Om Kumar Singh

[Email address]

Garbage Monitoring System

IOT Based Solution

Team Members

1.Om Kumar Singh (20JE0650)

2.Harsh Raj (20JE0401)

**Acknowledgement**

We would like to express our sincere gratitude to our project guide, Prof Tarachand Amgoth for his invaluable guidance and support throughout the project. We would also like to thank our college, IIT(ISM) Dhanbad and NVCTI, for providing us with the necessary resources and infrastructure to carry out this project successfully.

Lastly, we would like to thank our friends for their constant encouragement and motivation. Their unwavering support kept us motivated and helped us to overcome the challenges faced during the project.

**Abstract**

This documentation presents an app-based IoT solution for a garbage collection system. The solution is designed to provide an efficient and automated waste management system by leveraging the capabilities of IoT technology. The solution includes a mobile application that allows residents to schedule garbage pickups, track the collection process, and receive notifications. The system also comprises of sensors and IoT-enabled devices that monitor the garbage level in the bins and communicate with the app to optimize the collection process.

The solution is designed to address the challenges of traditional waste management systems, such as inefficient garbage collection, overflowing bins, and missed pickups. By leveraging the capabilities of IoT technology, the solution enables a smarter and more efficient waste management system, resulting in improved cleanliness, reduced environmental impact, and increased cost savings.

This documentation outlines the system architecture, hardware and software requirements, and the functionality of the app-based solution. Additionally, the documentation provides a detailed overview of the development process, including the design, implementation, and testing of the solution. Overall, the documentation serves as a comprehensive guide for the app-based IoT solution for a garbage collection system, providing a roadmap for municipalities and waste management companies looking to implement a smarter waste management system.

**Content**

1. Introduction
2. Problem Statement
3. Our Approach
4. Experiment
5. Hardware Explanation
6. Software Explanation
7. Results
8. Conclusion
9. References

**Introduction**

In our IoT class, our professor has assigned us an exciting project that challenges us to design and develop an IoT product that addresses a specific need or problem. This project requires us to leverage our knowledge of IoT technologies, including sensors, networks, and cloud computing, to create a product that is both innovative and practical.

As an IoT project, this endeavor provides us with a unique opportunity to work on cutting-edge technologies and develop a product that has the potential to impact people's lives in meaningful ways. Through this project, we will gain hands-on experience in the entire IoT product development process, from ideation and prototyping to testing and deployment. Additionally, we will learn about the key challenges and opportunities associated with developing and deploying IoT products in real-world settings.This project also requires us to work collaboratively in interdisciplinary teams, which is an essential skill in the field of IoT.

During our IoT class, our professor presented us with a range of IoT product examples, which sparked our imaginations and inspired us to develop our own unique idea for this project. Drawing on the lessons and insights we gained from these examples, we developed a vision for an IoT product that addresses a specific need in a novel way.

Overall, this IoT project provides an exciting opportunity for us to apply our knowledge of IoT technologies to real-world problem-solving, and to gain valuable experience in IoT product development. We are thrilled to take on this challenge and look forward to developing an innovative and impactful IoT product.

**Problem Statement**

**The improper disposal of waste has become a major environmental and health concern in urban areas. It leads to pollution, health hazards, and other negative impacts on the environment. One of the significant challenges faced by municipalities is the inefficient management of garbage disposal. Traditional garbage management systems rely on scheduled pickups or manual monitoring, which often results in overflowing bins, littering, and inefficient resource allocation. In this context, there is a need for a modern and efficient system that can monitor the garbage level in real-time and optimize garbage collection processes.**



**Our Approach**

The idea for the project came to our minds when we observed the inefficient garbage collection and management practices in our city. We noticed that garbage bins were often overflowing, leading to an unpleasant sight and odor. Also, the garbage collection schedule was not consistent, resulting in delayed pickups, which further added to the problem.

To address this issue, we brainstormed ideas and came up with the concept of developing a garbage monitoring system using IoT. We thought that real-time monitoring of the garbage level in bins could help plan the collection schedule accordingly and avoid the overflow of garbage.



We began by researching the various technologies available for garbage monitoring and management. We explored different types of sensors that could be used to measure the garbage level and the communication protocols that could be employed to send data to a central server.

Overall, the project was a success, and we believe that our garbage monitoring system using IoT can be a potential solution to the garbage management problem in cities and towns worldwide.

**Experiment**

* Hardware

Components Used :-

1. ESP 32

The ESP32 is a powerful and versatile microcontroller module developed by Espressif Systems. It is an improved version of the ESP8266 module, with more processing power, memory, and advanced features. The ESP32 module includes a dual-core Tensilica LX6 microprocessor, WiFi, Bluetooth, and a range of peripherals such as SPI, I2C, UART, and ADC.

The ESP32 also features low power consumption, making it suitable for battery-powered IoT applications. It can be programmed using the Arduino IDE, ESP-IDF (Espressif IoT Development Framework), and other programming languages such as MicroPython and JavaScript.

The ESP32 is widely used in IoT applications, including home automation, smart agriculture, industrial automation, and robotics. Its advanced features, low power consumption, and high processing power make it an ideal choice for developing IoT devices that require wireless connectivity, data processing, and sensor integration.

1. Ultrasonic Sensor

An ultrasonic sensor is a device that uses high-frequency sound waves to detect the presence and proximity of objects. The sensor emits a series of ultrasonic waves, which bounce off any nearby object and return to the sensor. The time taken for the waves to return is measured, and this data is used to calculate the distance between the sensor and the object.

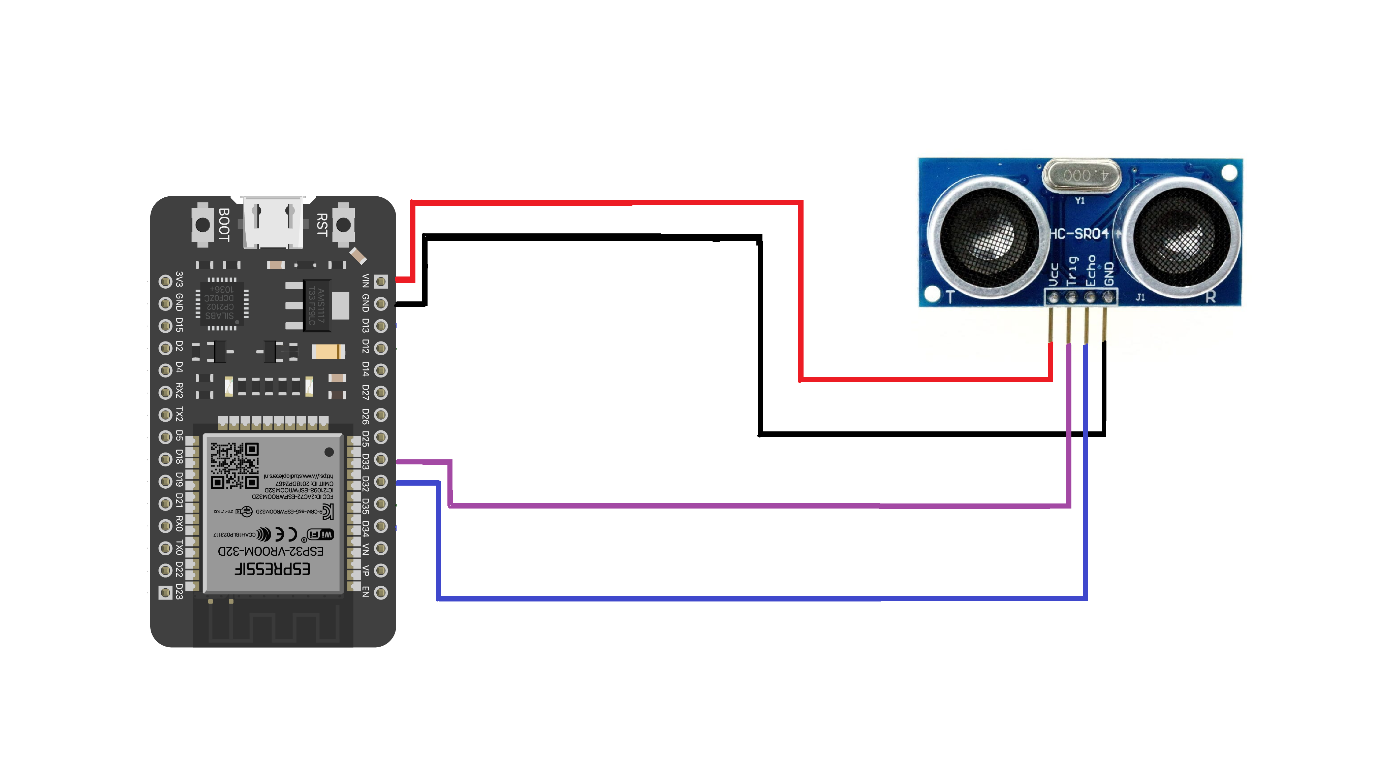
Ultrasonic sensors are commonly used in robotics, automation, and IoT applications for distance measurement, obstacle detection, and object tracking. They can operate at high frequencies, typically in the range of 20 kHz to 200 kHz, and can detect objects at a distance of up to several meters. They are also immune to color, transparency, and lighting conditions, making them ideal for use in a variety of environments.

Ultrasonic sensors can be either analog or digital, with the digital sensors providing a discrete output indicating the presence or absence of an object, while analog sensors provide a continuous output proportional to the distance between the sensor and the object. They can be interfaced with microcontrollers such as Arduino, Raspberry Pi, and ESP32, and programmed to perform various tasks based on the distance and presence of objects.

1. Jumper Wires

Jumper wires are essential components for prototyping and experimenting with electronics projects. They are easy to use, reliable, and can be reused multiple times. They are compatible with a wide range of electronic components such as resistors, capacitors, sensors, and microcontrollers, making them a versatile tool for electronics hobbyists and professionals alike.

Block Diagram



* **Software Used**

1. Arduino Ide

The Arduino Integrated Development Environment (IDE) is a software application used to program and develop applications for the Arduino microcontroller platform. The IDE provides a user-friendly interface for writing, compiling, and uploading code to an Arduino board.

The Arduino IDE is based on the Processing programming environment and uses the C++ programming language. It provides a set of libraries and functions to interface with the hardware and sensors connected to the Arduino board. The IDE also includes a serial monitor that allows users to communicate with the Arduino board and receive data from sensors in real-time.

1. ThingSpeak

ThingSpeak is an open-source IoT platform that allows developers to collect, store, analyze, and visualize data from connected devices or sensors. It provides a web-based interface and APIs (Application Programming Interfaces) to manage IoT data streams and create IoT applications.

ThingSpeak supports a wide range of IoT protocols, such as MQTT, HTTP, and TCP, making it easy to connect devices and sensors from different manufacturers. It also includes a cloud-based data storage system and real-time data analytics tools to help developers extract valuable insights from their data.

ThingSpeak provides an easy-to-use MATLAB Analytics Engine that enables developers to create complex algorithms and models for advanced data analysis. It also integrates with other popular IoT platforms such as Arduino, Raspberry Pi, and ESP8266.

The platform is highly customizable, allowing developers to build and deploy their own IoT applications tailored to their specific needs. ThingSpeak is free to use, but it also offers a paid version with additional features such as larger data storage and advanced analytics capabilities.

Overall, ThingSpeak is an excellent platform for developers and IoT enthusiasts who want to build and deploy IoT applications quickly and easily.

Code :-

1. #include <WiFi.h>
2. #include "secrets.h"
3. #include "ThingSpeak.h" // always include thingspeak header file after other header files and custom macros
4. char ssid[] = SECRET\_SSID;   // your network SSID (name)
5. char pass[] = SECRET\_PASS;   // your network password
6. int keyIndex = 0;            // your network key Index number (needed only for WEP)
7. WiFiClient  client;
8. #define TRIGGER\_PIN 33
9. #define ECHO\_PIN 32
10. unsigned long myChannelNumber = SECRET\_CH\_ID;
11. const char \* myWriteAPIKey = SECRET\_WRITE\_APIKEY;
12. float duration = 0;
13. float distance = 0;
14. void setup() {
15. Serial.begin(9600);  //Initialize serial
16. pinMode(TRIGGER\_PIN, OUTPUT);
17. pinMode(ECHO\_PIN, INPUT);
18. while (!Serial) {
19. ; // wait for serial port to connect. Needed for Leonardo native USB port only
20. }
22. WiFi.mode(WIFI\_STA);
23. ThingSpeak.begin(client);  // Initialize ThingSpeak
24. }
25. void loop() {
26. // Connect or reconnect to WiFi
27. if(WiFi.status() != WL\_CONNECTED){
28. Serial.print("Attempting to connect to SSID: ");
29. Serial.println(SECRET\_SSID);
30. while(WiFi.status() != WL\_CONNECTED){
31. WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP network
32. Serial.print(".");
33. delay(5000);
34. }
35. Serial.println("\nConnected.");
36. }
37. // Write to ThingSpeak. There are up to 8 fields in a channel, allowing you to store up to 8 different
38. // pieces of information in a channel.  Here, we write to field 1.
39. int x = ThingSpeak.writeField(myChannelNumber, 1, distance, myWriteAPIKey);
40. // - myChannelNumber: the ID number of the ThingSpeak channel to write to
41. // - 1: the field number to write the data to
42. // - distance: the value to write to the field (in this case, the distance measurement)
43. // - myWriteAPIKey: the API key that provides write access to the ThingSpeak channel
45. if(x == 200){
46. Serial.println("Channel update successful.");
47. }
48. else{
49. Serial.println("Problem updating channel. HTTP error code " + String(x));
50. }
52. // change the value
53. digitalWrite(TRIGGER\_PIN, LOW);
54. delayMicroseconds(2);
55. digitalWrite(TRIGGER\_PIN, HIGH);
56. delayMicroseconds(10);
57. digitalWrite(TRIGGER\_PIN, LOW);
58. // Read the duration of the ultrasonic pulse from the sensor
59. duration = pulseIn(ECHO\_PIN, HIGH);
60. // Calculating the distance
61. distance= duration\*0.034/2;
62. // The speed of sound in air is approximately 340 m/s, or 0.034 cm/us.
63. // The distance is calculated as half the total distance traveled by the sound wave: from the sensor to the object, and then back again.
64. // Therefore, the distance is calculated as (speed of sound \* duration) / 2, where duration is the time it took for the reflected sound wave to return to the sensor.
65. // The resulting distance value is in centimeters.
66. // Print the distance to the serial monitor
67. Serial.print("Distance: ");
68. Serial.print(distance);
69. Serial.println(" cm");
71. delay(20000); // Wait 20 seconds to update the channel again
72. }
73. MIT App Inventor

MIT App Inventor is a web-based platform that allows users to create mobile applications for Android devices using a visual programming language. It provides an intuitive drag-and-drop interface that allows users to build applications quickly and easily without the need for extensive programming knowledge.

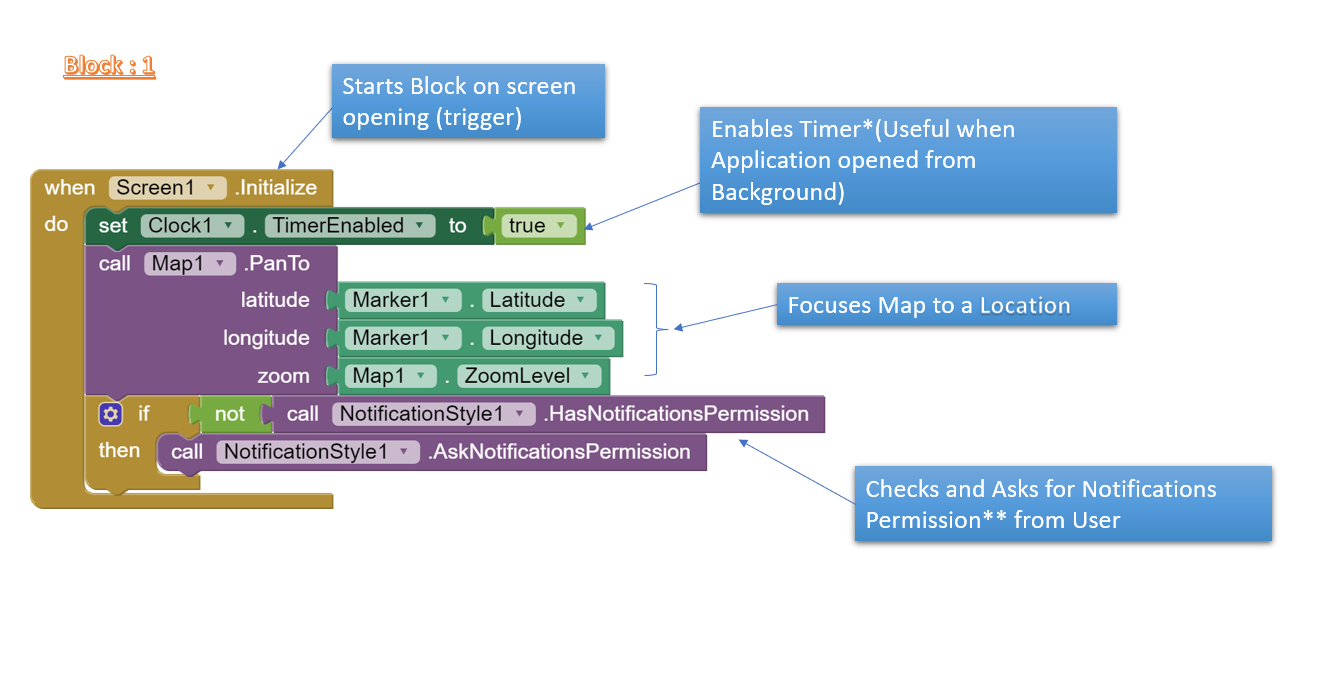
The platform is based on the block-based programming language Scratch, which uses colorful blocks to represent programming concepts such as loops, variables, and conditionals. This makes it easy for beginners to learn programming concepts and build their own mobile applications.

MIT App Inventor includes a wide range of pre-built components and modules that can be easily integrated into applications, including sensors, data storage, media, and social media integration. It also allows users to test and preview their applications in real-time on their Android devices using the App Inventor Companion App.

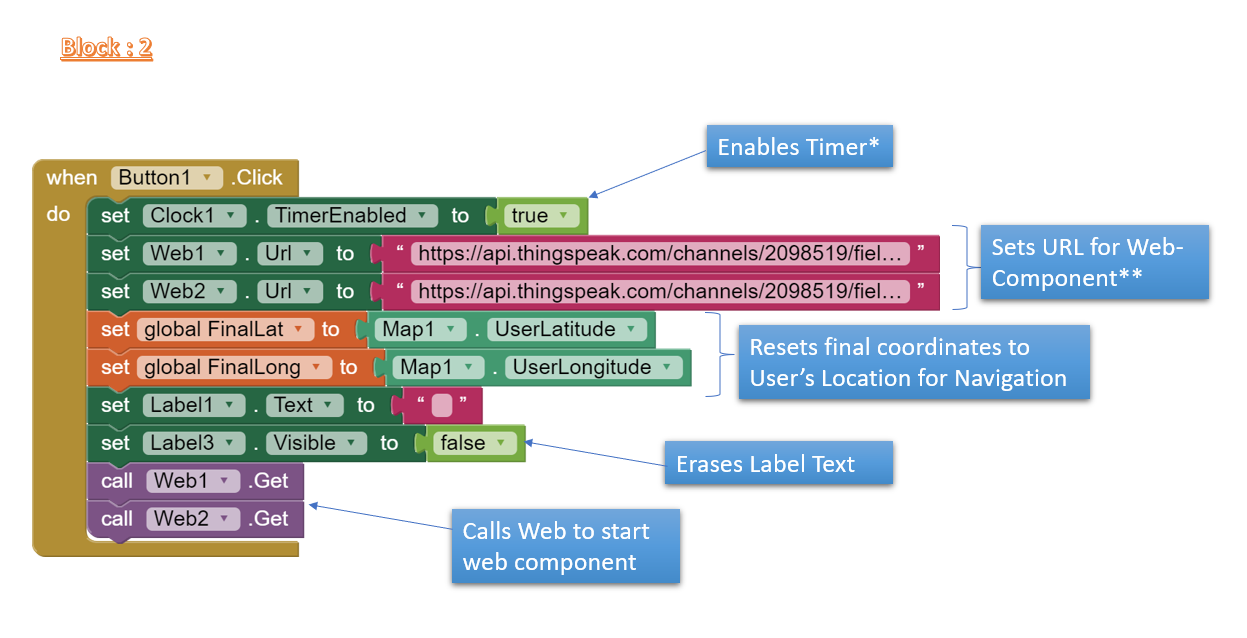
Overall, MIT App Inventor is an excellent platform for anyone who wants to learn programming and build their own mobile applications quickly and easily. Its intuitive interface and extensive library of components make it an ideal tool for beginners, while its flexibility and customizability make it suitable for more advanced users.

Blocks Used

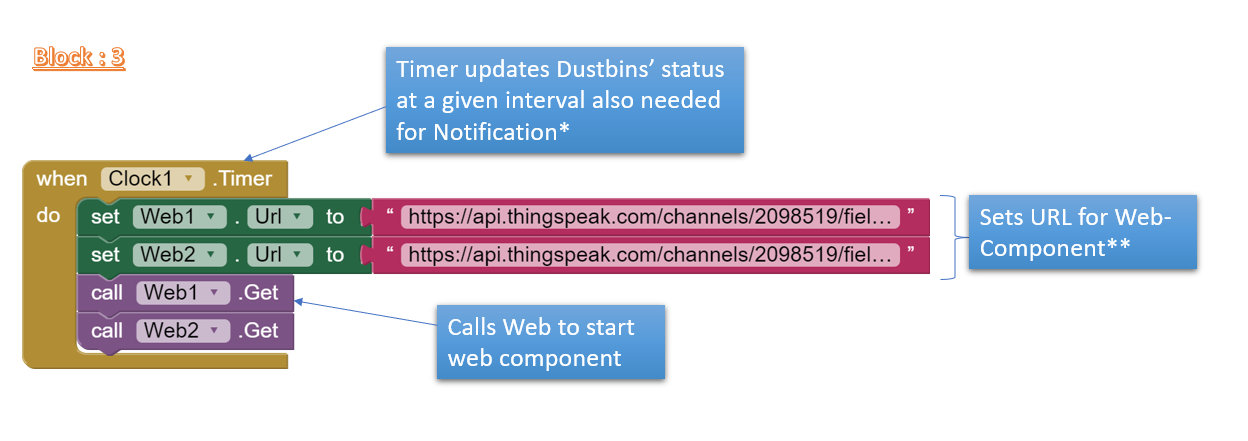
1.Screen Trigger Block



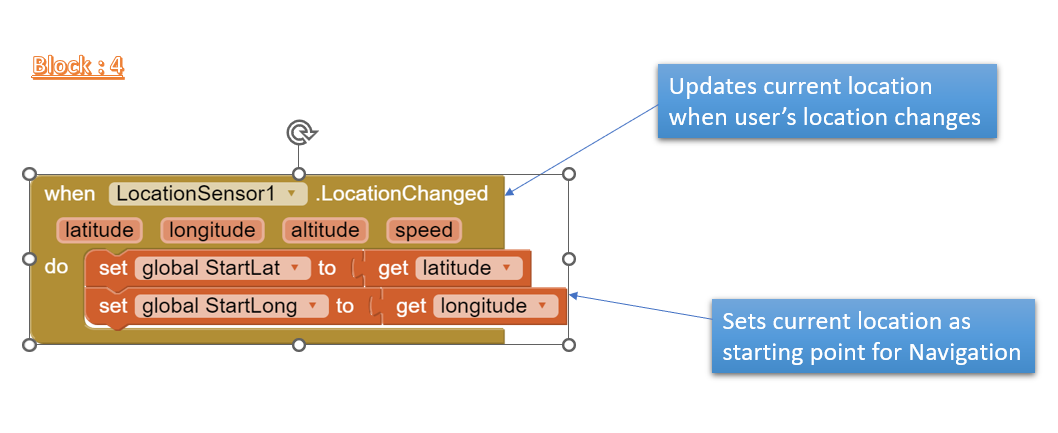
2.Refresh Button Trigger Block



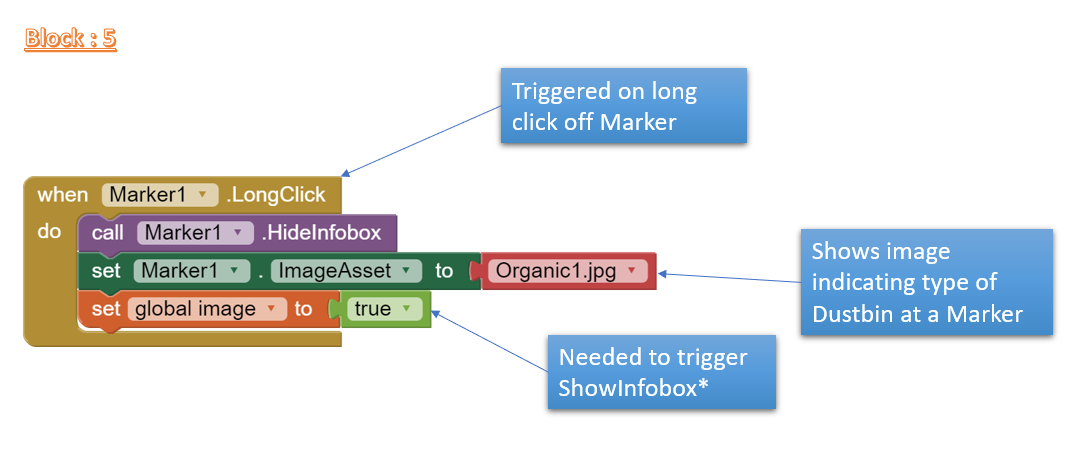
3.Clock Time Trigger Block



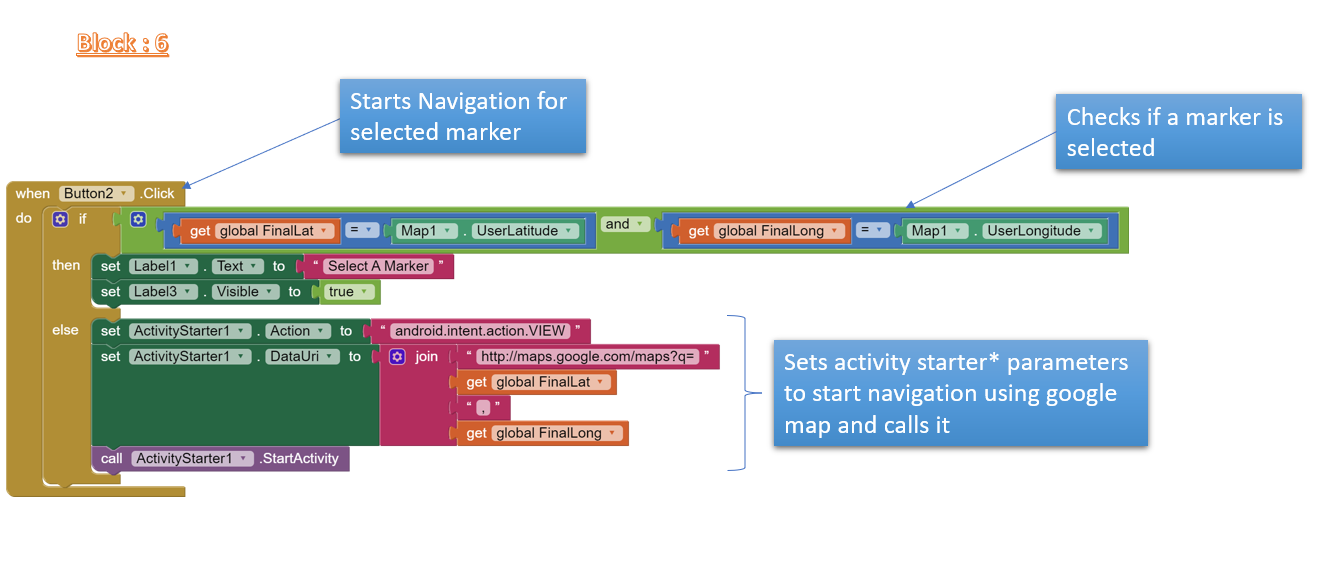
4.Location Change Trigger Block



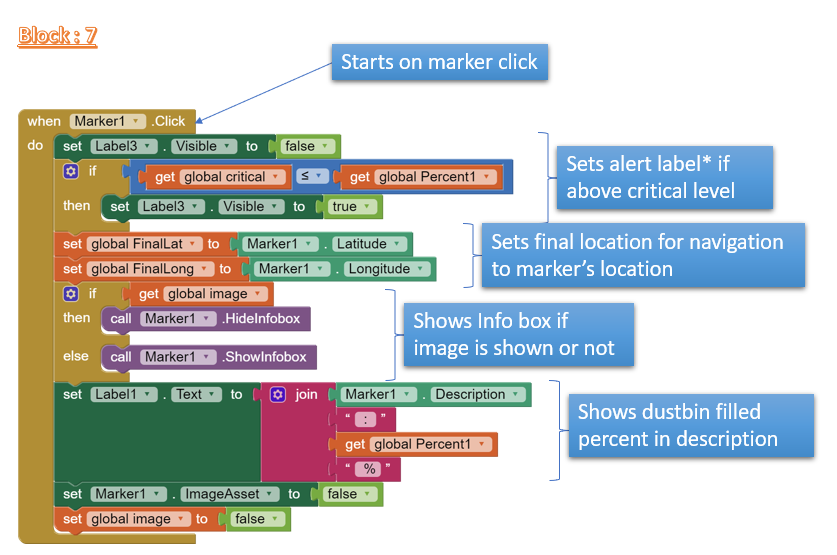
5.Marker Long Click Trigger Block



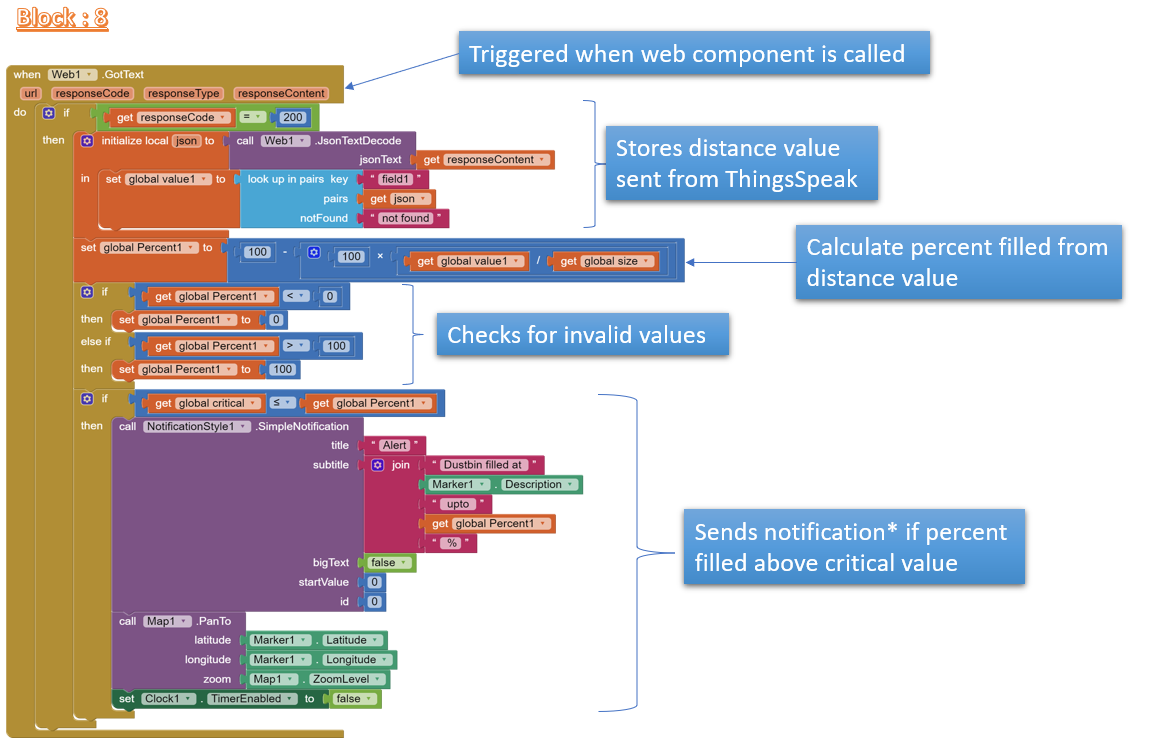
6.Navigation Button Trigger Block



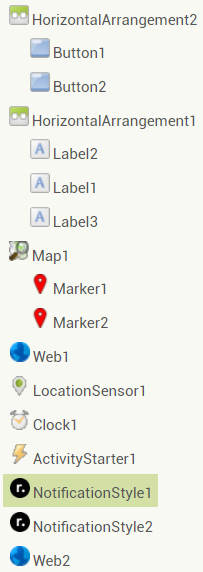
7.Marker Click Trigger Block



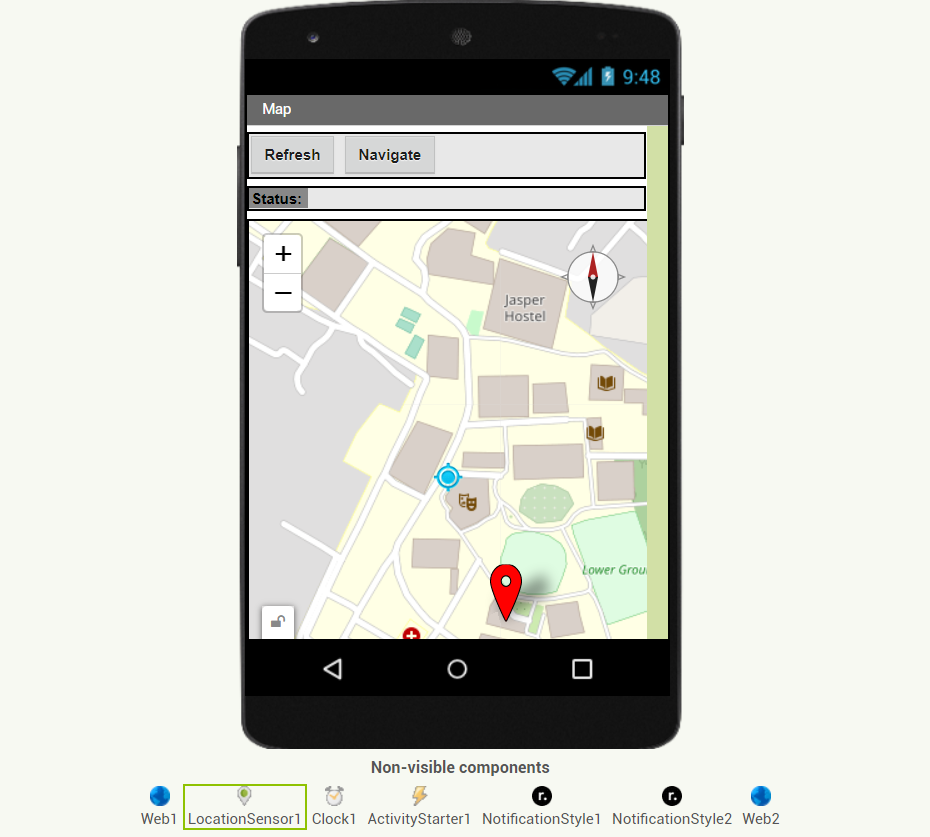
8.Web Component Block



8.Components used



10.MIT App Inventor Final Interface

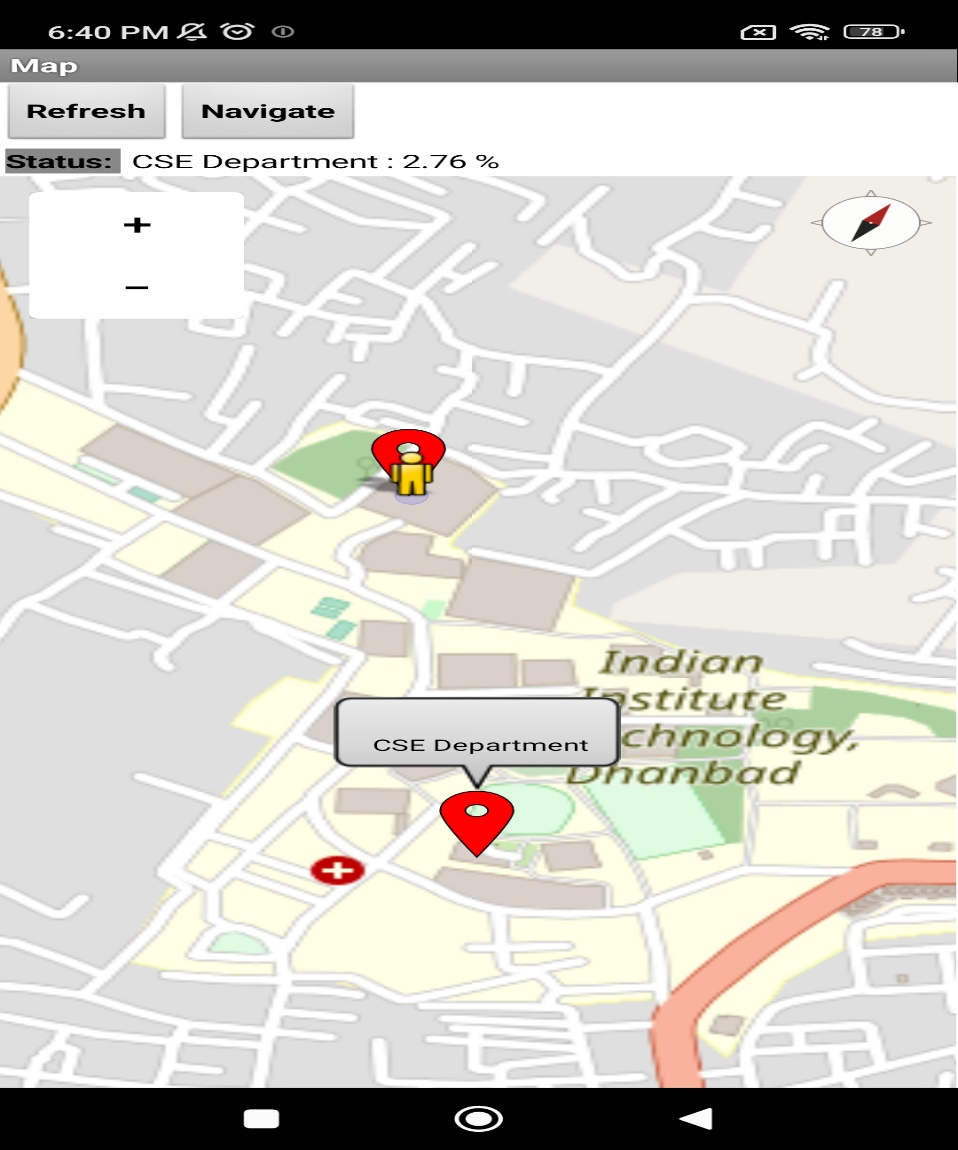


Results

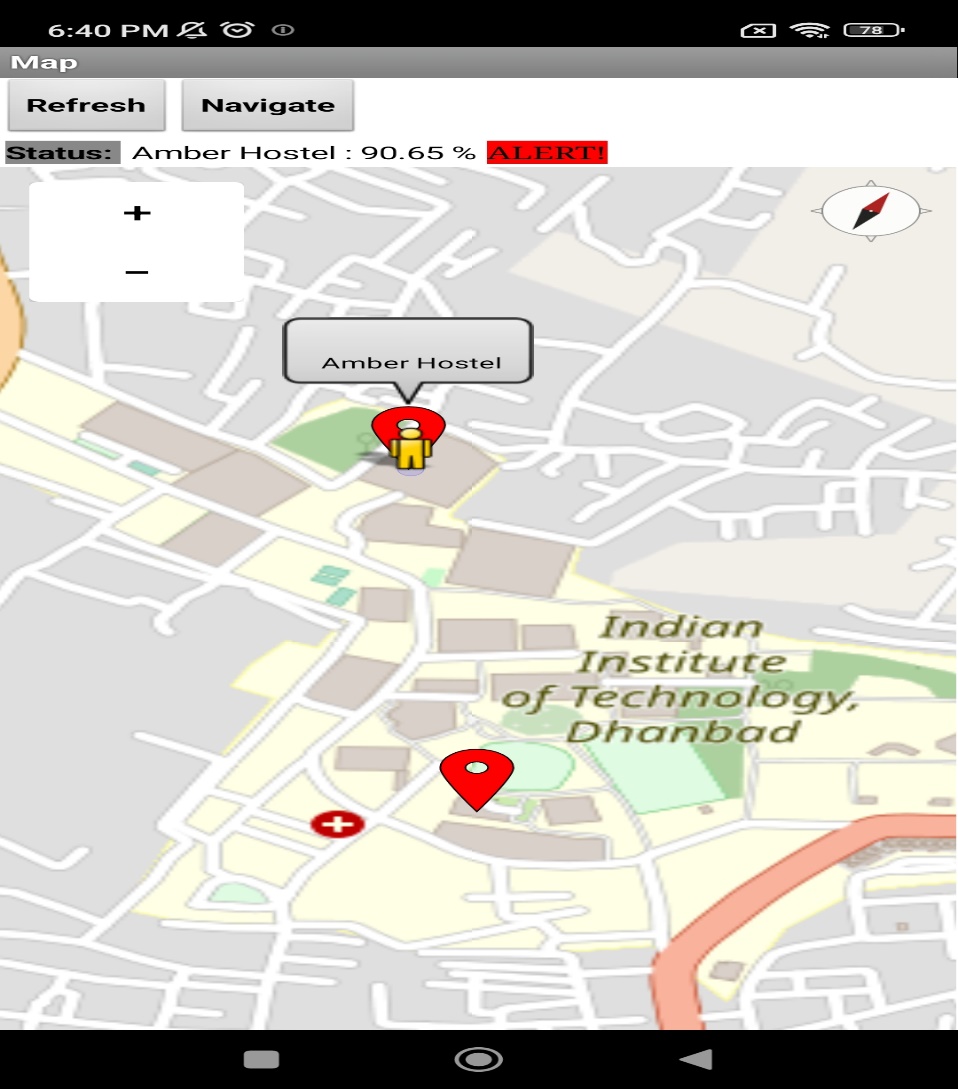
1.Permission On App Start



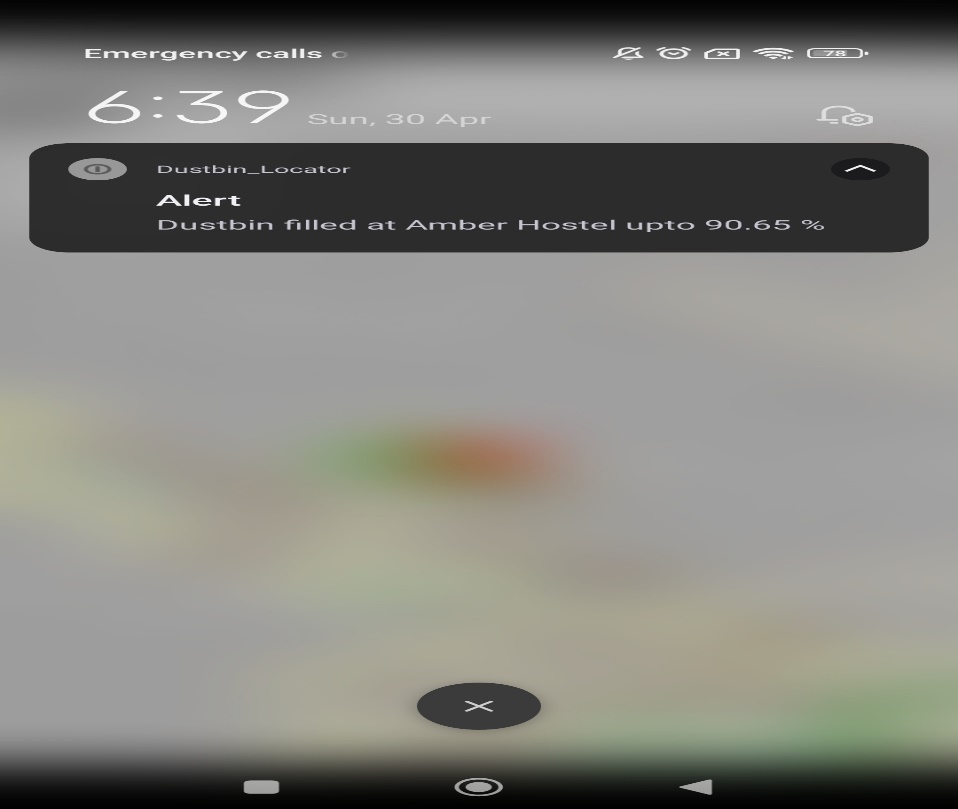
2.Dustbin Not Filled



3.Dustbin Filled



4.Notification



**Conclusion**

An app-based garbage status system is a digital platform designed to help communities manage their waste effectively. The system provides information about the status of garbage collection in specific areas, making it easier for people to keep their surroundings clean and organized. This technology allows residents to quickly and easily check the garbage collection schedule and plan accordingly. The app-based garbage status system simplifies waste management and can be an effective tool in promoting cleanliness and reducing the negative impact of waste on the environment.

If an app-based garbage status system is not connected to a database, it can cause several problems. One of the most significant issues is the inability to track the garbage collection schedule, which can lead to missed pickups, overflowing bins, and other problems. Additionally, a lack of connectivity with a database means that the system cannot collect and analyze data about garbage collection trends, which can be useful for local authorities in improving waste management. Finally, without a database, it may be challenging to scale the system to handle the growing demands of a community, potentially leading to performance issues and a lack of usability.

Connecting an app-based garbage system to AWS and its DynamoDB has many benefits and future possibilities. DynamoDB is a NoSQL database that can easily store and manage data from the app, including garbage collection schedules, collection trends, and feedback from residents. By connecting to AWS, the app-based garbage system can leverage the power of cloud computing, allowing for more significant scalability, improved performance, and higher reliability. Additionally, with the use of AWS Lambda, the app can run serverless code that can automate tasks, such as sending reminders to residents about their scheduled garbage collection day.

With the help of AWS Lambda, the app-based garbage system can automate tasks, including sending notifications to users when a dustbin is full. For example, the system can utilize a sensor in the dustbin that detects when it's reached its capacity, and then trigger a Lambda function to send a push notification to the app user. This automated process can help ensure that users are aware of the status of nearby dustbins, and can plan accordingly for disposal.

AWS SageMaker can be utilized to predict garbage collection routes for local authorities. SageMaker is an AWS machine learning service that can be used to build, train, and deploy machine learning models quickly and easily. By using historical data about garbage collection patterns, SageMaker can analyze and learn the optimal routes for garbage collection trucks. The system can take into account various factors, such as the size of the garbage truck, traffic patterns, and the density of garbage in different areas.

Once the model is trained, it can be integrated into the app-based garbage system, allowing local authorities to optimize their garbage collection routes. The system can predict the most efficient route for garbage trucks to follow, reducing travel time and fuel costs. This can lead to significant savings in the cost of garbage collection and a more efficient and reliable waste management system.

Moreover, AWS SageMaker can also be used to optimize garbage collection based on real-time data. For instance, if there is an unexpected road closure or a sudden increase in the amount of garbage in a particular area, SageMaker can quickly analyze this data and adjust the garbage collection route in real-time to ensure efficient and timely waste management.

In conclusion, an app-based garbage system connected to AWS and its various services can offer numerous benefits for waste management. With the help of DynamoDB, Lambda, and SageMaker, the app can become more intelligent, efficient, and sustainable. By automating tasks, predicting optimal garbage collection routes, and providing real-time data insights, the app-based garbage system can help local authorities reduce costs, improve their waste management practices, and promote a cleaner environment for all. By leveraging the power of cloud computing and machine learning, the future of waste management can be more efficient, effective, and sustainable than ever before.

**References**

1. <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

To Install ESP32 board

1. <https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>

If Port is not Detected for ESP32

1. For ThingSpeak Search in Library Manager ThingSpeak by MathWorks and install latest version. See example from Files > Examples > Thingspeak > WriteSingleField