

IBM ASSIGNMENT – 4

Assignment Date	06 NOVEMBER 2022
Student Name	SAKTHI KIRAN M B
Student Roll Number	710019106036
Team ID	PNT2022TMID42272

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cms send "alert" to ibm cloud and display in device recent events.

CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "26i696"//IBM ORGANITION ID
#define DEVICE_TYPE "Smart_Waste"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "036"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "Sakthi@2012" //Token
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("");

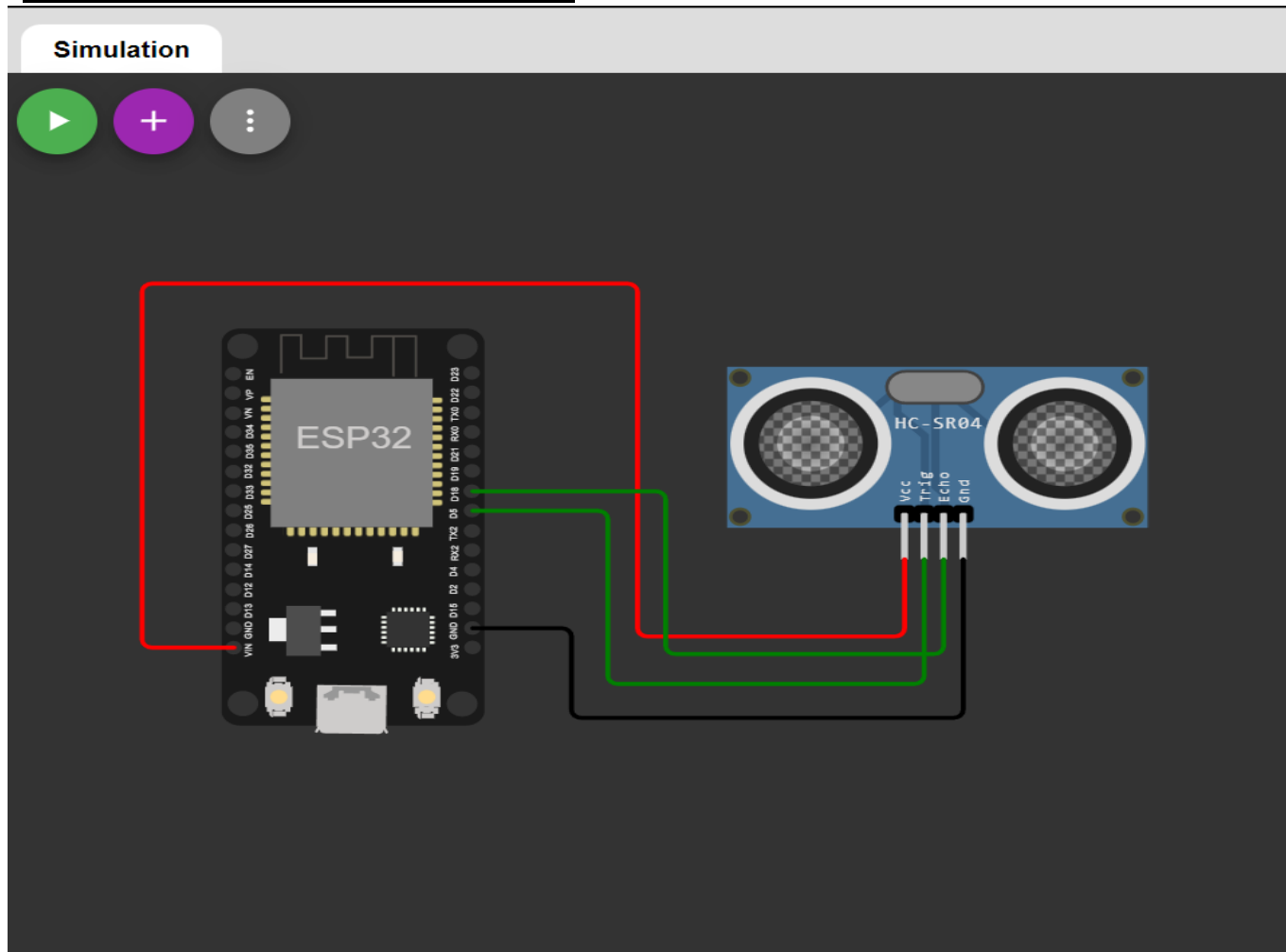
```

```

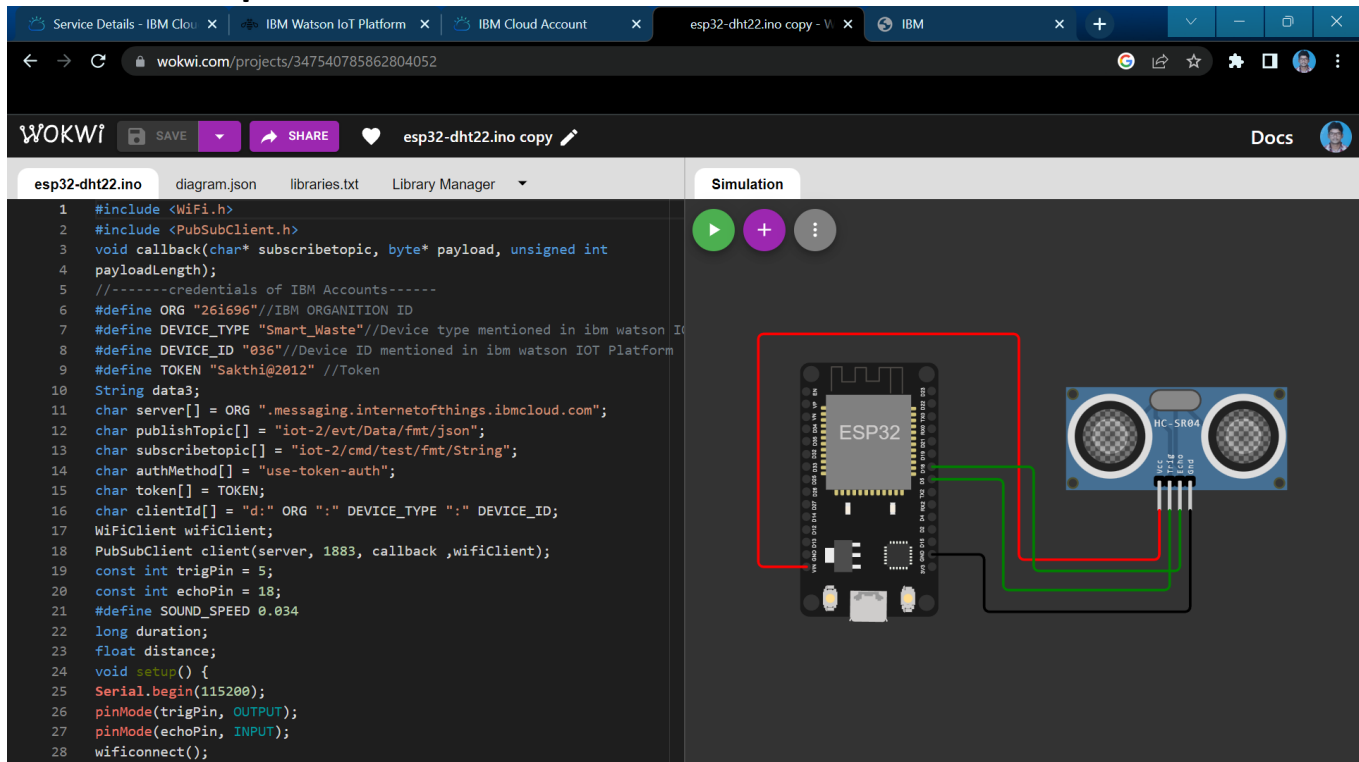
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
void initManagedDevice() {
if (client.subscribe(subscribetopic)) {
Serial.println((subscribetopic));
Serial.println("subscribe to cmd OK");
} else {
Serial.println("subscribe to cmd FAILED");
}
}
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
Serial.print("callback invoked for topic: ");
Serial.println(subscribetopic);
for (int i = 0; i < payloadLength; i++) {
//Serial.print((char)payload[i]);
data3 += (char)payload[i];
}
Serial.println("data: "+ data3);
data3="";
}

```

SCHEMATIC/CIRCUIT DIAGRAM:



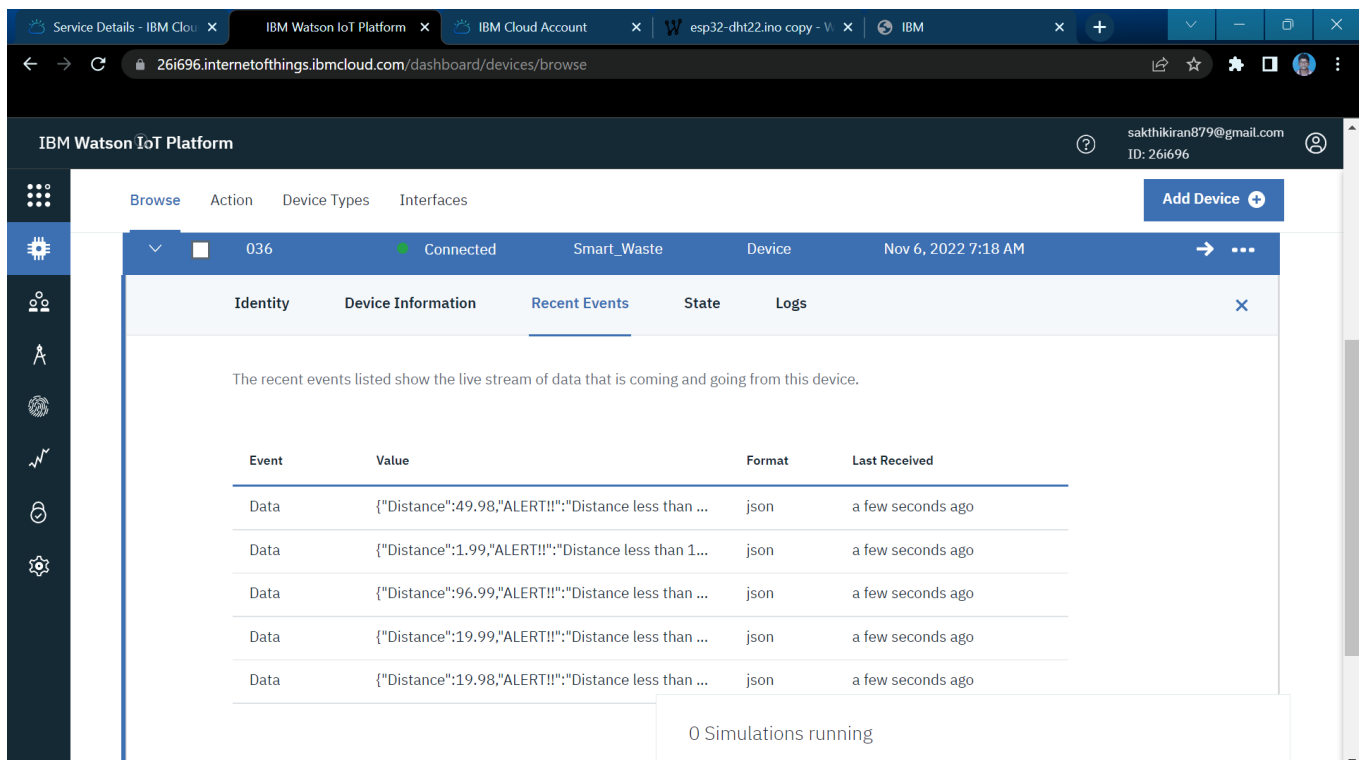
WOKWI Output:



The screenshot shows the Wokwi IDE interface. On the left, the code for `esp32-dht22.ino` is displayed. The code includes headers for `WiFi.h` and `PubSubClient.h`, and defines constants for the IBM Cloud organization ID, device type, device ID, and token. It sets up a `WiFiClient` and a `PubSubClient` to connect to the IBM Cloud IoT Platform. The `setup` function initializes the serial port and pins. The `loop` function (partially visible) would handle the data transmission.

On the right, the simulation shows an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The sensor is connected to the ESP32 via I2C or UART pins, as indicated by the wiring diagram.

IBM CLOUD OUTPUT:



The screenshot shows the IBM Watson IoT Platform dashboard. The device `036` is listed under the `Smart_Waste` device type. The device is connected and its details are shown. The `Recent Events` tab is selected, displaying a list of events received from the device.

Event	Value	Format	Last Received
Data	{"Distance":49.98,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":1.99,"ALERT!!":"Distance less than 1...	json	a few seconds ago
Data	{"Distance":96.99,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":19.99,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":19.98,"ALERT!!":"Distance less than ...	json	a few seconds ago

0 Simulations running

WOKWI LINK:

<https://wokwi.com/projects/347540785862804052>