**NAAN MUDHALVAN-IBM**

**ASSIGNMENT 3**

**TRACK -** IOT

**NAME –** DHILLI BABU

**COLLEGE -**MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE

**TOPIC-** Build wowki product, use ultrasonic sensor and detect the

distance from the object. Whenever distance is less than

100cms upload the value to the ibm cloud.in recent device

events upload the data from wokwi.

**Wokwi** [**-**](file:///C:\Users\sivaprakash\Desktop\ibm%20work\-%20https:\wokwi.com\projects\364172743294124033)**https://wokwi.com/projects/364763321030910977**

**PROGRAM:**

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

#include <PubSubClient.h>//library for MQtt

#define ECHO\_PIN 5

#define TRIG\_PIN 4

const int LED = 13;

void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength);

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

#define ORG "etuanl"//IBM ORGANITION ID

#define DEVICE\_TYPE "acbd"//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 distance;

//-------- 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);

  pinMode(LED, OUTPUT);

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

  wificonnect();

  mqttconnect();

}

float readDistanceCM() {

  digitalWrite(TRIG\_PIN, LOW);

  delayMicroseconds(2);

  digitalWrite(TRIG\_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG\_PIN, LOW);

  int duration = pulseIn(ECHO\_PIN, HIGH);

  return duration \* 0.034 / 2;

}

void loop()// Recursive Function

{

 float distance = readDistanceCM();

  bool isNearby = distance < 100;

  digitalWrite(LED, isNearby);

**Serial**.print("Measured distance: ");

**Serial**.println(readDistanceCM());

  delay(100);

  if(distance<=100.00)

  {

  PublishData(distance);

  }

  delay(1000);

  if (!client.loop()) {

    mqttconnect();

  }

}

/\*.....................................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);

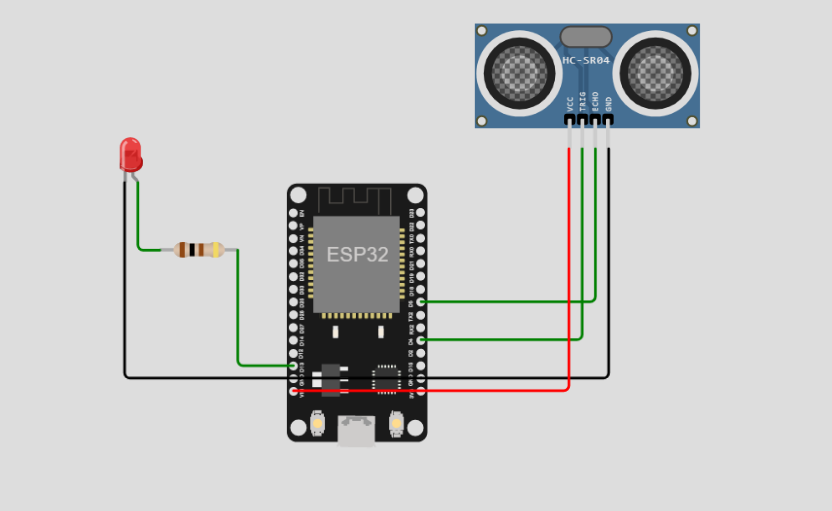
digitalWrite(LED,LOW);

  }

data3="";

}

**SIMULATION:**



**SIMULATION OUTPUT :**

Measured distance: 88.98

Sending payload: {"distance":88.96}

Publish ok

Measured distance: 88.98

Sending payload: {"distance":88.96}

Publish ok

Measured distance: 88.98

Sending payload: {"distance":88.96}

Publish ok

Measured distance: 88.98

Sending payload: {"distance":88.96}

Publish ok

Measured distance: 89.03

Sending payload: {"distance":88.96}

Publish ok

Measured distance: 193.95

Measured distance: 193.95

Measured distance: 4.98

Sending payload: {"distance":4.98}

Publish ok

Measured distance: 4.98

Sending payload: {"distance":4.98}

Publish ok

Measured distance: 4.98

Sending payload: {"distance":4.98}

Publish ok

Measured distance: 4.98

Sending payload: {"distance":4.98}

Publish ok

Measured distance: 399.99

**IBM CLOUD:**

