MAKE OUTPUT IN WATSON IBM CLOUD & NODE-RED CLOUD

Sprint_12

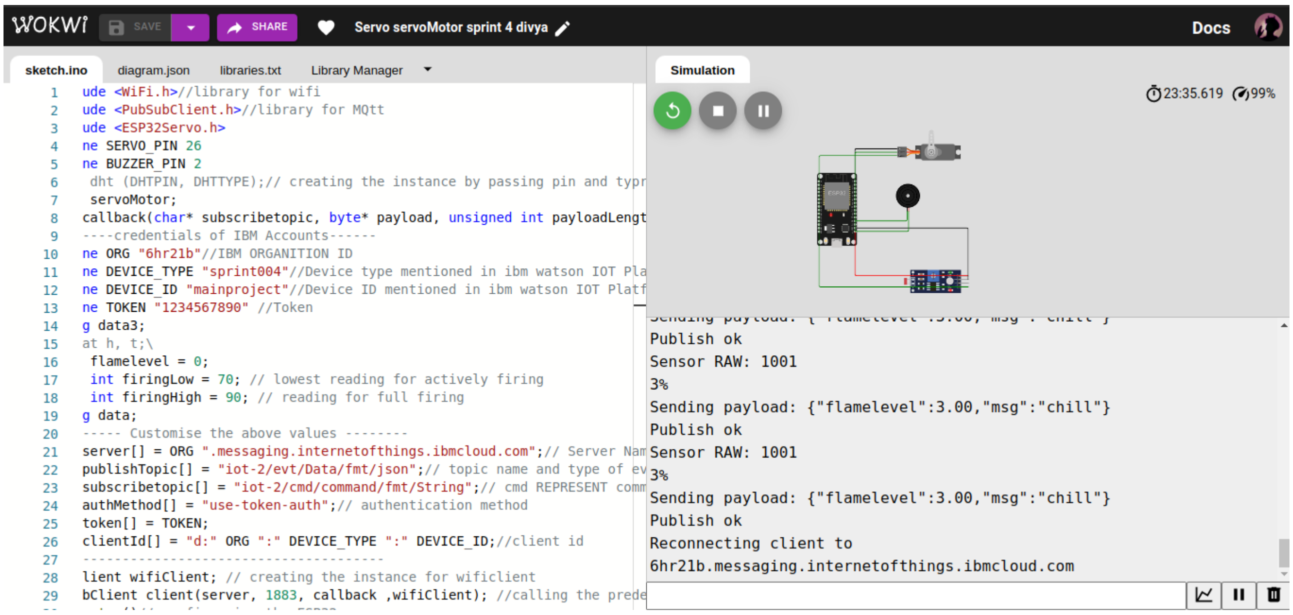
Team ID : PNT2022TMID29941
IBM ID : IBM-Project-31889-1660205917

Code;

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include <ESP32Servo.h>
#define SERVO_PIN 26
#define BUZZER_PIN 2
//DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht
connected
Servo servoMotor;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
//-------credentials of IBM Accounts------
#define ORG "6hr21b"//IBM ORGANITION ID
#define DEVICE_TYPE "sprint004"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "mainproject"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "1234567890" //Token
String data3;
//float h, t;\
float flamelevel = 0;
const int firingLow = 70; // lowest reading for actively firing
const int firingHigh = 90; // reading for full firing
String data;
//-------- 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(115200);
pinMode(BUZZER_PIN, OUTPUT);
servoMotor.attach(SERVO_PIN);
delay(10);
Serial.println();
wificonnect();
mqttconnect();
```

```
}
void loop()// Recursive Function
{
float analogValue = analogRead(36);
float flamelevel;
Serial.print("Sensor RAW: ");
Serial.println(analogValue, 0);
flamelevel = map(analogValue, 0, 1024, 100, 0);
Serial.print(flamelevel, 0);
Serial.println("%");
if (flamelevel >= firingHigh) { // stoker is fully firing
tone(BUZZER_PIN,2000);
servoMotor.write(180);
delay(300);
data="alert";
}
if (flamelevel < firingLow) { // fire out
data="chill";
noTone(BUZZER_PIN);
servoMotor.write(0);
// send alert
}
PublishData(flamelevel);
delay(1000);
if (!client.loop()) {
mqttconnect();
}
}
/*...................................retrieving to Cloud..............................*/
void PublishData(float flamelevel) {
mqttconnect();//function call for connecting to ibm
/*
creating the String in in form JSon to update the data to ibm cloud
*/
String payload = "{\"flamelevel\":";
payload += flamelevel;
payload += "," "\"msg\":\"";
payload += data;
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=="lighton")
{
Serial.println(data3);
digitalWrite(LED,HIGH);
}
else
{
Serial.println(data3);
digitalWrite(LED,LOW);
}
```

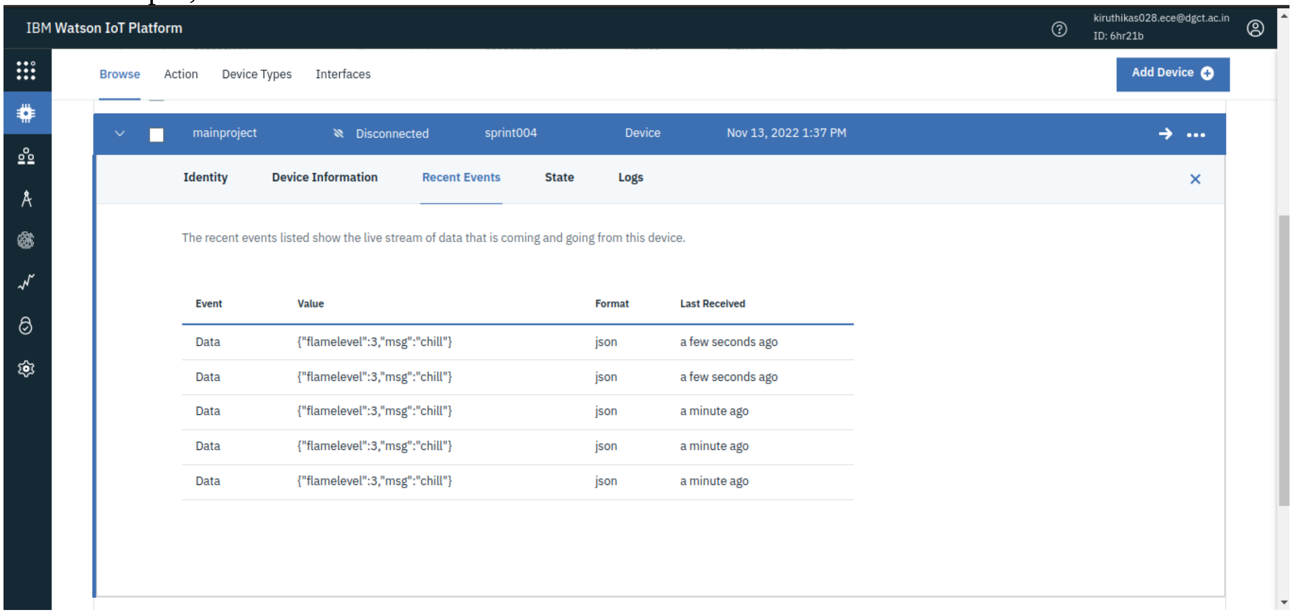
```
data3=""; */
}
```



```
ude <WiFi.h>//library for wifi
ude <PubSubClient.h>//library for MQtt
ude <ESP32Servo.h>
ne SERVO_PIN 26
ne BUZZER_PIN 2
 dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr
 servoMotor;
callback(char* subscribetopic, byte* payload, unsigned int payloadLengt
----credentials of IBM Accounts------
ne ORG "6hr21b"//IBM ORGANITION ID
ne DEVICE_TYPE "sprint004"//Device type mentioned in ibm watson IOT Pla
ne DEVICE_ID "mainproject"//Device ID mentioned in ibm watson IOT Platf
ne TOKEN "1234567890" //Token
g data3;
at h, t;\
 flamelevel = 0;
 int firingLow = 70; // lowest reading for actively firing
 int firingHigh = 90; // reading for full firing
g data;
----- Customise the above values --------
server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Nam
publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of ev
subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT comm
authMethod[] = "use-token-auth";// authentication method
token[] = TOKEN;
clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
---------------------------------------
lient wifiClient; // creating the instance for wificlient
bClient client(server, 1883, callback ,wifiClient); //calling the prede
```

```
Publish ok
Sensor RAW: 1001
3%
Sending payload: {"flamelevel":3.00,"msg":"chill"}
Publish ok
Sensor RAW: 1001
3%
Sending payload: {"flamelevel":3.00,"msg":"chill"}
Publish ok
Reconnecting client to
6hr21b.messaging.internetofthings.ibmcloud.com
```

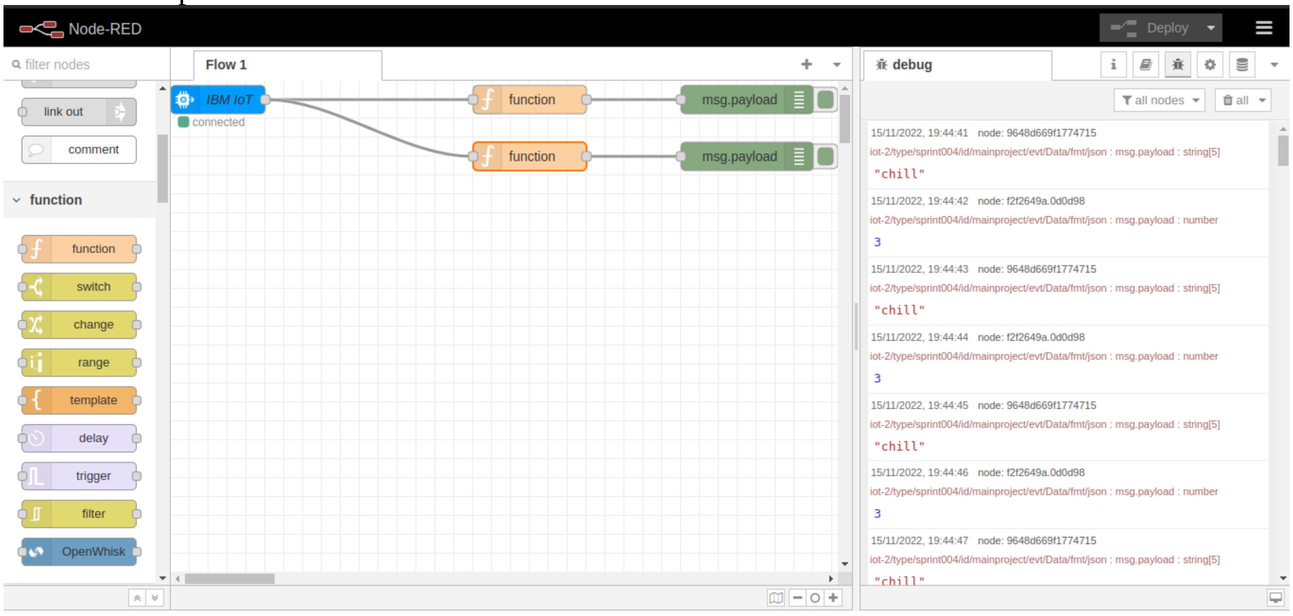
Watson output;



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

Event	Value	Format	Last Received
Data	{"flamelevel":3,"msg":"chill"}	json	a few seconds ago
Data	{"flamelevel":3,"msg":"chill"}	json	a few seconds ago
Data	{"flamelevel":3,"msg":"chill"}	json	a minute ago
Data	{"flamelevel":3,"msg":"chill"}	json	a minute ago
Data	{"flamelevel":3,"msg":"chill"}	json	a minute ago

Node red Output



```
15/11/2022, 19:44:41   node: 9648d669f1774715
iot-2/type/sprint004/id/mainproject/evt/Data/fmt/json : msg.payload : string[5]
"chill"
15/11/2022, 19:44:42   node: f2f2649a.0d0d98
iot-2/type/sprint004/id/mainproject/evt/Data/fmt/json : msg.payload : number
3
15/11/2022, 19:44:43   node: 9648d669f1774715
iot-2/type/sprint004/id/mainproject/evt/Data/fmt/json : msg.payload : string[5]
"chill"
15/11/2022, 19:44:44   node: f2f2649a.0d0d98
iot-2/type/sprint004/id/mainproject/evt/Data/fmt/json : msg.payload : number
3
15/11/2022, 19:44:45   node: 9648d669f1774715
iot-2/type/sprint004/id/mainproject/evt/Data/fmt/json : msg.payload : string[5]
"chill"
15/11/2022, 19:44:46   node: f2f2649a.0d0d98
iot-2/type/sprint004/id/mainproject/evt/Data/fmt/json : msg.payload : number
3
15/11/2022, 19:44:47   node: 9648d669f1774715
iot-2/type/sprint004/id/mainproject/evt/Data/fmt/json : msg.payload : string[5]
"chill"
```