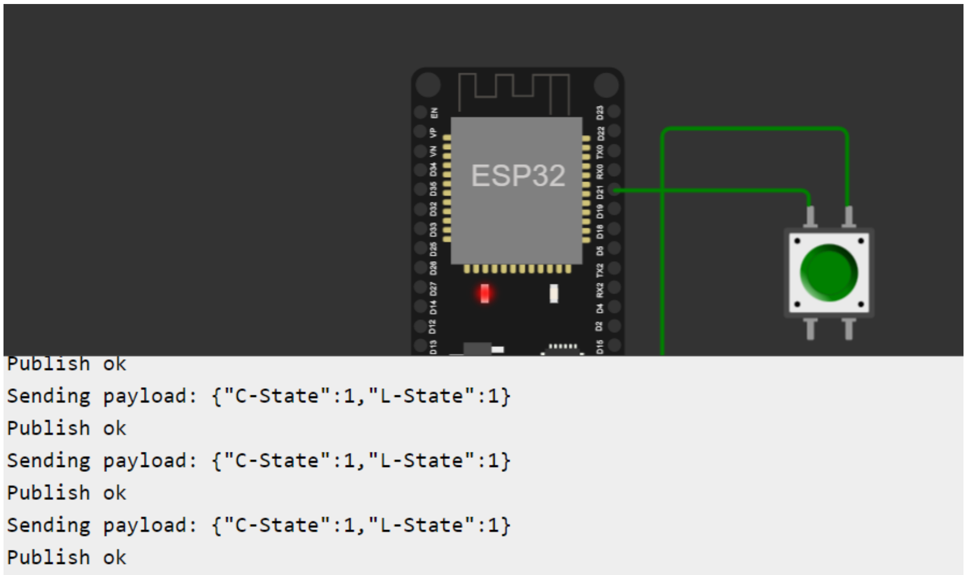




	□	Device ID	Status	Device Type	Class ID	Date Added
⌄	□	1234	● Connected	abcd	Device	May 24, 2023 7:50 PM

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"C-State":1,"L-State":1}	json	a few seconds ago
Data	{"C-State":1,"L-State":1}	json	a few seconds ago
Data	{"C-State":1,"L-State":1}	json	a few seconds ago
Data	{"C-State":1,"L-State":1}	json	a few seconds ago



```
Publish ok
Sending payload: {"C-State":1,"L-State":1}
Publish ok
Sending payload: {"C-State":1,"L-State":1}
Publish ok
Sending payload: {"C-State":1,"L-State":1}
Publish ok
```