

Sprint-3

Project Development

Date	16 November 2022
Team ID	PNT22022TMID14391
Project Name	INDUSTRY - SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

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()// configureing 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 displays the Wokwi simulation interface. On the left, the code for `sprint04.ino` is visible, which includes libraries for WiFi, PubSubClient, and DHT. It defines an ESP32 as the device and configures it to connect to a specific WiFi network. The code also sets up an IBM Watson IoT client with predefined credentials and topics. A callback function is defined to process incoming data, and the main loop publishes temperature and humidity data to the IoT cloud.

On the right, the simulation shows an ESP32 microcontroller connected to a DHT22 digital temperature and humidity sensor. The simulation output window at the bottom shows the following sequence of events:

```

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

```

Browse Action Device Types Interfaces

Add Device +

Device ID	Status	Name	Type	Date Added	Action
spsprint003	Connected	sprint003	Device	Nov 13, 2022 9:58 PM	→ ...

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				

Items per page 50 | 1–2 of 2 items

1 of 1 page < 1 >

Browse				Add Device	
Action		Device Types		Interfaces	
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