

Smart Farmer -IoT Enabled Smart Farming Application

ASSIGNMENT – 4

NAME: Arun Kumar S

Write Code and connections in wokwi for ultrasonic sensor. Whatever distance is less than 100 cm send “Alert” to ibm cloud and display in devicerecent events.

Solution:

```
//Pins
const int TRIG_PIN = 7 ;const int ECHO_PIN = 8;

//Anything over 400 cm (23200 us pulse) is "out of range"const unsigned int
MAX_DIST = 23200;

void setup() {

// The Trigger pin will tell the sensor to range find
Pin Mode(TRIG_PIN, OUTPUT);
digital Write(TRIG_PIN, LOW);

//Set Echo pin as input to measure the duration of
//pulses coming back from the distance sensor
pinMode(ECHO_PIN, INPUT ) ;

// We'll use the serial monitor to view the sensor output
Serial.begin(9600);

}
void loop()
{
  unsigned long t1;
```

```
unsigned long t2;
unsigned long
pulse_width;float cm;
float inches;
// Hold the trigger pin high for at least 10 us
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);

// Wait for pulse on echo pin
while (digitalRead( ECHO_PIN )==0 );

// Measure how long the echo pin was held high (pulse width)
// Note: the micros() counter will overflow after-70
mint1= micros ();
while (digitalRead(ECHO_PIN) == 1);t2= micros ();
pulse_width = t2-t1;

// Calculate distance in centimeters and inches. The constants
//are found in the datasheet, and calculated from the assumed speed
// of sound in air at sea level (- 340m/s)
cm=pulse_Width / 58 ;
inches = pulse_width/148.0;
```

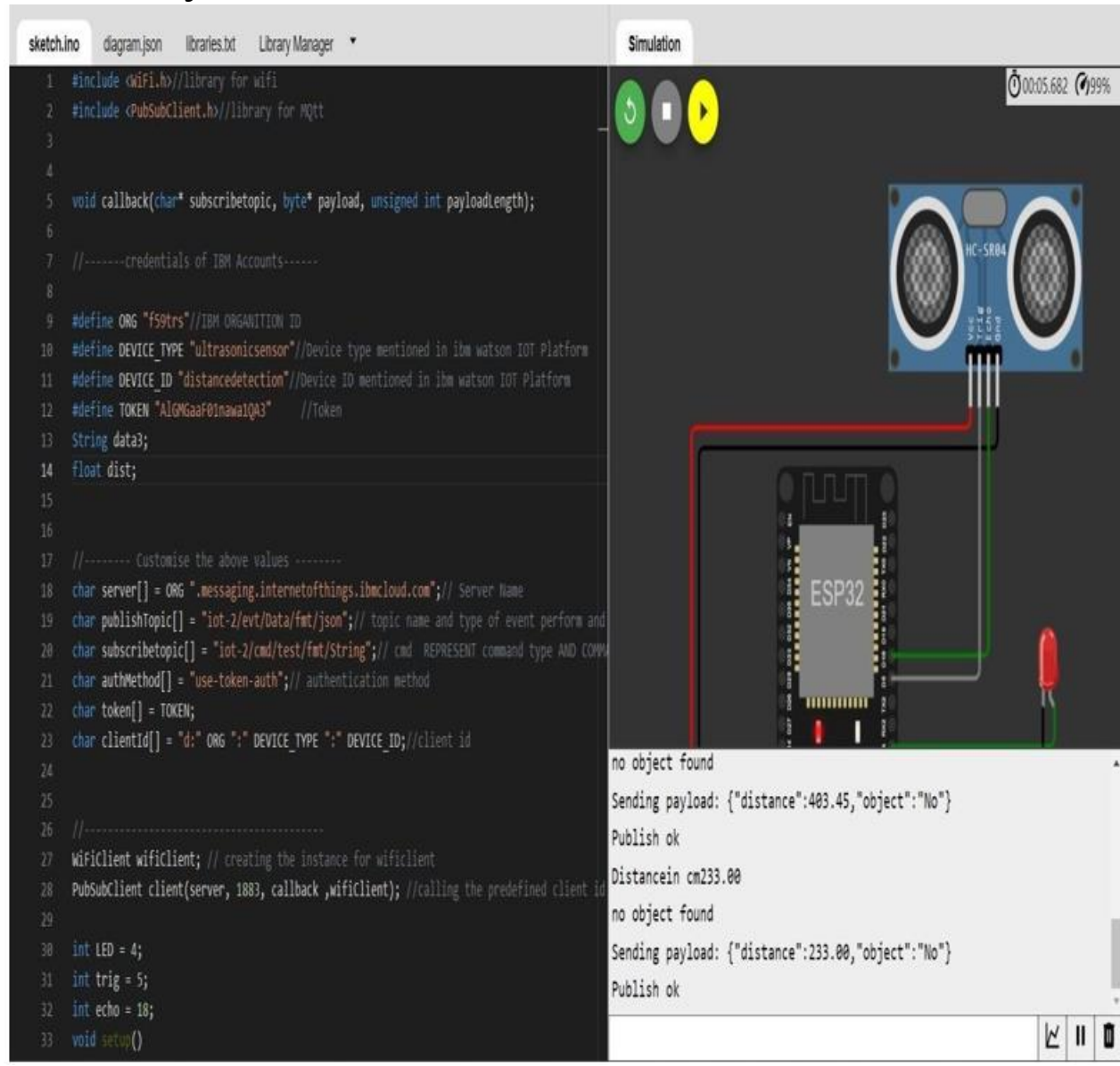
```
// Print out results
if (pulse_width > MAX_DIST
){Serial.println("Out of
range");
} else {
Serial.println("*****");
Serial.print("The Measured Distance in cm: ");
Serial.println(cm);

if( cm < 100 ){
    //while(true){
        Serial.println("Alert!!");
        //}
    }
Serial.print("*****");
}

//wait at least 1000ms before next
measurementDelay(1000);
}
```

OUTPUT :

When object is not near to the ultrasonic sensor



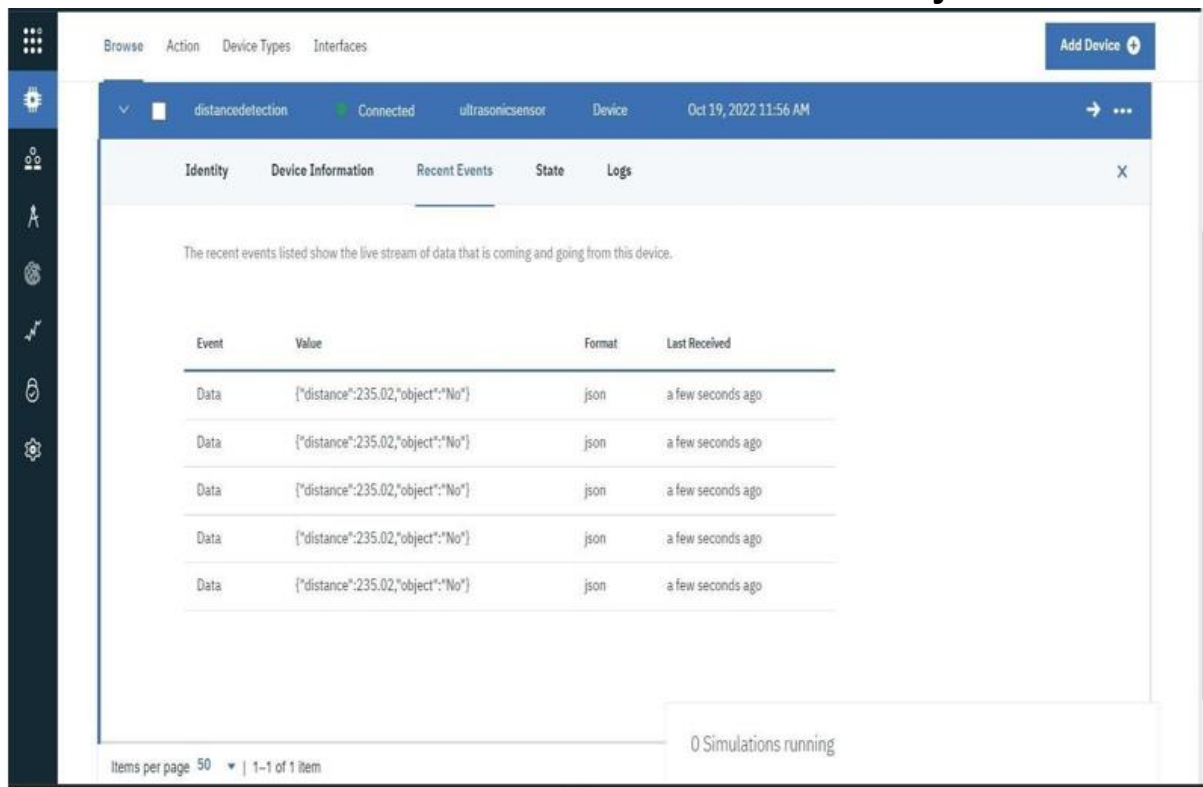
The screenshot displays the Arduino IDE interface with a sketch on the left and a simulation window on the right. The sketch is a C++ program for an ESP32 microcontroller, utilizing the WiFi and PubSubClient libraries to connect to an IBM Watson IoT Platform. It defines various constants for the organization, device type, ID, and token, and sets up a callback function for MQTT messages. The simulation window shows a 3D model of the ESP32 board connected to an HC-SR04 ultrasonic sensor and a red LED. The output console shows the results of the simulation, indicating that no object was found and sending a payload to the IoT platform.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4
5 void callback(char* subscribetopic, byte* payload, unsigned int payloadlength);
6
7 //-----credentials of IBM Accounts-----
8
9 #define ORG "fs9trs" //IBM ORGANITION ID
10 #define DEVICE_TYPE "ultrasonicsensor" //Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "distancedetection" //Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "AlGMGaaf01naw1QA3" //Token
13 String data3;
14 float dist;
15
16
17 //----- Customise the above values -----
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
19 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform and
20 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND COMM
21 char authMethod[] = "use-token-auth"; // authentication method
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
24
25
26 //-----
27 WiFiClient wificlient; // creating the instance for wificlient
28 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined client id
29
30 int LED = 4;
31 int trig = 5;
32 int echo = 18;
33 void setup()
```

Simulation

no object found
Sending payload: {"distance":403.45,"object":"No"}
Publish ok
Distancein cm233.00
no object found
Sending payload: {"distance":233.00,"object":"No"}
Publish ok

Data sent to the IBM cloud device when the object is far

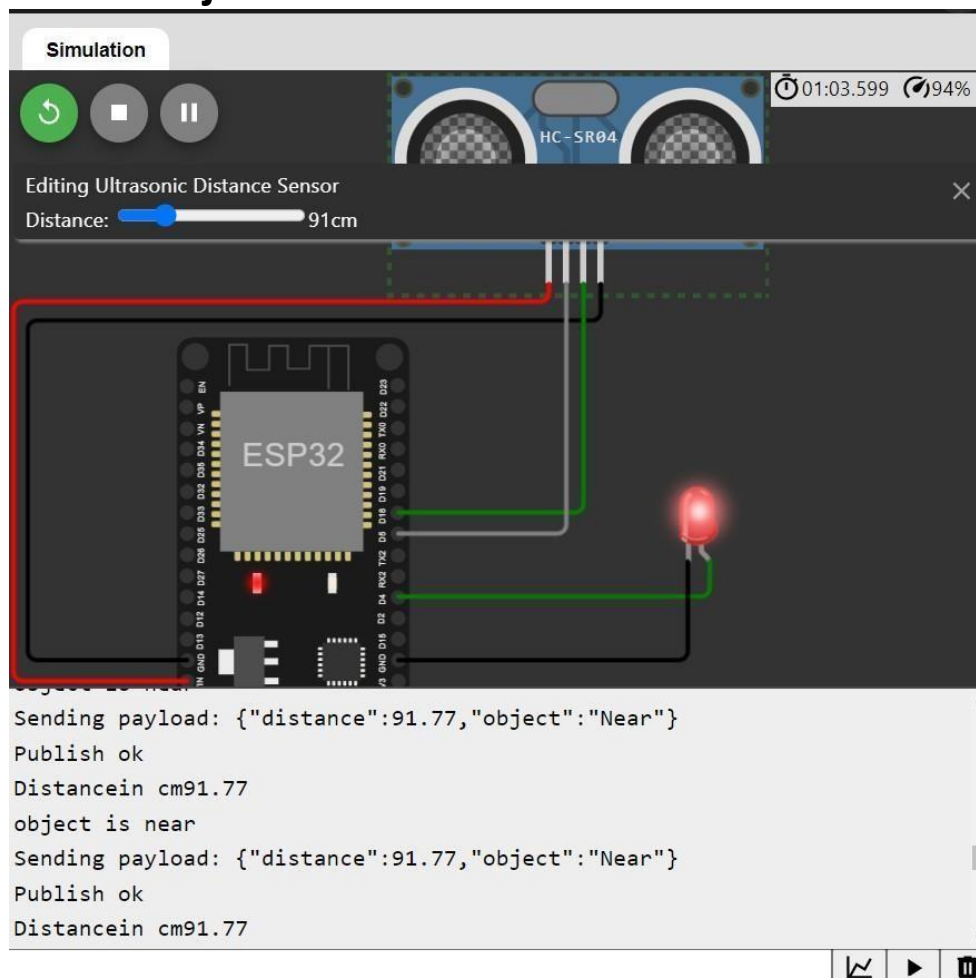


The screenshot shows the IBM Cloud IoT Platform console. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various functions. The main content area displays the 'Recent Events' tab for a device named 'distanceDetection'. The device is connected, and the last update was on Oct 19, 2022 at 11:56 AM. Below the tabs, a message states: 'The recent events listed show the live stream of data that is coming and going from this device.' A table lists the recent events:

Event	Value	Format	Last Received
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago

At the bottom, it indicates '0 Simulations running' and 'Items per page: 50 | 1-1 of 1 item'.

When object is nearer to the ultrasonic sensor



The screenshot shows a simulation environment. At the top, there's a 'Simulation' tab. Below it, a control bar includes a play button, a stop button, and a pause button. A timer shows '01:03.599' and a battery level of '94%'. The main area displays an 'Editing Ultrasonic Distance Sensor' dialog box with a 'Distance' slider set to '91cm'. Below the dialog, a circuit diagram shows an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor and a red LED. The sensor's VCC is connected to the ESP32's VCC, and its GND is connected to the ESP32's GND. The sensor's Trig pin is connected to the ESP32's D4 pin, and its Echo pin is connected to the ESP32's D5 pin. The red LED is connected to the ESP32's D6 pin. The bottom of the screen shows a terminal window with the following text:

```
Sending payload: {"distance":91.77,"object":"Near"}
Publish ok
Distancein cm91.77
object is near
Sending payload: {"distance":91.77,"object":"Near"}
Publish ok
Distancein cm91.77
```

Data sent to the IBM cloud device when the object is near

The screenshot displays the IBM Cloud IoT Platform console. On the left is a dark sidebar with navigation icons. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces', along with an 'Add Device' button. Below this is a search bar. The main content area shows a table of devices. The selected device, 'distancedetection', is shown in a detailed view with tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, showing a live stream of data events. A status box at the bottom right indicates '0 Simulations running'.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
distancedetection	Connected	ultrasonicsensor	Device	Oct 19, 2022 11:56 AM	

Event	Value	Format	Last Received
Data	{"distance":91.77,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.75,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.77,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.79,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.8,"object":"Near"}	json	a few seconds ago

0 Simulations running