**Smart Trashcan using IR Obstacle**

**Detection Sensor**



**Session**: 2022 – 2026

**Submitted by:**

Asad Ullah 2022-CS-11

Mudassir Naveed 2022-CS-32

Muhammad Kamran 2022-CS-53

Hamza Waseem 2022-CS-54

**Supervised by:**

Mr. Syed Tahseen ul Hassan Shah

**Department of Computer Science**

**University of Engineering and Technology,**

**Lahore, Pakistan**

**Acknowledgements:**

First and foremost, we would like to extend our heartfelt appreciation to our project supervisor, our instructor Mr. Tahseen ul Hassan Shah, whose guidance and mentorship were invaluable throughout the project. Your expertise, patience and constant support played a pivotal role in shaping this project and helping me tackling various challenges.

The project would not have been as comprehensive and well-executed without the hard work and commitment of the Team. The Team put valuable effort into the completion of the project. It has the sheer hard work that led to the completion of the project. Working on this final project of the semester has been a challenging yet rewarding experience for us. The experience gained from the project is invaluable and has helped us to strengthen the concept of Computer Organization and Assembly Language

**Abstract:**

In recent times, we've witnessed a remarkable revolution in our world, all thanks to the advancements in technology. As people are getting smarter so are the things. While the thought comes up for Smart cities there is a requirement for Smart waste management. The idea of Smart Dustbin is for the Smart buildings, Colleges, Hospitals, office, classroom, etc.

The idea behind Smart Dustbins is to enhance the functionality of traditional bins by infusing them with intelligent features. This transformation is achieved through the incorporation of various modules, such as infrared (IR) sensors. These sensors enable the dustbin to operate in a smart manner, responding to usage with predefined logical actions.

The Smart Dustbin is a significant improvement over its conventional counterpart, offering a seamless blend of technology and practicality. For instance, after the dustbin is utilized, it can be remotely returned to its designated position. This innovative approach eliminates the need for physical interaction, providing a more efficient and hygienic waste management solution.

This report delves into the significance, the construction, mechanism and overall functionality of the Smart Dustbin.

Contents

[**Introduction:** 5](#_Toc155333199)

[**Project Photo:** 6](#_Toc155333200)

[**Objectives:** 7](#_Toc155333201)

[**Methodology:** 8](#_Toc155333202)

[**Data Flow Diagram:** 9](#_Toc155333203)

[**Circuit Diagram:** 10](#_Toc155333204)

[**Components:** 11](#_Toc155333205)

[**Arduino UNO:** 11](#_Toc155333206)

[**ESP32 Development Board:** 12](#_Toc155333207)

[**Wires:** 13](#_Toc155333208)

[**Servo Motor:** 13](#_Toc155333209)

[**Buzzer Module:** 14](#_Toc155333210)

[**IR Obstacle Detection Module:** 14](#_Toc155333211)

[**Functionalities:** 15](#_Toc155333212)

[ **Automated Lid Control:** 15](#_Toc155333213)

[ **Mobile App Integration:** 15](#_Toc155333214)

[ **Adaptive Decision-Making:** 15](#_Toc155333215)

[ **Efficient Communication:** 15](#_Toc155333216)

[ **Rapid Prototyping with Breadboard:** 15](#_Toc155333217)

[ **User-Friendly Interface:** 15](#_Toc155333218)

[**AVR Code:** 16](#_Toc155333219)

[**IOT Code:** 18](#_Toc155333220)

[**Code Documentation:** 22](#_Toc155333221)

[**AVR Module** 22](#_Toc155333222)

[**IOT Module** 24](#_Toc155333223)

[**MQTT Dashboard:** 25](#_Toc155333224)

[**Future Enhancements:** 27](#_Toc155333225)

[**Conclusion:** 27](#_Toc155333226)

[**Project Links:** 27](#_Toc155333227)

[**GitHub** 27](#_Toc155333228)

[**References:** 28](#_Toc155333229)

# **Introduction:**

Dustbin is an important equipment; it helps to keep our neighborhood clean. It is very important equipment in a company or big supermarkets because they have a garbage disposal system. Garbage disposal system is important to secure their garbage waste orderly.

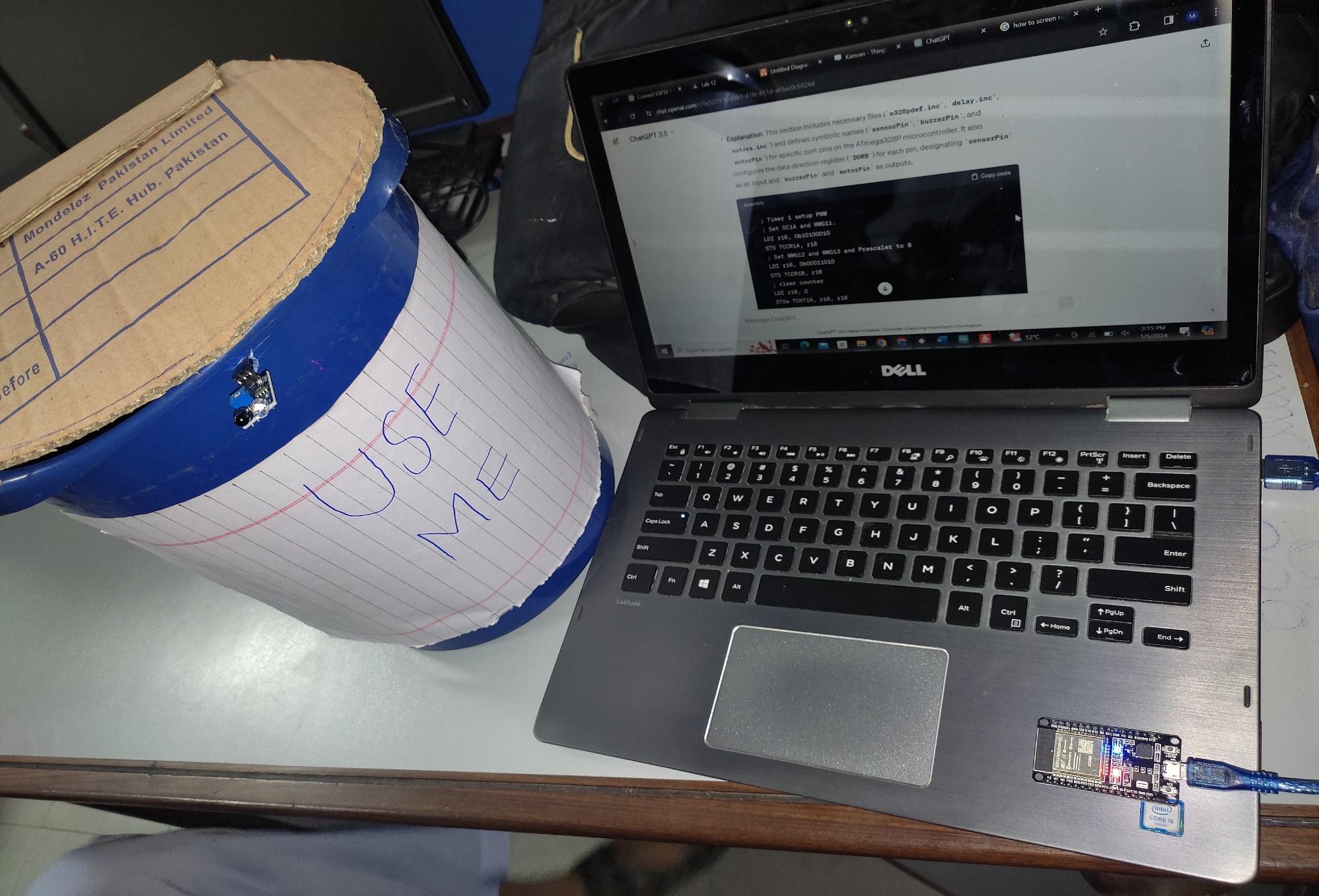
The Internet of Things (IOT) is the network of physical objects-devices, vehicles, buildings, and other items which are embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data. The Internet of Things allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy, and economic benefit.

Although an intrinsic part of our everyday routine, the dustbin's role as a mediator of changing waste particles has rarely been considered. As dustbin become reconfigured as environmental technologies for contemporary recycling programs, it is argued that they provide a revealing indicator of new waste relationships in society.

The dustbin has been implemented using combination of AVR Assembly and IOT. The IOT enable us to manage the dustbin efficiently using the internet. The physical implementation included a combination of various instruments (tools) which have been combined with code to keep the project up and running.

# 

# **Project Photo:**

****

# **Objectives:**

**User Convenience and Engagement:**

Design Smart Dustbins with user-friendly features to encourage public engagement and participation in efficient waste disposal practices.

**Efficiency Enhancement:**

Elevate the operational efficiency of Smart Dustbins through intelligent controls that respond dynamically to real-time usage patterns, ensuring optimal waste management and resource utilization.

**Environmental Impact:**

Advance environmental consciousness by integrating sensors into Smart Dustbins that adapt to usage patterns, optimizing waste collection processes and contributing to sustainable waste management practices.

**Health and Hygiene Standards:**

Prioritize health and hygiene by integrating touchless operation, antimicrobial surfaces, and other sanitary measures to enhance the overall cleanliness of Smart Dustbins.

**Data-driven Insights:**

Collect and analyze data from sensors to gain insights into usage patterns, aiding in future infrastructure planning and decision-making.

**Integration and Scalability:**

Design a system that can be easily integrated with existing infrastructure and scaled up for future expansion or integration of additional smart dustbin features.

# **Methodology:**

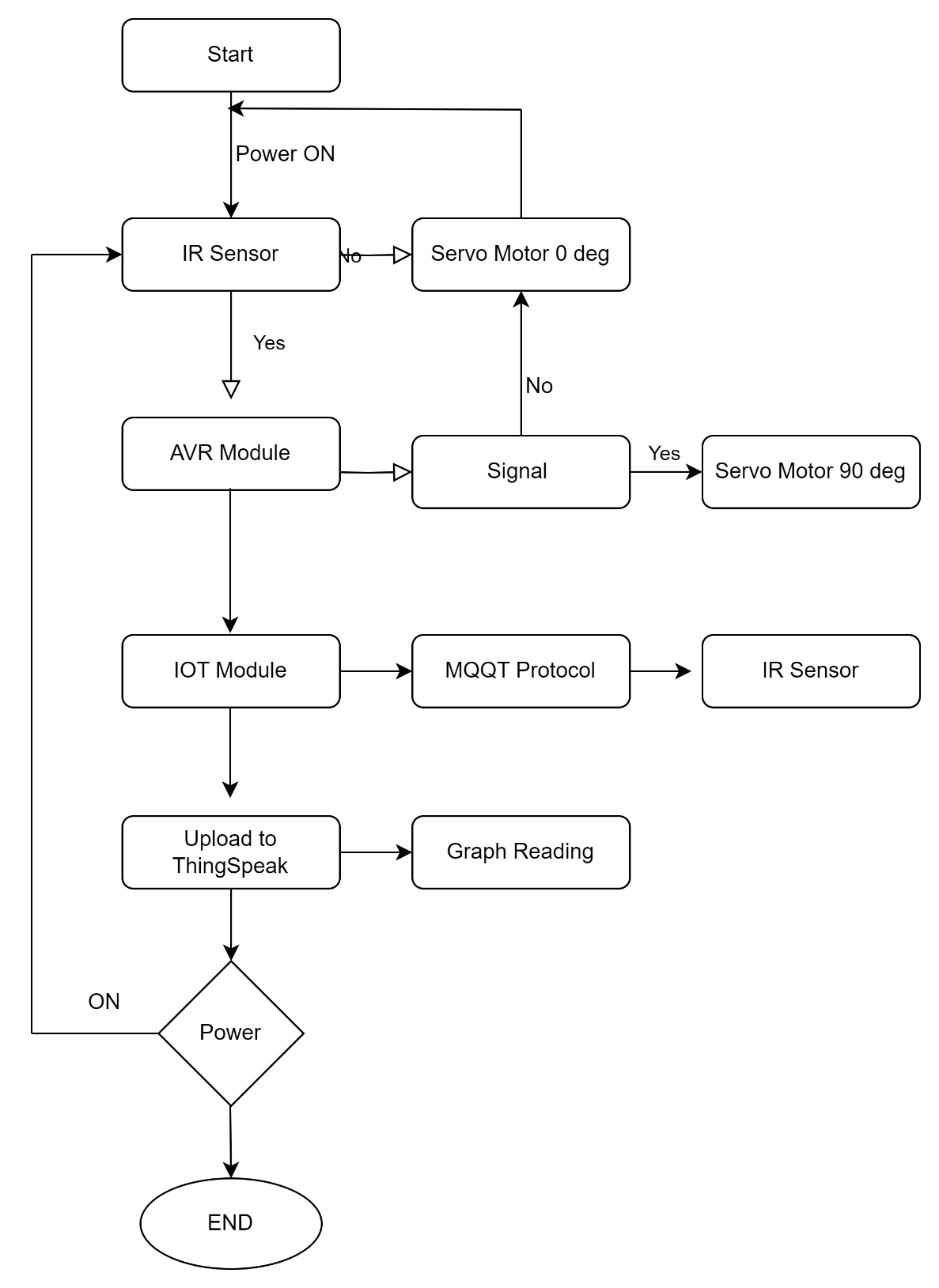
Below outlines the methodology employed in the development of our Smart dustbin Project.

* **Requirement Analysis:**

The project initiation involved an in-depth analysis of requirements, focusing on creating an intelligent and remotely manageable smart dustbin system. This phase identified key components, including the Arduino UNO, ESP32 Development Board, Breadboard, Pin-to-Hole Jumper Wires, Pin-to-Pin Jumper Wires, Servo Motor, Buzzer Module, and IR Obstacle Detection Module, based on specific needs related to connectivity, control, sensing, and automation.

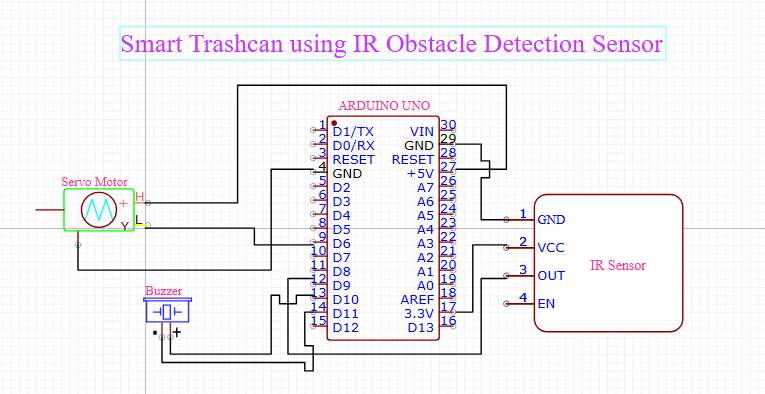
* **Component Selection and Integration:** Meticulous selection of components was based on their functionalities, compatibility, and alignment with the project objectives. Integration planning involved configuring the Arduino UNO as the processing unit, ESP32 for connectivity, Breadboard for prototyping, and incorporating additional components such as the Servo Motor, Buzzer Module, and IR Obstacle Detection Module for optimized smart dustbin functionality.
* **System Design and Prototyping:** The Breadboard served as a platform for visualizing and testing connections between the selected components. This facilitated the creation of a prototype, allowing for rapid iteration, testing, and refinement of connections and functionalities among the Arduino UNO, ESP32, Servo Motor, Buzzer Module, and IR Obstacle Detection Module.
* **Programming and Functional Testing:** Rigorous programming of the Arduino UNO and ESP32 was undertaken to enable communication with the various components, ensuring proper functionality. Functional testing verified the system's ability to detect obstacles using the IR module, triggered the Servo Motor for lid of trash can to open and close after a delay of our choice (we choose delay to be of 5 seconds), and activate the Buzzer Module as needed.
* **Integration with MQTT Dashboard:** The integration phase involved connecting the Smart Bin Management System to the Internet of Things (IoT) infrastructure, enabling remote monitoring and control via a dedicated mobile phone app. The Arduino UNO and ESP32 were programmed to communicate with the mobile app, allowing users to remotely check bin status, receive notifications, and even initiate lid-opening or closing actions.

# **Data Flow Diagram:**



# **Circuit Diagram:**

The Circuit Diagram provides easy understanding of the project on a computer screen. The circuit diagram of the project is presented below:



# **Components:**

The Smart Trash can use below components for its effective working:

* Arduino UNO with USB Cable
* ESP32 Development Board with Micro USB Cable
* Breadboard
* Pin-to-Hole Jumper Wires
* Pin-to-Pin Jumper Wires
* Servo Motor
* Buzzer Module
* IR Obstacle Detection Module
* **Description of Components:**

## **Arduino UNO:**

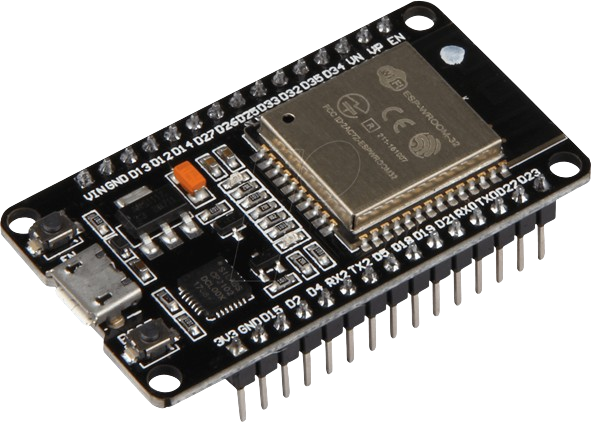
The Arduino UNO assumes a central role in the Smart Bin Management System by serving as the primary processing unit. Functioning as the brain of the system, it orchestrates the integration of key components, including the ESP32 for connectivity, sensors such as the IR Obstacle Detection Module, and actuators like the Servo Motor and Buzzer Module. Responsible for processing data from sensors, the Arduino UNO makes real-time decisions to initiate actions such as lid opening or closing based on predefined algorithms. Moreover, it establishes connectivity with the Internet of Things (IoT), enabling remote monitoring and control through the dedicated mobile app. As a versatile platform, the Arduino UNO facilitates seamless prototyping, testing, and execution of programmed instructions, ensuring the overall intelligence and efficiency of the Smart Bin Management System. Using a USB cable, we can attach it to the computer where the code is written.



*Figure.1: Arduino UNO*

## **ESP32 Development Board:**

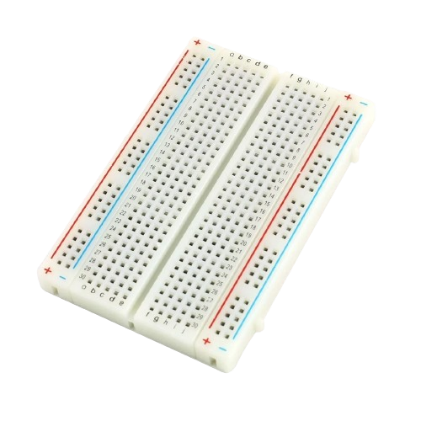
The ESP32 Development Board, paired with a Micro USB Cable, is a key component in the Smart Bin Management System. Functioning as the primary interface for wireless communication, it connects the smart dustbin to the Internet of Things (IoT) infrastructure. With built-in Wi-Fi capabilities, it facilitates seamless communication with the Arduino UNO and other system components. Powered through the Micro USB Cable, it establishes synchronized operation with the Arduino UNO, serving as a crucial link for remote monitoring, real-time alerts, and control of the dustbin's functions via the mobile app. The ESP32's role in providing connectivity enhances the overall efficiency and intelligence of the Smart Bin Management System.



*Figure.2: ESP Board*

**Bread Board:**

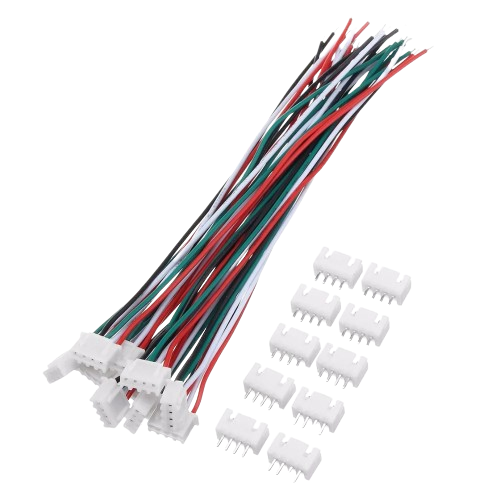
To make connections for the electronic circuit, we use Bread Board. The Breadboard in our Smart Bin project is like a testing playground for our electronic stuff. It helps us check if everything connects well before making it permanent. It's like a practice run where we can fix any problems easily. Once everything works smoothly on the Breadboard, we can set it up for real in the Smart Bin.



*Figure.3: Bread Board*

## **Wires:**

Wires are used to build a connection for the bread board. **Pin-to-Pin** wires have connectors on both ends, making them great for linking parts that are close together. On the other hand, **Pin-to-Hole** wires have a pin on one end and can go into the holes on the Breadboard or components, allowing us to connect things that are a bit farther apart.



*Figure.4: Wires*

## **Servo Motor:**

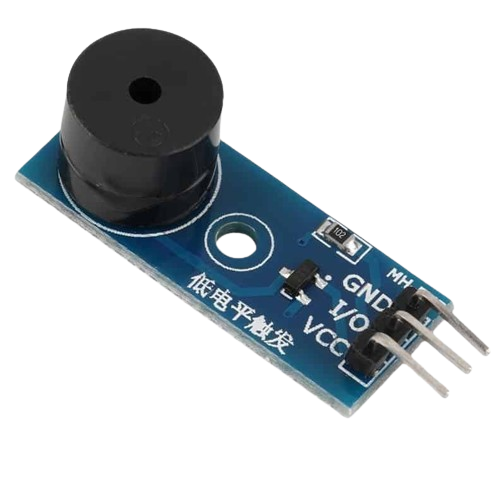
Servo Motor is an important and useful component in our project. The Servo Motor in our Smart Bin Management System project serves as the operator for dustbin lid, that makes it easy to automatically open and close whenever necessary. The Servo Motor causes the lid to open when the IR sensor senses something near the trash can. After the hand is removed from near the lid, the Servo Motor causes the lid to automatically close, ensuring hygiene and aesthetics within our project.



*Figure.5: Servo Motor*

## **Buzzer Module:**

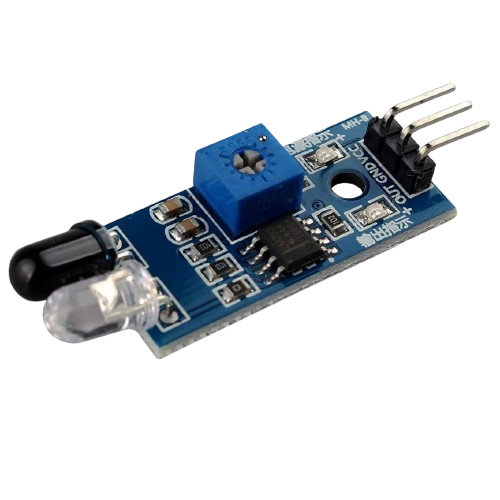
In our Smart Trash Can, we incorporate a Buzzer Module to provide audible alerts and notifications. It acts like a little speaker that makes sound when triggered by the system. Buzzer Module receives a signal from the Arduino UNO and produces a sound. It makes a sound when the lid remains open and goes silent when the lid closes. This audible feedback serves as a helpful indicator for users or maintenance personnel, enhancing the overall functionality and user experience of the smart dustbin.



*Figure.5: Buzzer Module*

## **IR Obstacle Detection Module:**

The IR Obstacle Detection Module in our Smart Bin acts as a smart sensor. It helps the system "see" if there's anything in front of the dustbin. When someone approaches or when there is an obstacle, the IR module sends a signal to the system. This signal triggers actions like opening the lid through the Servo Motor. Essentially, the IR Obstacle Detection Module enhances the dustbin's awareness, enabling it to respond intelligently to user interaction and facilitating a hands-free and efficient waste disposal experience.



*Figure.6: IR obstacle detector*

# **Functionalities:**

## **Automated Lid Control:**

The system integrates sensors like the IR Obstacle Detection Module to detect the presence of waste. When triggered, the Arduino UNO commands the Servo Motor to automate the opening and closing of the dustbin lid, enhancing user convenience and promoting touchless waste disposal.

## **Mobile App Integration:**

Through IoT connectivity, the system interfaces with a mobile app, allowing users to remotely monitor the dustbin's fill level, receive notifications, and initiate lid actions. This feature enhances user engagement and provides real-time control over the smart waste disposal process.

## **Adaptive Decision-Making:**

The Arduino Uno, acting as the brain of the system, makes intelligent decisions based on data from sensors. It compares the received light intensity with predefined thresholds and dynamically adjusts the lighting, ensuring a responsive and adaptive lighting system.

## **Efficient Communication:**

The MQTT server facilitates seamless communication between the various components of the system, including the Arduino Uno, ESP microcontroller, and the MQTT dashboard. This ensures reliable data exchange, enabling swift decision-making and control.

## **Rapid Prototyping with Breadboard:**

During development, the breadboard serves as a versatile testing platform, allowing for quick and easy rearrangement of components. This facilitates rapid prototyping, troubleshooting, and experimentation with different configurations before final implementation.

## **User-Friendly Interface:**

The MQTT dashboard provides a user-friendly interface for visualizing and controlling street light functionalities. Users can easily understand and interact with the system, making it accessible to a broad audience.

# **AVR Code:**

; Code by CS 53, 54, 11, 32

.include "m328pdef.inc"

.include "delay.inc"

.include "motors.inc"

.equ sensorPin = PB4

.equ buzzerPin = PB5

.equ motorPin = PB1

.cseg

.org 0x0000

CBI DDRB,sensorPin ; input IR

SBI DDRB,buzzerPin ; buzzer output

SBI DDRB, motorPin ; motor Pin

; Timer 1 setup PWM

; Set OC1A and WMG11.

LDI r16, 0b10100010

STS TCCR1A, r16

; Set WMG12 and WMG13 and Prescalar to 8

LDI r16, 0b00011010

STS TCCR1B, r16

; clear counter

LDI r16, 0

STSw TCNT1H, r16, r16

; count of 40000 for a 20ms period or 50 Hz cycle

LDI r16, LOW(40000)

LDI r17, HIGH(40000)

STSw ICR1H,r17,r16

loop:

;0 degree

LDI r16, LOW(900)

LDI r17, HIGH(900)

STSw OCR1AH,r17,r16

SBIS PINB,sensorPin ;compare sensor input

CALL L1 ;jump to L1 label below

rjmp loop

L1:

CALL motor

ret

motor:

sbi PORTB,buzzerPin ; set buzzer as ON

; 90 degree

LDI r16, LOW(2900)

LDI r17, HIGH(2900)

STSw OCR1AH,r17,r16

delay 2000

CBI PORTB,buzzerPin ;set buzzer as OFF

ret

; Macro to add delay (blocking) upto 2500ms

.macro delay

push r18

push r24

push r25

ldi r18,@0/10

L1:

ldi r24,LOW(39998) ; intialize inner loop count in inner

ldi r25,HIGH(39998) ; loop high and low registers

L2:

sbiw r24,1 ; decrement inner loop registers

brne L2 ; branch to L2 if iLoop registers != 0

dec r18 ; decrement outer loop register

brne L1 ; branch to L1 if outer loop register != 0

nop ; no operation

pop r25

pop r24

pop r18

.endmacro

.macro STSw

cli

STS @0, @1

STS @0-1, @2

sei

.endmacro

.macro LDSw

cli

LDS @1, @2-1

LDS @0, @2

sei

.endmacro

# **IOT Code:**

#include <WiFi.h>

#include <PubSubClient.h>

#include <ThingSpeak.h>

const char \*ssid = "Kami";

const char \*password = "Someone56";

const char \*mqttServer = "test.mosquitto.org";

const int mqttPort = 1883;

const char \*mqttClientId = "COAL\_GROUP\_8\_A";

const char \*outputTopic = "esp32/output\_53";

const char \*inputTopic = "esp32/input\_53";

static unsigned long lastMillis = 0;

WiFiClient espClient;

PubSubClient client(espClient);

// ThingSpeak settings

unsigned long channelID = 2395229;

const char \*writeAPIKey = "CGGRUSO3CDXQX4DJ";

void setup() {

Serial.begin(115200);

Serial2.begin(9600);

WifiSetup();

// Configure MQTT

client.setServer(mqttServer, mqttPort);

client.setCallback(callBack);

connectToMQTT();

// Initialize ThingSpeak

ThingSpeak.begin(espClient);

}

void WifiSetup() {

// Connect to Wi-Fi

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected with IP address: ");

Serial.println(WiFi.localIP());

}

void callBack(char \*inputTopic, byte \*message, unsigned int length) {

Serial.print("Message arrived on topic: ");

Serial.print(inputTopic);

Serial.print(". Message : ");

String messageTemp;

for (int i = 0; i < length; i++) {

messageTemp += (char)message[i];

}

Serial.println(messageTemp);

Serial.print("Message Sent to UART : ");

}

void loop() {

if (WiFi.status() != WL\_CONNECTED) {

WifiSetup();

}

if (Serial2.available() > 0) {

// String receivedChar = Serial2.readString();

// Serial.print("Message Received through UART : ");

// Serial.println(receivedChar);

}

// Handle MQTT events

if (!client.connected()) {

connectToMQTT();

}

client.loop();

// Message Publishing on app with delay of 5s

if (millis() - lastMillis > 5000) {

String resultString = " working ";

const char \*message = resultString.c\_str();

publishMessage(message);

// Send data to ThingSpeak

sendToThingSpeak();

lastMillis = millis();

}

}

void connectToMQTT() {

while (!client.connected()) {

Serial.println("Connecting to MQTT...");

if (client.connect(mqttClientId)) {

Serial.println("Connected to MQTT");

client.subscribe(inputTopic);

} else {

Serial.print("Failed with state : ");

Serial.println(client.state());

delay(2000);

}

}

}

void publishMessage(const char \*message) {

if (client.connected()) {

client.publish(outputTopic, message);

Serial.print("Message Published : ");

Serial.println(message);

}

}

void sendToThingSpeak() {

// Create data for Field1 and Field2

int valueField1 = random(100);

int valueField2 = random(100);

// Write to ThingSpeak

ThingSpeak.writeField(channelID, 1, valueField1, writeAPIKey);

ThingSpeak.writeField(channelID, 2, valueField2, writeAPIKey);

int status = ThingSpeak.writeFields(channelID, writeAPIKey);

if (status == 200) {

Serial.println("ThingSpeak update successful!");

} else {

Serial.println("Error updating ThingSpeak. HTTP status code: " + String(status));

}

}

# **Code Documentation:**

## **AVR Module**

.include "m328pdef.inc"

.include "delay.inc"

.include "motors.inc"

.equ sensorPin = PB4

.equ buzzerPin = PB5

.equ motorPin = PB1

This section includes necessary external files (m328pdef.inc, delay.inc, and motors.inc) and defines symbolic names (sensorPin, buzzerPin, and motorPin) for specific port pins on the ATmega328P.

CBI DDRB,sensorPin ; input IR

SBI DDRB,buzzerPin ; buzzer output

SBI DDRB, motorPin ; motor Pin

Configures the Data Direction Register (DDRB) to set sensorPin as an input for an IR sensor, buzzerPin as an output for a buzzer, and motorPin as an output for a motor.

; Timer 1 setup PWM

; Set OC1A and WMG11.

LDI r16, 0b10100010

STS TCCR1A, r16

; Set WMG12 and WMG13 and Prescalar to 8

LDI r16, 0b00011010

STS TCCR1B, r16

; clear counter

LDI r16, 0

STSw TCNT1H, r16, r16

; count of 40000 for a 20ms period or 50 Hz cycle

LDI r16, LOW(40000)

LDI r17, HIGH(40000)

STSw ICR1H,r17,r16

Sets up Timer 1 for PWM generation. Configures the mode and prescaler, clears the counter, and sets the top value (ICR1) for a 20ms period or 50 Hz cycle.

loop:

;0 degree

LDI r16, LOW(900)

LDI r17, HIGH(900)

STSw OCR1AH,r17,r16

SBIS PINB,sensorPin ;compare sensor input

CALL L1 ;jump to L1 label below

rjmp loop

L1:

CALL motor

ret

Defines the main loop. Sets the PWM value for a 0-degree position, checks the input from the IR sensor, and jumps to label L1 if the sensor input is high. Calls the motor subroutine.

motor:

sbi PORTB,buzzerPin ; set buzzer as ON

; 90 degree

LDI r16, LOW(2900)

LDI r17, HIGH(2900)

STSw OCR1AH,r17,r16

delay 2000

CBI PORTB,buzzerPin ;set buzzer as OFF

ret

Defines the motor subroutine. Turns on the buzzer, sets the PWM value for a 90-degree position, calls a delay function, and turns off the buzzer.

## **IOT Module**

The setup() function is a standard Arduino function that is executed once when the microcontroller starts or resets. It is typically used for initializing variables, configuring pin modes, and setting up initial conditions for the program.

Serial.begin(115200);

Initializes the serial communication with a baud rate of 115200. This is commonly used for debugging and communication with a connected computer.

lidServo.attach(14);

lidServo.write(closedAngle);

Configures a servo motor attached to pin 14. Sets the initial position of the servo to closedAngle (closed position).

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

Sets the trigger pin (trigPin) as an output and the echo pin (echoPin) as an input, preparing the pins for interfacing with an ultrasonic sensor.

The setup() function is automatically called once at the beginning of the program and is essential for initializing components and setting up the initial state of the system.

void setup() {

Serial.begin(115200);

lidServo.attach(14);

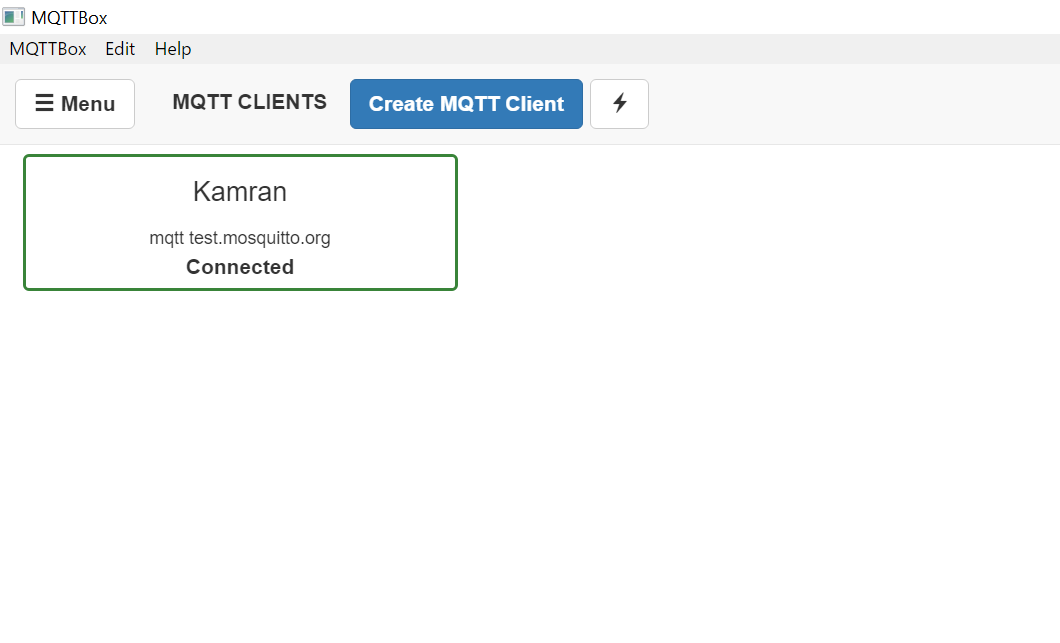
lidServo.write(closedAngle);

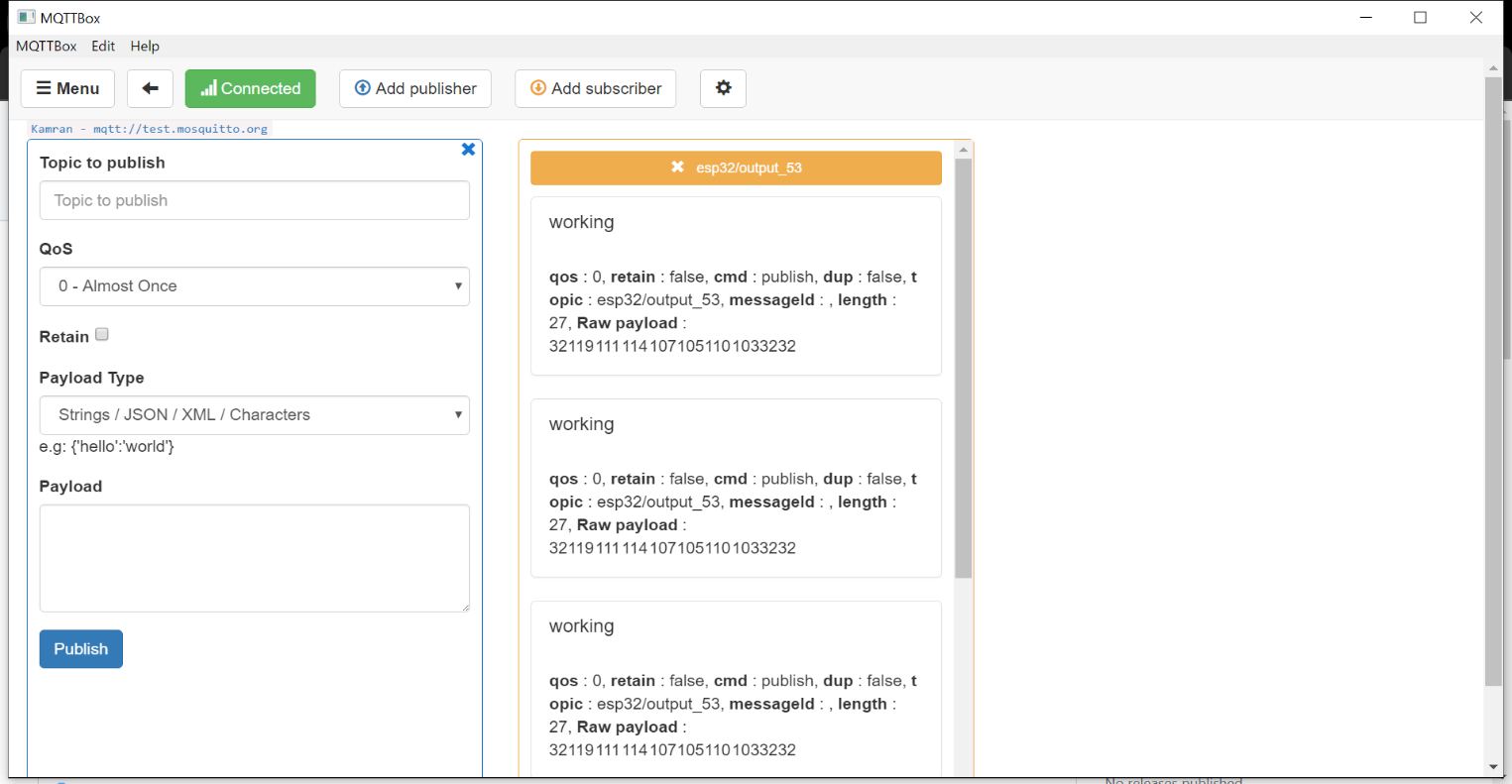
pinMode(trigPin, OUTPUT);

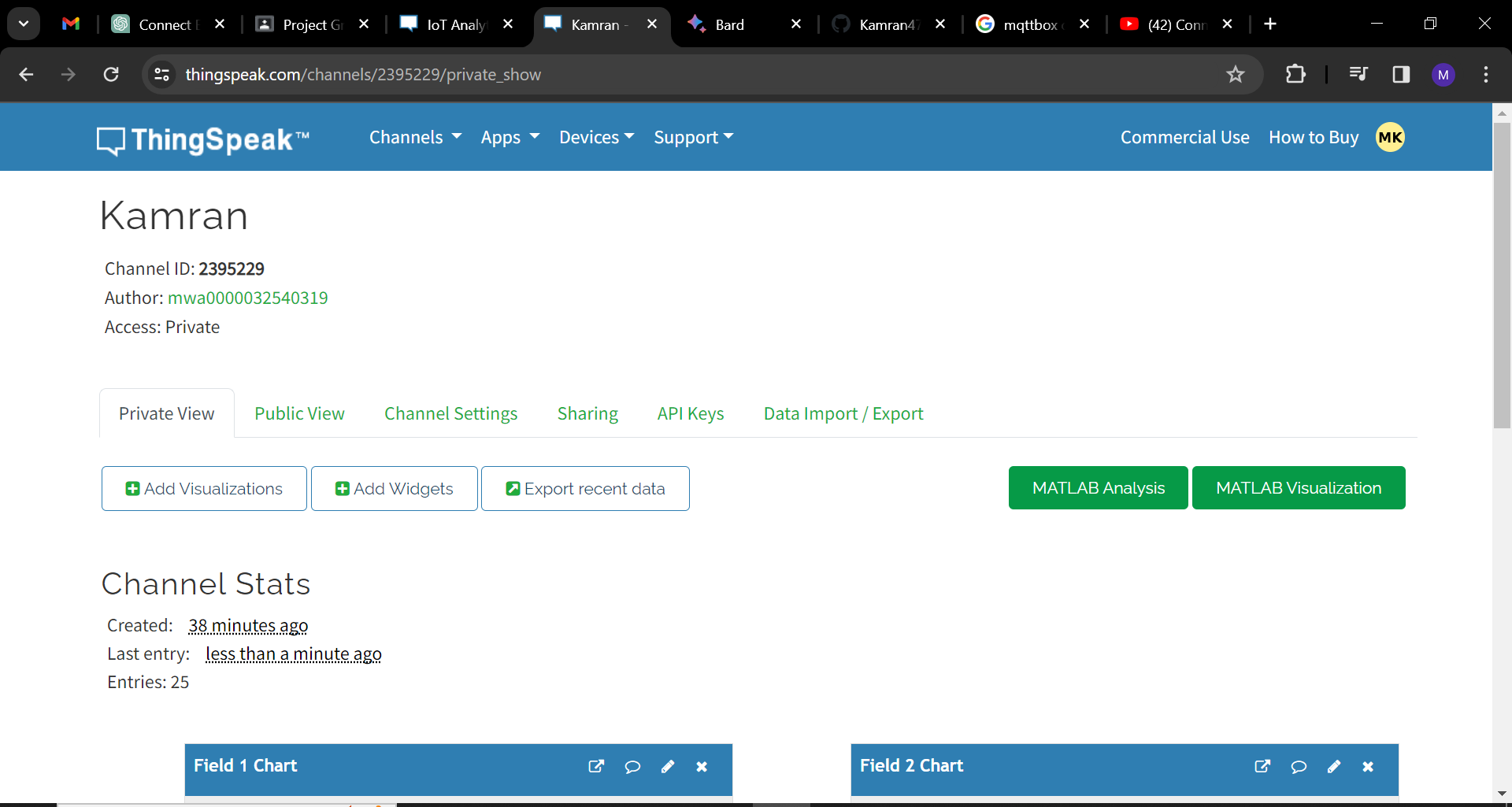
pinMode(echoPin, INPUT);

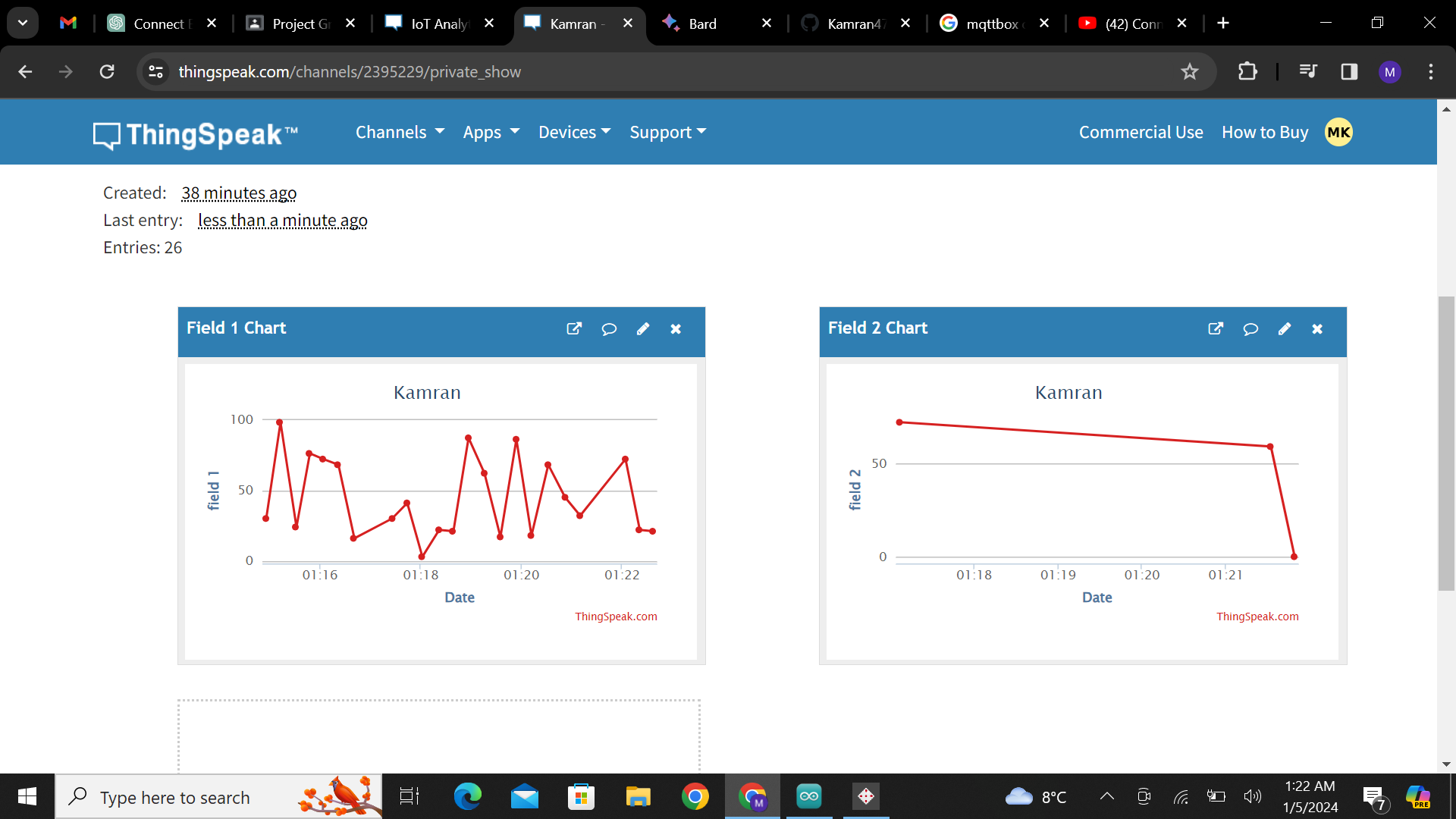
}

# **MQTT Dashboard:**









# **Future Enhancements:**

The future enhancements for the smart dustbin project involving an ESP32 could include the integration of a mobile app for remote monitoring and control, cloud connectivity for data storage and analysis, advanced sensor integration for more comprehensive data collection, machine learning algorithms for predictive analysis and optimization, enhanced security measures, customizable lid control settings, collaboration with waste management systems, user feedback mechanisms, solar power integration for sustainability, localization and internationalization for broader accessibility, robust error handling, community engagement platforms, integration with smart home systems, and improvements in physical design and aesthetics. These enhancements aim to enhance functionality, user experience, and overall efficiency, ensuring a comprehensive and evolving smart dustbin system.

# **Conclusion:**

In conclusion, our Trash Can project represents a practical and innovative approach to modernizing waste disposal. By incorporating smart features such as automated lid control, we've not only enhanced user convenience but also contributed to more efficient and sustainable waste management practices.

The utilization of sensors and a user-friendly mobile app interface adds a layer of intelligence to the traditional trash can, allowing for real-time monitoring and control. The integration of these technologies not only improves the overall user experience but also aligns with the growing trend towards smart solutions in various urban environments.

These enhancements not only elevate the functionality and efficiency of the smart dustbin but also pave the way for sustainable practices, community engagement, and a more aesthetically appealing and user-oriented solution for modern waste disposal needs. The continual development and implementation of these features promise a smarter, more adaptable, and environmentally conscious approach to waste management in the future.

## **Project Links:**

**GitHub:**  Here’s the GitHub link of our project: <https://github.com/Kamran47t6/Smart_Dustbin_using_IR_Obstacle_Detection.git>

**LinkedIn:** The video Link of our project:

<https://www.linkedin.com/posts/muhammad-kamran-4a9539257_thrilled-to-unveil-our-groundbreaking-project-activity-7148969509951893504-acsx?utm_source=share&utm_medium=member_desktop>

# **References:**

1. GeekforGeeks,Intro to MQTT Protocol [online] <https://www.geeksforgeeks.org/introduction-of-message-queue-telemetry-transport-protocol-mqtt/>
2. GeekforGeeks, Encapsulating Security Payload [online]

<https://www.geeksforgeeks.org/what-is-encapsulating-security-payload/>

1. Connect ESP with MQTT, [online]

<https://www.makeuseof.com/using-esp01-with-mqtt-how-to-connect-and-control-iot-devices/>

1. Serial connection B/W ESP and Arduino, [online] <https://www.instructables.com/Serial-Communication-Between-Arduino-and-ESP-01/>
2. Connect ESP to ThingSpeak Cloud, <https://www.electronicshub.org/connect-esp8266-to-thingspeak/>
3. Data Flow Diagram (DFD) ,[online] <https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1#G1J94sirpjxK_pDK1u-mBqufI0ETe2_cZe>