

SPRINT 2

Date	13 th November 2022
Team ID	PNT2022TMID34110
Project Name	SmartFarmer –IoT Enabled Smart Farming Application

Develop a python script to publish random sensor data :

Program:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "6pjvs7"
deviceType = "Arshidevicetype"
deviceId = "Arshideviceid"
authMethod = "token"
authToken = "tGfGvVl-F2luRl2bsG"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
```

```

status=cmd.data['command']
if status=="lighton":
    print ("led is on")
else :
    print ("led is off")
#print(cmd)

try:
deviceOptions = {"org": organization, "type": deviceType, "id":
deviceId, "auth-method": authMethod, "auth-token":
authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

#.....

except Exception as e:
print("Caught exception connecting device: %s" % str(e))
sys.exit()

# Connect and send a datapoint "hello" with value "world" into
the cloud as an event of type "greeting" 10 times

deviceCli.connect()

while True:

    #Get Sensor Data from DHT11
    temp=random.randint(0,100)
    Humid=random.randint(0,100)
    data = { 'Temperature' : temp, 'Humidity': Humid }

```

```

#print data

def myOnPublishCallback():

    print ("Published Temperature = %s C" % temp,
"Humidity = %s %% " % Humid, "to IBM Watson")

    success = deviceCli.publishEvent("IoTSensor", "json",
data, qos=0, on_publish=myOnPublishCallback)

    if not success:

        print("Not connected to IoTf")

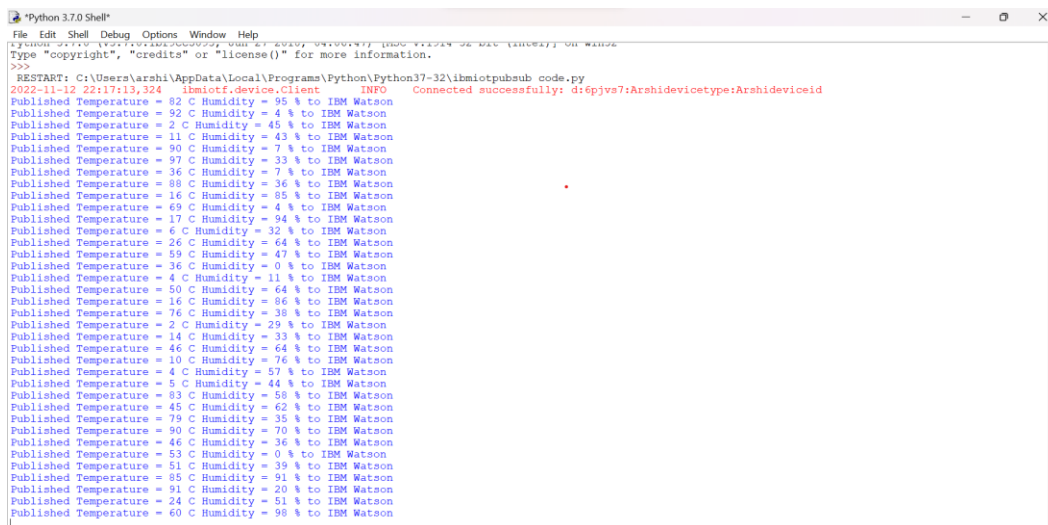
    time.sleep(1)

    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()

```

OUTPUT:

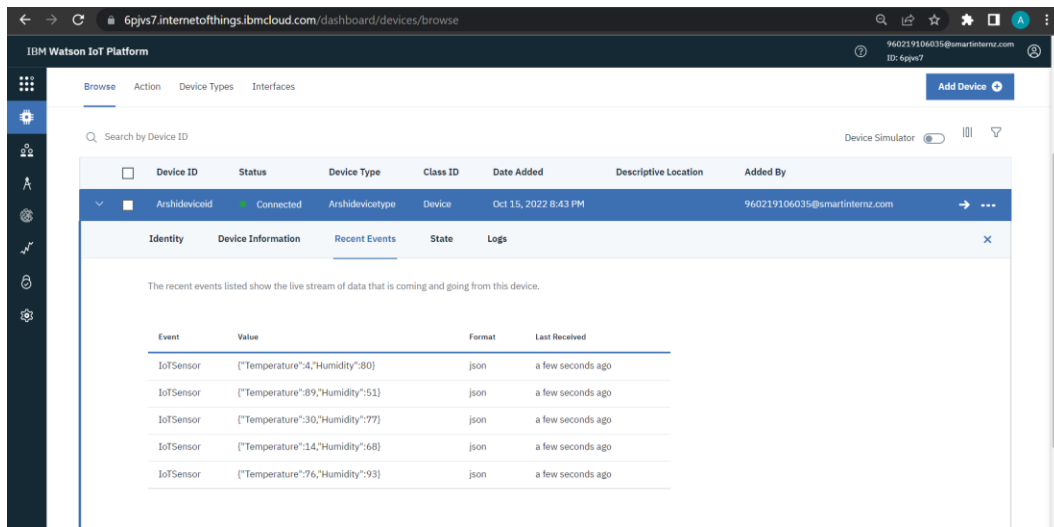


```

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>>
RESTART: C:\Users\arshi\AppData\Local\Programs\Python\Python37-32\ibmiotpubsub code.py
2022-11-12 22:17:13.324 ibmiotf.device.Client INFO Connected successfully: d16pjvs7Arshidevicetype:Arshideviceid
Published Temperature = 82 C Humidity = 95 % to IBM Watson
Published Temperature = 92 C Humidity = 4 % to IBM Watson
Published Temperature = 2 C Humidity = 45 % to IBM Watson
Published Temperature = 11 C Humidity = 43 % to IBM Watson
Published Temperature = 90 C Humidity = 7 % to IBM Watson
Published Temperature = 97 C Humidity = 33 % to IBM Watson
Published Temperature = 36 C Humidity = 7 % to IBM Watson
Published Temperature = 98 C Humidity = 36 % to IBM Watson
Published Temperature = 16 C Humidity = 85 % to IBM Watson
Published Temperature = 69 C Humidity = 4 % to IBM Watson
Published Temperature = 17 C Humidity = 94 % to IBM Watson
Published Temperature = 6 C Humidity = 32 % to IBM Watson
Published Temperature = 26 C Humidity = 64 % to IBM Watson
Published Temperature = 59 C Humidity = 47 % to IBM Watson
Published Temperature = 36 C Humidity = 0 % to IBM Watson
Published Temperature = 4 C Humidity = 11 % to IBM Watson
Published Temperature = 50 C Humidity = 64 % to IBM Watson
Published Temperature = 16 C Humidity = 86 % to IBM Watson
Published Temperature = 76 C Humidity = 38 % to IBM Watson
Published Temperature = 2 C Humidity = 29 % to IBM Watson
Published Temperature = 14 C Humidity = 33 % to IBM Watson
Published Temperature = 46 C Humidity = 64 % to IBM Watson
Published Temperature = 10 C Humidity = 76 % to IBM Watson
Published Temperature = 4 C Humidity = 57 % to IBM Watson
Published Temperature = 5 C Humidity = 44 % to IBM Watson
Published Temperature = 83 C Humidity = 58 % to IBM Watson
Published Temperature = 45 C Humidity = 62 % to IBM Watson
Published Temperature = 79 C Humidity = 35 % to IBM Watson
Published Temperature = 90 C Humidity = 70 % to IBM Watson
Published Temperature = 46 C Humidity = 36 % to IBM Watson
Published Temperature = 53 C Humidity = 0 % to IBM Watson
Published Temperature = 51 C Humidity = 39 % to IBM Watson
Published Temperature = 85 C Humidity = 91 % to IBM Watson
Published Temperature = 91 C Humidity = 20 % to IBM Watson
Published Temperature = 24 C Humidity = 51 % to IBM Watson
Published Temperature = 60 C Humidity = 98 % to IBM Watson

```

Publish data to IBM cloud



Connect the circuit with the IBM Cloudant API integration

Program:

```
#include <WiFi.h>//library for wifi
```

```
#include <PubSubClient.h>//library for MQTT
```

```
#include "DHT.h"// Library for dht11
```

```
#define DHTPIN 15 // what pin we're connected to
```

```
#define DHTTYPE DHT22 // define type of sensor DHT 11
```

```
#define LED 2
```

```
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 "6pjvs7"//IBM ORGANITION ID
#define DEVICE_TYPE "Arshidevicetype"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "Arshideviceid"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "tGfGvVl-F2luRl2bsG" //Token
String data3;
float h, t;

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

```

```

wificonnect();
mqttconnect();
}
void loop()// Recursive Function
{
  h = dht.readHumidity();
  t = dht.readTemperature();
  Serial.print("Temperature:");
  Serial.println(t);
  Serial.print("Humidity:");
  Serial.println(h);
  PublishData(t, h);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}
/*.....retrieving to Cloud.....*/
void PublishData(float temp, float humid) {
  mqttconnect();//function call for connecting to ibm
  /*
    creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Temperature\":";
  payload += temp;
  payload += "," "\"Humidity\":";
  payload += humid;
  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());
}

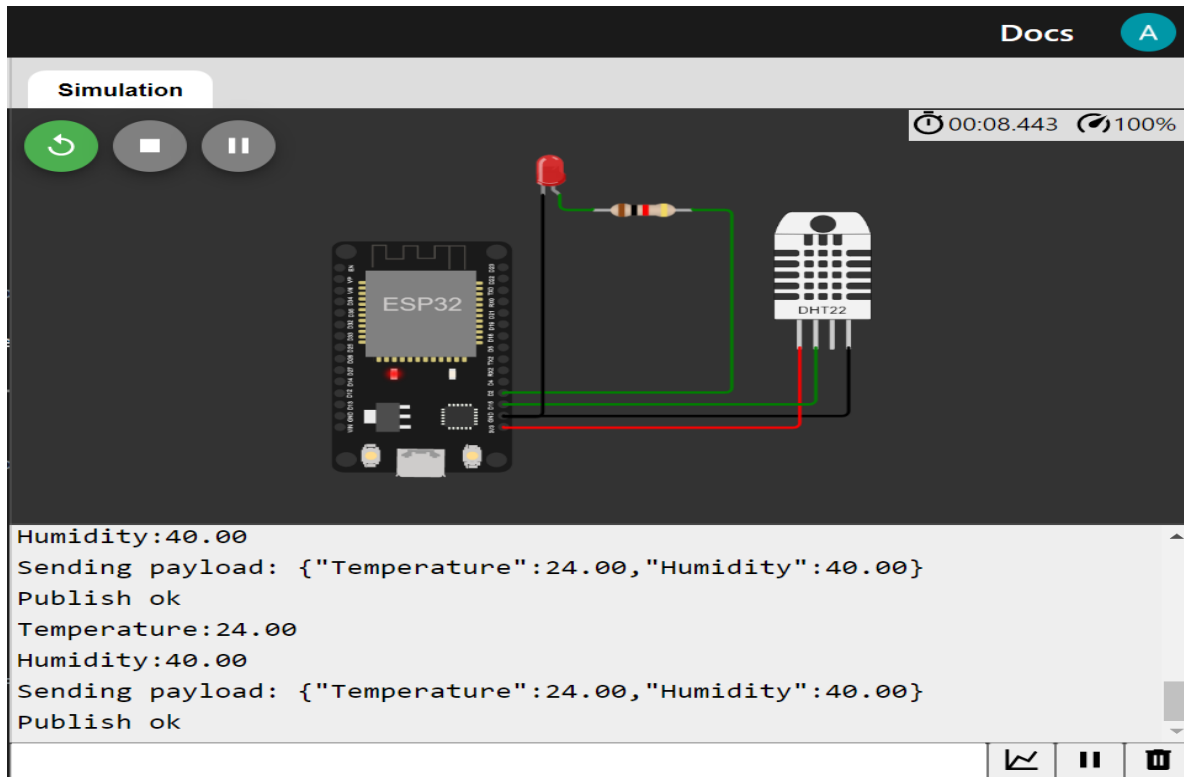
void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

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
  {
```



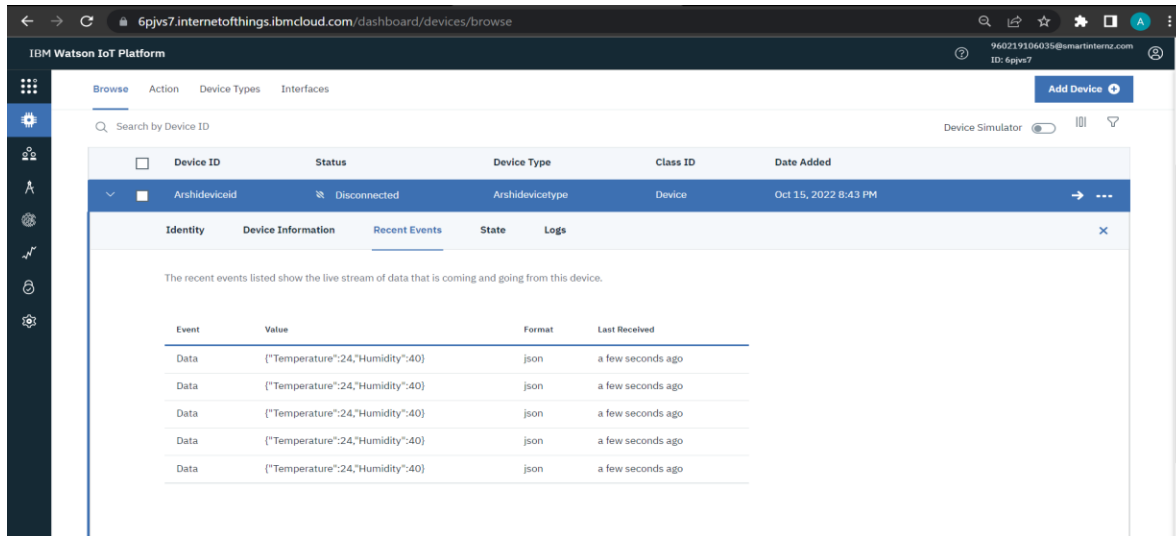
```
Serial.println(data3);  
digitalWrite(LED,LOW);  
}  
data3="";  
}
```

Circuit & Output:

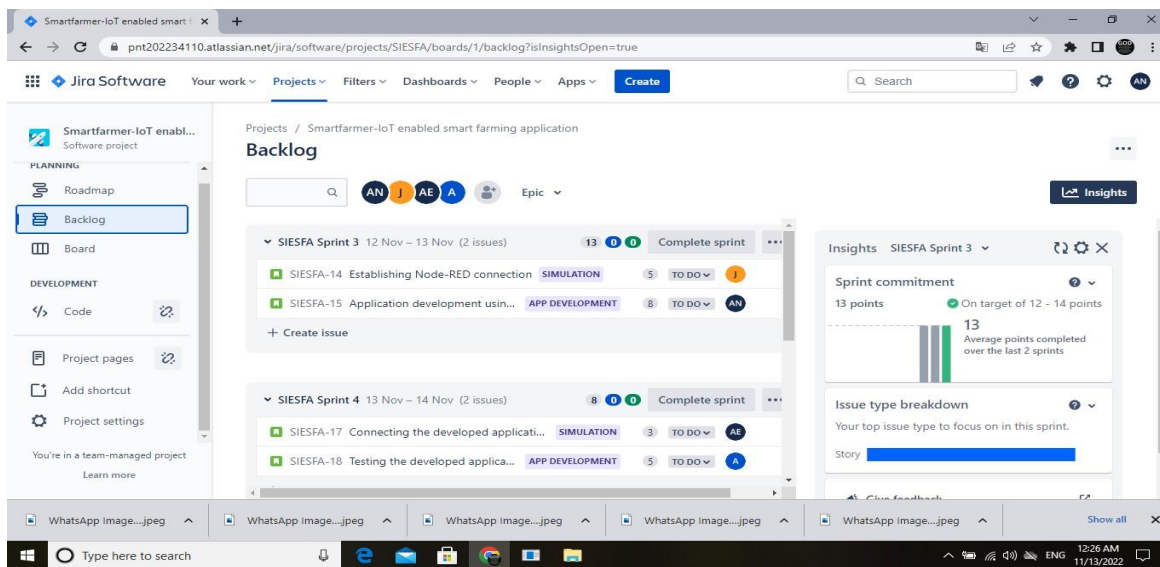


The simulation window displays a circuit with an ESP32 microcontroller, a red LED, and a DHT22 sensor. The LED is connected to a digital pin and a resistor. The DHT22 is connected to the ESP32. The simulation is running, and the console shows the output of the code.

Humidity:40.00
Sending payload: {"Temperature":24.00,"Humidity":40.00}
Publish ok
Temperature:24.00
Humidity:40.00
Sending payload: {"Temperature":24.00,"Humidity":40.00}
Publish ok



REPORTS:



SIESFA board - Agile board - Jira

Projects / Smartfarmer-IoT enabled smart farming application

SIESFA Sprint 2

0 days remaining [Complete sprint](#)

Search [] Filters: Epic Sprint 1 Clear filters GROUP BY: None Insights

TO DO

IN PROGRESS

DONE 3

Develop publish

SOFTWA

SIES

Publish

SOFTWA

SIES

Connect Cloudar

SIMULA

SIES

Insights SIESFA SPRINT 2

Sprint progress

100% done

Sprint burndown

Add estimates to manage and maintain scope. This insight helps you compare planned work against completed work, so you can track scope and pivot as needed. [Learn more](#)

Epic progress

This sprint is working towards 2 epics

SIESFA-10 Software 100% done

SIESFA-5 Simulation 100% done

Type here to search

9:23 PM 11/12/2022

Smartfarmer-IoT enabled smart

Projects / Smartfarmer-IoT enabled smart farming application

Roadmap

Give feedback Share Export

Search [] Status category Epic

	T	NOV	DEC	JAN
Sprints				
> SIESFA-5 Simulation				
> SIESFA-6 Login				
> SIESFA-10 Software				
> SIESFA-16 App development				
+ Create Epic				

Today Weeks Months Quarters

Type here to search

9:32 PM 11/12/2022