

Sprint-3

Team ID : PNT2022TMID20931

Project Title : Industry-specific intelligent fire management system

Project Development

CODE:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h>
#include "DHT.h" // Library for dht11
#define DHTPIN 15 // what pin we're connected to
#define DHTTYPE DHT22
DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and typr of dht
connected

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

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

#define ORG "vg9s67" //IBM ORGANITION ID
#define DEVICE_TYPE "sprint003" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "spsprint003" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "1234567890" //Token
String data3;
float Humidity, Temp;

//----- 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() // configuring the ESP32
{
  Serial.begin(115200);
```

```

dht.begin();
delay(10);
Serial.println();
wificonnect();
mqttconnect();
}

void loop() // Recursive Function
{
Humidity = dht.readHumidity();
Temp = dht.readTemperature();
Serial.print("Temp:");
Serial.println(Temp);
Serial.print("Humidity:");
Serial.println(Humidity);
PublishData(Temp,Humidity);
delay(1000);
if (!client.loop()) {
mqttconnect();
}
}

/* .....retrieving to Cloud ..... */

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

```

Simulation:

The screenshot shows the Wokwi IoT simulator interface. On the left is a code editor for a file named 'sprint04.ino'. The code includes headers for WiFi, PubSubClient, and DHT, and defines constants for the DHT sensor pin and type. It also includes credentials for an IBM Watson IoT account. The main function 'void callback' is defined to handle incoming data. On the right is a 'Simulation' window showing a virtual hardware setup with an ESP32 microcontroller, a DHT22 temperature and humidity sensor, and a USB connection. Below the hardware diagram, the simulation output shows the device sending a JSON payload: {"Temp":-40.00,"Humidity":0.00}.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h>
3 #include "DHT.h" // Library for dht11
4 #define DHTPIN 15 // what pin we're connected to
5 #define DHTTYPE DHT22
6 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and t
7
8 void callback(char* subscribetopic, byte* payload, unsigned int payload
9
10 //-----credentials of IBM Accounts-----
11
12 #define ORG "vg9s67" //IBM ORGANITION ID
13 #define DEVICE_TYPE "sprint003" //Device type mentioned in ibm watson IoT
14 #define DEVICE_ID "spsprint003" //Device ID mentioned in ibm watson IoT
15 #define TOKEN "1234567890" //Token
16 String data3;
17 float Humidity, Temp;
18
19 //----- Customise the above values -----
20
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Serve
22 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type
23 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT
24 char authMethod[] = "use-token-auth"; // authentication method
25 char token[] = TOKEN;
26 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
27
28
29
```

Simulation output:

```
Humidity:0.00
Sending payload: {"Temp":-40.00,"Humidity":0.00}
Publish ok
Temp:-40.00
Humidity:0.00
Sending payload: {"Temp":-40.00,"Humidity":0.00}
Publish ok
```

ibm cloud connection:

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various IoT functions. The main content area displays a table of devices, with one device 'spsprint003' highlighted. Below the table, a detailed view of the device is shown, including its identity, device information, recent events, state, and logs. The device is currently 'Connected'.

Identity	Device Information	Recent Events	State	Logs
Device ID	spsprint003			
Device Type	sprint003			
Date Added	Nov 13, 2022 9:58 PM			
Added By	nspgipsy@gmail.com			
Connection Status	Connected Connection Time: Nov 13, 2022 9:59 PM Client Address: 50.31.197.64 Insecure			

output in ibm cloud:

IBM Watson IoT Platform

nsppjpsy@gmail.com
ID: vg9s67

Browse

Action

Device Types

Interfaces

Add Device +

Event	Value	Format	Last Received
Data	{"Temp":-40,"Humidity":0}	json	a few seconds ago
Data	{"Temp":-40,"Humidity":0}	json	a few seconds ago
Data	{"Temp":-40,"Humidity":0}	json	a few seconds ago
Data	{"Temp":-40,"Humidity":0}	json	a few seconds ago
Data	{"Temp":-40,"Humidity":0}	json	a few seconds ago

Items per page 50 | 1-2 of 2 items

1 of 1 page < 1 >