Table of Contents

Executive Summary

For this project we implemented a remote garbage sensing system, with the aimto alert a user when a bin is full. The system is designed to be fitted to almost any garbage system, be it an office trash can or a dumpster in the street, it is portable only being the sizeof a palm and can be battery-powered or hard-wired.

The sensor system made use of two ultrasonic sensors in different locations, both posting the data using an ESP32 development board to our created database hosted in the cloud. We made use of the Firebase cloud storage to host our data in a database, and Asksensors cloud analytics API to visualize the data retrieved from the database. We performed edge analytics and calculated the retrieved data’s average, mean, standard deviation and variance. The last task was to activate an actuator based on the data retrievedfrom the database, for this we implemented a green and red LED to display if the bin is full orempty.

Task 1 - Develop an IoT System

Two garbage sensor devices were installed at two different locations which return two separate readings. To create the garbage sensor device we made use of an ultrasonic sensor module, HC-SR04 (Figure 1), an ESP32 development board (Figure 2), running Arduino code. We chose to use the ESP32 module, instead of the provided Arduino Uno, because of the Wi-Fi capabilities of the ESP32, not needing an extra Wi-Fi module, as we would have with the Uno, means the system can be smaller and more portable.

Figure 1 - HR-SR04 Ultrasonic sensor module

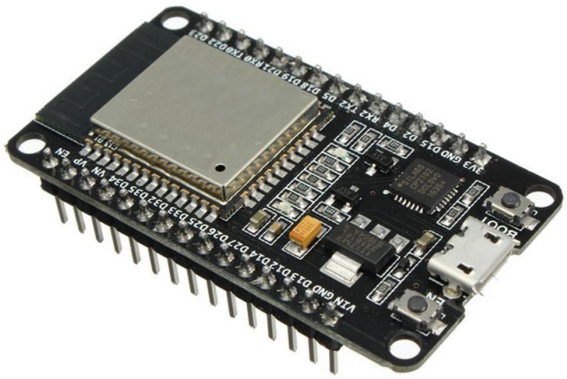

Figure 2 - ESP32 development board

The sensor system detects the rubbish by using the HC-SR04 sensor which emits a sound wave, called the trigger signal, above human hearing frequencies, and then the module listens for the return (bounced back) sound, called the ‘echo’ signal.

This sensing system makes use of two garbage sensors in different countries, writing to a database hosted in the cloud. This data is then retrieved from the cloud in order to complete Task 3 data analysis and visualisation. Kain handled the sensor design and setup, Dev handled the cloud connectivity and data management and handled the actuator activation.

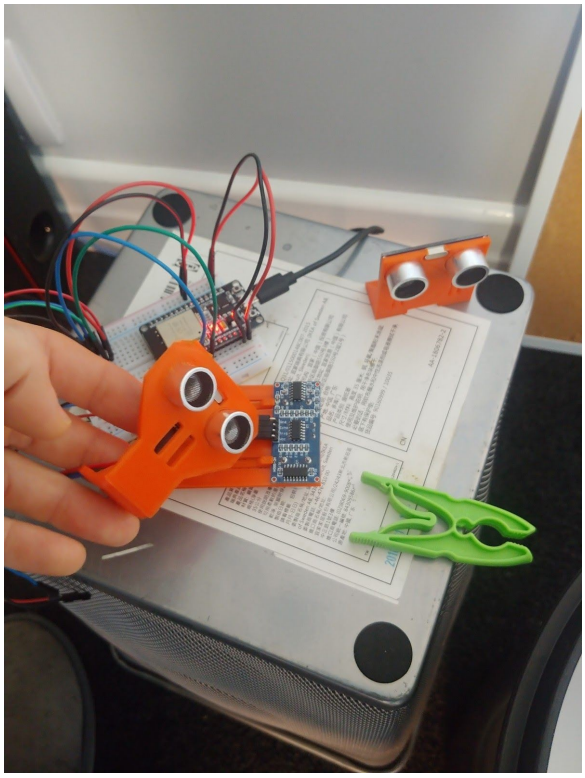

Figure 4 - Ultrasonic sensor testing with 3D printed mounts

Figure 5 - Trash can sensor setup

Figure 6 - Trash can sensor setup

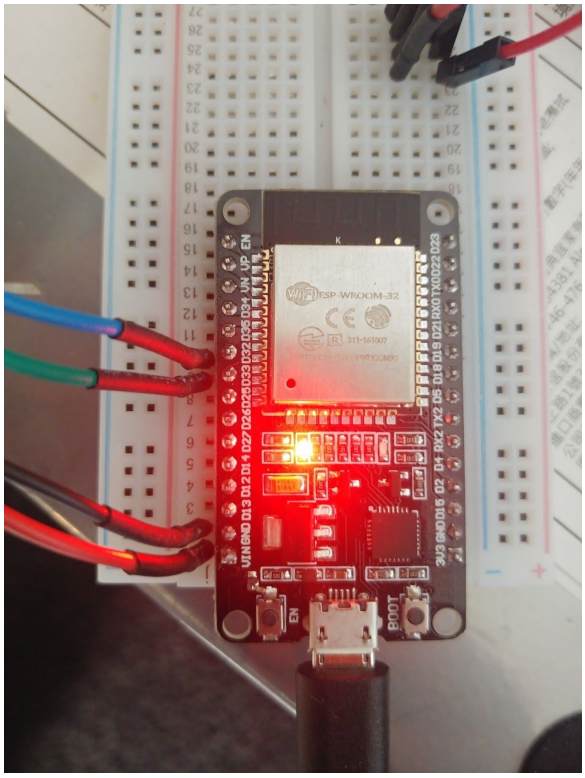

Figure 7 - Trash can ESP32 board setup

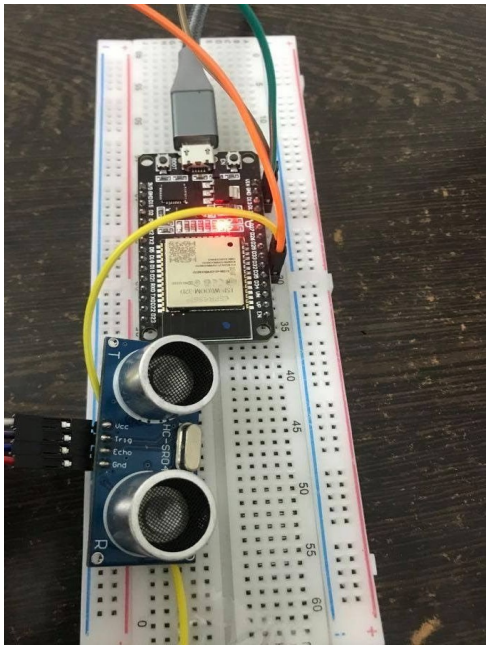

Figure 8 - Ultrasonic Sensor HC-SR04 from second location

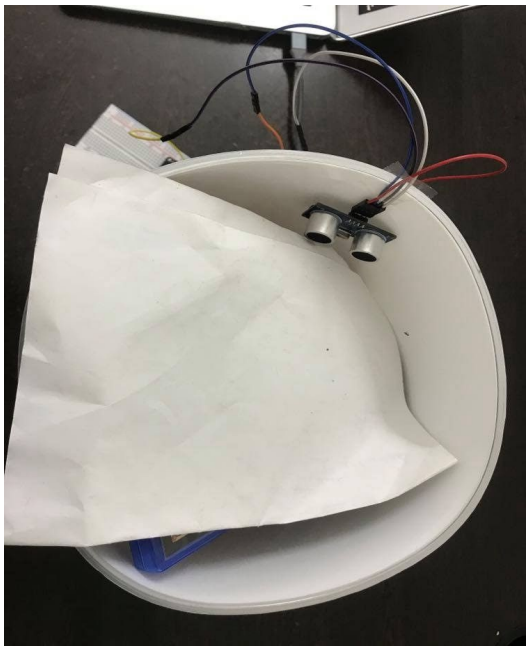

Figure 9 - Trash can setup from second location

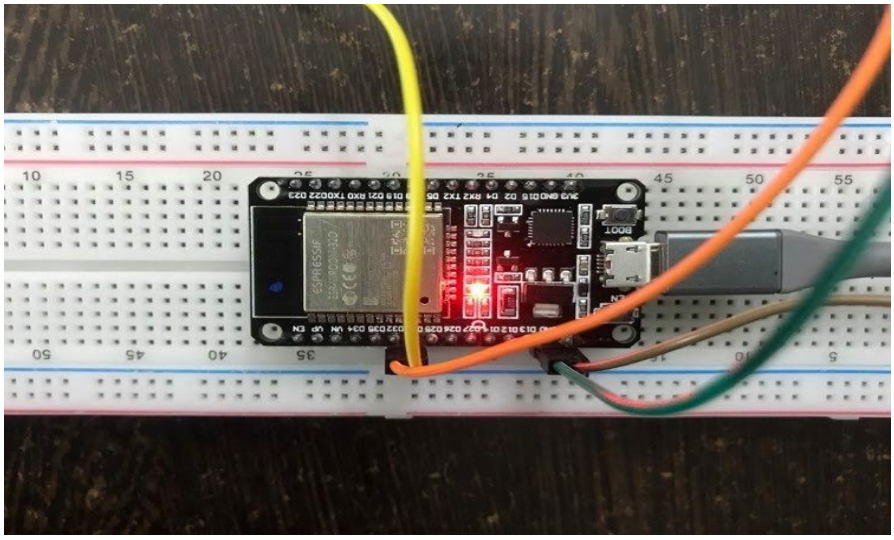

Figure 10 - ESP 32 Board Setup from Second location

Figure 8 shows a block diagram of how the sensing system data passes.

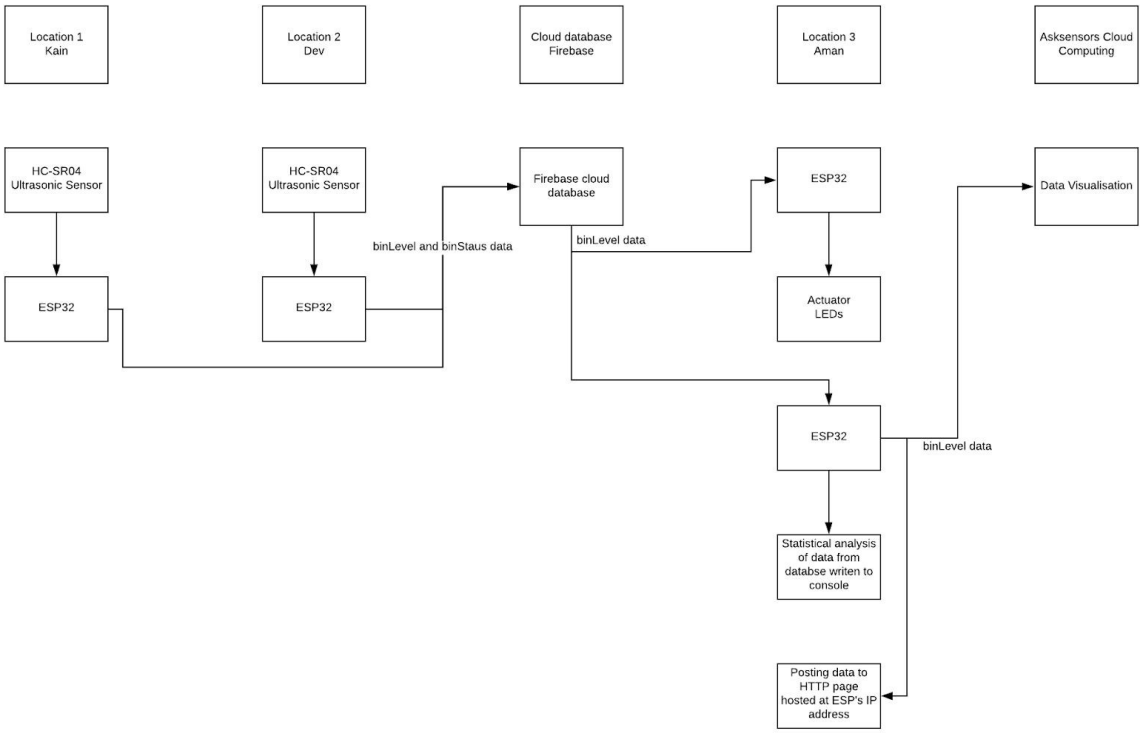

Figure 11 - Work flow block diagram of the IoT sensor network

The arduino code, Code Snippet 1, that the ESP32 board is running, this device is capable of completing both sides of the actuator and that of the sensor at the same time, both sending and writing to the database, it can also send the database data to a cloud computing API for graphical analysis, as well as post the data as HTML via HTTP to a website hosted at the ESP32's IP address.

Features:

- Receive data from Ultrasonic sensor
- Connect to Wi-Fi, Firebase API and Asksensors API
- Send data to Firebase database
- Retrieve data from Firebase database
- Complete edge computing on data received from Firebase database
- Turn on actuator based on condition from sensor data
- Post HTML data via HTTP to vistualise the data at the device's IP address

Code Snippet 1 - Arduino code for ESP32 with sensor

```
//.....//
// Garbage Bin Ultrasonic Sensor HC-SR04
// Author: Kain Burgemeister 100662073
//.....//
```

```
//----------- Libraries
#include <WiFi.h>
#include <WiFiMulti.h>
#include <WebServer.h>
#include <Wire.h>
#include <FirebaseESP32.h>
#include <FirebaseESP32HTTPClient.h>
#include <FirebaseJson.h>

//----------- Deffinitions
#define echoPin 32 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 33 //attach pin D3 Arduino to pin Trig of HC-SR04

//----------- Firebase
#define FIREBASE_HOST "https://garbagedisposal-8d74b.firebaseio.com/"
#define FIREBASE_AUTH "znYg2kfgZLK9b39JE8HXCVezg9SnpGKCdtQHpomu"

//----------- Ask Sensors
const char* https_host = "api.asksensors.com";          // ASKSENSORS host name
const int https_port = 443;                             // https port
const char* https_fingerprint = "B5 C3 1B 2C 0D 5D 9B E5 D6 7C B6 EF 50 3A AD 3F 9F
1E 44 75";     // ASKSENSORS HTTPS SHA1 certificate
const char* apiKeyIn = "JWprULpuQGBSaqhtSk2r74emngnfWmzl"; // API KEY IN
const unsigned int writeInterval = 10000; // write interval (in ms)

//----------Wi-Fi
#define WIFI_SSID       "********************"      //Enter SSID here
#define WIFI_PASSWORD   "********************"              //Enter Password here

//----------- Variables
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement
String status; // variable for the full/empty status of bin
//For statistical analysis
int average2 = 0;
int average10 = 0;
float difference2 = 0.00;
float squares2 = 0.00;
float sum_of_squares2 = 0.00;
float variance2 = 0.00;
float standard_deviation2 = 0.00;

//----------- Wi-Fi and Setup
WebServer httpServer(80);
WiFiClient client;
FirebaseData firebaseData;
WiFiMulti WiFiMulti;
HTTPClient ask;

//----------- Setup
```

```
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

  Serial.begin(9600); // Serial communication is started with a baudrate speed of 9600
  Serial.println("Ultrasonic Sensor HC-SR04 Garbage Bin");
  Serial.println("");

  //Wi-Fi
  Serial.println("Connecting to ");
  Serial.println(WIFI_SSID);

  //Sets to station mode
  WiFi.mode(WIFI_STA);
  //Connect to your specified Wi-Fi network
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  //Checks if Wi-Fi is connected
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP());

  //Firebase
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); //Set Firebase info
  Firebase.reconnectWiFi(true); //Enables reconnectto Wi-Fi if connection lost

  //HTTP
  httpServer.on("/", handle_OnConnect);
  httpServer.onNotFound(handle_NotFound);
  httpServer.begin();
  Serial.println("HTTP server started");
}

//------------ Loop
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delay(2000);

  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
```

```
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = (duration-10) * 0.034 / 2;
//  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("\nDistance: ");
  Serial.print(distance);
  Serial.println(" cm");
  Serial.print("The Bin is: ");
  if (distance < 10) {
     Serial.print("Full!");
     status = "Full";
  } else if (distance < 30) {
    Serial.print("Not full, but contains trash.");
    status = "Not full, but contains trash";
  } else {
    Serial.print("Empty.");
    status = "Empty";
  }

  //Firebase
  //Send
  if((Firebase.push(firebaseData, "/binLevel_kain", distance)) &&
(Firebase.push(firebaseData, "/binStatus_kain", status)))
  {
    //Success
     Serial.println("Set Firebase data success ");
  } else {
    //Failed?, get the error reason from firebaseData

    Serial.print("Error in setting Firebase data ");
    Serial.println(firebaseData.errorReason());
  }

  //Get
    if(Firebase.get(firebaseData, "/binLevel_kain"))
  {
    //Success
    Serial.print("Get variant data success, type = ");
    Serial.println(firebaseData.dataType());
 if (firebaseData.dataType() == "json")
    {
        Serial.println();
        FirebaseJson &json = firebaseData.jsonObject();
        //Print all object data
        Serial.println("Pretty printed JSON data:");
        String jsonStr;
        json.toString(jsonStr, true);
        Serial.println(jsonStr);
        Serial.println();
```

```
      Serial.println("Iterate JSON data:");
      Serial.println();
      size_t len = json.iteratorBegin();
      String key, value = "";
      int type = 0;
      int sum2 = 0;

      for (size_t i = 0; i < len; i++)
      {
        json.iteratorGet(i, type, key, value);
        sum2 = sum2 + value.toInt();
      }
      Serial.print("Length = ");
      Serial.println(len);

      average2 = sum2/len;
      Serial.print("Average = ");
      Serial.println(average2);


      for (size_t i = 0; i < len; i++)
      {
         json.iteratorGet(i, type, key, value);
         difference2 = value.toInt() - average2;
         squares2 = sq(difference2);
         sum_of_squares2 = sum_of_squares2 + squares2;
      }

       variance2 = sum_of_squares2/len;
       Serial.print("Variance = ");
       Serial.println(variance2);

       standard_deviation2 = sqrt(variance2);
       Serial.print("Standard Deviation = ");
       Serial.println(standard_deviation2);
    }
  } else {
    //Failed?, get the error reason from firebaseData
    Serial.print("\nError in getting Firebase data, reason: ");
    Serial.println(firebaseData.errorReason());
  }

//Asksensors
  if (!client.connect(https_host, https_port)) {
  Serial.println("connection failed");
  return;
}else {

  // Create a URL for the request
  String url = "https://api.asksensors.com/write/";
```

```
    url += apiKeyIn;
    url += "?module1=";
    url += distance;
    url += "&module2=";
    url += average2;
    url += "&module3=";
    url += variance2;
    url += "&module4=";
    url += standard_deviation2;

    Serial.print("********** requesting URL: ");
    Serial.println(url);

    ask.begin(url); //Specify the URL

    //Check for the returning code
    int httpCode = ask.GET();

    if (httpCode > 0) {

        String payload = ask.getString();
        Serial.println(httpCode);
        Serial.println(payload);
      } else {
      Serial.println("Error on HTTP request");
    }

    ask.end(); //End
    Serial.println("********** End ");
    Serial.println("*****************************************************");

  }

  client.stop(); // stop client

  delay(writeInterval);    // delay

  //HTTP
  httpServer.handleClient();
}

//----------- HTML
void handle_OnConnect() {
  httpServer.send(200, "text/html", Sendhtml(distance, status));
}

void handle_NotFound(){
  httpServer.send(404, "text/plain", "Not found");
}
```

```
String Sendhtml(int distance, String binStaus){
  String htmlDec = "<!DOCTYPE html> <html>\n";
  htmlDec +="<head><meta http-equiv=\"refresh\" content=\"10\" name=\"viewport\"
content=\"width=device-width, initial-scale=1.0, user-scalable=no\">\n";
  htmlDec +="<title>ESP32 Sensors</title>\n";
  htmlDec +="<style>html { font-family: Arial; display: inline-block; margin: 0px auto;
text-align: center;}\n";
  htmlDec +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;}\n";
  htmlDec +="p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
  htmlDec  +="</style>\n";
  htmlDec +="</head>\n";
  htmlDec +="<body>\n";
  htmlDec +="<div id=\"webpage\">\n";
  htmlDec +="<h1>ESP32 Trash Bin Ultrasonic Sensor</h1>\n";
  htmlDec +="<p>The trash level is: ";
  htmlDec +=distance;
  htmlDec +=" cm</p>";
  htmlDec +="<p>The bin status is: ";
  htmlDec +=status;
  htmlDec +=".</p>";
  htmlDec +="</body>\n";
  htmlDec +="</html>\n";
  return htmlDec;
}
```

For the actuator part, we are using 2 green and 2 red LEDs. 1 red and 1 green LEDs will work based on Devaesh's sensor readings and the other pair will work based on Kain's sensor reading, all connected to the esp32 development board. In the code, Code Snippet 2, firstly we connect to Firebase and local WiFi. In setup(), we initialize the serial communication and connect to WiFi and Firebase. Also, initialize the digital pins for LEDs.
In loop(), to retrieve Devaesh's sensor data, we pull the data from the database using the get() function and verify that the datatype is 'json'. Once verified, the data is stored in a string and printed in the serial monitor. Next step is calculations. To perform calculations, only the value part is needed and we use the len variable to calculate the number of observations. To calculate the sum, the values are converted to int and then, added together, afterwards, various formulas are used to calculate average, variance and standard deviation. The same logic is repeated to retrieve Kain's sensor data and perform calculations.
For the LEDs to work, if the average is more than 10, the green LED turns on, otherwise, red LED turns on.

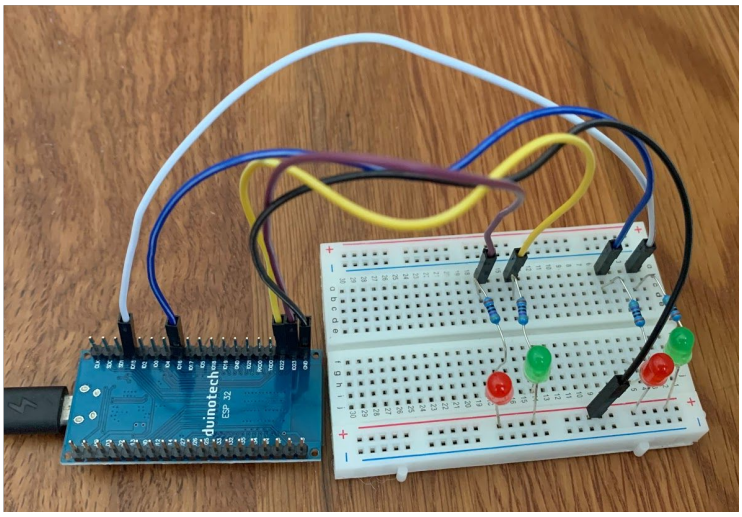

Figure 12 - Actuator (LED) setup using ESP32

Code Snippet 2 - Arduino code for ESP32 with actuator

```
#include <FirebaseESP32.h>
#include <WiFi.h>

//defining the Firebase and WiFi connection information
#define FIREBASE_HOST "https://garbagedisposal-8d74b.firebaseio.com"
#define FIREBASE_AUTH "znYg2kfgZLK9b39JE8HXCVezg9SnpGKCdtQHpomu"
#define WIFI_SSID "********************"
#define WIFI_PASSWORD "********************"

//Defining FirebaseESP32 data object for data sending and receiving
FirebaseData firebaseData;

// LED connections
//LEDs for dev's data
int greenLED1=15;
int redLED1=16;
//LEDs for kain's data
int redLED = 23;
int greenLED = 22;

void setup()
{
  Serial.begin(9600);  // Initiating a serial communication

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
```

```
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); //setting up Firebase info

  Firebase.reconnectWiFi(true);  //Enable auto reconnect the WiFi when connection lost

  //initialize the digital pins for LEDs
  pinMode(greenLED1, OUTPUT);
  pinMode(redLED1, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);


}

void loop()
  // Dev's data
  if(Firebase.get(firebaseData, "/Binlevel_Dev"))
  {
    //Success
    Serial.print("Get variant data success, type = ");
    Serial.println(firebaseData.dataType()); //prints the data type
 if (firebaseData.dataType() == "json")
    {
        Serial.println();
        FirebaseJson &json = firebaseData.jsonObject();
        //Print all object data
        Serial.println("Pretty printed JSON data:");
        String jsonStr;
        json.toString(jsonStr, true);
        Serial.println(jsonStr);
        Serial.println();

        //calculations
        Serial.println("Iterate JSON data:");
        Serial.println();
        size_t len = json.iteratorBegin();
        String key, value = "";
        int type = 0;
        int sum1 = 0;
         int average1 = 0;
        for (size_t i = 0; i < len; i++)
        {
            json.iteratorGet(i, type, key, value);
            sum1 = sum1 + value.toInt();
        }
```

```
      average1 = sum1/len;
      Serial.print("Average = ");
      Serial.println(average1);

      float difference1 = 0.00;
      float squares1 = 0.00;
      float sum_of_squares1 = 0.00;
      float variance1 = 0.00;
      for (size_t i = 0; i < len; i++)
      {
         json.iteratorGet(i, type, key, value);
         difference1 = value.toInt() - average1;
         squares1 = sq(difference1);
         sum_of_squares1 = sum_of_squares1 + squares1;
      }
       variance1 = sum_of_squares1/len;
       Serial.print("Variance =");
       Serial.println(variance1);

       float standard_deviation1 = sqrt(variance1);
       Serial.print("Standard deviation = ");
       Serial.println(standard_deviation1);

       //LED code
      if(average1>10)
       {
        digitalWrite(greenLED1, HIGH); // turn the LED on (HIGH is the voltage level)
        delay(1000);               // wait for a second
        digitalWrite(greenLED1, LOW);    // turn the LED off by making the voltage LOW
        delay(1000);
        }
        else
        {
        digitalWrite(redLED1, HIGH); // turn the LED on (HIGH is the voltage level)
        delay(1000);               // wait for a second
        digitalWrite(redLED1, LOW);    // turn the LED off by making the voltage LOW
        delay(1000);
  }

   }
 }

 //Kain's data
 if(Firebase.get(firebaseData, "/binLevel_kain"))
 {
   //Success
   Serial.print("Get variant data success, type = ");
   Serial.println(firebaseData.dataType()); //prints the data type
if (firebaseData.dataType() == "json")
```

```
{
    Serial.println();
    FirebaseJson &json = firebaseData.jsonObject();
    //Print all object data
    Serial.println("Pretty printed JSON data:");
    String jsonStr;
    json.toString(jsonStr, true);
    Serial.println(jsonStr);
    Serial.println();

    //calculations
    Serial.println("Iterate JSON data:");
    Serial.println();
    size_t len = json.iteratorBegin();
    String key, value = "";
    int type = 0;
    int sum2 = 0;
    int average2 = 0;
    for (size_t i = 0; i < len; i++)
    {
       json.iteratorGet(i, type, key, value);
       sum2 = sum2 + value.toInt();
    }

    average2 = sum2/len;
    Serial.print("Average = ");
    Serial.println(average2);

    float difference2 = 0.00;
    float squares2 = 0.00;
    float sum_of_squares2 = 0.00;
    float variance2 = 0.00;
    for (size_t i = 0; i < len; i++)
    {
       json.iteratorGet(i, type, key, value);
       difference2 = value.toInt() - average2;
       squares2 = sq(difference2);
       sum_of_squares2 = sum_of_squares2 + squares2;
    }
     variance2 = sum_of_squares2/len;
     Serial.print("Variance =");
     Serial.println(variance2);

     float standard_deviation2 = sqrt(variance2);
     Serial.print("Standard deviation = ");
     Serial.println(standard_deviation2);

     //LED code
    if(average2>10)
     {
```

```
        digitalWrite(greenLED, HIGH); // turn the LED on (HIGH is the voltage level)
        delay(1000);                  // wait for a second
        digitalWrite(greenLED, LOW);    // turn the LED off by making the voltage LOW
        delay(1000);
        }
        else
        {
        digitalWrite(redLED, HIGH); // turn the LED on (HIGH is the voltage level)
        delay(1000);                  // wait for a second
        digitalWrite(redLED, LOW);    // turn the LED off by making the voltage LOW
        delay(1000);
  }

   }
 }

}
```

Task 2 - Demonstrated understanding of API

In order to store the data that we are receiving from the sensor network, the use of a database was required. Firebase was the cloud storage solution that we chose, it allowed us to send our data using the Firebase API to a cloud-based database that we created and hosted on their website. The garbage sensor device recorded data and stored it in two variables called “binlevel” and “binstatus”. This data, from the ESP32 and Ultrasonic sensor, is pushed onto the firebase real-time database and is stored in JSON format. The variables ‘binLevel_Kain’ and ‘binStatus_kain’ stores data from the first location and ‘BinStatus_Dev’ and ‘Binlevel_Dev’ stores data from the second location.

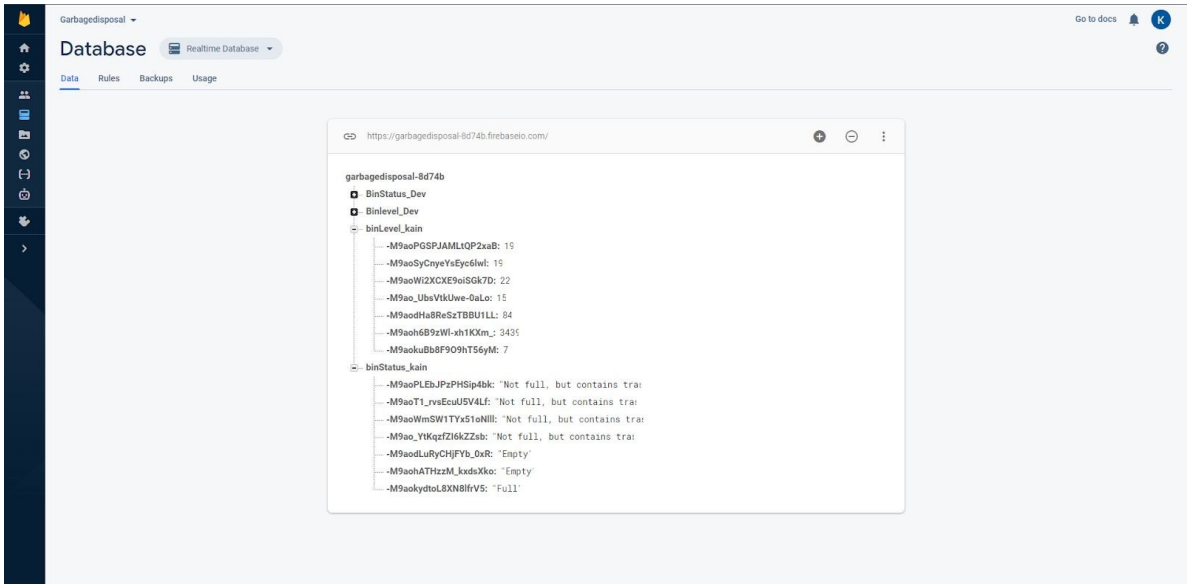

Figure 13 - Firebase cloud-based real-time database

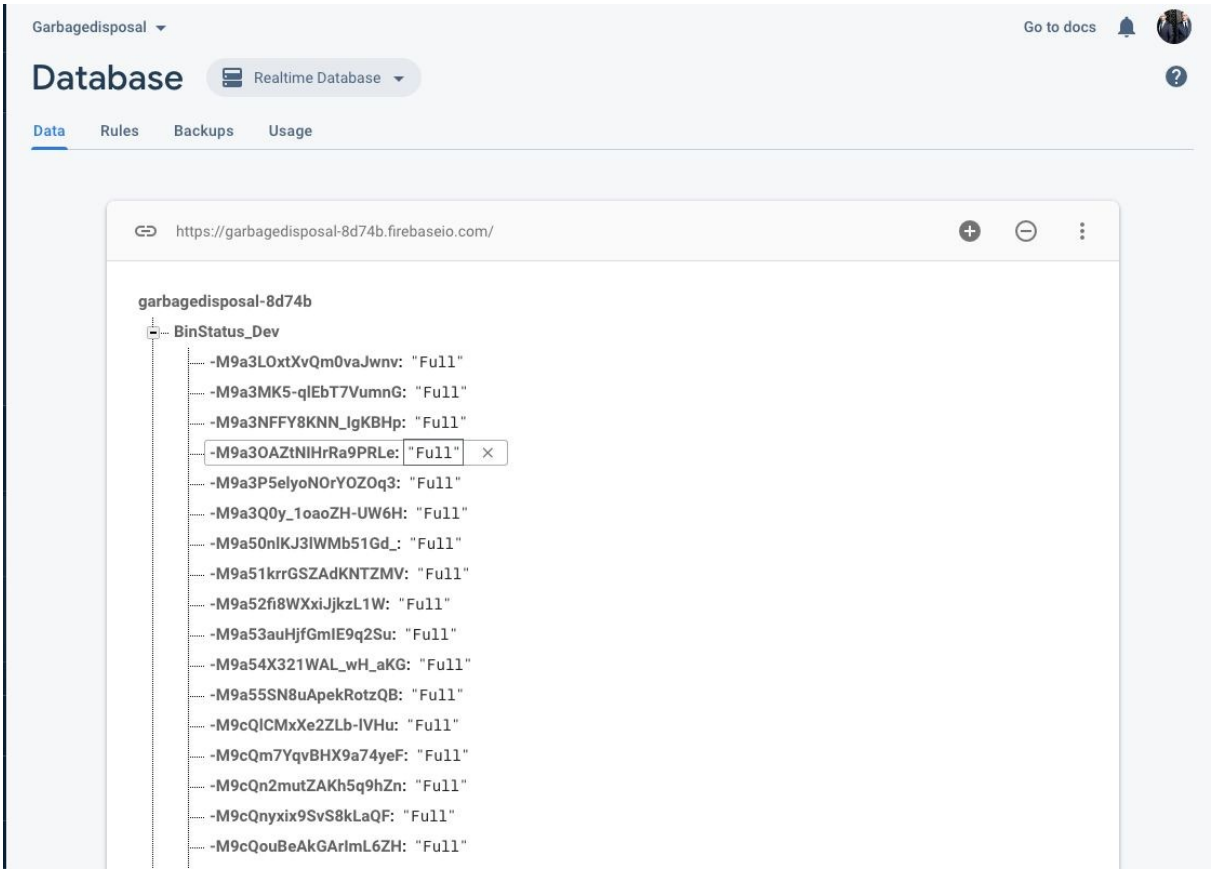

Figure 14 - Firebase cloud-based real-time database

Figure 13 and 14 shows the Firebase database that we implemented to store our data from the sensors. The image shows the values of both location 1 and 2's sensors, with a set of sensor data shown minimised.

Figure 15 - HTTP website hosted at ESP32's IP address

Figure 15 shows the IP address of the ESP32 board, the board is sending HTML via HTTP to the site to allow us to visualise the data.

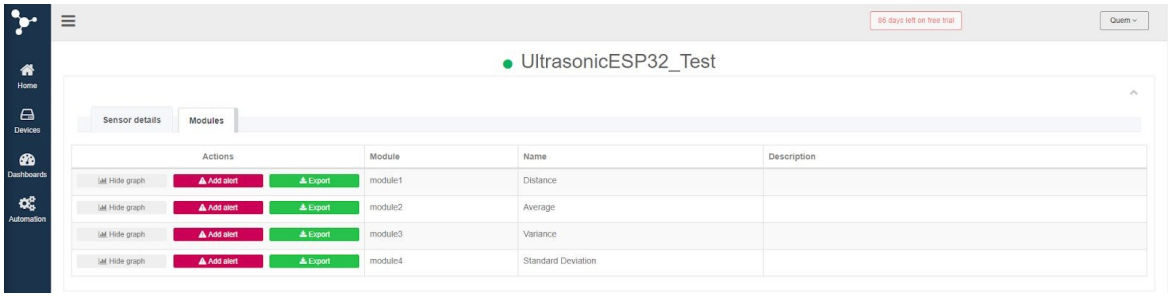

Figure 16 - Asksensors cloud visualization services

Figure 16 shows the cloud data visualisation API by Asksensors, the four values we were plotting are in the figures below, Distance, Average, Variance, Standard Deviation.

Part 3 - Statistical data analysis

In order to be able to perform a statistical analysis of the data from the sensors, remembering that they are in different locations, we would need to first retrieve the data from the cloud database, we used the Firebase API to do this by 'get'ing the data from the database, the parsing the JSON, through a unwrapper in the Arduino code, we received back into a more suitable data type (eg int, float).

Figure 17 - Average graph

Figure 18 - Average graph

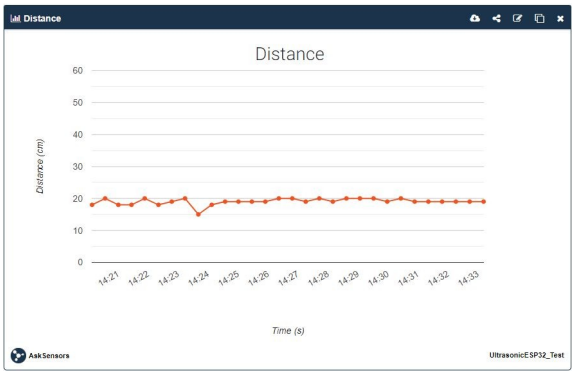

Figure 19 - Distance graph

Figure 20 - Distance graph

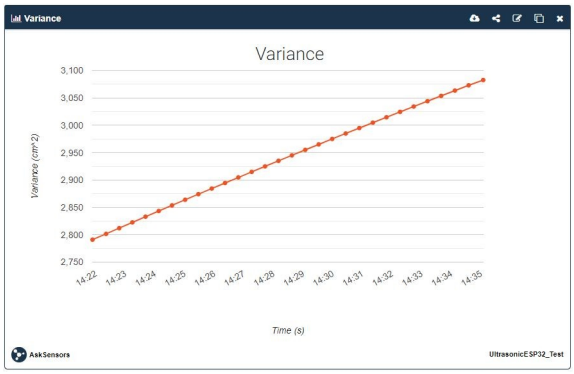

Figure 21 - Variance graph

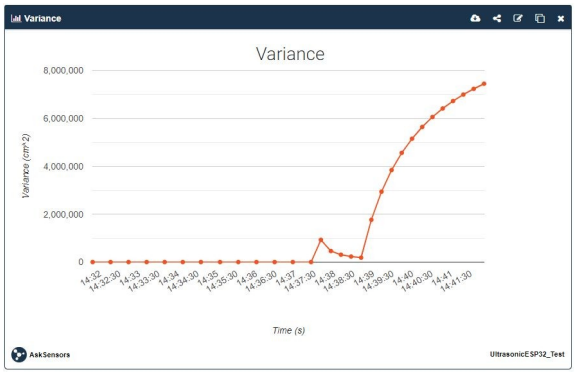

Figure 22 - Variance graph

Figure 23 - Standard deviation graph

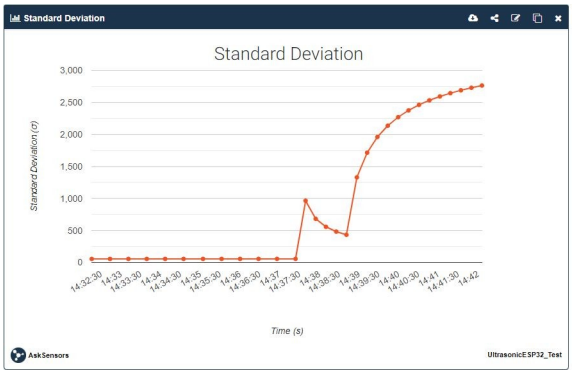

Figure 24 - Standard deviation graph

The figures above, Figures 17 - 24, show the visualised data of the values from the statistical data analysis portion of this project, this is done using the Asksensors API.

Console output from the data fetching and calculating can be seen Figure 25. The JSON can be seen first, then the length of the database record, and the calculated statistical analysis values.
Finally it shows the URL that is being passed via HTTPS to a cloud based sensor visualisation API called Asksensors

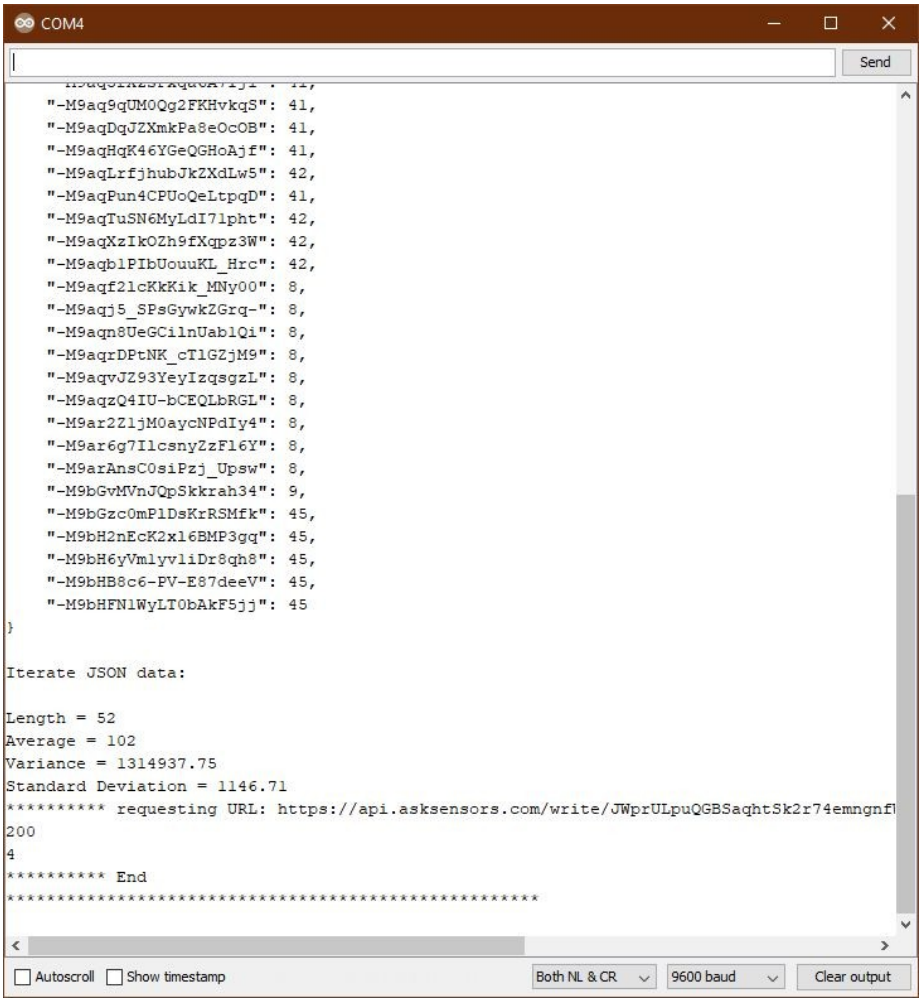

Figure 25 - Console output of the statistical data analysis and visualisation

```
    "-M9a44qJ5pnlsjoUAVIg": 16,
    "-M9a48tMgBWa9FL9qzRQ": 16,
    "-M9a4CwSf9-E-ay6gY58": 15,
    "-M9a4H0234KsyNitcyFt": 16,
    "-M9a4L32ITLwuipLtApZ": 15,
    "-M9a4P66QWJ7NYbAthLN": 16,
    "-M9a4TAhMK3ulVxyvMqH": 16,
    "-M9a4XFP3YevclDIulES": 16,
    "-M9a4aLZMwBSBH_KbuSC": 55,
    "-M9a4eRyN2lUeEDLy-QT": 55,
    "-M9a4iZZsIykfOHhz0lV": 41,
    "-M9a4mgjdciiqvtNM_x6": 56,
    "-M9a4qoJEMGsKXGfUljQ": 55
}

Iterate JSON data:

Average = 21
Variance =127.59
Standard deviation = 11.30
```

Figure 26 - Console output of the statistical data analysis

References

Anon n.d., ‘AskSensors - An Amazing IoT platform’, *AskSensors*, viewed 12 June, 2020a, <https://asksensors.com>.

Anon n.d., ‘binSensor: Lucidchart’, viewed 12 June, 2020b, <https://app.lucidchart.com/documents/edit/2f6439e3-1fe9-4485-a5da-37c96395d963/0_0?beaconFlowId=942A37B1DEBA4A63>.

Anon n.d., ‘Garbagedisposal – Firebase console’, viewed 12 June, 2020c, <https://console.firebase.google.com/project/garbagedisposal-8d74b/database/garbagedisposal-8d74b/data/~2F>.

Anon n.d., ‘Stack Overflow - Where Developers Learn, Share, & Build Careers’, *Stack Overflow*, viewed 12 June, 2020d, <https://stackoverflow.com/>.

Anon n.d., ‘The world’s leading software development platform · GitHub’, viewed 12 June, 2020e, <https://github.com/>.

Anon n.d., ‘Understanding How Ultrasonic Sensors Work | MaxBotix Inc.’, viewed 12 June, 2020f, <https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm>.