### ASSIGNMENT 2

**NAME:** SREEMATHI SIVAKUMAR **REGISTRATION NO:** 20BEC1174 **DATE:** 27/05/2022

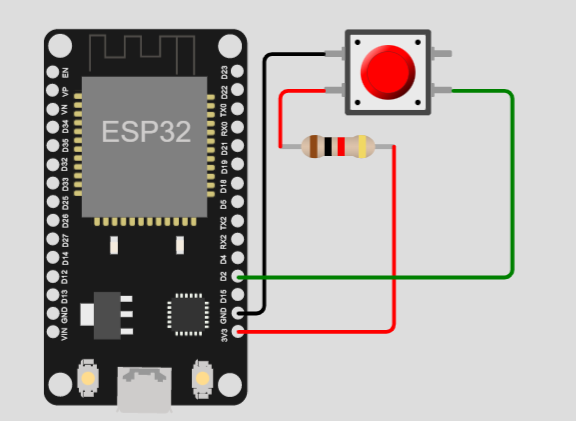
**WOKWI AND IBM CLOUD**

**Aim:** To construct a circuit with push button using WOKWI and send the data (1s and 0s) to IBM cloud

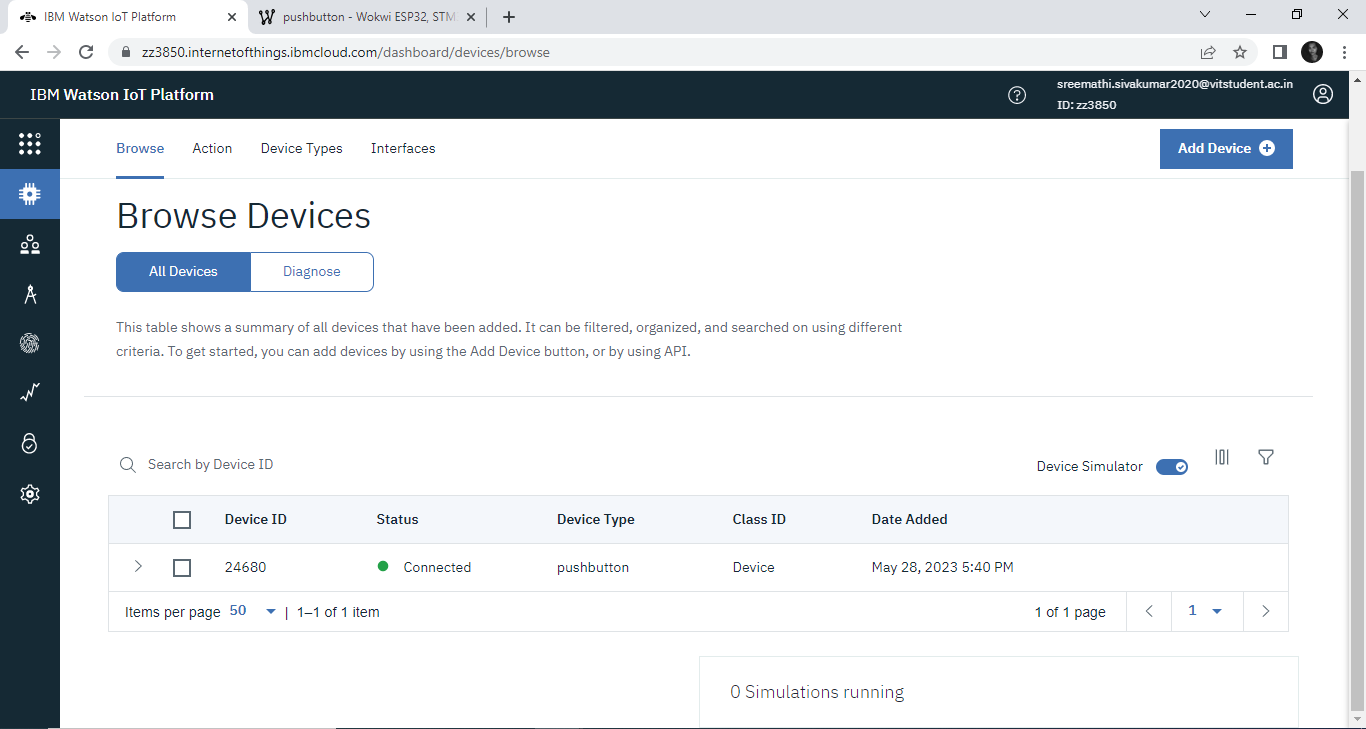
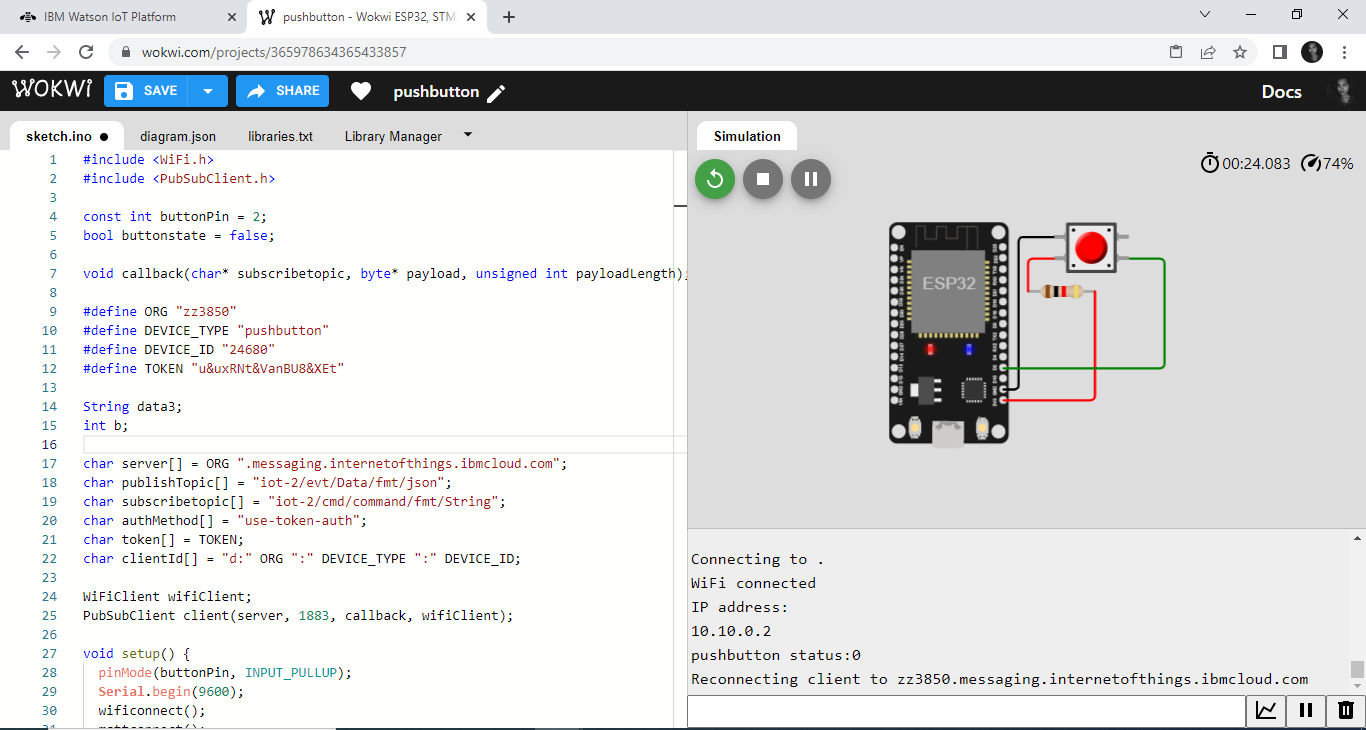
**Software required:** WOKWI , IBM WATSON IoT

**Components Required:** ESP32, 1K Resistor, Push button, Connecting wires

## Circuit Diagram:



**Simulation:**

****

**Procedure:**

1. Place the "ESP32 Devkit" board component on the breadboard.
2. Place the push button component on the breadboard.
3. Connect the push button's left pin to one leg of the 1K resistor.
4. Connect the other leg of the resistor to the ESP32's 3V3 pin.
5. Connect the push button's left pin to the ESP32's GND pin.
6. Connect the push button's right pin to the ESP32's D2 pin.
7. Write the necessary code
8. Create a new device in IBM IoT and give the device details in the code
9. Run the simulation

**Code :**

#include <WiFi.h>

#include <PubSubClient.h>

const int buttonPin = 2;

bool buttonstate = false;

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

#define ORG "zz3850"

#define DEVICE\_TYPE "pushbutton"

#define DEVICE\_ID "24680"

#define TOKEN "u&uxRNt&VanBU8&XEt"

String data3;

int b;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char publishTopic[] = "iot-2/evt/Data/fmt/json";

char subscribetopic[] = "iot-2/cmd/command/fmt/String";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

PubSubClient client(server, 1883, callback, wifiClient);

void setup() {

  pinMode(buttonPin, INPUT\_PULLUP);

**Serial**.begin(9600);

  wificonnect();

  mqttconnect();

}

void loop() {

  buttonstate = digitalRead(buttonPin);

  if (buttonstate == LOW) {

    b = 1;

  } else {

    b = 0;

  }

  wificonnect();

**Serial**.print("pushbutton status:");

**Serial**.println(b);

  PublishData(b);

  delay(1000);

  if (!client.loop()) {

    mqttconnect();

  }

}

void PublishData(int button) {

  mqttconnect();

  String payload = "{\"pushbutton status\":" + String(button) + "}";

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

**Serial**.println(payload);

  if (client.publish(publishTopic, (char\*)payload.c\_str())) {

**Serial**.println("Publish ok");

  } else {

**Serial**.println("Publish failed");

  }

}

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

  String data3;

  for (int i = 0; i < payloadLength; i++) {

    data3 += (char)payload[i];

  }

  int startIndex = data3.indexOf(":") + 1;

  int endIndex = data3.length() - 1;

  String valueString = data3.substring(startIndex, endIndex);

  int value = valueString.toInt();

**Serial**.print("Received value: ");

**Serial**.println(value);

}

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() {

**Serial**.println();

**Serial**.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);

  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 {

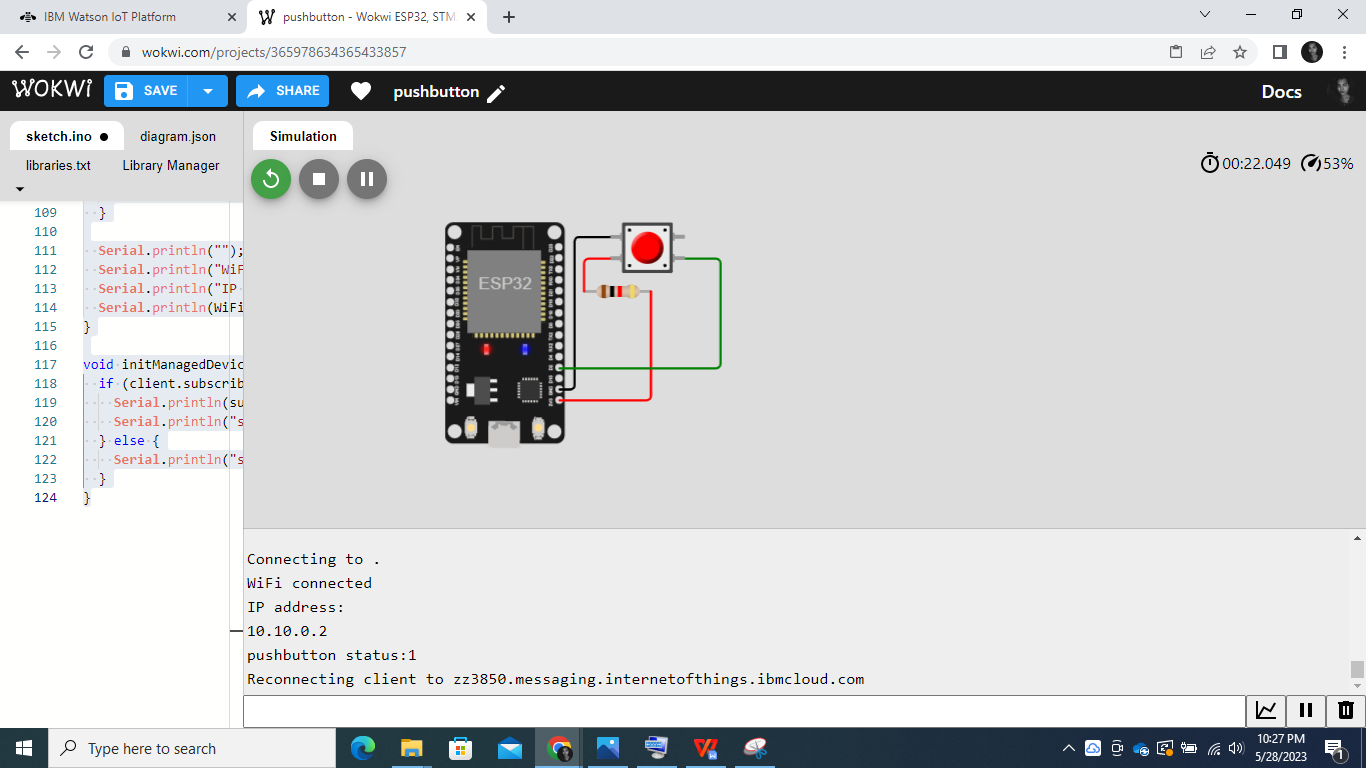
**Serial**.println("subscribe to cmd FAILED");

  }

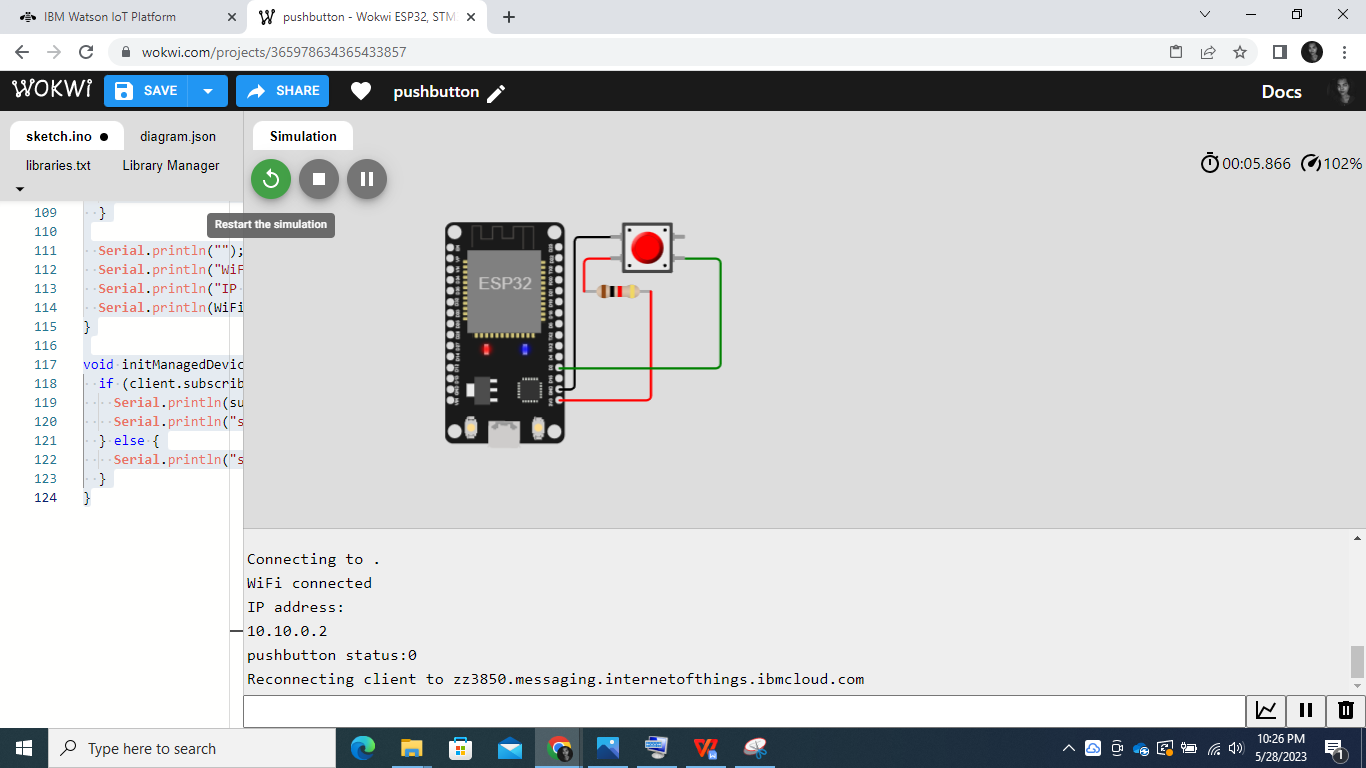
}

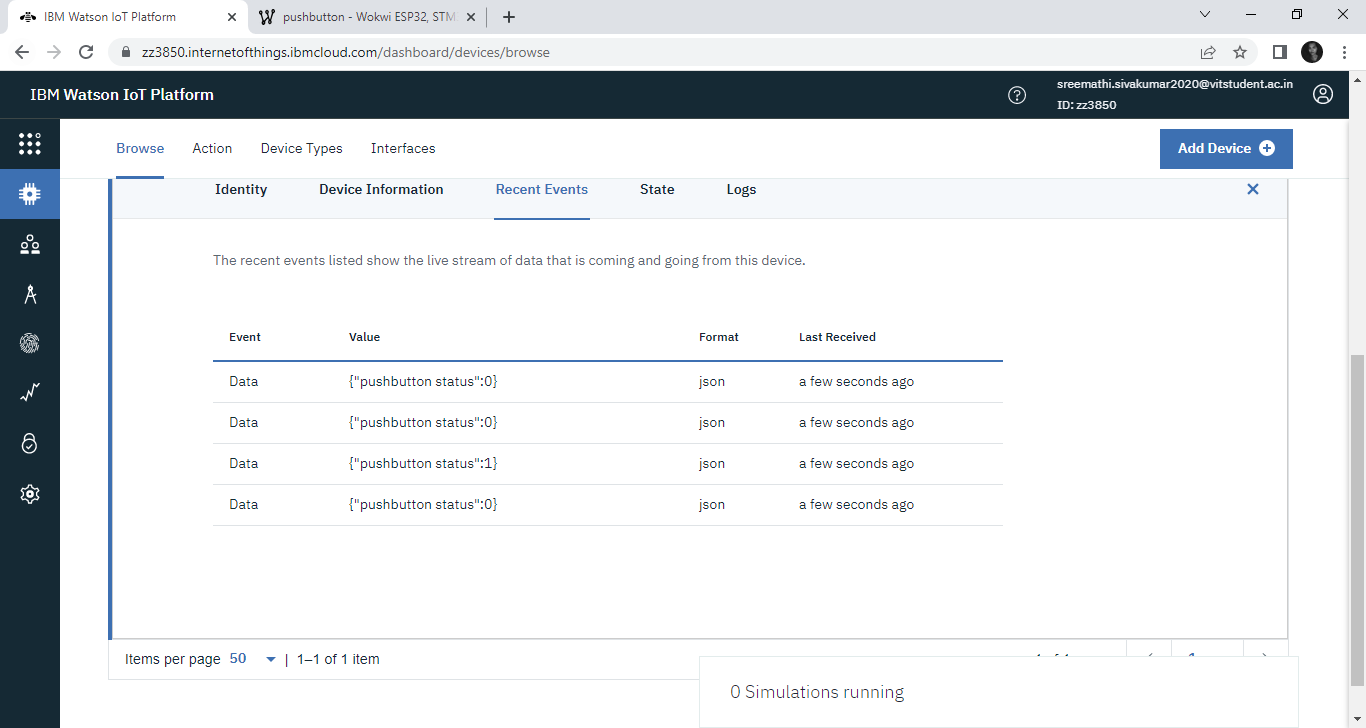
**Outputs :**

**When Button not pushed = 0**

****

**When Button pushed = 1**





The circuit has been constructed in WOKWI and the status of the push buttons have been uploaded on IBM IoT Cloud as 1s and 0s.

**Result:**

Thus, the values from WOKWI has been uploaded to IBM cloud using wifi module of ESP32 successfully and outputs are verified.