IBM Nalaiyathiran

Assignment-4

Question:

Write code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events. Upload document with wokwi share link and images of IBM cloud

Program:

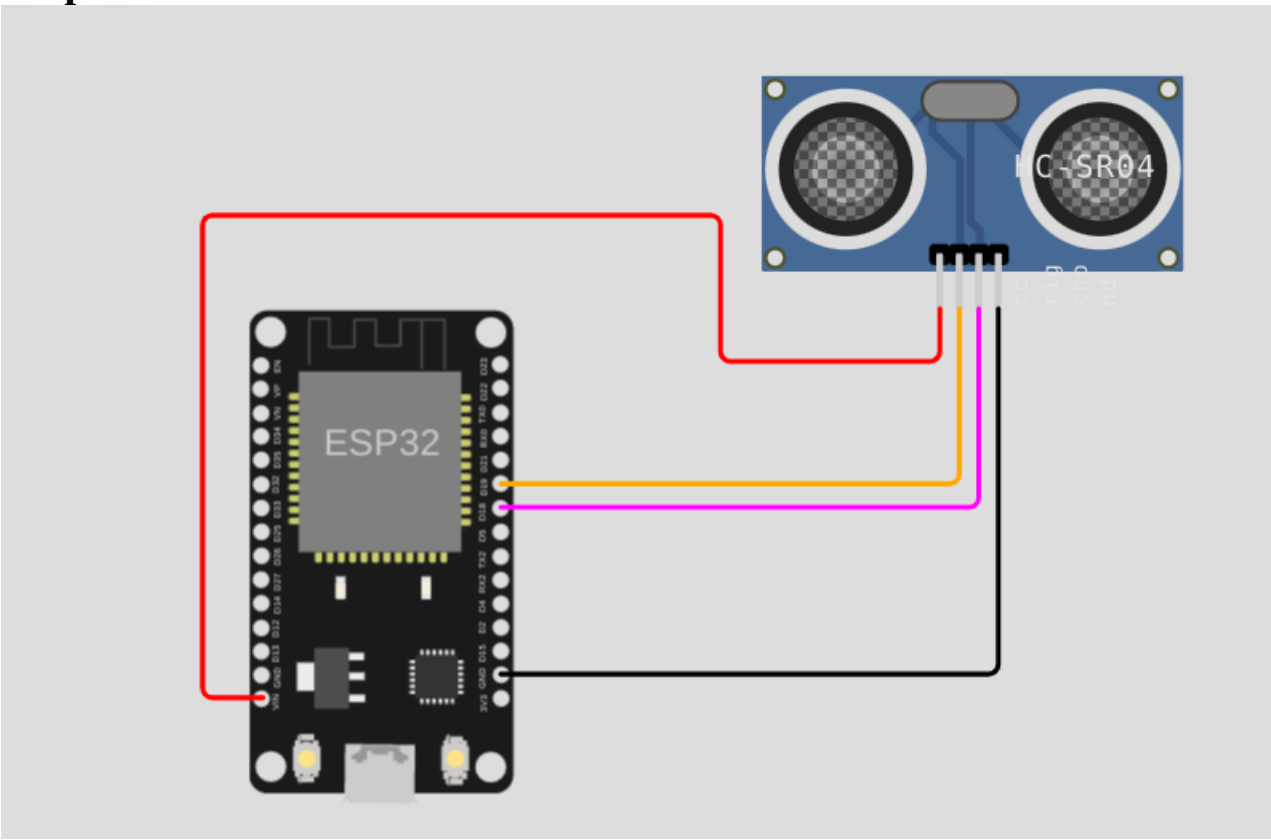
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15 // what pin we're connected to
#define DHTTYPE DHT22 // define type of sensor DHT 11
#define LED 2

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 "gcxa4l"//IBM ORGANITION ID
#define DEVICE_TYPE "Type1"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //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
```

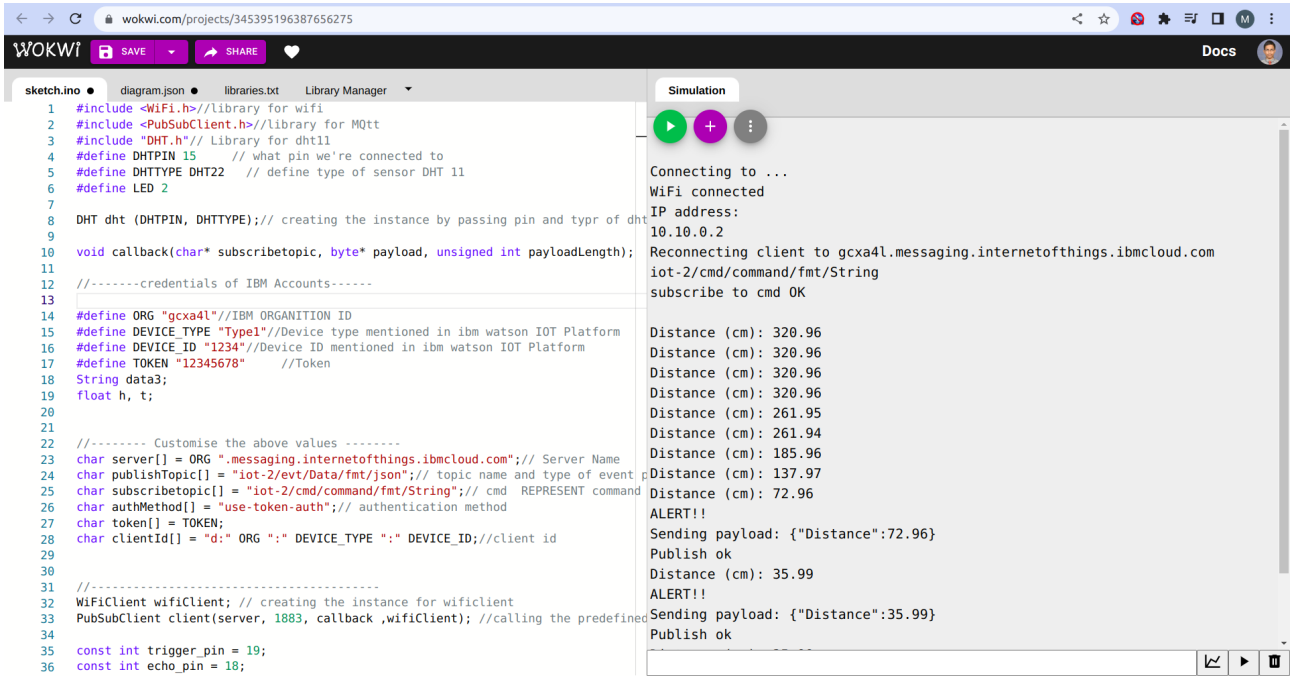
```
//---------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined
client id by passing parameter like server id,portand wificredential

const int trigger_pin = 19;
const int echo_pin = 18;
const float SOUND_SPEED = 0.034;

void setup()// configureing the ESP32
{
Serial.begin(115200);
pinMode(trigger_pin,OUTPUT);
pinMode(echo_pin, INPUT);
delay(10);
Serial.println();
wificonnect();
mqttconnect();
}

void loop()// Recursive Function
{
digitalWrite(trigger_pin, LOW);
delayMicroseconds(2);
digitalWrite(trigger_pin, HIGH);
delayMicroseconds(10);
digitalWrite(trigger_pin, LOW);
float duration = pulseIn(echo_pin, HIGH);
float 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);
}



/*.....retrieving to Cloud.....*/

void PublishData(float distance) {
mqttconnect();//function call for connecting to ibm
/*
creating the String in in form JSon to update the data to ibm cloud
```

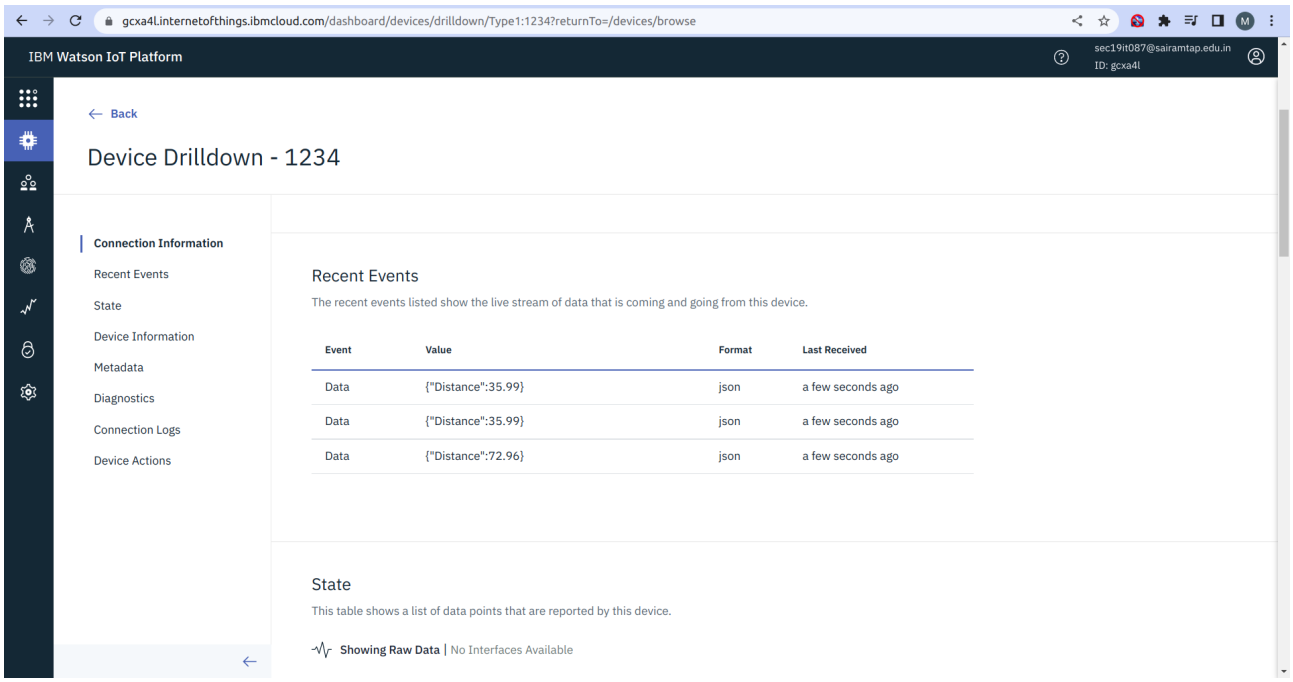
```
*/
String payload = "{\"Distance\":";
payload += distance;
payload += "}";

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");
```

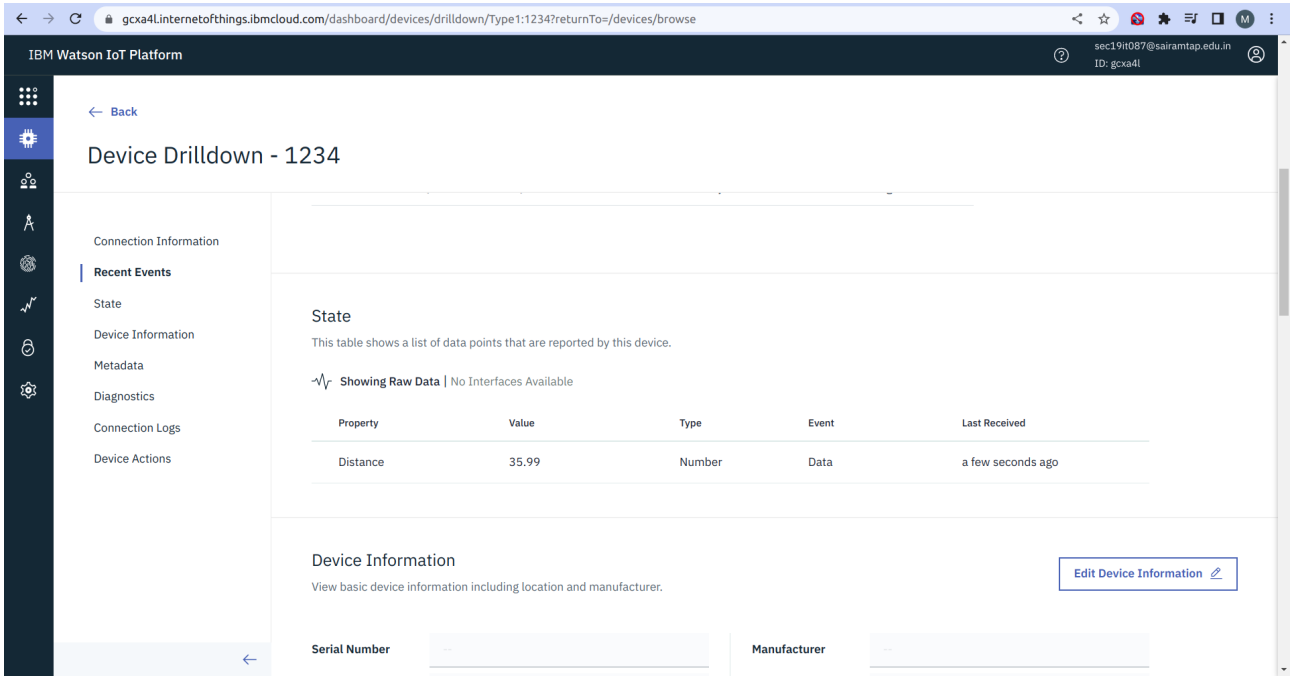
```
}
}

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="";
}
```

Output:




wokwi.com/projects/345395196387656275
WOKWI
SAVE
SHARE
Docs
sketch.ino
diagram.json
libraries.txt
Library Manager
#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
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
//-------credentials of IBM Accounts------
#define ORG "gcxa4l"//IBM ORGANITION ID
#define DEVICE_TYPE "Type1"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //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
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command
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
const int trigger_pin = 19;
const int echo_pin = 18;
Simulation
Connecting to ...
WiFi connected
IP address:
10.10.0.2
Reconnecting client to gcxa4l.messaging.internetofthings.ibmcloud.com
iot-2/cmd/command/fmt/String
subscribe to cmd OK
Distance (cm): 320.96
Distance (cm): 320.96
Distance (cm): 320.96
Distance (cm): 320.96
Distance (cm): 261.95
Distance (cm): 261.94
Distance (cm): 185.96
Distance (cm): 137.97
Distance (cm): 72.96
ALERT!!
Sending payload: {"Distance":72.96}
Publish ok
Distance (cm): 35.99
ALERT!!
Sending payload: {"Distance":35.99}
Publish ok


gcxa4l.internetofthings.ibmcloud.com/dashboard/devices/drilldown/Type1:1234?returnTo=/devices/browse
IBM Watson IoT Platform
sec19it087@sairamtap.edu.in
ID: gcxa4l
Back
Device Drilldown - 1234
Connection Information
Recent Events
State
Device Information
Metadata
Diagnostics
Connection Logs
Device Actions
Recent Events
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":35.99}
json
a few seconds ago
Data
{"Distance":35.99}
json
a few seconds ago
Data
{"Distance":72.96}
json
a few seconds ago
State
This table shows a list of data points that are reported by this device.
Showing Raw Data | No Interfaces Available


gcxa4l.internetofthings.ibmcloud.com/dashboard/devices/drilldown/Type1:1234?returnTo=/devices/browse
IBM Watson IoT Platform
sec19it087@sairamtap.edu.in
ID: gcxa4l
Back
Device Drilldown - 1234
Connection Information
Recent Events
State
Device Information
Metadata
Diagnostics
Connection Logs
Device Actions
State
This table shows a list of data points that are reported by this device.
Showing Raw Data | No Interfaces Available
Property
Value
Type
Event
Last Received
Distance
35.99
Number
Data
a few seconds ago
Device Information
View basic device information including location and manufacturer.
Edit Device Information
Serial Number
Manufacturer