KD34303 Internet of Things

Semester 1 (2022/2023)

Group 1

Project Report
Topic: Smart Dustbin

Name	Matric Num.
CHAN KAI KHEE	BI20110118
ADIANA BINTI ROSLEH	BI20110094
JOANNE PATRIC@PATRICK	BI20110280
LIM HONG YAO	BI20110046

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

The Internet of Things (IoT) is a concept that is becoming more and more common in our daily lives. It is also positioned as one of the most advanced technologies, which means it has the ability to significantly affect everything we do and how we connect with those around us. IoT is a new revolution that makes it possible for electrical gadgets and sensors to communicate with one another over the internet to make our lives easier. IoT utilizes smart devices and the internet to offer creative solutions to a range of problems and difficulties affecting various sectors worldwide (Kumar et al., 2019).

The idea behind the Internet of Things is to make the web even more pervasive and immersive. Furthermore, by facilitating simple access to and interaction with a wide range of devices, for example, household appliances and sensors The IoT will advance the creation of several applications that employ the enormous volume and diversity of data that is created by objects to implement new services for businesses, individuals, and governmental agencies.

The country's population is growing quickly, thus the amount of rubbish is also increasing significantly. As of November 15 2022, there are about 8 billion people on earth (*World Population Clock: 8 Billion People (LIVE, 2022) - Worldometer*). About 33 million people are living in Malaysia (*Malaysia Population (2022) - Worldometer*). The amount of trash each household produces every week would be notable. People would often use the rubbish bin to throw their trash but do not realize when it is full. This resulted in rubbish spilling out of the dustbin. The surrounding people would be exposed to bacteria and viruses which can lead to harmful diseases.

The purpose for this smart dustbin is to alert users that their dustbin is full. This can reduce the spread of diseases and avoid air pollution. This smart dustbin is made out of ESP32, Ultrasonic Sensor, Arduino Uno, Jumper wires (Male-To-Male & Female-To-Male), LED. Arduino USB, Mini Gear Servo, Battery, Battery plug, Breadboard and a dustbin. This smart dustbin not only opens the lid when it senses someone is near but also provides the bin's status. For example, the application will show status 'Not Full', 'Half Full' and 'Full'. The smart dustbin can be best used in packed areas for instance residential areas, inside the house, offices and even schools.

PROJECT OBJECTIVES

1. To create a cleaner, safer and hygienic environment.
2. To enhance operational efficiency

PROBLEM STATEMENT

With the increased number of people in an area, the amount of rubbish thrown would also increase. People would often forget to clean out their rubbish bin and it is hard to accurately predict when the dustbin is full, resulting in trash spilling out of the bin. This can create a filthy and smelly surrounding and encourage the birth of bacteria viruses. When it is hard to predict the amount of waste inside the bin, the trash would be collected when it is not full.

PROJECT BACKGROUND

It is important to have a smart dustbin to ensure a cleaner environment. A filthy environment can cause multiple health problems, for example, it can promote respiratory diseases because waste can contaminate the air with carbon dioxide, nitrous oxide, and methane. For that, the smart dustbin is created to control the amount of waste. Smart dustbin is recommended to be used in residential areas, schools and crowded places.

In this project, 'Mr Bin' is a smart dustbin that enables users to monitor the amount of trash inside their dustbin. Mr Bin can also open its lid when someone approaches it and closes when no one is nearby. The user will be able to see the status of the dustbin whether it is 'Not Full', 'Half Full' or 'Full' through an application made with Flutter. For this project, we also use Firebase, to collect and store data from the device.

Smart dustbin also aims to improve operational efficiency. As mentioned, this dustbin has a function of informing the user on the amount of waste. By doing so, fewer garbage collections are made. This reduces manpower and fuel use. The garbage collectors do not have to waste time checking every dustbin in the area to see if there is rubbish that needs to be thrown away.

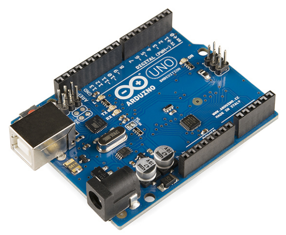
To make this dustbin, we would need ESP32, Ultrasonic Sensor, Arduino Uno, Jumper wires (Male-To-Male & Female-To-Male), LED. Arduino USB, Mini Gear Servo, Battery, Battery plug, Breadboard and a dustbin. Visual Studio is used to code the Arduino.

CHAPTER 2

HARDWARE

Components

The group members bought all of the electronics to build this 'smart dustbin' via Einstronic Enterprise at KK Plaza Mall, Kota Kinabalu. Another thing that the group bought but is not listed in the table is the dustbin itself. The total cost for this project is RM100.20.

No.	Hardware	Diagram	Quantity	Function
1	ESP32 (NodeMCU)		1	To store collected data to cloud database
2	Ultrasonic sensor		2	To send signal according to motion or given threshold
3	Arduino UNO		1	Works as the motherboard

4	Jumper (Male-To-Male & Female-To-Male)		1 Pack	To connect between components
5	LED		2	To produce light
6	Aduino USB		1	To connect Arduino with PC
7	Mini Gear Servo		1	To spin and open the dustbin lid

8	Battery – DC 9V		1	To provide power supply
9	Battery plug – 9V		1	To connect battery with arduino
10	Breadboard		1	Act as extension to connect components to arduino

Circuit Diagram

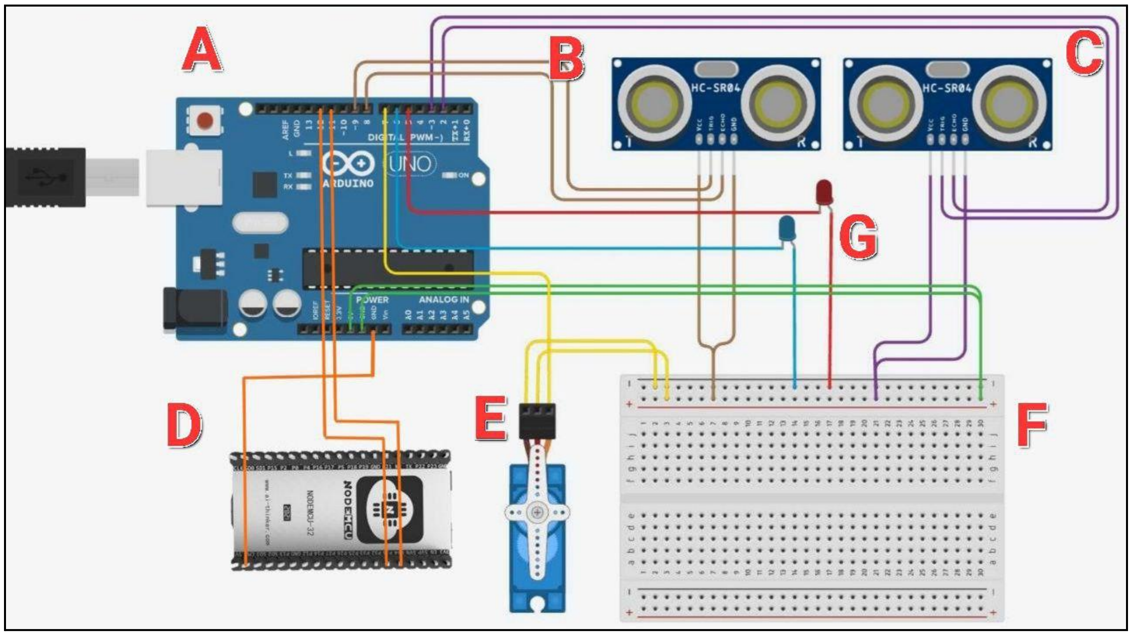


Figure 1.0: Circuit Diagram of the Smart Dustbin

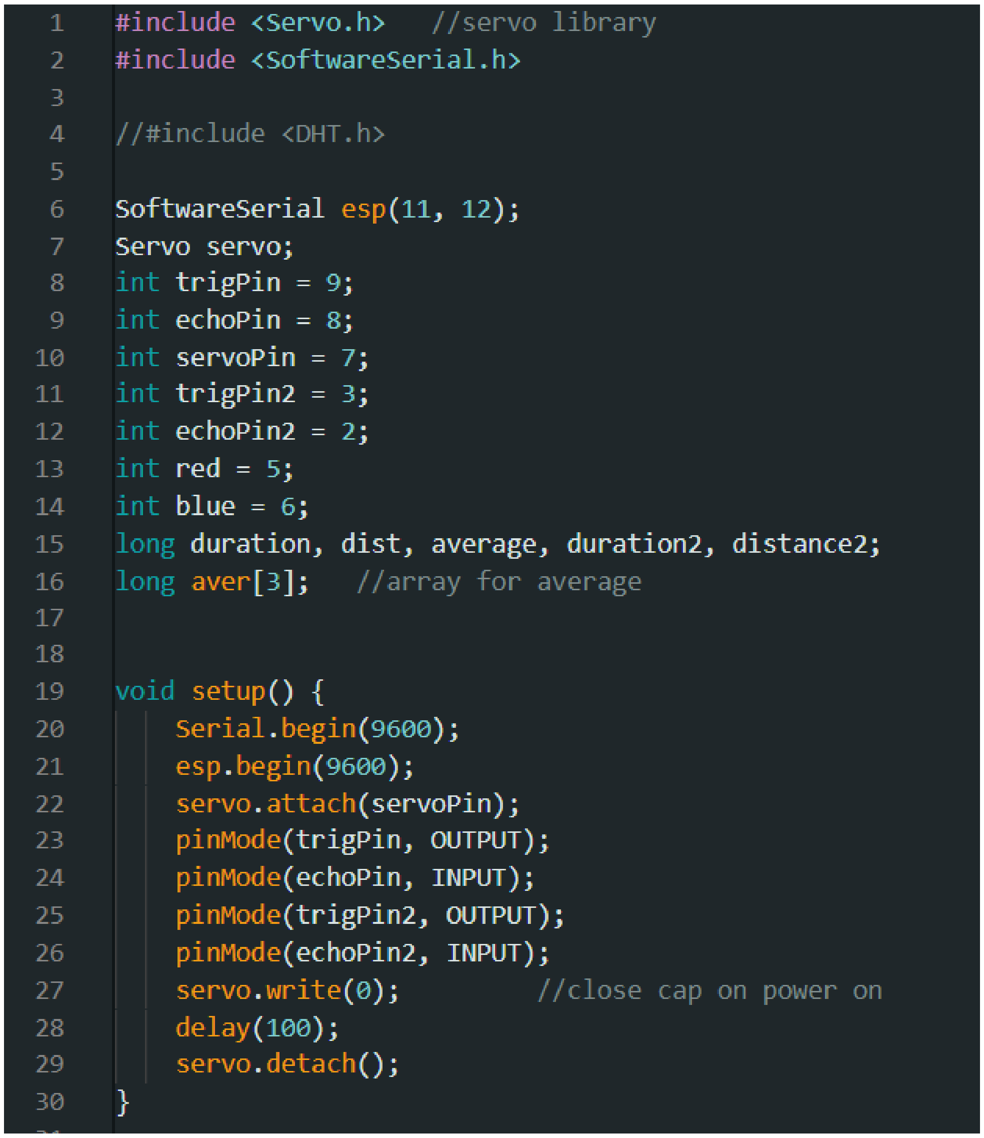
Circuit diagram above shows the layout of the electronic components in the circuit. This diagram was drawn by using online drawing tool tinkercad.com. Tinkercad allows users to drag and drop the desired components and organize the connection by assigning colors to the wires. This way, it is easier to differentiate between different component's connections.

Indicator	Name of the component	Assigned wire color
A	Arduino Uno	Green
B	Sensor (status monitoring)	Brown
C	Sensor (motion detector)	Purple
D	ESP 32 - NodeMCU	Orange
E	Mini Gear Servo	Yellow
F	Breadboard	Green
G	LEDs	Red & Blue

SOFTWARE

Arduino IDE

Arduino IDE is used to write programs.

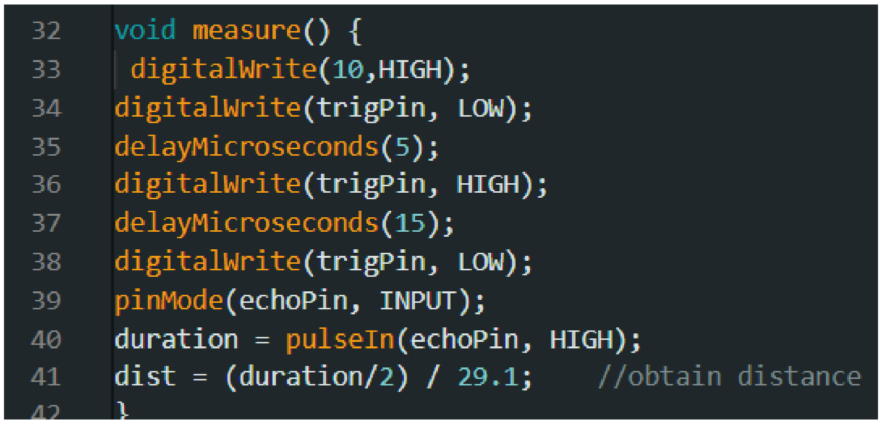


```
#include <Servo.h>   //servo library
#include <SoftwareSerial.h>

//#include <DHT.h>

SoftwareSerial esp(11, 12);
Servo servo;
int trigPin = 9;
int echoPin = 8;
int servoPin = 7;
int trigPin2 = 3;
int echoPin2 = 2;
int red = 5;
int blue = 6;
long duration, dist, average, duration2, distance2;
long aver[3];   //array for average


void setup() {
    Serial.begin(9600);
    esp.begin(9600);
    servo.attach(servoPin);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(trigPin2, OUTPUT);
    pinMode(echoPin2, INPUT);
    servo.write(0);         //close cap on power on
    delay(100);
    servo.detach();
}
```

These lines of code include the Servo library and the SoftwareSerial library. The Servo.h library allows the code to control a servo motor, while the SoftwareSerial library allows the code to communicate with a device over serial communication. By including these libraries, the code can use the functions and methods defined in them to control a servo motor and communicate with a device over serial respectively. The variable esp is an instance of the SoftwareSerial class, which is used to communicate with a device over serial communication using digital pins 11 and

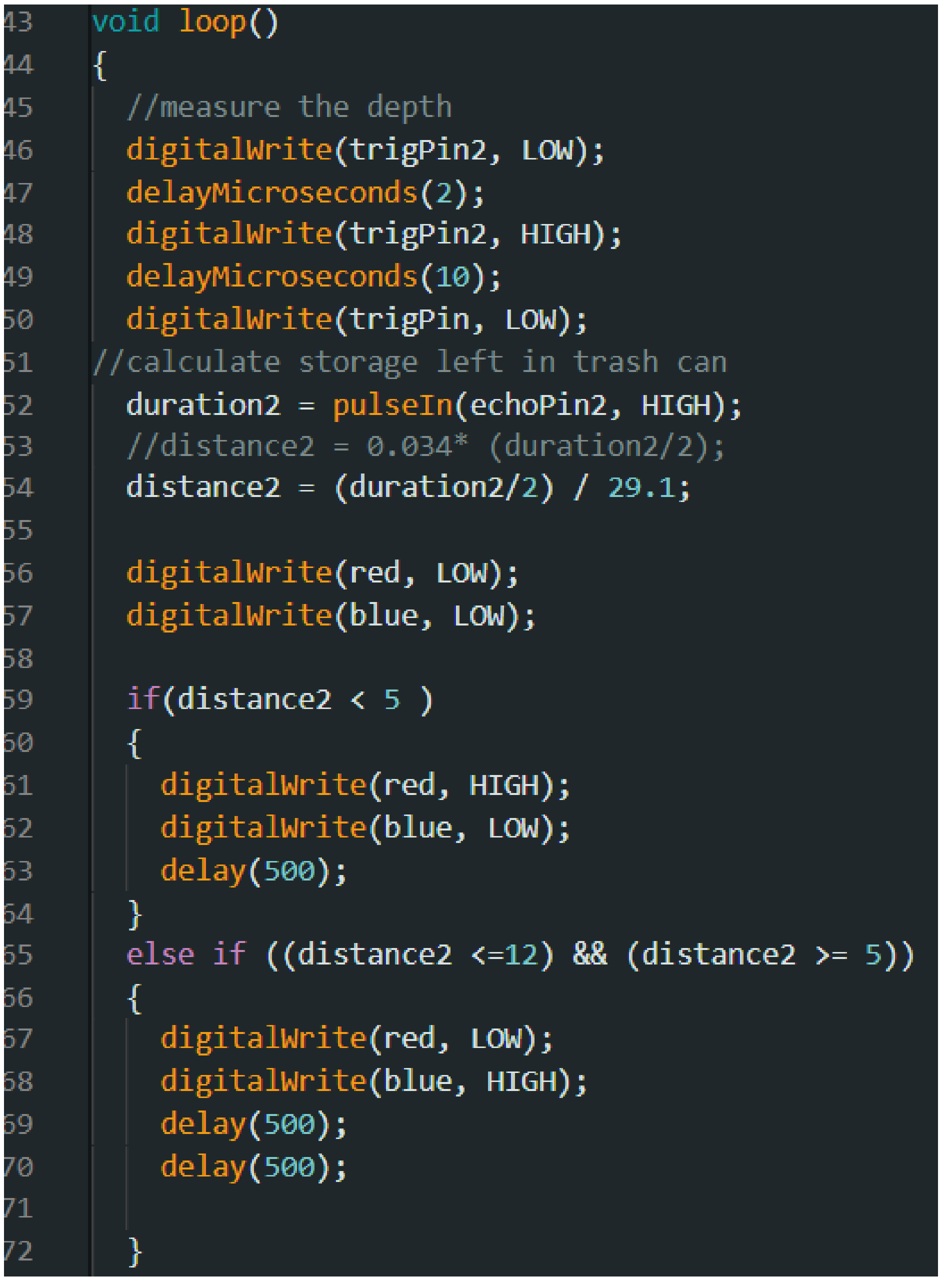
12 as the RX and TX respectively. The variable servo is an instance of the Servo class, which is used to control a servo motor. The variables trigPin, echoPin, servoPin, trigPin2, echoPin2, red and blue are integers that are used to store the digital pin numbers that the HC-SR04 ultrasonic sensor, servo motor and the LEDs are connected to. The variable aver is an array of 3 long integers that are used to store the multiple measurements taken by the ultrasonic sensor for calculating the average distance.

In setup() function Serial.begin(9600) is called to start the serial communication with the computer using a baud rate of 9600 bps. Next it starts software serial communication with the device connected to digital pins 11 and 12 using a baud rate of 9600 bps by calling esp.begin(9600). servo.attach(servoPin) is called to attach the servo motor to the digital pin. The setup function sets the trigPin and trigPin2 as outputs and echoPin and echoPin2 as inputs by calling pinMode(trigPin, OUTPUT) and pinMode(echoPin, INPUT) and pinMode(trigPin2, OUTPUT) and pinMode(echoPin2, INPUT) respectively. Next, it sets the servo to the initial position of 0 degrees by calling servo.write(0). Then It waits for 100 milliseconds by calling delay(100) to give the servo time to move to the initial position. Moving on, it detaches the servo by calling servo.detach(). This stops the power to the servo motor and prevents it from holding its position and drawing power when not in use.

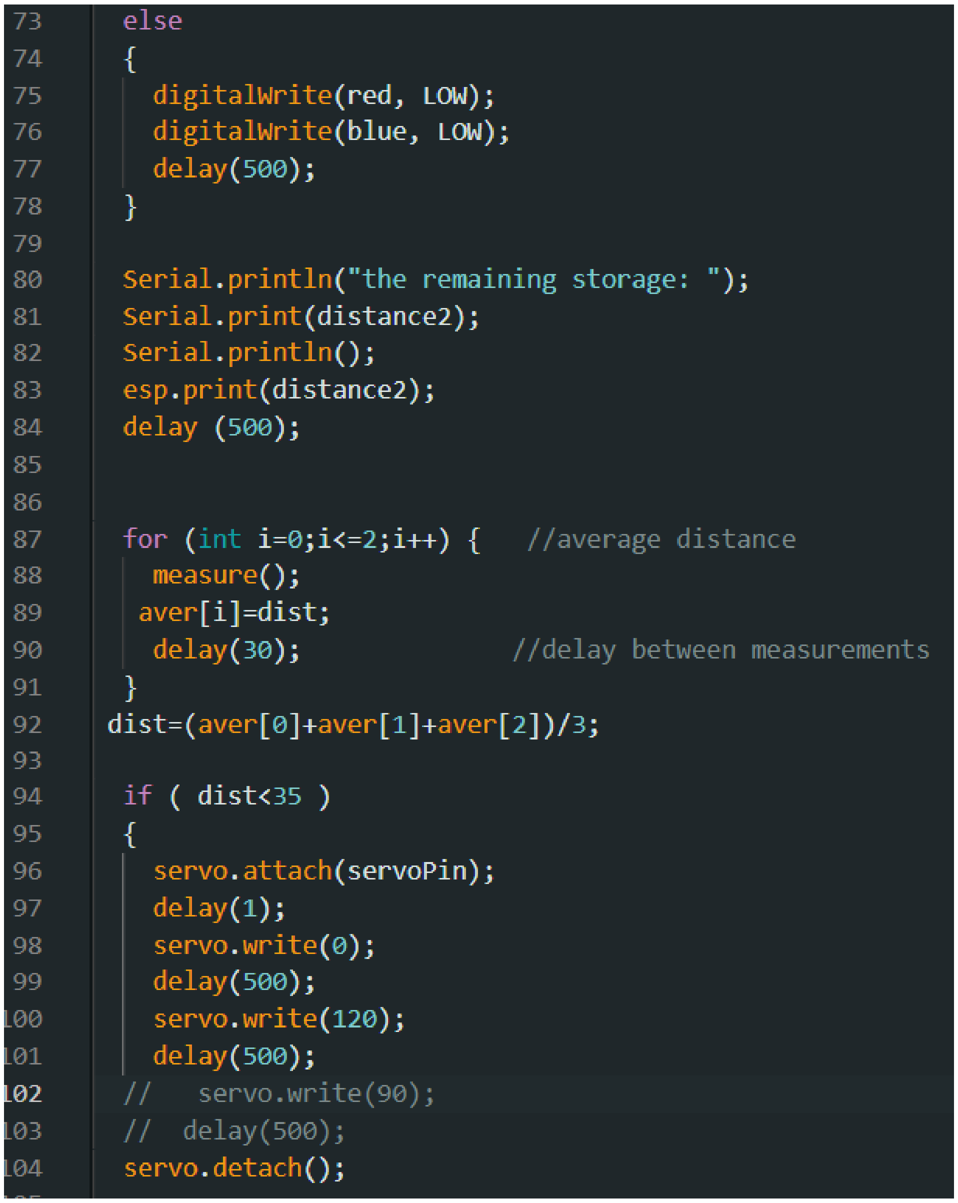


```
void measure() {
 digitalWrite(10,HIGH);
digitalWrite(trigPin, LOW);
delayMicroseconds(5);
digitalWrite(trigPin, HIGH);
delayMicroseconds(15);
digitalWrite(trigPin, LOW);
pinMode(echoPin, INPUT);
duration = pulseIn(echoPin, HIGH);
dist = (duration/2) / 29.1;    //obtain distance
}
```

This is a function called measure() that is used to measure the distance of an object from the ultrasonic sensor.



```
void loop()
{
  //measure the depth
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
//calculate storage left in trash can
  duration2 = pulseIn(echoPin2, HIGH);
  //distance2 = 0.034* (duration2/2);
  distance2 = (duration2/2) / 29.1;

  digitalWrite(red, LOW);
  digitalWrite(blue, LOW);

  if(distance2 < 5 )
  {
    digitalWrite(red, HIGH);
    digitalWrite(blue, LOW);
    delay(500);
  }
  else if ((distance2 <=12) && (distance2 >= 5))
  {
    digitalWrite(red, LOW);
    digitalWrite(blue, HIGH);
    delay(500);
    delay(500);

  }
```

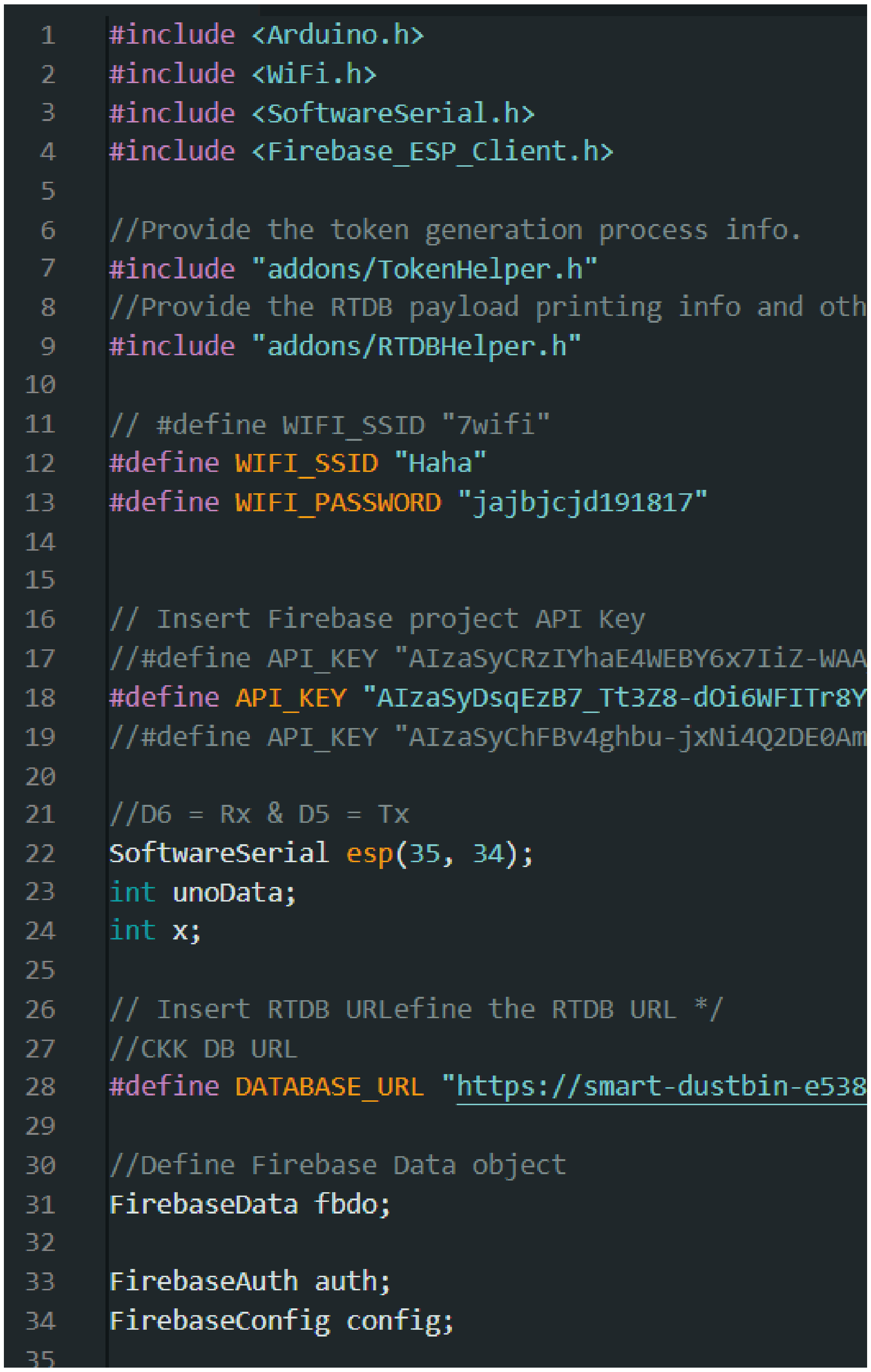


```
  else
  {
    digitalWrite(red, LOW);
    digitalWrite(blue, LOW);
    delay(500);
  }

  Serial.println("the remaining storage: ");
  Serial.print(distance2);
  Serial.println();
  esp.print(distance2);
  delay (500);


  for (int i=0;i<=2;i++) {   //average distance
    measure();
   aver[i]=dist;
    delay(30);              //delay between measurements
  }
 dist=(aver[0]+aver[1]+aver[2])/3;

  if ( dist<35 )
  {
    servo.attach(servoPin);
    delay(1);
    servo.write(0);
    delay(500);
    servo.write(120);
    delay(500);
  //   servo.write(90);
  //  delay(500);
  servo.detach();
```

In the loop() function digitalWrite(trigPin2, LOW) and delayMicroseconds(2) to set the trigPin2 to LOW and wait for 2 microseconds. It sets the trigPin2 to HIGH and waits for 10 microseconds by calling digitalWrite(trigPin2, HIGH) and delayMicroseconds(10). Then it set the trigPin2 to low by calling the digitalWrite(trigPin, LOW). Next, the function calculates the distance from the sensor to the object by dividing the duration by 2 and then dividing that by 29.1 and stores the result in the distance2 variable. Then the function sets both the red and blue LEDs to LOW by calling digitalWrite(red, LOW) and digitalWrite(blue, LOW) respectively. To determine the remaining space of the dustbin. The function uses an if-else statement to check the value of distance2. If distance2 is less than 5, it turns on the red LED and turns off the blue LED by calling digitalWrite(red, HIGH) and digitalWrite(blue, LOW). This shows that the dustbin is full. If distance2 is between 5 and 12, it turns on the blue LED and turns off the red LED by calling digitalWrite(red, LOW) and digitalWrite(blue, HIGH). This means that the dustbin is half full. If distance2 is greater than 12, it turns off both the red and blue LEDs by calling digitalWrite(red, LOW) and digitalWrite(blue, LOW). This means that the dustbin is not full. After that, it will print the remaining storage.

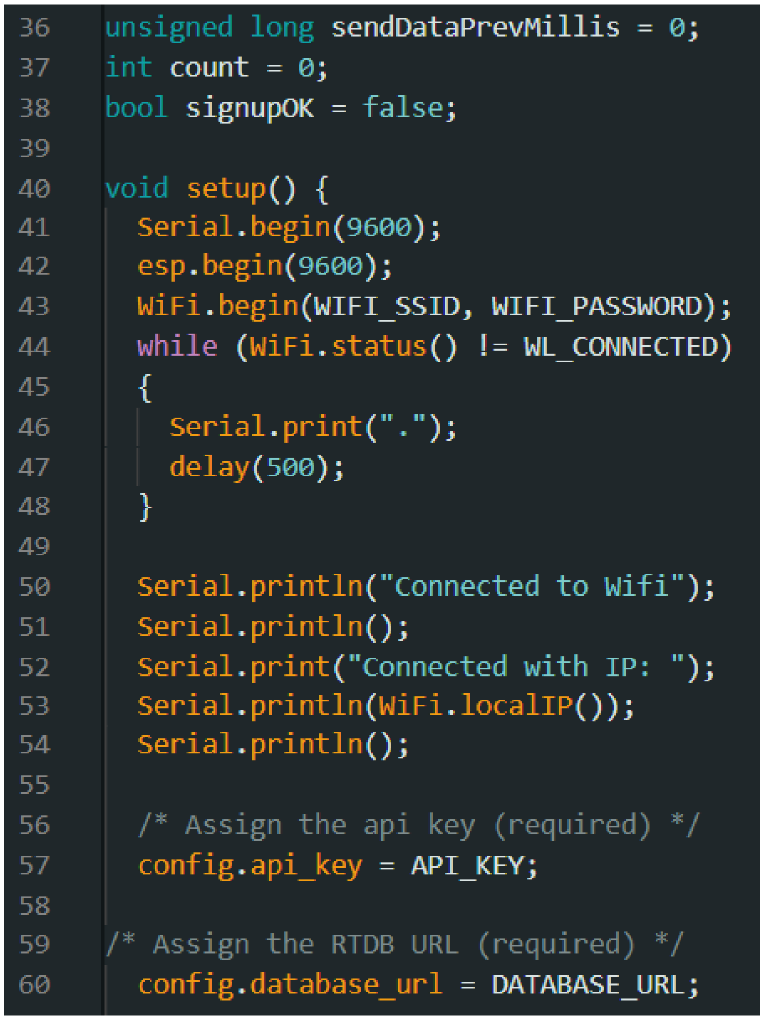
To allow the sensor to detect motion when throwing rubbish, measure() function is called to take multiple measurements and obtain average distance. It uses an if statement to check if the average distance is less than 35cm, if true the dustbin cap will open.

ESP

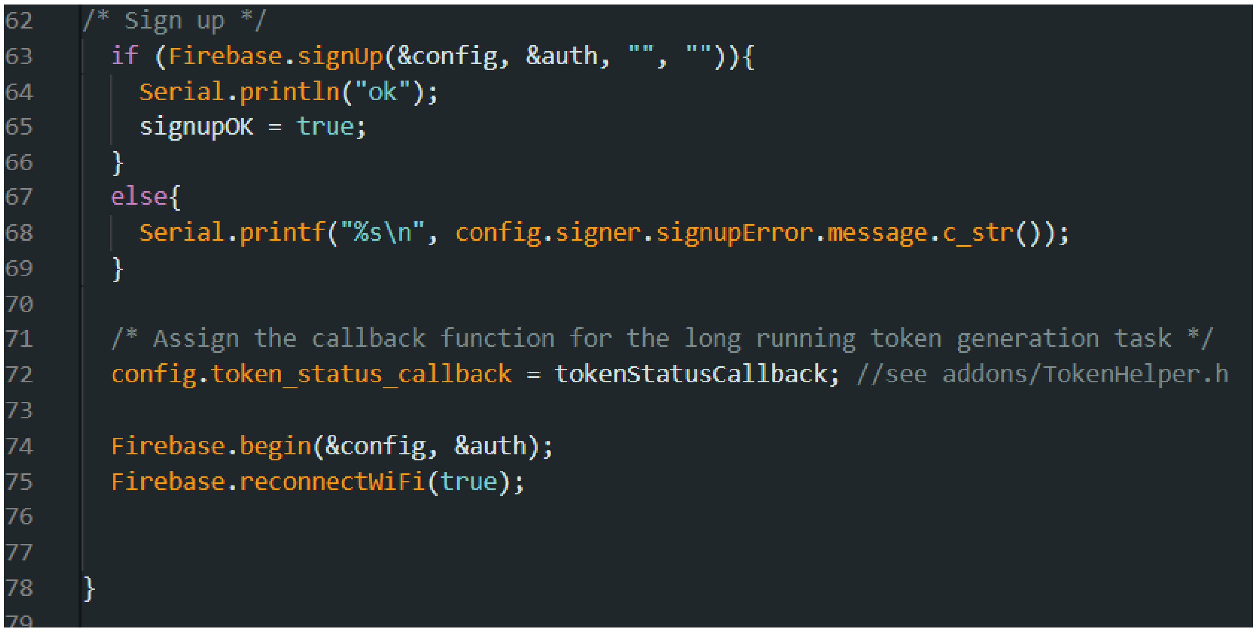


```
#include <Arduino.h>
#include <WiFi.h>
#include <SoftwareSerial.h>
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and oth
#include "addons/RTDBHelper.h"

// #define WIFI_SSID "7wifi"
#define WIFI_SSID "Haha"
#define WIFI_PASSWORD "jajbjcjd191817"


// Insert Firebase project API Key
//#define API_KEY "AIzaSyCRzIYhaE4WEBY6x7IiZ-WAA
#define API_KEY "AIzaSyDsqEzB7_Tt3Z8-dOi6WFITr8Y
//#define API_KEY "AIzaSyChFBv4ghbu-jxNi4Q2DE0Am

//D6 = Rx & D5 = Tx
SoftwareSerial esp(35, 34);
int unoData;
int x;

// Insert RTDB URLefine the RTDB URL */
//CKK DB URL
#define DATABASE_URL "https://smart-dustbin-e538

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

```

Above is the library used for the connection ofESP and Arduino UNO code.

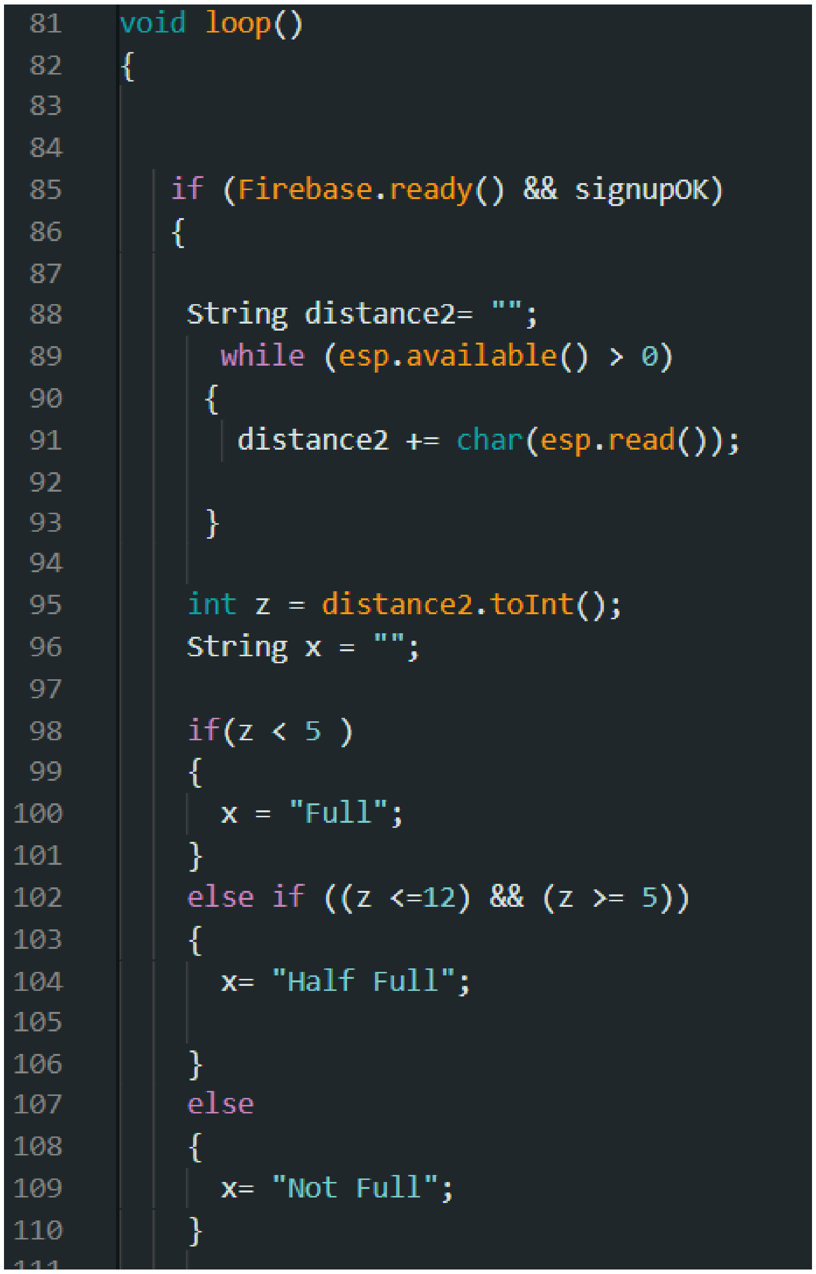


```
unsigned long sendDataPrevMillis = 0;
int count = 0;
bool signupOK = false;

void setup() {
  Serial.begin(9600);
  esp.begin(9600);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }

  Serial.println("Connected to Wifi");
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  /* Assign the api key (required) */
  config.api_key = API_KEY;

/* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;
```

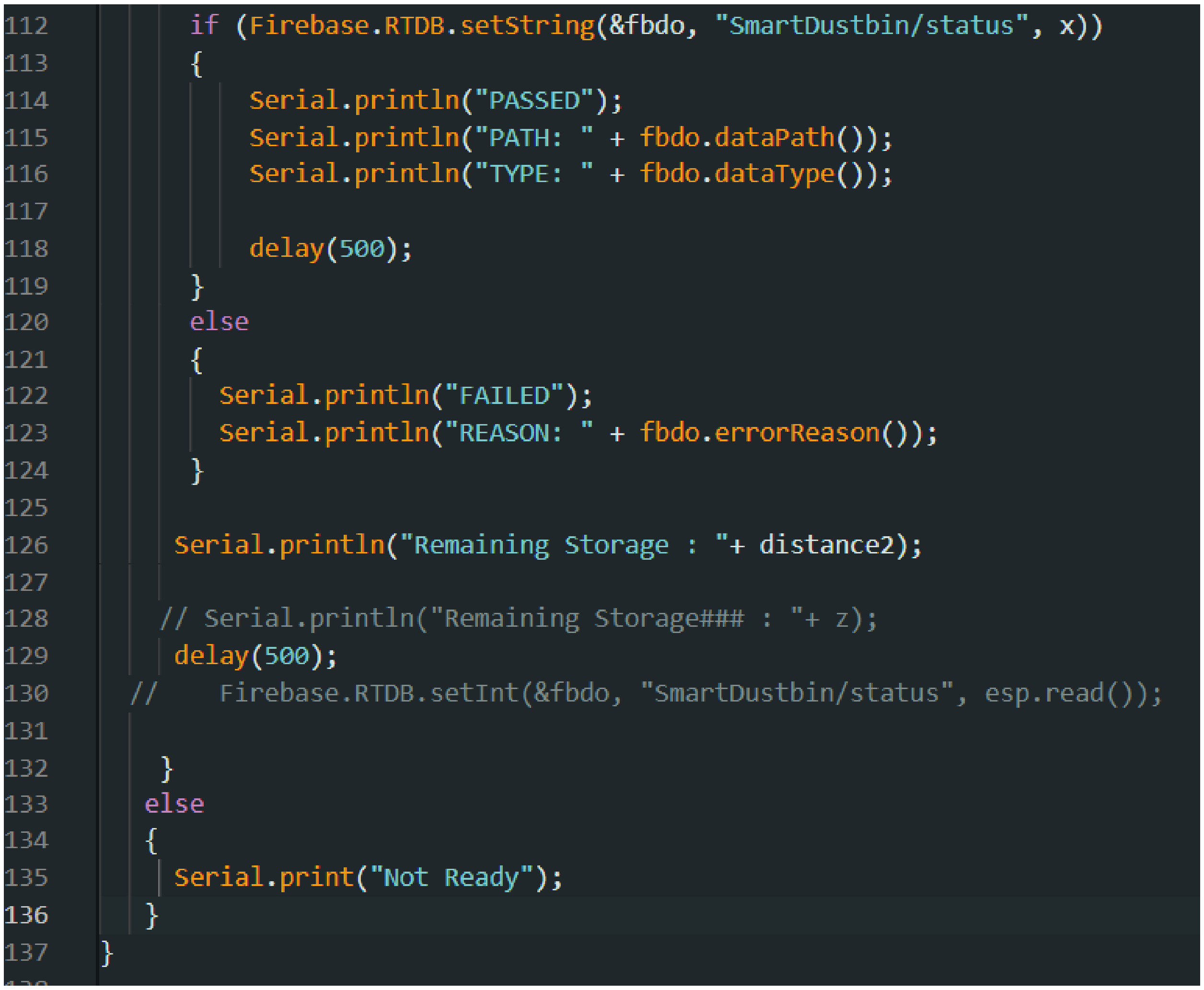


```
/* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
  else{
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);


}
```

The setup() function starts serial communication with the computer by calling Serial.begin(9600). At the same time, starts software serial communication with the ESP32 by calling esp.begin(9600). WiFi.begin(WIFI_SSID, WIFI_PASSWORD) is called to connects the specified WiFi network. Next the status of the connection to the serial monitor is printed out. Generally this function sets up the communication and authentication with Firebase, as well as establishes a connection to the WiFi network. If the connection is lost, Firebase.reconnectWiFi(true) will be called to reconnect to the WIFI.



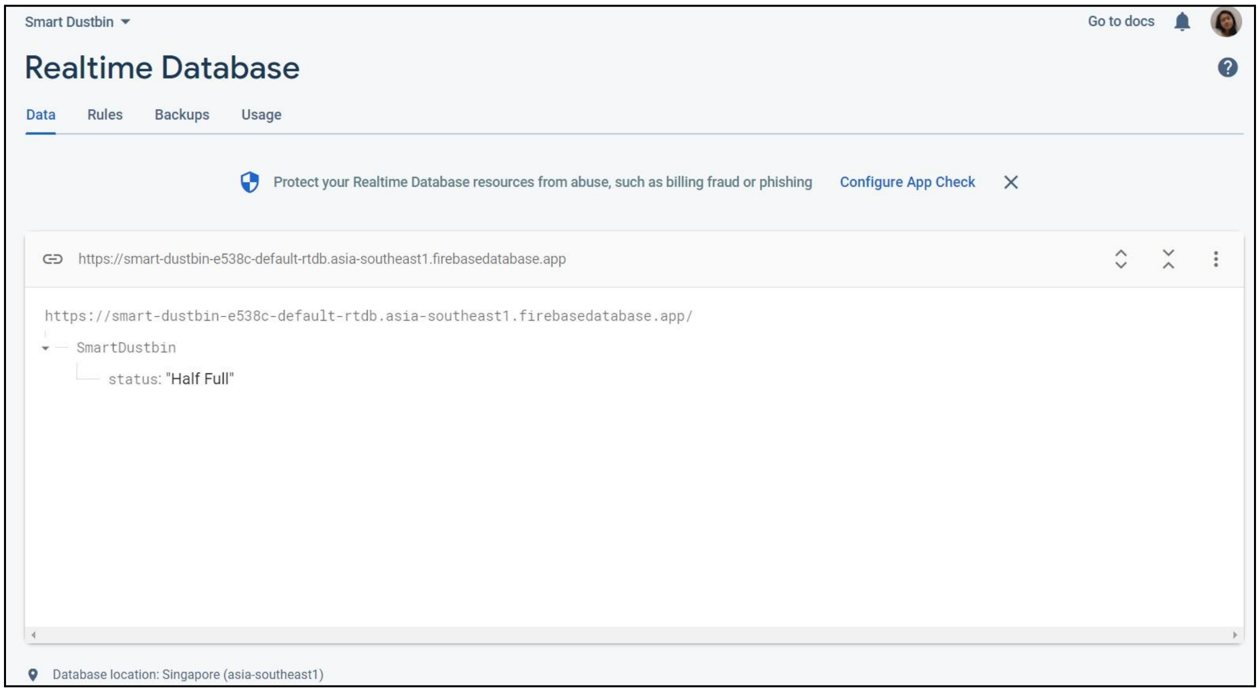
```
void loop()
{


   if (Firebase.ready() && signupOK)
   {

    String distance2= "";
      while (esp.available() > 0)
     {
      distance2 += char(esp.read());

     }

    int z = distance2.toInt();
    String x = "";

    if(z < 5 )
    {
      x = "Full";
    }
    else if ((z <=12) && (z >= 5))
    {
      x= "Half Full";

    }
    else
    {
      x= "Not Full";
    }
```



```
        if (Firebase.RTDB.setString(&fbdo, "SmartDustbin/status", x))
        {
          Serial.println("PASSED");
          Serial.println("PATH: " + fbdo.dataPath());
          Serial.println("TYPE: " + fbdo.dataType());

          delay(500);
        }
        else
        {
          Serial.println("FAILED");
          Serial.println("REASON: " + fbdo.errorReason());
        }

      Serial.println("Remaining Storage : "+ distance2);

     // Serial.println("Remaining Storage### : "+ z);
      delay(500);
    //    Firebase.RTDB.setInt(&fbdo, "SmartDustbin/status", esp.read());

     }
    else
    {
      Serial.print("Not Ready");
    }
}
```

In the loop() function, the code checks if the ESP32 is connected to the WiFi network and the Firebase connection is ready. Then it reads the value of the distance2 variable from the serial communication with the ESP32. Then it maps this distance to one of the three possible states (Full, Half Full, Not Full) of the dustbin and updates the corresponding value of the status variable in the Firebase RTDB.

Firebase


Smart Dustbin
Go to docs
Realtime Database
Data
Rules
Backups
Usage
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check

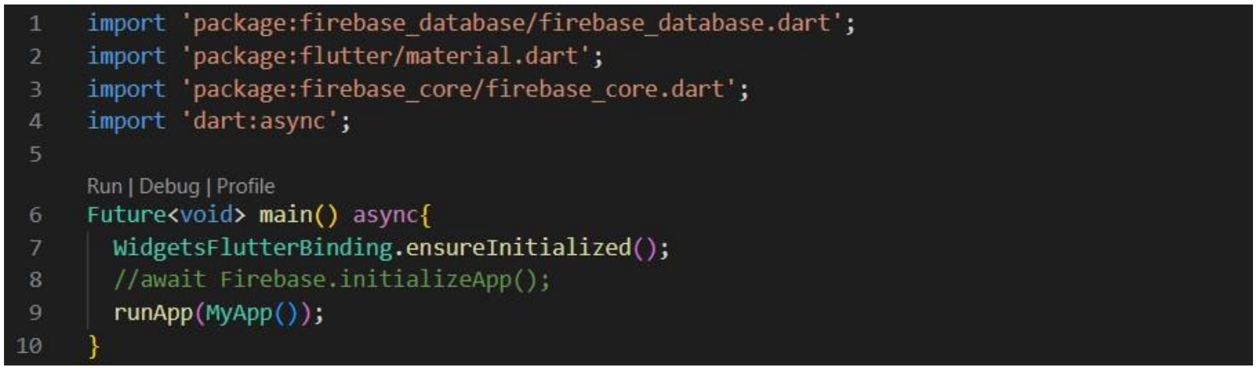
https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app

```
https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app/
▾ — SmartDustbin
      └── status: "Half Full"
```

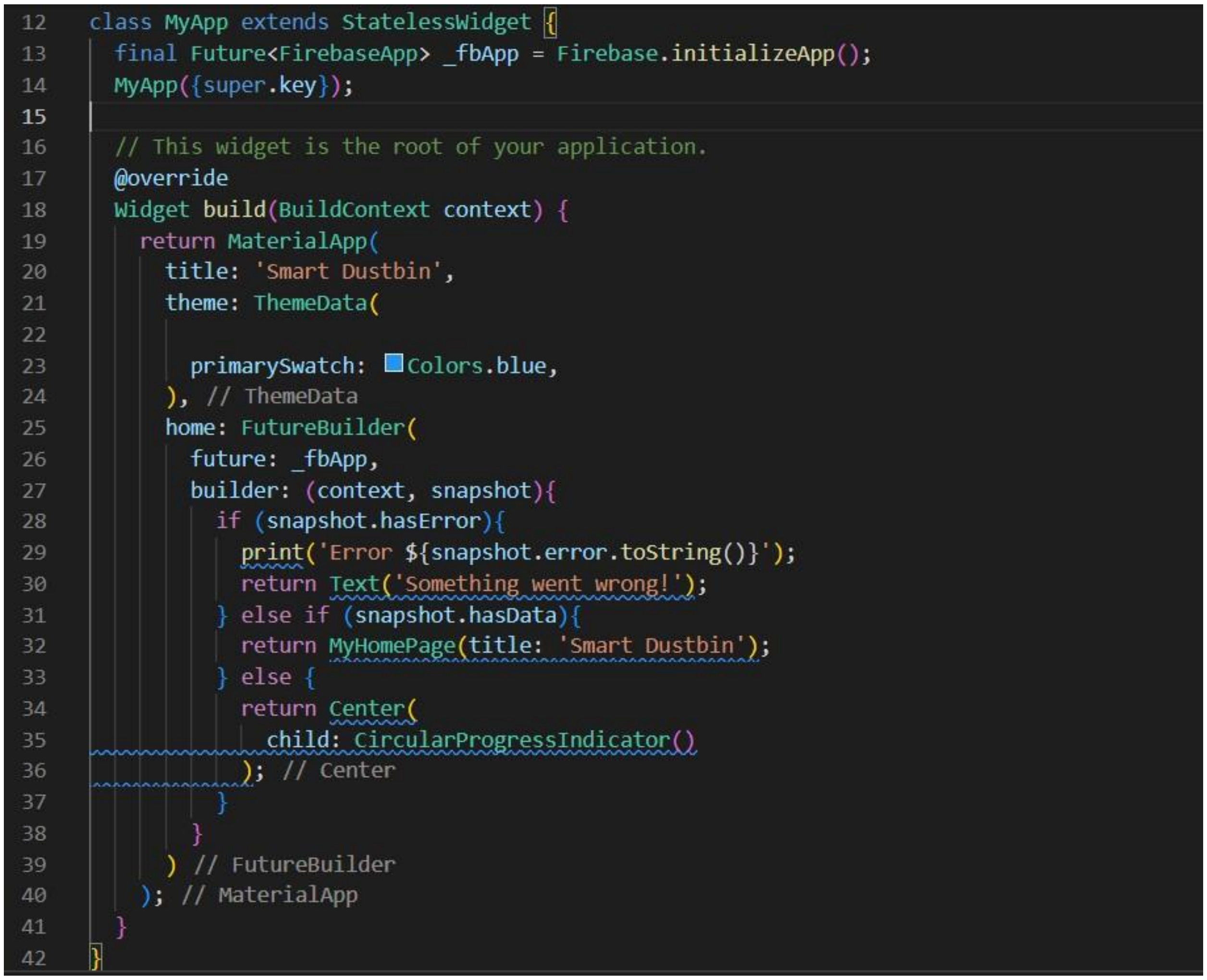
Database location: Singapore (asia-southeast1)

Firebase is a mobile and web application development platform developed by Google. It provides a number of tools and services to help developers build apps. In this project, a real time database is used. Real-time database is a cloud-hosted NoSQL database that allows developers to store and sync data across multiple clients in real-time.

Flutter



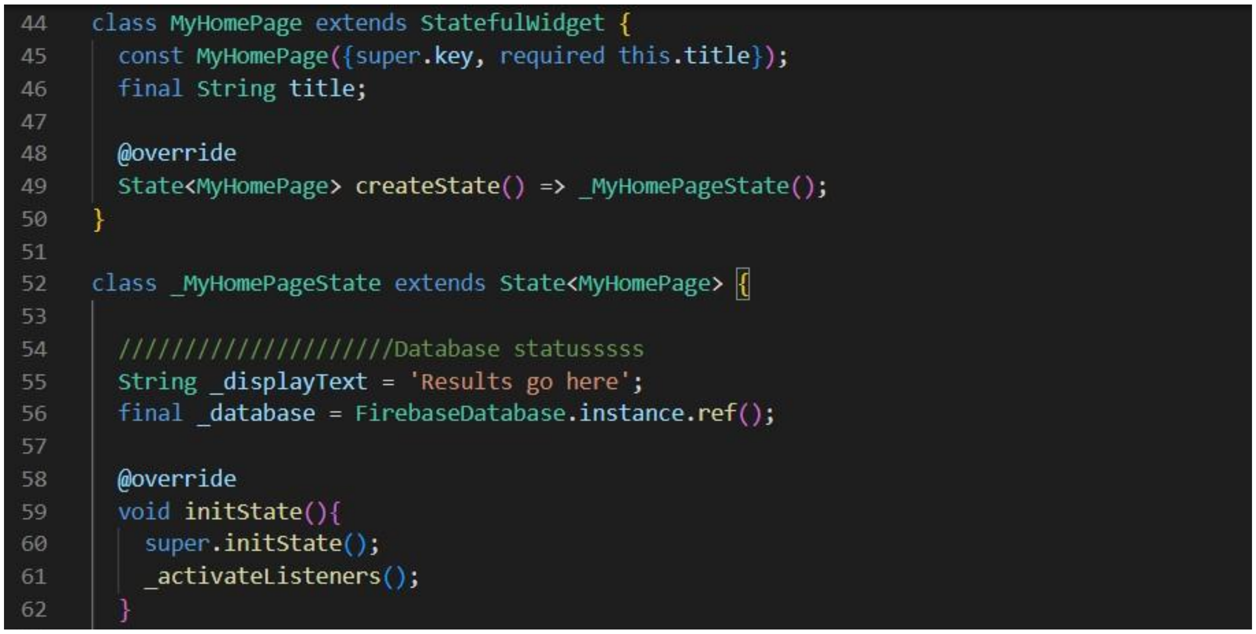
```
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'dart:async';

Run | Debug | Profile
Future<void> main() async{
  WidgetsFlutterBinding.ensureInitialized();
  //await Firebase.initializeApp();
  runApp(MyApp());
}
```



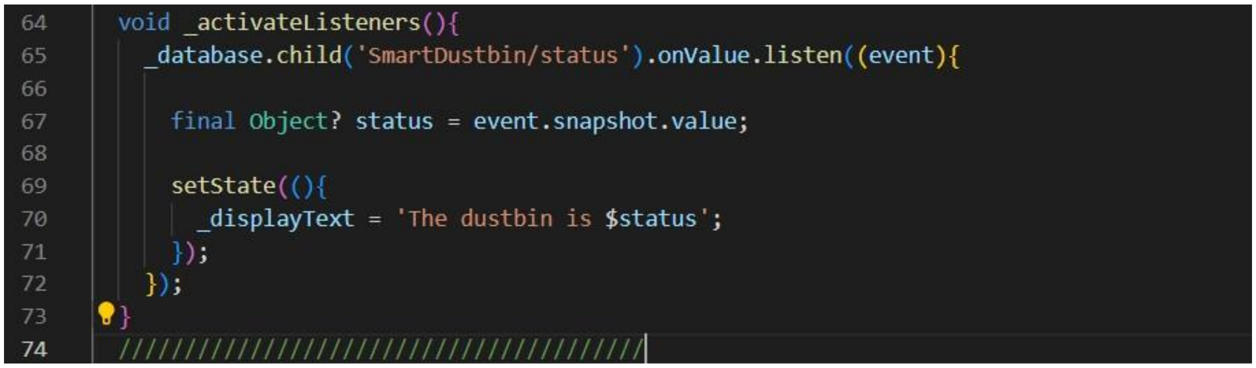
```
class MyApp extends StatelessWidget {
  final Future<FirebaseApp> _fbApp = Firebase.initializeApp();
  MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Smart Dustbin',
      theme: ThemeData(

        primarySwatch: Colors.blue,
      ), // ThemeData
      home: FutureBuilder(
        future: _fbApp,
        builder: (context, snapshot){
          if (snapshot.hasError){
            print('Error ${snapshot.error.toString()}');
            return Text('Something went wrong!');
          } else if (snapshot.hasData){
            return MyHomePage(title: 'Smart Dustbin');
          } else {
            return Center(
              child: CircularProgressIndicator()
            ); // Center
          }
        }
      ) // FutureBuilder
    ); // MaterialApp
  }
}
```

This code is a Flutter application that uses Firebase to display the status of a "smart dustbin." The main function ‘main()’ initializes the Flutter app and runs it. The ‘MyApp’ class is the root of the application and it returns a ‘MaterialApp’ widget with a title 'Smart Dustbin', a theme and a home page. The home page is a ‘FutureBuilder’ widget that waits for Firebase to initialize and then returns ‘MyHomePage’ widget.

It's also using ‘CircularProgressIndicator’ if Firebase is still initializing and if an error occurs during the Firebase initialization, it will print the error to the console and display 'Something went wrong!' on the screen.



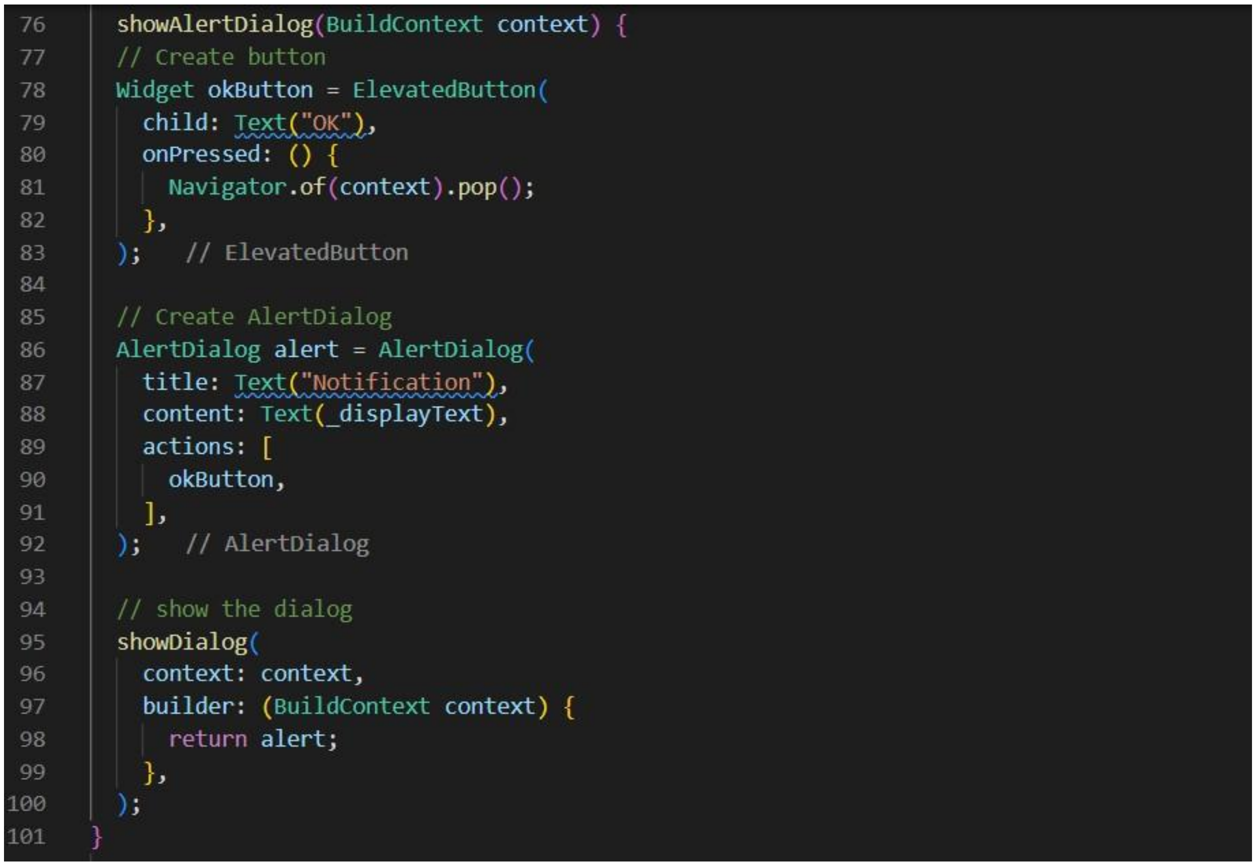
```dart
class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});
  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {

  ////////////////////Database statusssss
  String _displayText = 'Results go here';
  final _database = FirebaseDatabase.instance.ref();

  @override
  void initState(){
    super.initState();
    _activateListeners();
  }
```



```dart
  void _activateListeners(){
    _database.child('SmartDustbin/status').onValue.listen((event){

      final Object? status = event.snapshot.value;

      setState((){
        _displayText = 'The dustbin is $status';
      });
    });
  }
  ////////////////////////////////////////
```

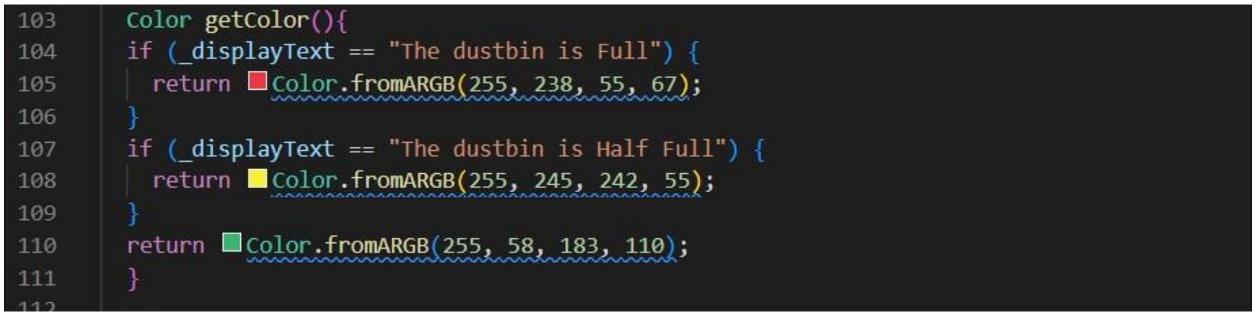
‘MyHomePage’ class is a stateful widget that displays the status of the dustbin in the form of a text and a color-coded circle. The status is obtained by listening to the 'SmartDustbin/status' node of the Firebase Realtime Database and displaying the value in the form of text. The color of the circle is determined by the status of the dustbin, which can be "Full", "Half Full" or "Not Full".

The '_MyHomePageState' class has a method '_activateListeners()' which is called in 'initState()' method, that listens to the 'SmartDustbin/status' node of the Realtime Database and updates the '_displayText' variable with the value.



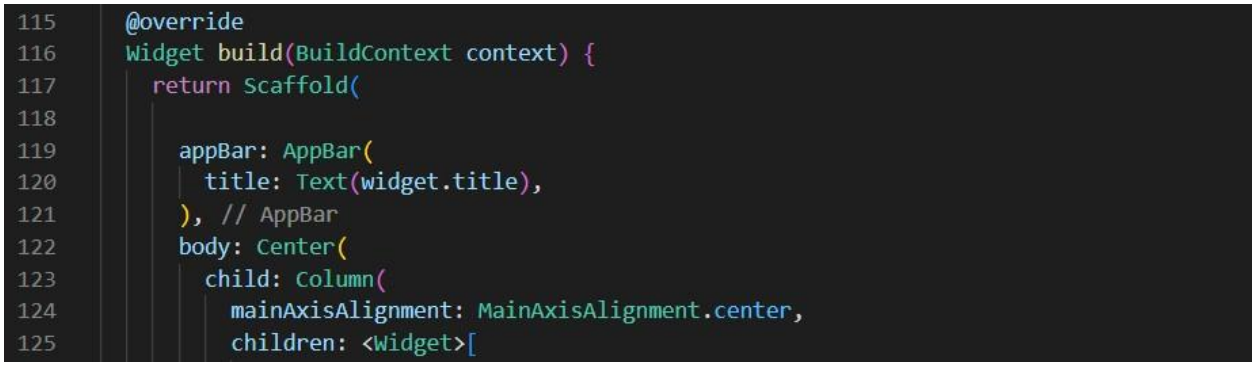
```
  showAlertDialog(BuildContext context) {
  // Create button
  Widget okButton = ElevatedButton(
    child: Text("OK"),
    onPressed: () {
      Navigator.of(context).pop();
    },
  );   // ElevatedButton

  // Create AlertDialog
  AlertDialog alert = AlertDialog(
    title: Text("Notification"),
    content: Text(_displayText),
    actions: [
      okButton,
    ],
  );   // AlertDialog

  // show the dialog
  showDialog(
    context: context,
    builder: (BuildContext context) {
      return alert;
    },
  );
}
```

The 'showAlertDialog(BuildContext context)' method is used to show the value of '_displayText' in an AlertDialog when the user clicks on the 'ElevatedButton'.

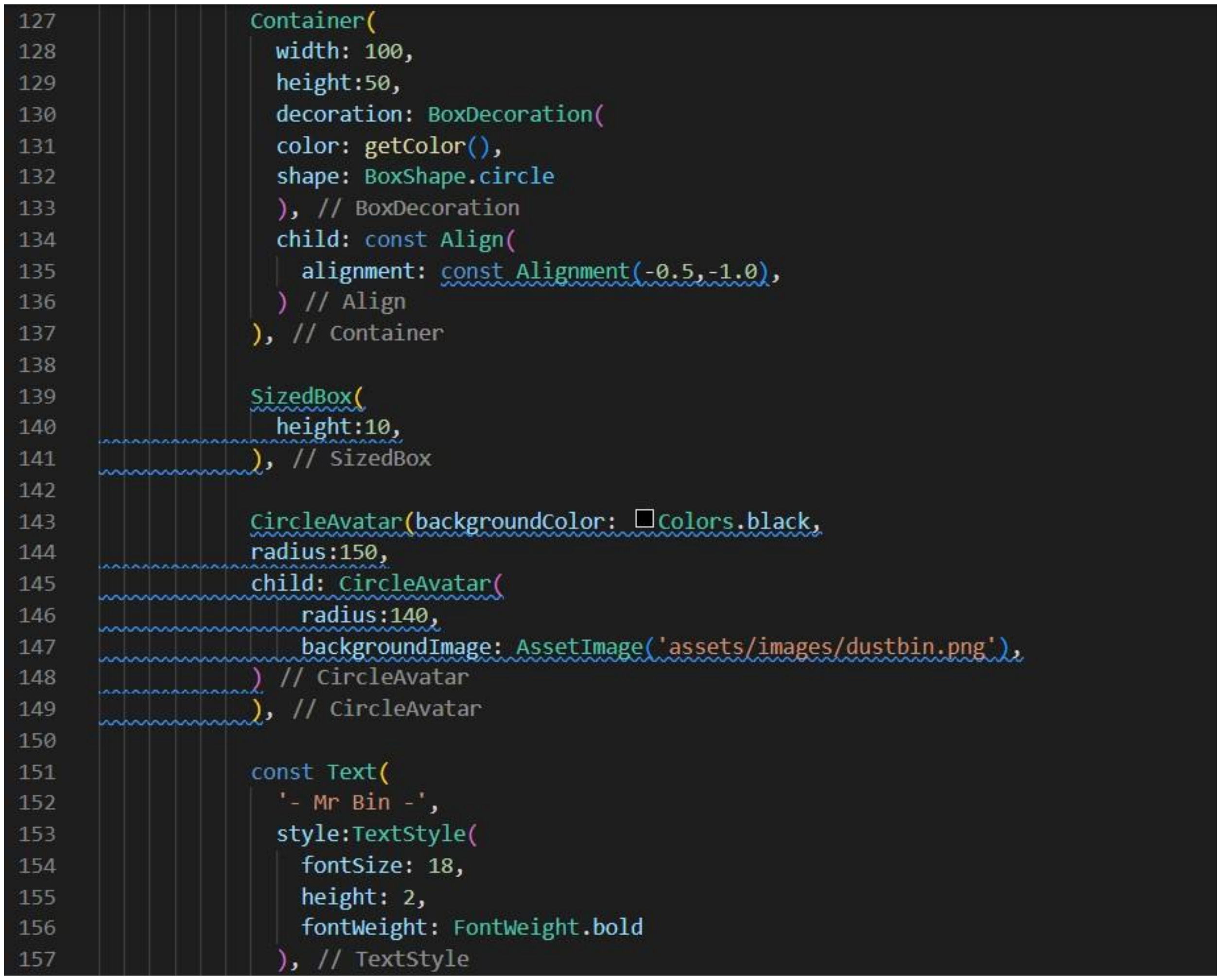


```
  Color getColor(){
  if (_displayText == "The dustbin is Full") {
    return Color.fromARGB(255, 238, 55, 67);
  }
  if (_displayText == "The dustbin is Half Full") {
    return Color.fromARGB(255, 245, 242, 55);
  }
  return Color.fromARGB(255, 58, 183, 110);
  }
```

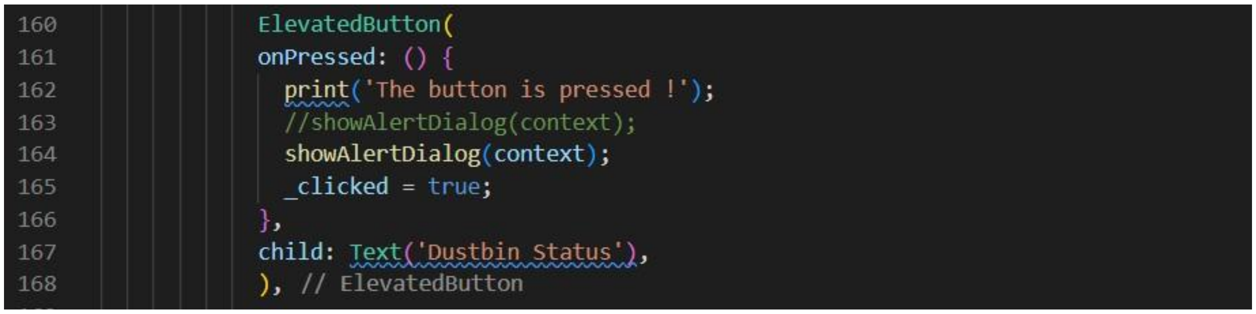
The 'getColor()' method is used to define the color of the circle to determine whether the dustbin is "Full", "Half Full" or "Not Full".



```dart
@override
Widget build(BuildContext context) {
  return Scaffold(

    appBar: AppBar(
      title: Text(widget.title),
    ), // AppBar
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
```



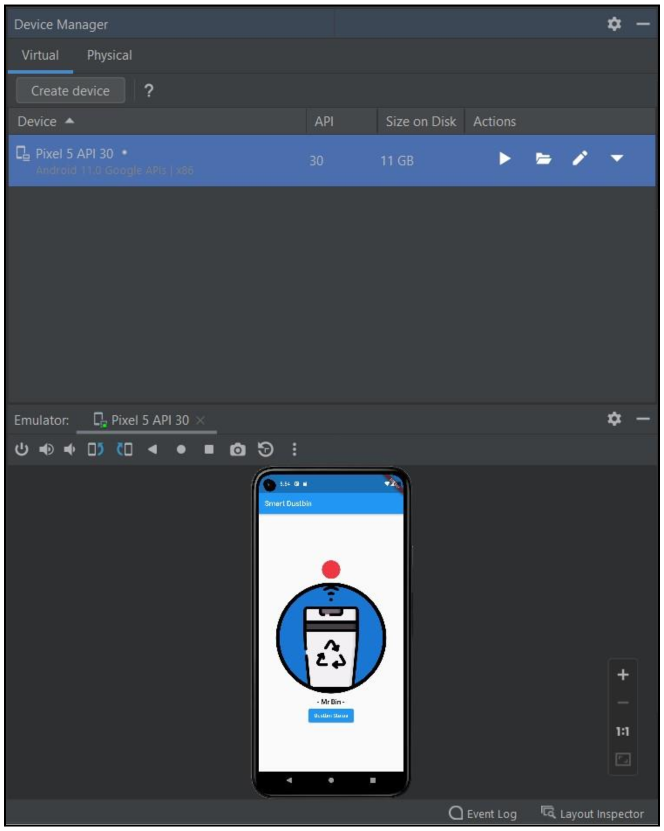
```dart
          Container(
            width: 100,
            height:50,
            decoration: BoxDecoration(
            color: getColor(),
            shape: BoxShape.circle
            ), // BoxDecoration
            child: const Align(
              alignment: const Alignment(-0.5,-1.0),
            ) // Align
          ), // Container

          SizedBox(
            height:10,
          ), // SizedBox

          CircleAvatar(backgroundColor: Colors.black,
          radius:150,
          child: CircleAvatar(
              radius:140,
              backgroundImage: AssetImage('assets/images/dustbin.png'),
          ) // CircleAvatar
          ), // CircleAvatar

          const Text(
            '- Mr Bin -',
            style:TextStyle(
              fontSize: 18,
              height: 2,
              fontWeight: FontWeight.bold
            ), // TextStyle
```



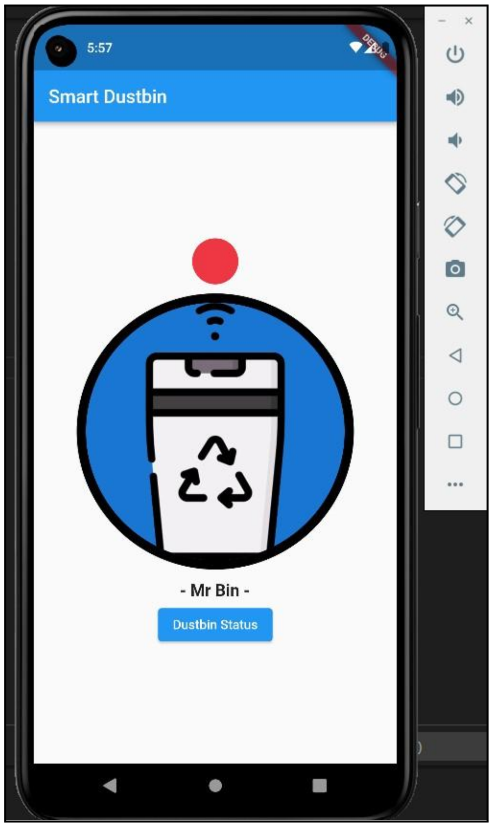
```dart
          ElevatedButton(
          onPressed: () {
            print('The button is pressed !');
            //showAlertDialog(context);
            showAlertDialog(context);
            _clicked = true;
          },
          child: Text('Dustbin Status'),
          ), // ElevatedButton
```

This code creates a 'Scaffold' widget in the 'build' method of a Flutter app, which is the basic layout structure of the app. The 'Scaffold' widget contains an 'AppBar' with a title, a 'Center' widget that aligns its children to the center, and a 'Column' widget that organizes its children in a vertical array. The 'Column' widget has several children, including a 'Container' widget with a 'BoxDecoration', a 'SizedBox' widget, a 'CircleAvatar' widget, a 'Text' widget, and an 'ElevatedButton' widget. The 'ElevatedButton' widget has an 'onPressed' callback that calls the 'showAlertDialog' method.

Android Studio



Device	API	Size on Disk	Actions
Pixel 5 API 30 • Android 11.0 Google APIs | x86	30	11 GB	



Android Studio is used to create an emulator. The emulator is able to test the application on many different devices virtually. The device that has been chosen is Pixel 5 API 30. The figures above show the Android Studio and the emulator. It enables the developers to test the application efficiently without keeping installing the application to the phone.

APPLICATION AND DATABASE ANALYSIS

With the rise of technology, humans are able to do anything and perform any tasks no matter where they are and where they go. Nowadays, it is undeniable that most people are dependent on technology as the existence of technology has made life become easier. One of the major growth can be easily seen is the development of applications. Nowadays almost everything can be accessed through smartphones with the help of internet connection.

The outcome of this project is the development of an application named Mr.Bin. This project was believed to be beneficial to any sector, mainly those in the cleaning service sector. Giving an apartment as an example, moving from one floor to another could be very troublesome as the cleaning workers need to check the status of the public dustbin at every level. This problem will be solved with the use of 'Mr. Bin' application where the real time data will be collected and the status of the dustbin will be updated through the application.

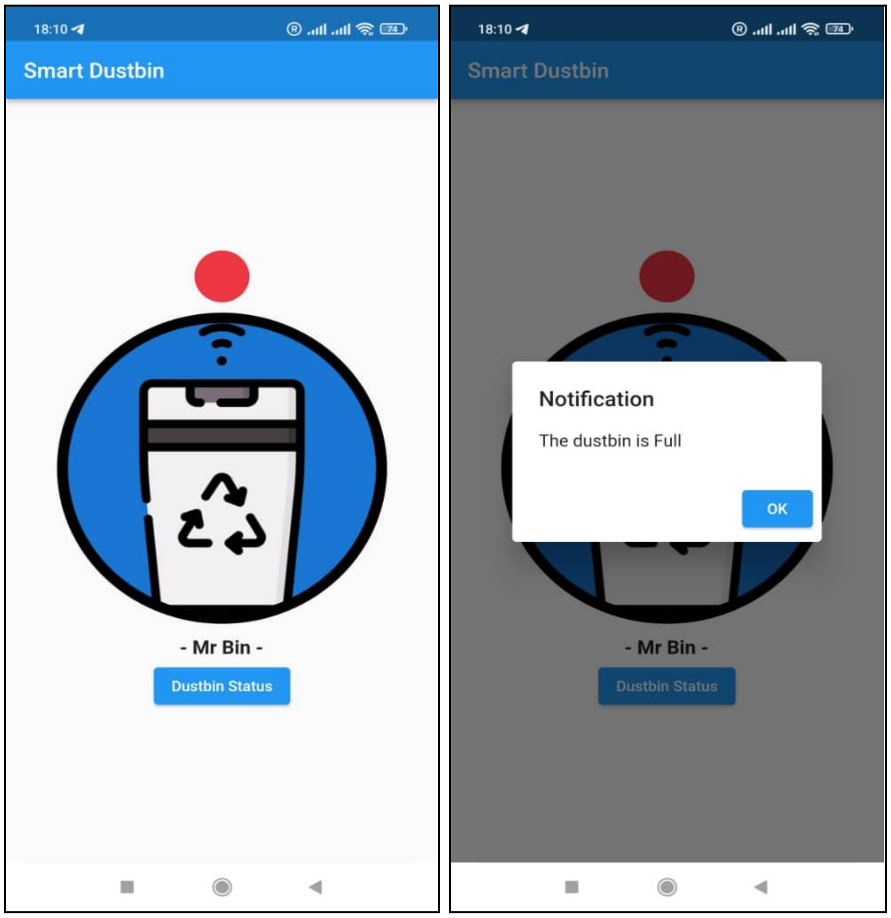
Features of the smart dustbin:

1) Motion detection to automatically open the dustbin lid when anyone is near.
2) Real time update of the dustbin status.

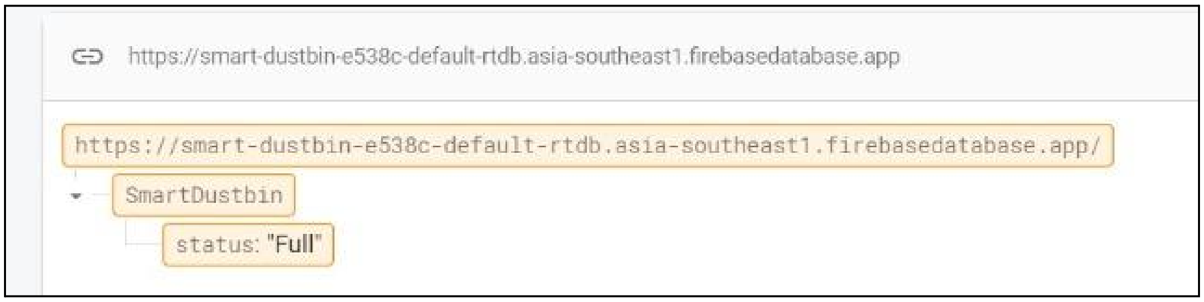
Dustbin status and database indicators:

1) When the dustbin is 'FULL', the RED LED on the dustbin will glow and the application will show a RED indicator with notification saying that the dustbin is already full.

Apps interface:



Database interface:

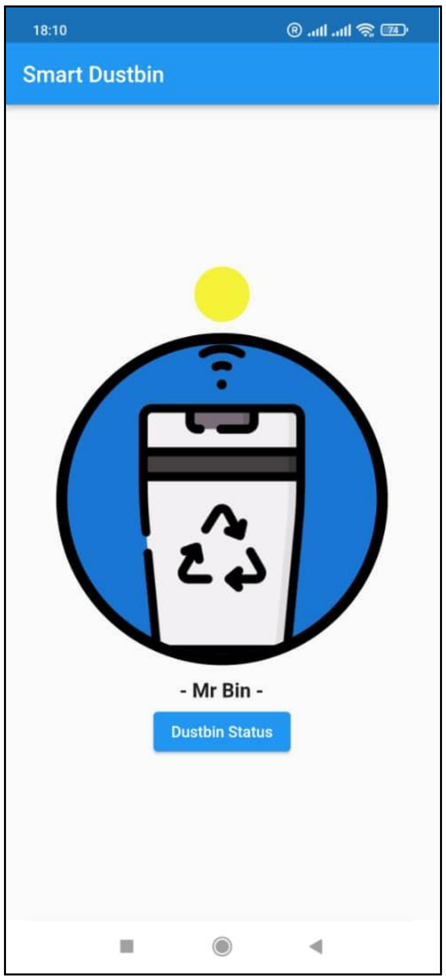


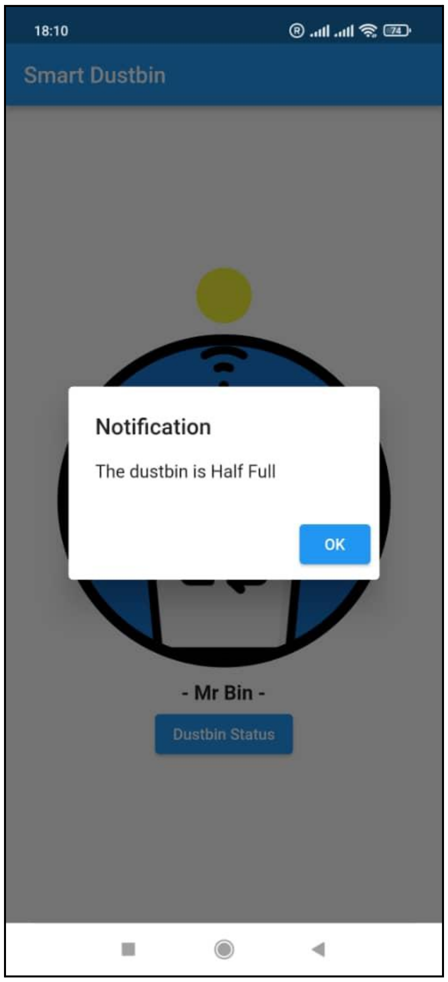
https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app

```
https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app/
  SmartDustbin
    status: "Full"
```

2) When the dustbin is ‘HALF FULL’, the BLUE LED on the dustbin will glow and the application will show a YELLOW indicator with notification saying that the dustbin is already half full.

Apps interface:

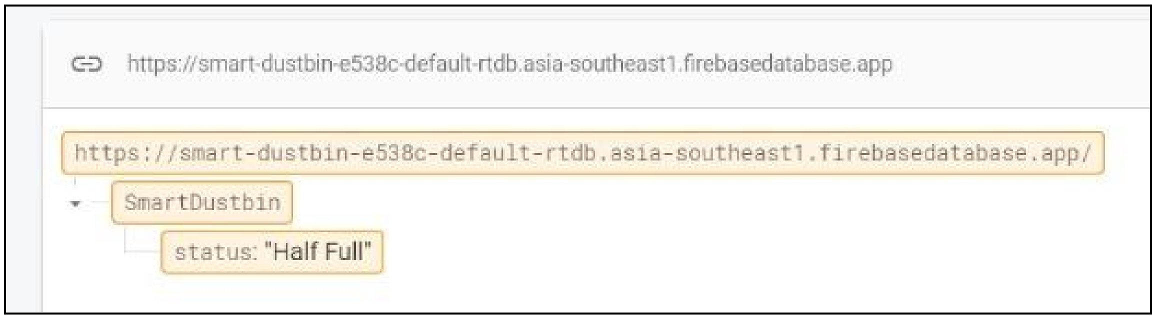




Notification

The dustbin is Half Full

Database interface:

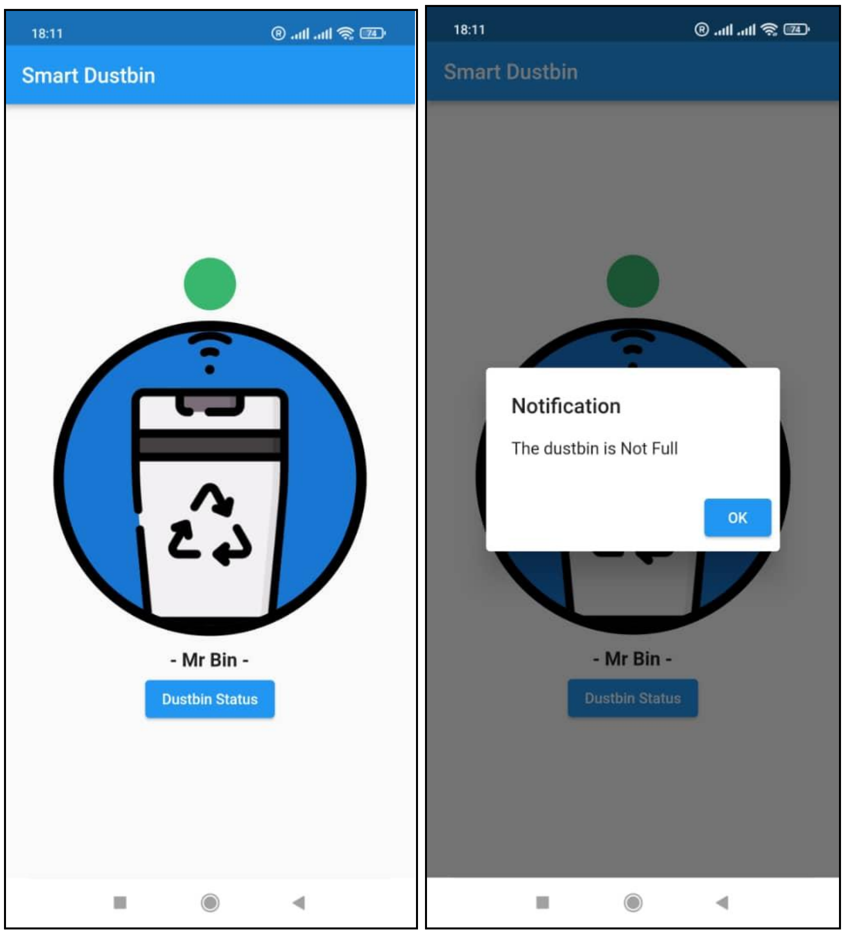


https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app

```
https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app/
  SmartDustbin
    status: "Half Full"
```

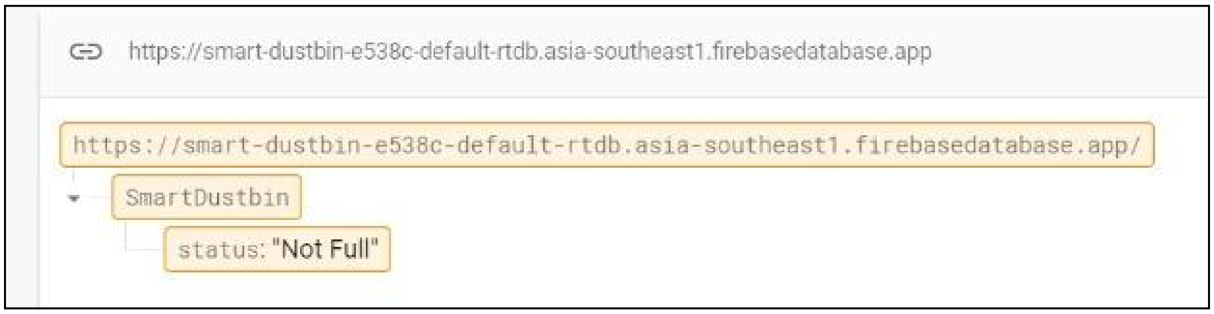
3) When the dustbin is ‘NOT FULL’, NO LED on the dustbin will glow and the application will show a GREEN indicator with notification saying that the dustbin is still not full.

Apps interface:



The dustbin is Not Full

Database interface:

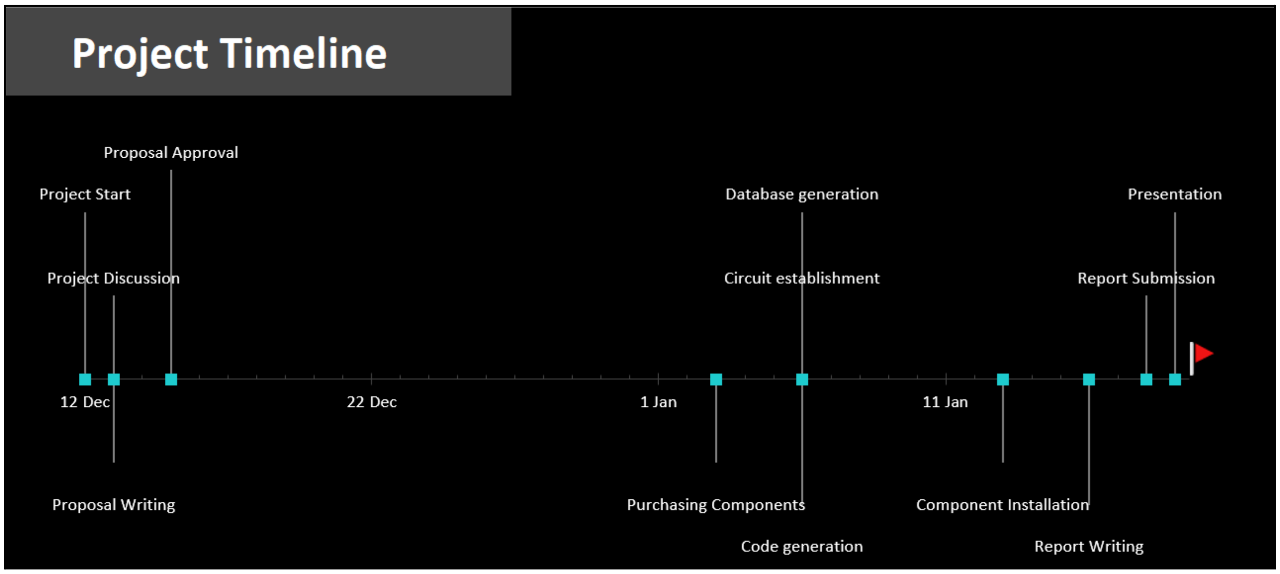


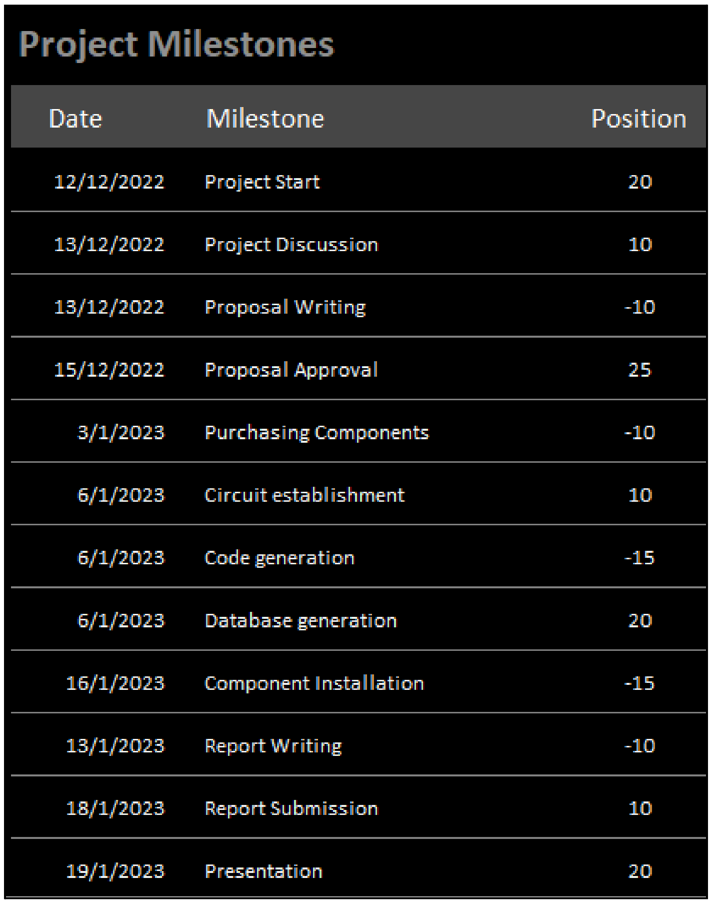
https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app

```
https://smart-dustbin-e538c-default-rtdb.asia-southeast1.firebasedatabase.app/
  SmartDustbin
    status: "Not Full"
```

CHAPTER 3

PROJECT TIMELINE

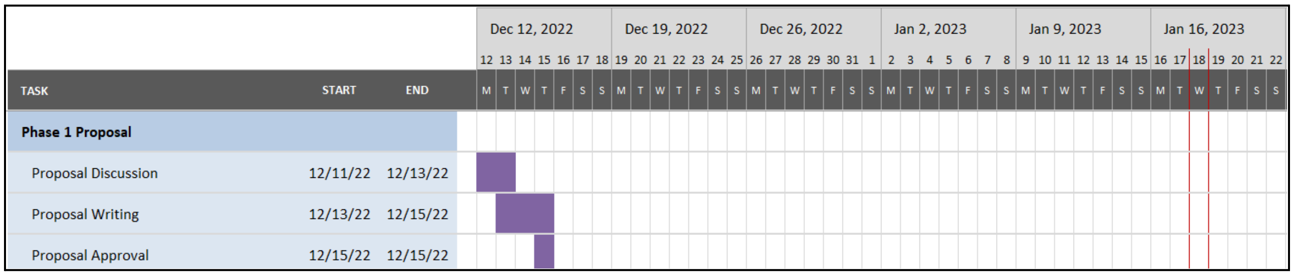




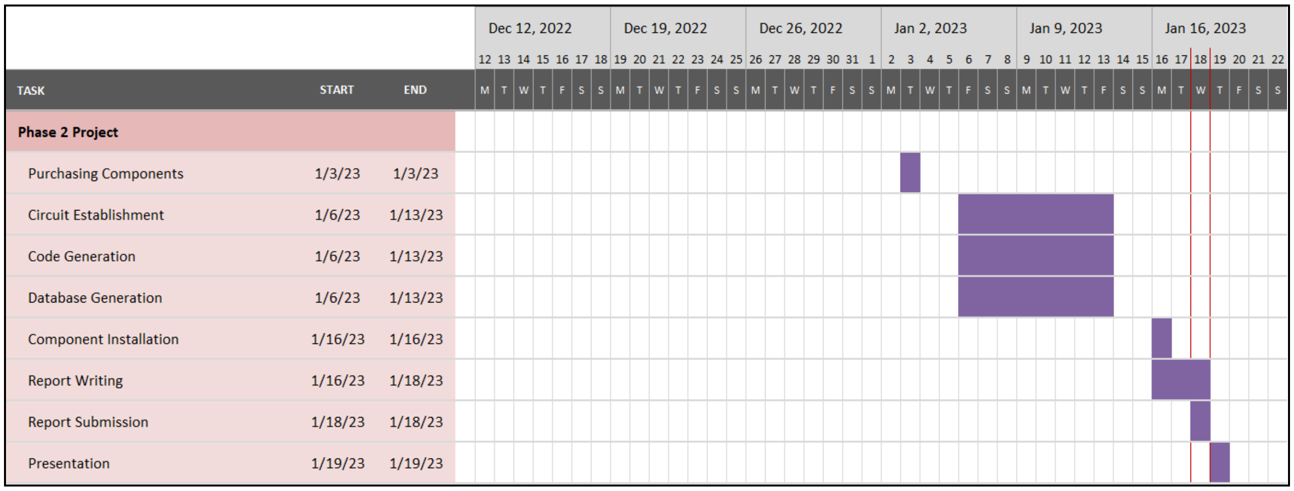
Project Milestones

Date	Milestone	Position
12/12/2022	Project Start	20
13/12/2022	Project Discussion	10
13/12/2022	Proposal Writing	-10
15/12/2022	Proposal Approval	25
3/1/2023	Purchasing Components	-10
6/1/2023	Circuit establishment	10
6/1/2023	Code generation	-15
6/1/2023	Database generation	20
16/1/2023	Component Installation	-15
13/1/2023	Report Writing	-10
18/1/2023	Report Submission	10
19/1/2023	Presentation	20

GANTT CHART



TASK	START	END
Phase 1 Proposal		
Proposal Discussion	12/11/22	12/13/22
Proposal Writing	12/13/22	12/15/22
Proposal Approval	12/15/22	12/15/22



TASK	START	END
Phase 2 Project		
Purchasing Components	1/3/23	1/3/23
Circuit Establishment	1/6/23	1/13/23
Code Generation	1/6/23	1/13/23
Database Generation	1/6/23	1/13/23
Component Installation	1/16/23	1/16/23
Report Writing	1/16/23	1/18/23
Report Submission	1/18/23	1/18/23
Presentation	1/19/23	1/19/23

CHAPTER 4

CONCLUSION

In modern life, IoT devices have become essentials to human beings. Be it a school, office, house, room and even a car uses electronic devices to operate smoothly these days. As there are a lot of sectors that are using devices to help with daily chores, the workload is also increasing. This leads to the evolution of IoT being implemented to current devices. In this project, the everyday dustbin will be added with a smart function to reduce user's time and energy to open the dustbin lid while also alerting them to throw away the rubbish if the bin is full.

Despite having the challenges such as having difficulties to build this project, gathering the components and time constraint, this project is believed to benefit a lot of users as it will give a lot of advantages in many ways. One of them is that it reduces human contact with the dustbin which can be quite an issue these days especially due to the spark of covid-19 in Malaysia. Next, this can help those with health issues to not be hunched over to reach the dustbin lid. This will also motivate the youngsters to always throw away their rubbish in the dustbin since it is easier as the lids automatically open.

This world has always been clean and beautiful. Humans are the ones that produce waste and humans too need to take care of them. Therefore, to maintain a clean and healthy ecosystem, each and everyone needs to play their role. With this project, researchers hope to serve their part by making throwing rubbish easier with the implementation of automatic lid and to remind human beings to always send their collected rubbish to the rubbish collector for proper rubbish disposal.

REFERENCE

Kumar, S., Tiwari, P., & Zymbler, M. (2019). Internet of Things is a revolutionary approach for future technology enhancement: a review. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0268-2

World Population Clock: 8 Billion People (LIVE, 2022) - Worldometer. (n.d.). https://www.worldometers.info/world-population/

Malaysia Population (2022) - Worldometer. (n.d.). https://www.worldometers.info/world-population/malaysia-population/

Appendix