

SPRINT 1

SIMULATION CREATION

1. AN EXAMPLE FOR SIMULATION OF SENSOR AND TRANSFER THE DATA TO IBM IOT DEVICE:

The screenshot displays the WOKWI simulation interface. On the left, the 'sketch.ino' file is open, showing an Arduino sketch that configures an ESP32 to connect to an IBM IoT device and read an ultrasonic sensor. The sketch includes the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength)
4 #define ORG "k01pgl"
5 #define DEVICE_TYPE "ESP32"
6 #define DEVICE_ID "123456"
7 #define TOKEN "1234567890"
8 String data3;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/Data/fmt/json";
11 char subscribTopic[] = "iot-2/cmd/test/fmt/String";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
15 WiFiClient wifiClient;
16 PubSubClient client(server, 1883, callback, wifiClient);
17 const int trigPin = 5;
18 const int echoPin = 18;
19 #define SOUND_SPEED 0.034
20 long duration;
21 float distance;
22 void setup() {
23   Serial.begin(115200);
24   pinMode(trigPin, OUTPUT);
25 }
26
27
28 }
29 void loop()
```

On the right, the 'Simulation' window shows a visual representation of the ESP32 and the HC-SR04 ultrasonic sensor connected by wires. Below the visual, the console output displays the simulation's progress:

Connecting to
WiFi connected
IP address:
10.10.0.2
Reconnecting client to k01pgl.messaging.internetofthings.ibmcloud.com
iot-2/cmd/test/fmt/String
subscribe to cmd OK

Distance (cm): 249.00
Distance (cm): 248.95
Distance (cm): 248.93
Distance (cm): 248.93
Distance (cm): 248.93
Distance (cm): 248.93
Distance (cm): 248.93
Distance (cm): 248.93
Distance (cm): 19.98
ALERT!!
Sending payload: {"Distance":19.98,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 19.98
ALERT!!
Sending payload: {"Distance":19.98,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 19.98
ALERT!!
Sending payload: {"Distance":19.98,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 19.98
ALERT!!
Sending payload: {"Distance":19.98,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 121.97
Distance (cm): 121.97
Distance (cm): 121.97
Distance (cm): 121.97
Distance (cm): 121.97
Distance (cm): 121.97

RECEPTION OF DATA IN IBM IOT DEVICE:

IBM Watson IoT Platform

manu.1914125@gst.ac.in
ID: k0ipgl

Browse Action Device Types Interfaces

Add Device

Browse Devices

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

Device ID	Status	Device Type	Class ID	Date Added
123456	Disconnected	ESP32	Device	7 Nov 2022 18:40

Identity Device Information Recent Events State Logs

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

Event	Value	Format	Last Received
Data	{"Distance":19.98,"ALERT!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":19.98,"ALERT!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":19.98,"ALERT!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":19.98,"ALERT!":"Distance less than ...	json	a few seconds ago

Items per page 50 | 1-1 of 1 item

1 of 1 page

0 Simulations running

SOURCE CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
#define ORG "k0ipgl"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "123456"
#define TOKEN "1234567890"
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
```

```

char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;
void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  wificonnect();
  mqttconnect();
}
void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * SOUND_SPEED/2;
  Serial.print("Distance (cm): ");
  Serial.println(distance);
  if(distance<100)
  {
    Serial.println("ALERT!!");
    delay(1000);
    PublishData(distance);
    delay(1000);
    if (!client.loop()) {
      mqttconnect();
    }
  }
  delay(1000);
}
void PublishData(float dist) {
  mqttconnect();
  String payload = "{\"Distance\":\"";
  payload += dist;
  payload += "\",\"ALERT!!\":\"\"Distance less than 100cms\"";
  payload += "\"}";
  Serial.print("Sending payload: ");
  Serial.println(payload);
  if (client.publish(publishTopic, (char*) payload.c_str())) {

```

```

Serial.println("Publish ok");
} 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()
{
Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);
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++)
{
data3 += (char)payload[i];
}
Serial.println("data: "+ data3);
}

```

```
data3="";  
}
```

2. SIMULATION OF SENSOR DATA FOR PROJECT:

The screenshot displays the IBM Watson IoT Platform interface. The main section is titled "Browse Devices" and shows a table of devices. A device with ID 123456, status "Disconnected", and type "ESP32" is selected. A modal window titled "Device Type: ESP32" is open, showing the configuration for a new event type named "event_1". The event is scheduled to occur every minute. The payload is a JSON object with three fields: "temperature", "humidity", and "objectTemp", each with a random value between 0 and 100. Below the modal, the "Recent Events" tab is selected, showing a list of events received from the device. The events are listed in a table with columns for Event, Value, Format, and Last Received. The events are all of type "event_1" and contain random sensor data.

Event	Value	Format	Last Received
event_1	{"temperature":45,"humidity":3,"objectTemp":61}	json	a few seconds ago
event_1	{"temperature":3,"humidity":51,"objectTemp":77}	json	a few seconds ago
event_1	{"temperature":52,"humidity":57,"objectTemp":4...	json	a few seconds ago
event_1	{"temperature":37,"humidity":39,"objectTemp":8...	json	a few seconds ago
event_1	{"temperature":59,"humidity":1,"objectTemp":37}	json	a few seconds ago

PAYLOAD CODE:

```
{  
  "temperature": random(0,100),  
  "humidity": random(0, 100),  
  "objectTemp": random(0, 100)  
}
```

3. PYTHON CODE TO CONTROL OF MOTOR THROUGH DATA SNT BY IBM IOT DEVICE:

```
import time
import sys
import ibmiotf.application # to install pip install ibmiotf
import ibmiotf.device

#Provide your IBM Watson Device Credentials
organization = "k0ipgl" #replace the ORG ID
deviceType = "ESP32"#replace the Device type wi
deviceId = "123456"#replace Device ID
authMethod = "token"
authToken = "1234567890" #Replace the authtoken

def myCommandCallback(cmd): # function for Callback
    print("Command received: %s" % cmd.data)
    if cmd.data['command']=='motoron':
        print("Motor On IS RECEIVED")

    elif cmd.data['command']=='motoroff':
        print("Motor Off IS RECEIVED")

    if cmd.command == "setInterval":

        if 'interval' not in cmd.data:
            print("Error - command is missing required info
rmation: 'interval'")
        else:
            interval = cmd.data['interval']
    elif cmd.command == "print":
        if 'message' not in cmd.data:
            print("Error - command is missing required info
rmation: 'message'")
        else:
            output=cmd.data['message']
            print(output)

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": devic
eId, "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:

    deviceCli.commandCallback = myCommandCallback

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