

FINAL DELIVERY

Project Title:	Personal Assistant for Senior Citizen
Team ID:	PNT2022TMID50622

Code to subscribe topic in IBM watson:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
#include <LiquidCrystal_I2C.h>
#include "DHT.h" // Library for dht11
#define DHTPIN 15 // what pin we're connected to
#define DHTTYPE DHT11 // define type of sensor DHT 11
#define LED 2
DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of dht connected
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

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

#define ORG "1161vg" //IBM ORGANITION ID
#define DEVICE_TYPE "nodeMCU" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "?nUW@lkY)OglhHt)i6" //Token
String data3="";

//----- 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
LiquidCrystal_I2C lcd(0x27,16,2);

//-----
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();
    pinMode(LED,OUTPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{
    if (!client.loop()) {
        mqttconnect();
    }
}

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("Medicine Name: "+ data3);
    if(data3 != "")
    {
        lcd.init();
        lcd.print(data3);
    }
}

```

```

digitalWrite(LED,HIGH);
delay(20000);
digitalWrite(LED,LOW);
}

else
{
digitalWrite(LED,LOW);

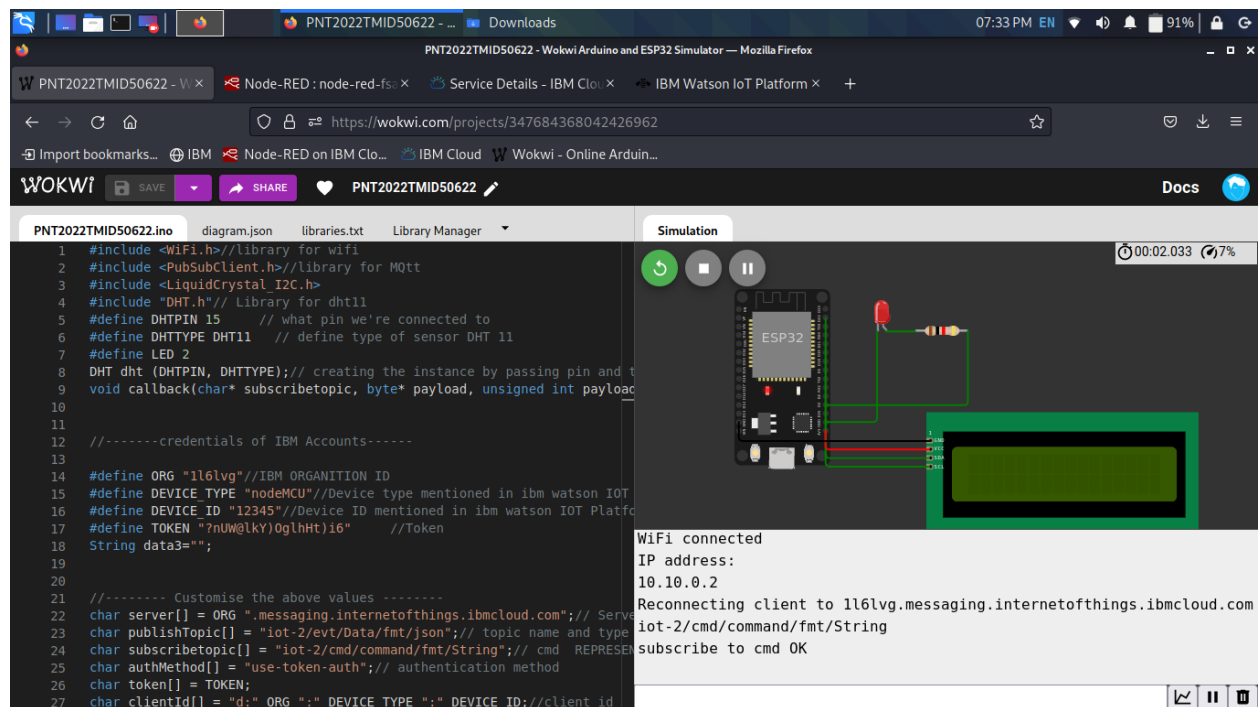
}
data3="";

}

```

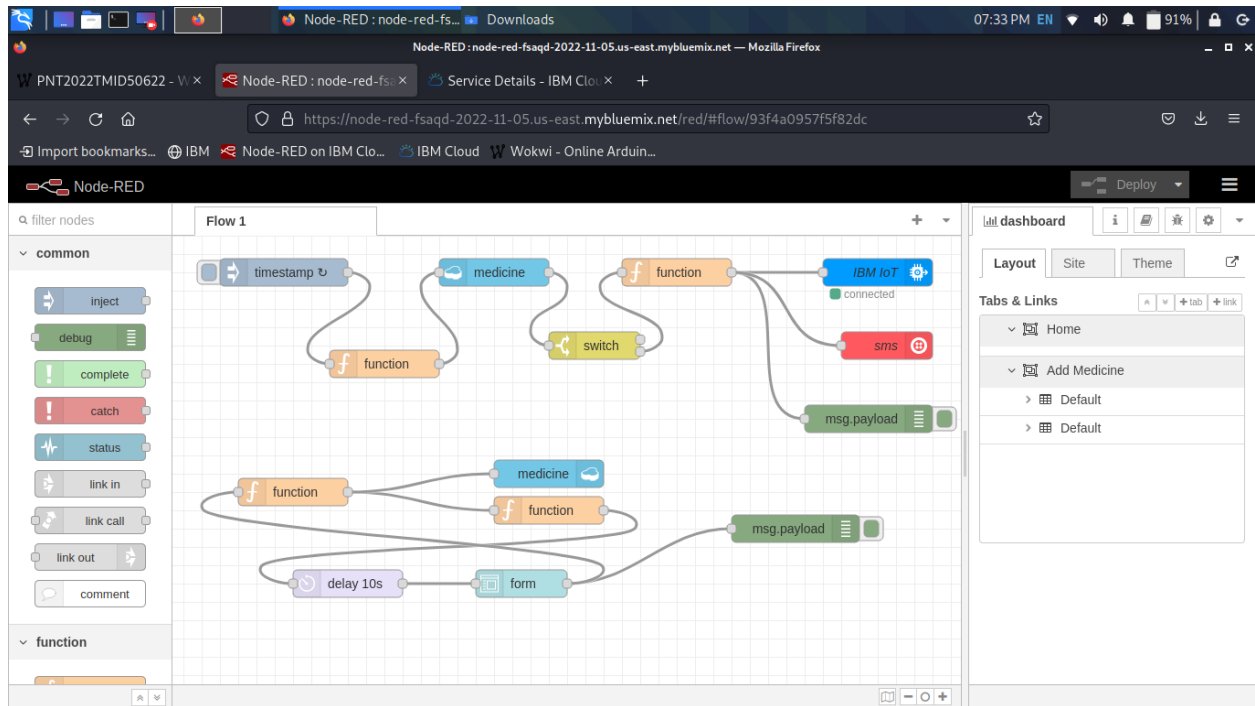
Simulation on Wokwi before medicine added :

Wokwi url: <https://wokwi.com/projects/347684368042426962>



Node-red configuration:

Node-red url: <https://node-red-fsaqd-2022-11-05.us-east.mybluemix.net/red/>



Added Medicine details in node-red form:

The screenshot displays the Node-RED web interface in a Mozilla Firefox browser, showing the 'Add Medicine' form. The form is titled 'Add Medicine' and has a 'Default' tab selected. The form contains the following fields:

- Medicine name: DEXORANGE 30
- Time: 07:36 pm
- Date: 16/11/2022

At the bottom of the form, there are two buttons: 'SUBMIT' and 'CANCEL'.

Simulation on wokwi after medicine added:

The screenshot shows the Wokwi online Arduino and ESP32 simulator. The code in the editor includes libraries for WiFi, MQTT, and DHT, and defines credentials for IBM Cloud IoT. The simulation output shows the device reconnecting to the IBM Cloud IoT platform and receiving a callback for the 'DEXORANGE 30' medicine.

```
1 #include <WiFi.h> // Library for wifi
2 #include <PubSubClient.h> // Library for MQTT
3 #include <LiquidCrystal_I2C.h>
4 #include "DHT.h" // Library for dht11
5 #define DHTPIN 15 // what pin we're connected to
6 #define DHTTYPE DHT11 // define type of sensor DHT 11
7 #define LED 2
8 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type
9 void callback(char* subscribtopic, byte* payload, unsigned int payloadLength) {
10
11
12 //-----credentials of IBM Accounts-----
13
14 #define ORG "1l6lvq" // IBM ORGANITION ID
15 #define DEVICE_TYPE "nodeMCU" // Device type mentioned in ibm watson IoT
16 #define DEVICE_ID "12345" // Device ID mentioned in ibm watson IoT Platform
17 #define TOKEN "?nUW@lky)OglhHT)i6" // Token
18 String data3="";
19
20
21 //----- Customise the above values -----
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server address
23 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type
24 char subscribtopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENTATION
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; // client id
```

Simulation output:

```
Reconnecting client to 1l6lvq.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK

callback invoked for topic: iot-2/cmd/command/fmt/String
Medicine Name: DEXORANGE 30
```

Medicine details added to the IBM cloud:

The screenshot shows the Cloudant Dashboard for the 'medicine' database. The details for the medicine 'DEXORANGE 30' are displayed, including its ID, revision, and name.

```
1 {
2   "_id": "2022-11-16 19:36",
3   "_rev": "1-66020b44317525c29505c6733a7517b1",
4   "name": "DEXORANGE 30"
5 }
```