**Assignment-4**

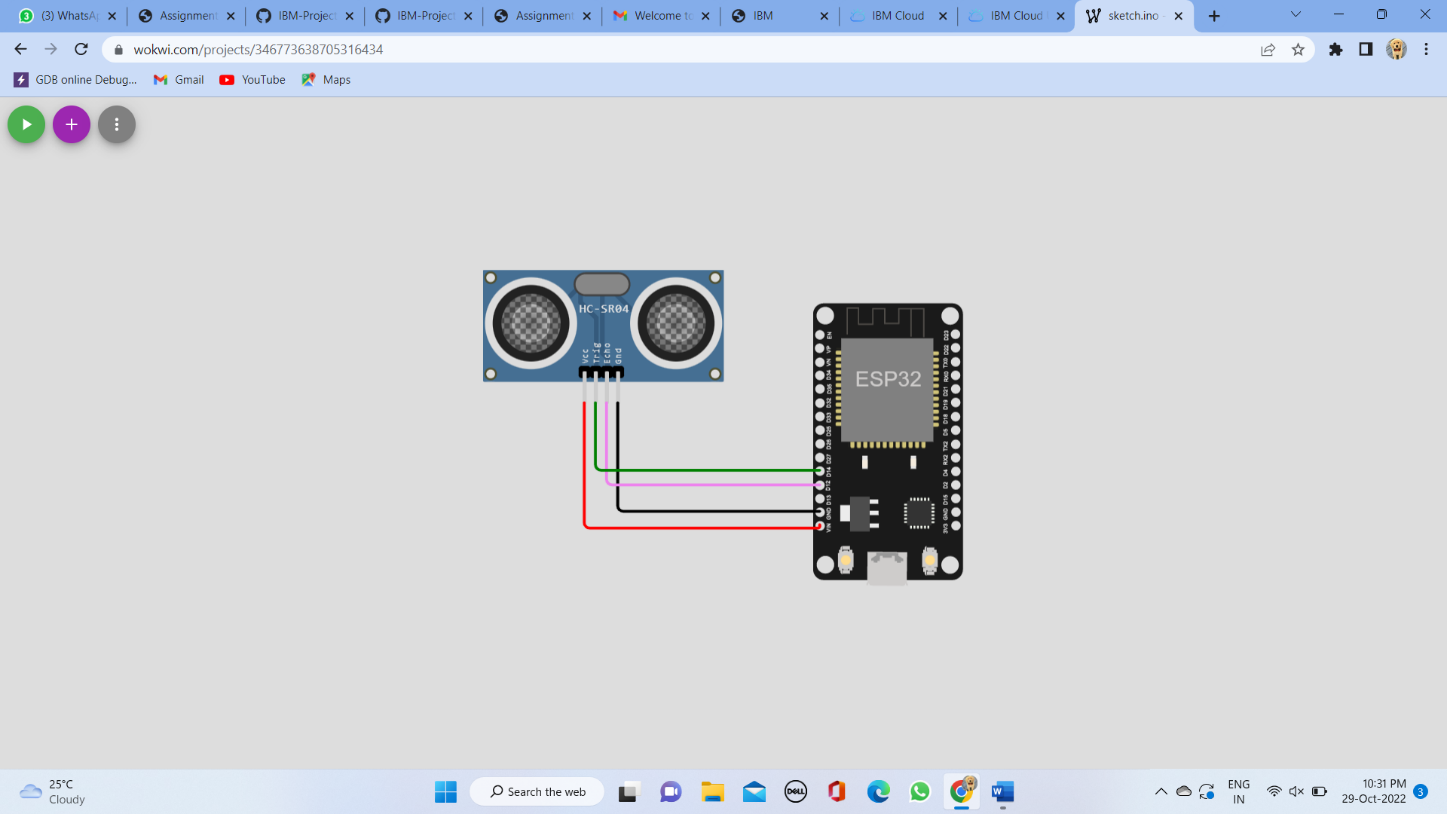
|  |  |
| --- | --- |
| **Date** | **28 October 2022** |
| **Team ID** | PNT2022TMID23797 |
| **Project Name** | Real time river water quality monitoring and Control System |
| **Maximum Marks** | **4 Marks** |

**Question:**

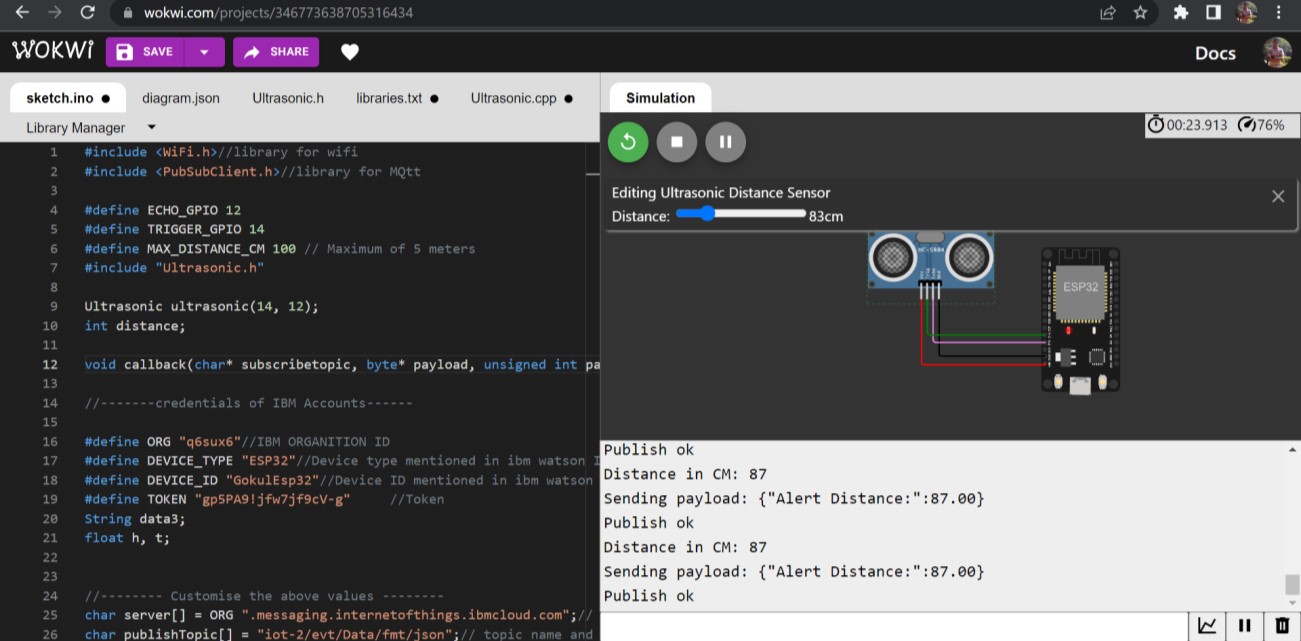
**Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cms send "alert" to ibm cloud and display in device recent events.**

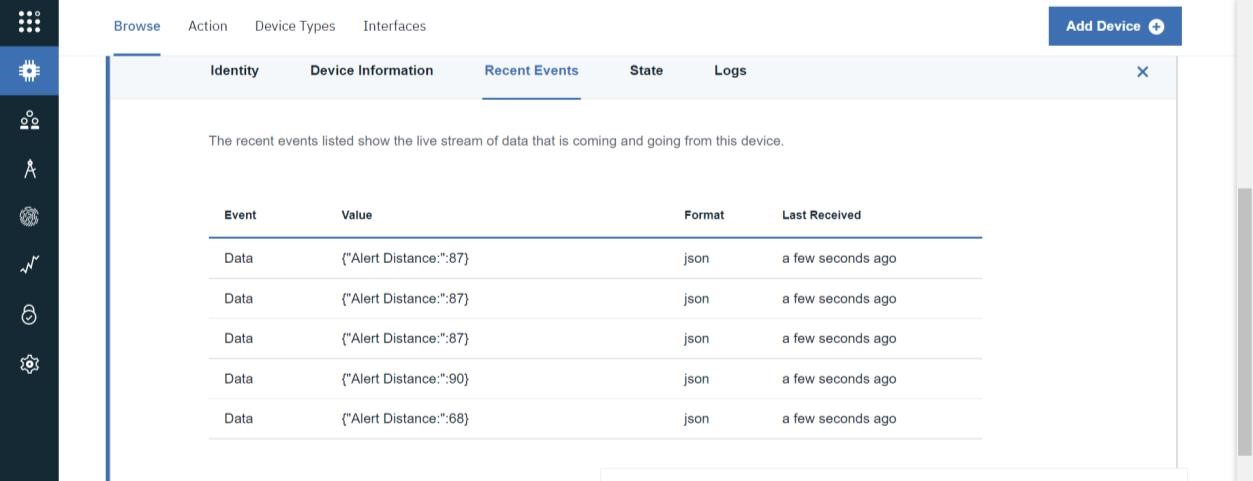
**CODE:**

|  |
| --- |
| #include <WiFi.h>//library for wifi  #include <PubSubClient.h>//library for MQtt    #define ECHO\_GPIO 12  #define TRIGGER\_GPIO 14  #define MAX\_DISTANCE\_CM 100 // Maximum of 5 meters  #include "Ultrasonic.h" Ultrasonic ultrasonic(14, 12); int distance;  void callback(char\* subscribetopic, byte\* payload, unsigned int payloadLength);    //-------credentials of IBM Accounts------    #define ORG "q6sux6"//IBM ORGANITION ID  #define DEVICE\_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform  #define DEVICE\_ID "Devadharshinim11"//Device ID mentioned in ibm watson IOT Platform  #define TOKEN "gp5PA9!jfw7jf9cV-g" //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); delay(10); **Serial**.println(); wificonnect(); mqttconnect();  } void loop()// Recursive Function  { distance = ultrasonic.read(CM); if(distance < 100){  **Serial**.print("Distance in CM: ");  **Serial**.println(distance); PublishData(distance); delay(1000); if (!client.loop()) { mqttconnect();  }  } delay(1000);      }    /\*.....................................retrieving to  Cloud...............................\*/  void PublishData(float temp) { mqttconnect();//function call for connecting to ibm  /\*  creating the String in in form JSon to update the data to ibm cloud \*/  String payload = "{\"Alert Distance:\":"; payload += temp; 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++) { data3 += (char)payload[i];  }  **Serial**.println("data: "+ data3); if(data3=="lighton") {  **Serial**.println(data3);  } else  {  **Serial**.println(data3);  }  data3="";  } |

**CIRCUIT DIAGRAM:**

**OUTPUT:**





**Project Link:**

https://wokwi.com/projects/346773638705316434