

## **Smart Campus Waste Management System - Smart Waste Management System for University Campuses**

**Project Created by: Christina Sharanyaa Selvakumari S, S. Janani, Krishna Priya V.,  
Mirudhula S.V.**

**Project Reviewed by: Mrs. K .Amudha**

**Project Created Date: 21/May/2024**

**Project Code: IOT001**

**College Code: 3135**

**College Name: Panimalar Engineering College Chennai City Campus**

**Team Name: IoT 1533**

## Executive Summary

Waste management in universities plays a pivotal role in fostering sustainability on campus and beyond. By implementing comprehensive waste reduction, recycling, and composting programs, universities minimize their environmental impact for campus landscaping and agriculture. Moreover, effective waste management educates students and staff about the importance of sustainability, encouraging responsible consumption and waste disposal habits. Ultimately, waste management in universities serves as a model for sustainable practices, shaping a future where environmental stewardship is integral to societal values and actions. Integrating smart waste management systems amplifies the sustainability efforts of universities. By leveraging IoT sensors, data analytics, and automation, universities can optimize waste collection routes, reduce operational costs, and enhance resource efficiency. Smart bins equipped with fill-level sensors enable real-time monitoring, ensuring timely collection and preventing overflowing bins, which can lead to littering and environmental hazards. Furthermore, data-driven insights from smart waste management systems allow universities to identify trends, measure progress, and refine waste management strategies for maximum sustainability impact. Predictive analytics can forecast waste generation patterns, enabling proactive planning and resource allocation. Additionally, mobile applications linked to smart waste systems engage students and staff, providing real-time updates on bin statuses, educational resources on waste reduction, and incentives for sustainable behaviors. Overall, smart waste management transforms universities into dynamic hubs of sustainability innovation, demonstrating how technology can be harnessed to create cleaner, greener campuses and inspire positive environmental action in the broader community.

## Table of Contents

### Contents

Executive Summary.....	2
Table of Contents.....	3
Project Objective.....	4
Scope.....	5
Methodology.....	6
Artifacts used.....	10
Technical coverage.....	15
Results.....	16
Challenges and Resolutions.....	17
Conclusion.....	20
References.....	21

## Project Objective

The objective of this project is to design and implement a comprehensive IoT-based smart waste management system tailored for a university campus. This system aims to address the inefficiencies and environmental concerns associated with traditional waste management practices. By leveraging the capabilities of modern sensors, microcontrollers, and wireless communication technologies, the project seeks to optimize waste collection processes, enhance recycling efforts, and foster a culture of sustainability among students and staff. At its core, the project is motivated by the need to solve several critical problems. Firstly, inefficient waste collection schedules often lead to either overflowing bins or unnecessary collection rounds, both of which incur additional operational costs and environmental impacts. Secondly, improper waste segregation hampers recycling efforts and contributes to increased landfill usage. Lastly, there is a growing need for real-time data to inform decision-making and strategic planning in waste management. To address these issues, the project will integrate various sensors, including ultrasonic sensors for measuring fill levels, weight sensors for monitoring the mass of waste, gas sensors for detecting decomposing gases, and temperature and humidity sensors for tracking environmental conditions. Additionally, accelerometers will be used to detect bin movements, and RFID/NFC readers will assist in user identification and waste tracking. The system will also incorporate GPS modules for location tracking and a camera module for visual monitoring. The data collected by these sensors will be processed and analyzed using microcontrollers such as Arduino and Raspberry Pi. This data will then be transmitted in real-time to a centralized platform via communication protocols like Wi-Fi and 5G. The centralized system will employ advanced algorithms to optimize waste collection routes, predict waste generation patterns, and provide actionable insights for better resource allocation.

## Scope

The scope of this project encompasses the design, development, and implementation of an IoT-based smart waste management system specifically for a university campus. This includes a comprehensive evaluation of the existing waste management practices to identify inefficiencies and areas for improvement. The project will involve the selection and integration of various sensors and microcontrollers to monitor waste levels, environmental conditions, and bin movements. A significant part of the scope involves the development of a robust data collection and transmission infrastructure. This will entail setting up communication protocols, such as Wi-Fi and 5G, to ensure real-time data flow from the bins to a centralized platform. The platform will be equipped with advanced analytics capabilities to process the data and generate actionable insights for optimizing waste collection routes and schedules. The project also includes the development of user-friendly software applications, such as mobile apps and web dashboards, to provide real-time monitoring, notifications, and reports. These applications will be designed to engage the campus community, promoting responsible waste disposal and recycling practices. Furthermore, the scope extends to the testing and deployment phases, where the system will be implemented in a phased manner across different zones of the campus. This will involve close collaboration with the university's facilities management team to ensure seamless integration with existing operations. Training and support for the campus staff and students are also part of the project scope. This will include developing educational materials and conducting workshops to familiarize users with the new system and encourage their participation in sustainability initiatives. Finally, the project scope includes provisions for future scalability and integration with other smart campus initiatives. This ensures that the system can evolve and adapt to changing needs and technological advancements, maintaining its effectiveness and relevance in the long term.

## **Methodology**

The methodology for smart waste management with IoT involves sensor selection, hardware integration, software development, and iterative testing. Sensors like ultrasonic and weight sensors measure fill levels and weight, while gas sensors detect decomposition gases. Temperature and humidity sensors monitor environmental conditions. Data collected by these sensors is analyzed to optimize waste collection routes and schedules. Prototyping and testing ensure system reliability, followed by deployment across the campus. Continuous monitoring and refinement improve efficiency, while staff training ensures smooth operation. This methodology ensures a comprehensive and effective implementation of IoT-based smart waste management systems in university campuses.

### **Block 1: Preliminary stage**

In this initial phase, the project is carefully planned, and the system requirements are established. This involves defining the hardware components such as ultrasonic sensors (HC-SR04), weight sensors (Load cell with HX711), gas sensors (MQ-135), temperature and humidity sensors (DHT22 or DHT11), RFID/NFC readers, GPS modules (NEO-6M), accelerometers (ADXL345), and camera modules (ESP32-CAM). The software stack and overall system design are also outlined during this stage.

### **Block 2: Purchasing Components**

All necessary hardware, sensors, and electronic components are procured at this stage. This ensures that all required supplies are available for subsequent project phases, avoiding delays and facilitating smooth progression.

### **Block 3: Development of Circuit Schematics**

A precise schematic diagram of the electronic circuit is developed. This involves designing the power supply arrangements, specifying component connections, and determining circuit board placements. This schematic serves as a blueprint for the hardware assembly.

#### **Block 4: Hardware Assembly and Integration**

During this phase, the hardware components are assembled according to the schematic. Sensors like ultrasonic, weight, gas, temperature/humidity, accelerometer, GPS modules, and RFID/NFC readers are integrated onto waste bins. Microcontrollers such as Arduino or Raspberry Pi are used to process data, and modules like HX711 and ESP32-CAM facilitate sensor communication.

#### **Block 5: Development of software**

Software development is conducted using platforms like wokwi and Tinkercad, which enables simulation and design of electronic circuits. The user-friendly interface helps in creating custom applications for sensor interfacing and system control.

#### **Block 6: Data Processing and Acquisition**

This phase involves collecting data from various sensors, such as ultrasonic, weight, and gas sensors. The data is then analyzed to optimize waste management strategies. Real-time monitoring and analysis algorithms are integrated with centralized systems to provide actionable insights.

#### **Block 7: Implementing Real-time Monitoring**

The system's real-time monitoring features are implemented, allowing users to obtain real-time data through a web portal. This ensures continuous data updates and enhances the system's responsiveness.

#### **Block 8: Developing Web Interface**

A Node-Red web interface is developed to enable users to browse and retrieve data from the waste monitoring system. This interface includes interactive features, data presentation, and user authentication mechanisms.

#### **Block 9: Implementing security**

Robust security measures are incorporated to protect user data and the system. This includes defense against cyber threats, encryption, and authentication protocols to ensure data integrity and privacy.

**Block 10: Analysis and Optimization of Efficiency**

The system's efficiency and power consumption are evaluated. Optimization techniques are applied to improve energy efficiency, ensuring the system operates effectively with minimal power usage.

**Block 11: Validation and Testing**

Rigorous testing procedures are applied to ensure the system's reliability, accuracy, and functionality. Various test scenarios are documented to verify the system's performance under different conditions.

**Block 12: Supporting Evidence**

Comprehensive documentation is produced, including user guides, technical specifications, project reports, and technical information. This documentation serves as a reference for users and potential developers.

**Block 13: Integration and Future Scalability**

Future scalability and integration with emerging technologies are considered. The system is designed to support future upgrades and improvements, ensuring it remains effective as needs evolve.

**Block 14: Knowledge Exchange and Spread**

Knowledge sharing and dissemination of research findings are emphasized through presentations, articles, and discussions. This phase highlights the importance of educating the public about the project's findings and promoting broader adoption of smart waste management practices.



# Artifacts used in Smart Waste Management System for University Campuses:

## Components used:

ESP32 Microcontroller, Ultrasonic Sensor (HC-SR04), RFID Reader, GPS Module, Load Cell (with HX711 Amplifier), Gas Sensor (MQ135), Camera Module, 5G Module, NPC Reader, Temperature Sensor (DS18B20)

## Software Libraries:

### 1. HX711.h :

A library used to interface with the HX711 load cell amplifier.

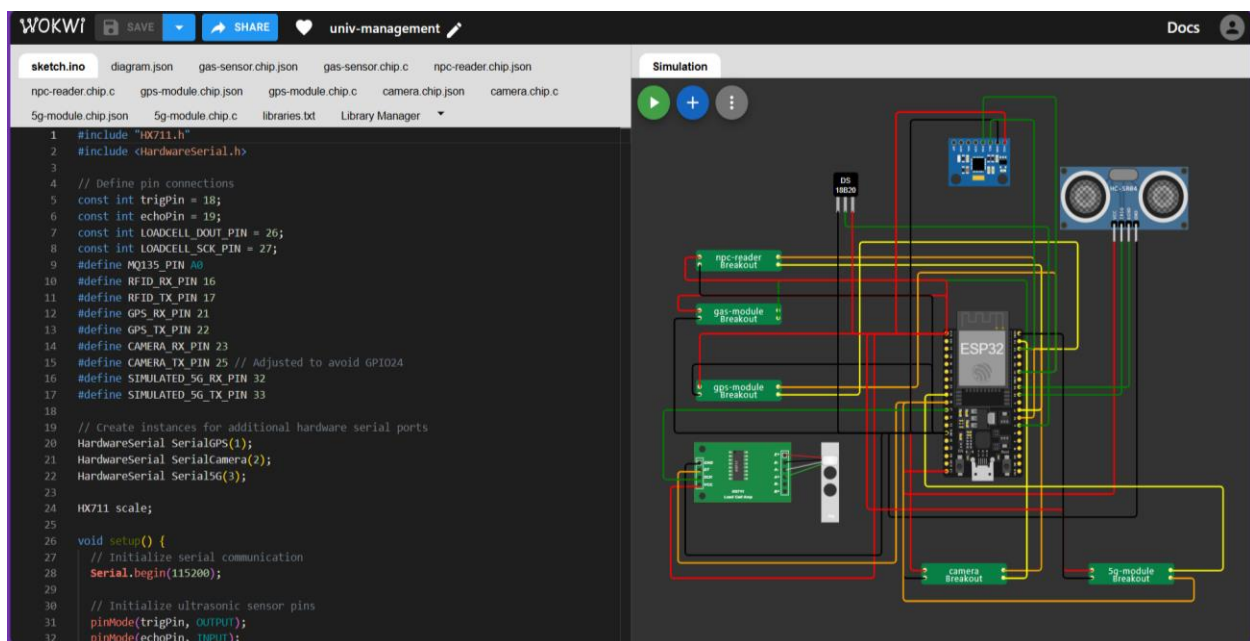
### 2. HardwareSerial.h :

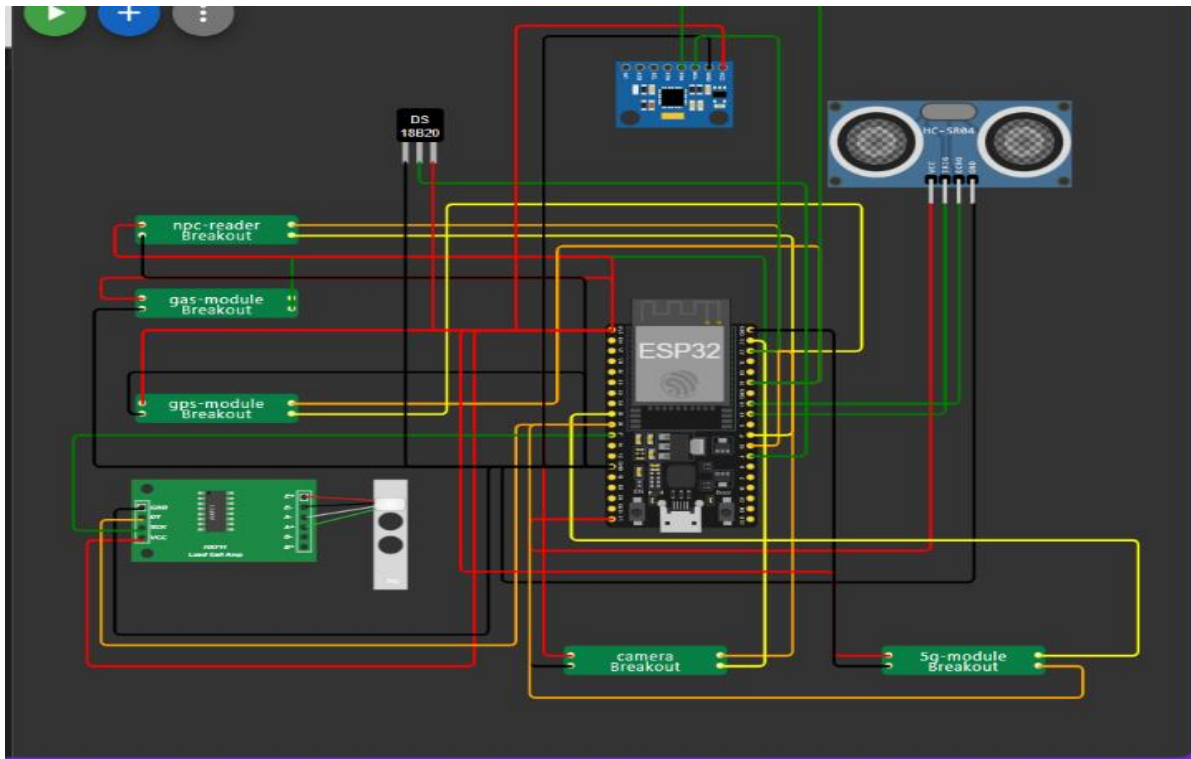
A library that allows for the creation and management of multiple hardware serial ports on the ESP32.

## Simulation and Development Environment:

### Wokwi :

An online simulator for Arduino and other microcontroller projects. It provides a virtual environment to build and test circuits and write code.





## Coding:

```
#include "HX711.h"
#include <HardwareSerial.h>

// Define pin connections
const int trigPin = 18;
const int echoPin = 19;
const int LOADCELL_DOUT_PIN = 26;
const int LOADCELL_SCK_PIN = 27;
#define MQ135_PIN A0
#define RFID_RX_PIN 16
#define RFID_TX_PIN 17
#define GPS_RX_PIN 21
#define GPS_TX_PIN 22
```

```

#define CAMERA_RX_PIN 23
#define CAMERA_TX_PIN 25 // Adjusted to avoid GPIO24
#define SIMULATED_5G_RX_PIN 32
#define SIMULATED_5G_TX_PIN 33

// Create instances for additional hardware serial ports
HardwareSerial SerialGPS(1);
HardwareSerial SerialCamera(2);
HardwareSerial Serial5G(3);

HX711 scale;

void setup() {
  // Initialize serial communication
  Serial.begin(115200);

  // Initialize ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Initialize HX711 weight sensor
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);

  // Initialize RFID reader
  Serial2.begin(9600, SERIAL_8N1, RFID_RX_PIN, RFID_TX_PIN);

  // Initialize GPS module
  SerialGPS.begin(9600, SERIAL_8N1, GPS_RX_PIN, GPS_TX_PIN);

  // Initialize Camera module
  SerialCamera.begin(9600, SERIAL_8N1, CAMERA_RX_PIN, CAMERA_TX_PIN);

  // Initialize 5G module
  Serial5G.begin(9600, SERIAL_8N1, SIMULATED_5G_RX_PIN, SIMULATED_5G_TX_PIN);
}

```

```
void loop() {  
  // Simulate Ultrasonic Sensor  
  int distance = simulateUltrasonic();  
  Serial.print("Distance: ");  
  Serial.println(distance);  
  
  // Simulate Weight Sensor  
  long weight = simulateWeight();  
  Serial.print("Weight: ");  
  Serial.println(weight);  
  
  // Simulate Gas Sensor  
  int gasValue = simulateGas();  
  Serial.print("Gas Value: ");  
  Serial.println(gasValue);  
  
  // Simulate RFID/NFC Reader  
  String rfidData = simulateRFID();  
  Serial.print("RFID Data: ");  
  Serial.println(rfidData);  
  
  // Simulate GPS Module  
  String gpsData = simulateGPS();  
  Serial.print("GPS Data: ");  
  Serial.println(gpsData);  
  
  // Simulate Camera Module  
  String cameraData = simulateCamera();  
  Serial.print("Camera Data: ");  
  Serial.println(cameraData);  
  
  // Simulate 5G Module
```

```

    String dataPacket = "Distance: " + String(distance) + ", Weight: " + String(weight) + ", Gas: " + String(gasValue) + ", RFID: " + rfidData + ", GPS: " + gpsData + ", Camera: " + cameraData;
    simulate5GTransmission(dataPacket);

    delay(1000);
}

// Simulated Functions
int simulateUltrasonic() {
    // Generate a random distance between 10 and 100 cm
    return random(10, 101);
}

long simulateWeight() {
    // Generate a random weight between 0 and 1000 grams
    return random(0, 1001);
}

int simulateGas() {
    // Generate a random gas concentration value between 0 and 1023
    return analogRead(MQ135_PIN);
}

String simulateRFID() {
    // Simulate reading a fixed RFID tag data
    return "E2 9A 3C 58";
}

String simulateGPS() {
    // Simulate GPS coordinates
    return "Lat: 37.7749, Long: -122.4194";
}

String simulateCamera() {

```

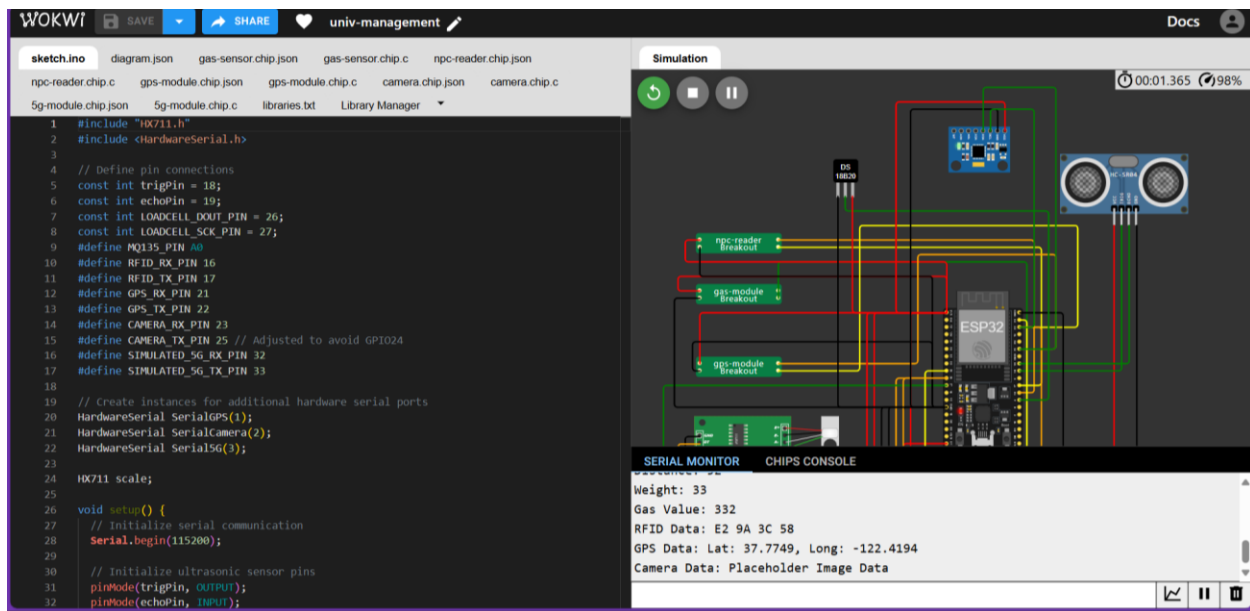
```

// Simulate placeholder image data
return "Placeholder Image Data";
}

void simulate5GTransmission(String data) {
  // Simulate data transmission over 5G
  Serial5G.println(data);
}

```

## OUTPUT:



## COMPONENTS SIMULATION USING SEPARATE SIMULATIONS:

1. GPS MODULE
2. GAS MODULE

### 1. GPS MODULE:

Components Used:

1. Arduino Uno
2. Liquid Crystal Display (LCD)
3. Red LED
4. Green LED

5. Buzzer
6. Fan (represented by a motor or a similar component)
7. Gas Sensor (analog sensor)
8. Relay Module
9. Breadboard
10. Resistors

#### Software Libraries and Tools:

1. LiquidCrystal.h

#### Project Functional Overview:

1. Pin Configuration
2. LCD Initialization
3. System Status Indication
4. Gas Detection and Alarm Activation
5. Fan Activation via Relay

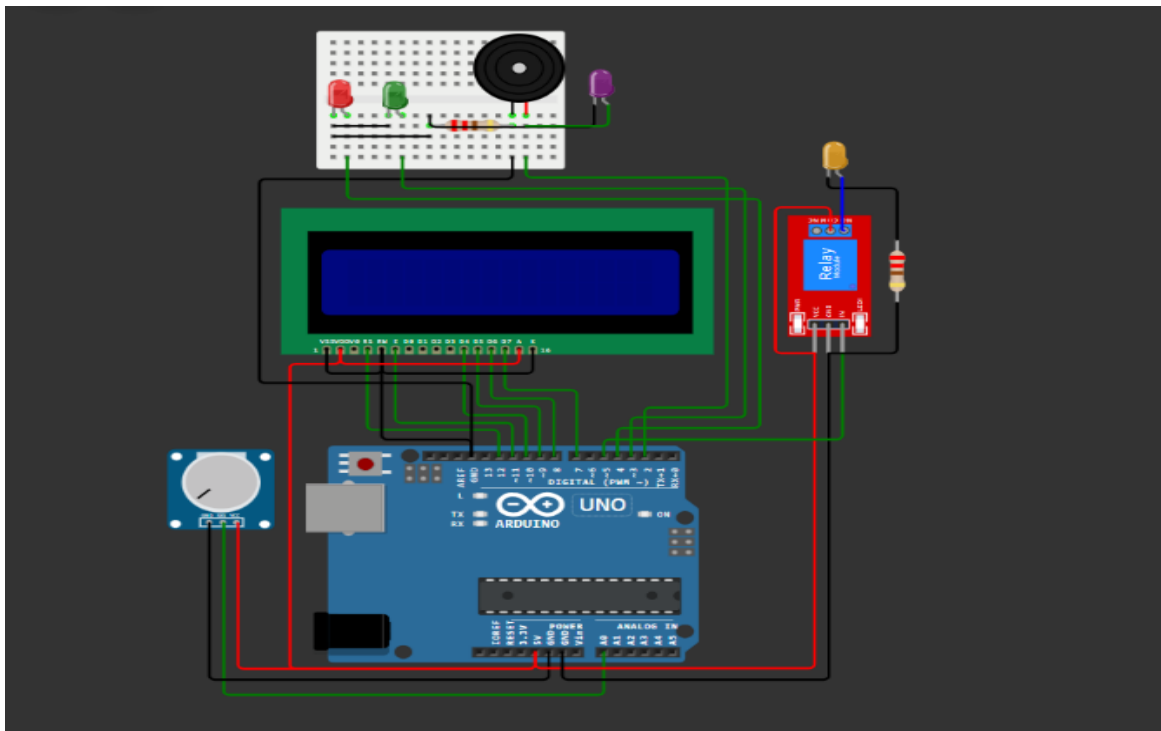
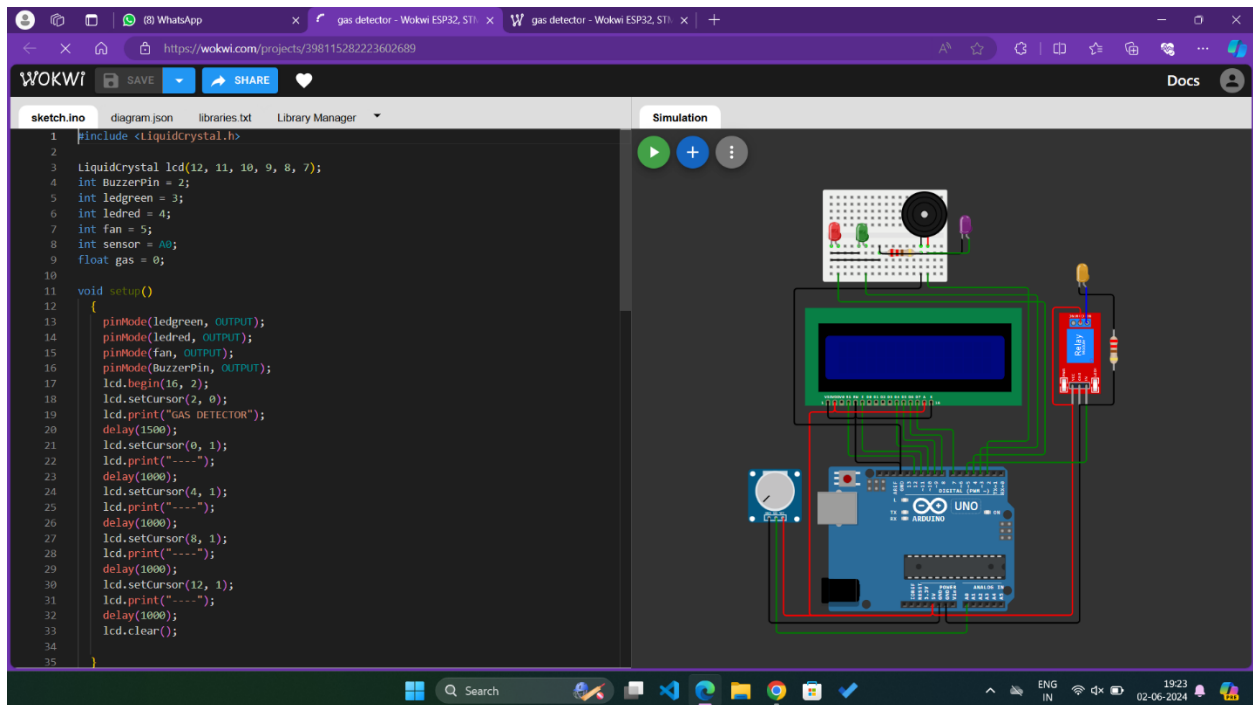
#### Code Snippets:

1. Pin Mode Setup
2. LCD Display Messages
3. Gas Sensor Reading
4. LED and Buzzer Control
5. Fan Control via Relay

#### Integration:

1. Gas Sensor Connection
2. LED and Buzzer Connection
3. Fan and Relay Connection
4. LCD Connection

## Simulation:





**Coding:**

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

int BuzzerPin = 2;
int ledgreen = 3;
int ledred = 4;
int fan = 5;
int sensor = A0;
float gas = 0;

void setup()
{
  pinMode(ledgreen, OUTPUT);
  pinMode(ledred, OUTPUT);
  pinMode(fan, OUTPUT);
  pinMode(BuzzerPin, OUTPUT);
  lcd.begin(16, 2);
  lcd.setCursor(2, 0);
  lcd.print("GAS DETECTOR");
  delay(1500);
  lcd.setCursor(0, 1);
  lcd.print("----");
  delay(1000);
  lcd.setCursor(4, 1);
  lcd.print("----");
```

```

    delay(1000);
    lcd.setCursor(8, 1);
    lcd.print("----");
    delay(1000);
    lcd.setCursor(12, 1);
    lcd.print("----");
    delay(1000);
    lcd.clear();

}

void loop() {

    gas = analogRead(sensor);
    gas = gas*100/1023;

    lcd.setCursor(0, 0);
    lcd.println("GAS =");
    lcd.print(gas);
    lcd.print(" % ");

    if (gas > 50)
    {
        digitalWrite(BuzzerPin, HIGH);
        digitalWrite(ledred, HIGH);
        digitalWrite(ledgreen, LOW);
        digitalWrite(fan, HIGH);
    }
}

```

```

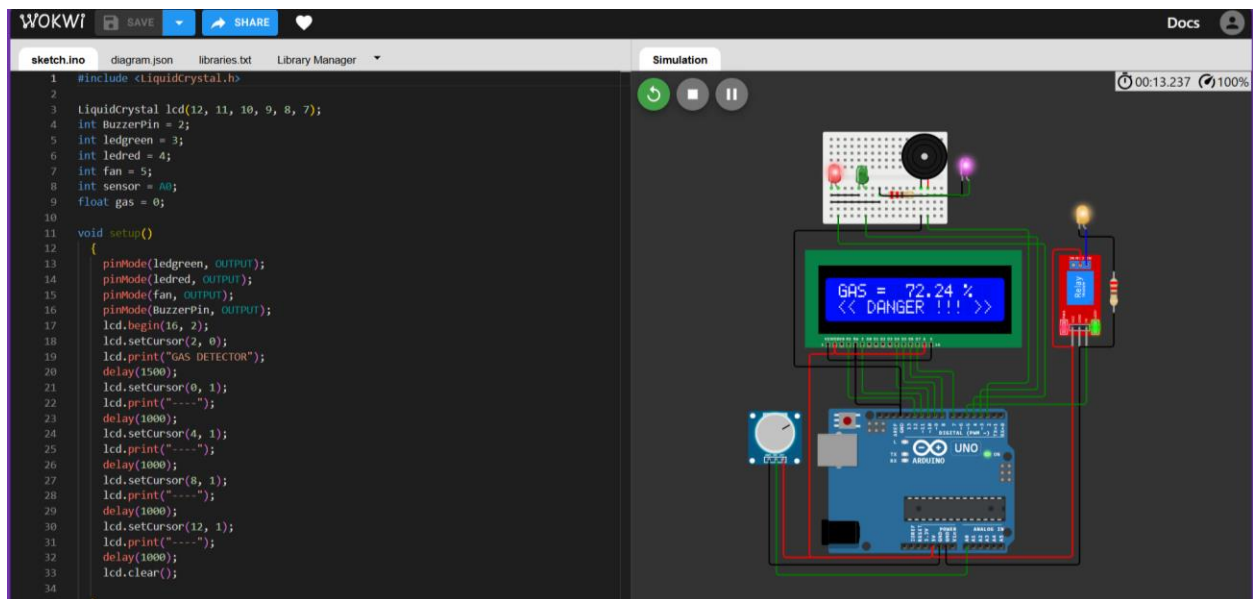
    lcd.setCursor(0, 1);
    lcd.print("<< DANGER !!! >>");

}
else
{
    digitalWrite(BuzzerPin, LOW);
    digitalWrite(ledred, LOW);
    digitalWrite(ledgreen, HIGH);
    digitalWrite(fan, LOW);

    lcd.setCursor(0, 1);
    lcd.print("<OUT OF DANGER.>");
}
}
delay(100);
}

```

## OUTPUT:



## **2. GPS MODULE**

Components Used:

1. ESP32 Microcontroller
2. GPS Module

Software Libraries and Tools:

1. WiFi.h
2. ArduinoHttpClient.h
3. NMEA.h

Project Functional Overview:

1. Wi-Fi Connection
2. GPS Data Parsing
3. WebSocket Communication
4. Data Transmission

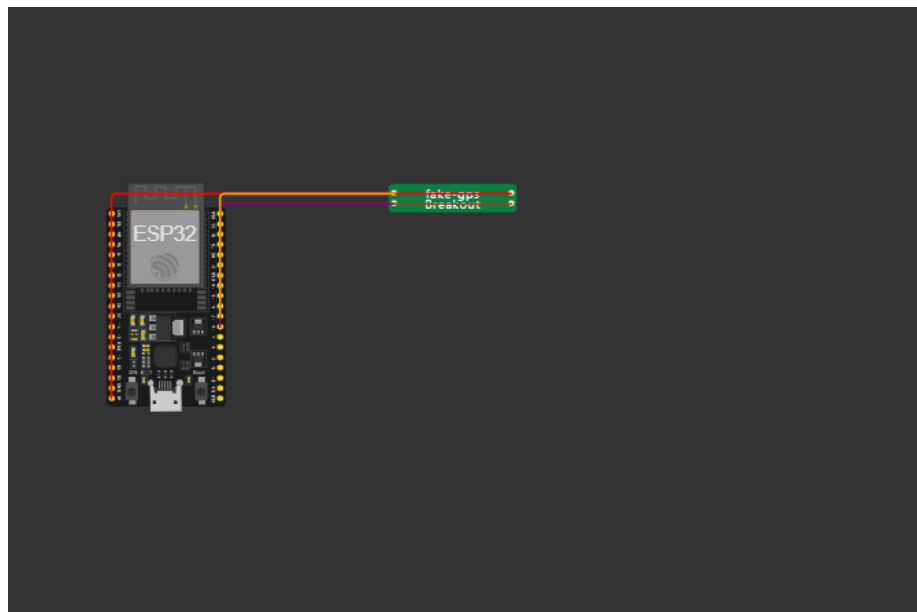
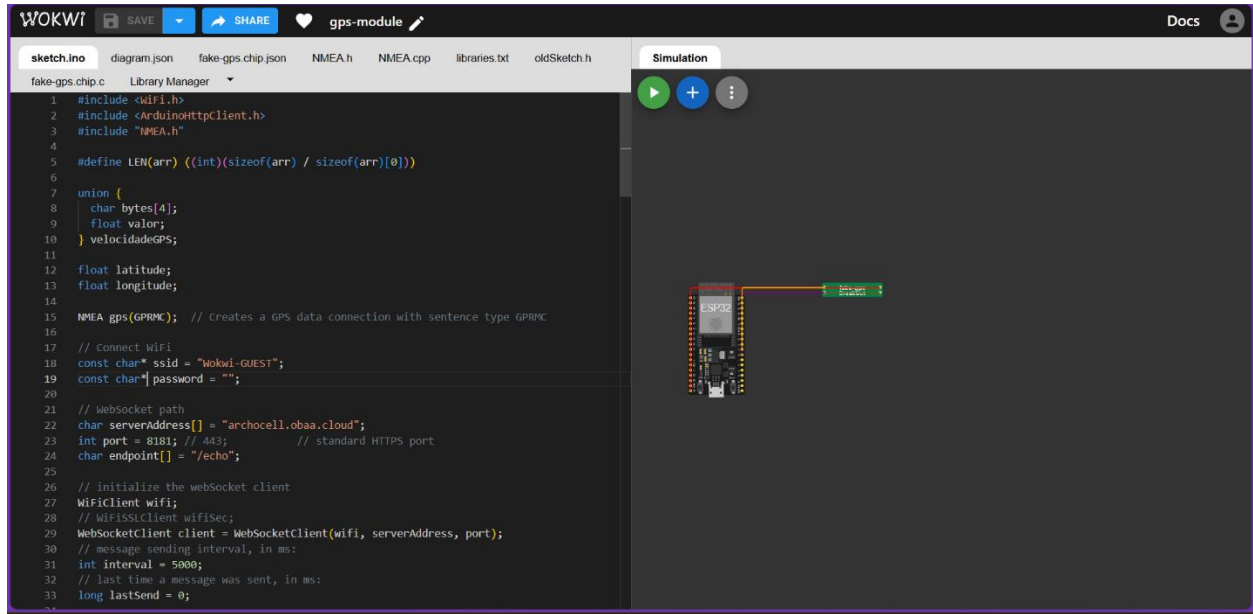
Code Snippets:

1. Union Definition
2. Global Variables

Integration:

1. GPS Module Connection

## Simulation:



### Coding:

```
#include <WiFi.h>
#include <ArduinoHttpClient.h>
#include "NMEA.h"

#define LEN(arr) ((int)(sizeof(arr) / sizeof(arr)[0]))

union {
    char bytes[4];
    float valor;
} velocidadeGPS;

float latitude;
float longitude;

NMEA gps(GPRMC); // Creates a GPS data connection with sentence type GPRMC

// Connect WiFi
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// WebSocket path
char serverAddress[] = "archocell.obaa.cloud";
int port = 8181; // 443; // standard HTTPS port
char endpoint[] = "/echo";

// initialize the websocket client
WiFiClient wifi;
// WiFiSSLClient wifiSec;
WebSocketClient client = WebSocketClient(wifi, serverAddress, port);
// message sending interval, in ms:
int interval = 5000;
// last time a message was sent, in ms:
long lastSend = 0;
```

```

// JSON string to store the location data
String jsonMessage;

void setup() {
  Serial.begin(115200);
  Serial2.begin(9600); // Serial2 is connected to the custom chip

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Connect to WS server
  client.begin(endpoint);

  Serial.println("Ready to receive data from GPS!");
  Serial.println();
  Serial.println();
}

void loop() {
  while (Serial2.available()) {
    char serialData = Serial2.read();
    Serial.print(serialData);

    if (gps.decode(serialData)) {
      if (gps.gprmc_status() == 'A') {
        velocidadeGPS.valor = gps.gprmc_speed(KMPH);
      } else {
        velocidadeGPS.valor = 0;
      }
    }
  }
}

```

```

latitude = gps.gprmc_latitude();
longitude = gps.gprmc_longitude();

// Add line break
Serial.println();
Serial.println();

// Show Latitude
Serial.print(" Latitude: ");
Serial.println(latitude, 8);

// Show Longitude
Serial.print("Longitude: ");
Serial.println(longitude, 8);

// Show Speed in km/h
Serial.print("  Speed: ");
Serial.print(velocidadeGPS.valor);
Serial.println(" Km/h");

// Converts Geographic Coordinates to Cartesian Plane
convertCoordinatesToCartesian(latitude, longitude);

// Create JSON message
jsonMessage = "{\"latitude\":\"" + String(latitude, 8) +
               "\",\"longitude\":\"" + String(longitude, 8) +
               "\",\"speed\":\"" + String(velocidadeGPS.valor) + "\"}";

// Send JSON message to WebSocket server
client.beginMessage(TYPE_TEXT);
client.print(jsonMessage);
client.endMessage();
Serial.print("Sending: ");
Serial.println(jsonMessage);

```



```

// Check if it's time to send the message
if (millis() - lastSend > interval) {

    // Update last send time
    lastSend = millis();
}
}
}
}

void convertCoordinatesToCartesian(float latitude, float longitude) {
    float latRadius = latitude * (PI) / 180;
    float lonRadius = longitude * (PI) / 180;

    int earthRadius = 6371;

    float posX = earthRadius * cos(latRadius) * cos(lonRadius);
    float posY = earthRadius * cos(latRadius) * sin(lonRadius);

    Serial.println("Cartesian Plane");
    Serial.print("    X: ");
    Serial.println(posX);
    Serial.print("    Y: ");
    Serial.println(posY);
}

```

## OUTPUT:

The screenshot displays the Wokwi IDE interface. On the left, the sketch editor shows the following code:

```
1 #include <WiFi.h>
2 #include <ArduinoHttpClient.h>
3 #include "NMEA.h"
4
5 #define LEN(arr) ((int)(sizeof(arr) / sizeof(arr)[0]))
6
7 union {
8   char bytes[4];
9   float valor;
10 } velocidadGPS;
11
12 float latitude;
13 float longitude;
14
15 NMEA gps(GPRMC); // Creates a GPS data connection with sentence type GPRMC.
16
17 // Connect WiFi
18 const char* ssid = "Wokwi-GUEST";
19 const char* password = "";
20
21 // WebSocket path
22 char serverAddress[] = "archocell.obaa.cloud";
23 int port = 8181; // 443; // standard HTTPS port
24 char endpoint[] = "/echo";
25
26 // initialize the websocket client
27 WiFiClient wifi;
28 // WiFiSSLClient wifiSec;
29 WebSocketClient client = WebSocketClient(wifi, serverAddress, port);
30 // message sending interval, in ms:
31 int interval = 5000;
32 // last time a message was sent, in ms:
33 long lastSend = 0;
```

The simulation window on the right shows a visual representation of the ESP32 board connected to a 'fake\_gps\_breakout' module. Below the simulation, the 'SERIAL MONITOR' and 'CHIPS CONSOLE' tabs are active. The serial monitor displays the following output:

```
Y: -4595.14
Sending: {"latitude":-23.46636581,"longitude":-51.84015656,"speed":17.96}

$GPGGA,172916.049,2327.980,S,05150.408,W,1,12,1.0,0.0,M,0.0,M,,*6E
$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30
```

## **Artifacts used in Lid Opener:**

### **1. Arduino Uno Board:**

- A microcontroller board based on the ATmega328P. It is the central unit for processing input from sensors and controlling the output devices such as LEDs and the servo motor.

### **2. Ultrasonic Sensor (HC-SR04):**

- Used to measure the distance of an object from the sensor. It has four pins: VCC, Trig, Echo, and GND. The Trig pin is used to send out a pulse, and the Echo pin receives the pulse reflected back from the object.

### **3. Servo Motor:**

- A small motor with a control circuit that allows precise control of the angular position. It is used here to open and close the lid of the waste bin.

### **4. Red LED:**

- Indicates a specific status (e.g., when the lid is closed or if an error occurs). It is connected to a digital output pin on the Arduino through a current-limiting resistor.

### **5. Green LED:**

- Indicates another status (e.g., when the lid is open). It is also connected to a digital output pin on the Arduino through a current-limiting resistor.

### **6. Resistors:**

- Used to limit the current flowing through the LEDs to prevent damage. Typically, these are 220-ohm resistors.

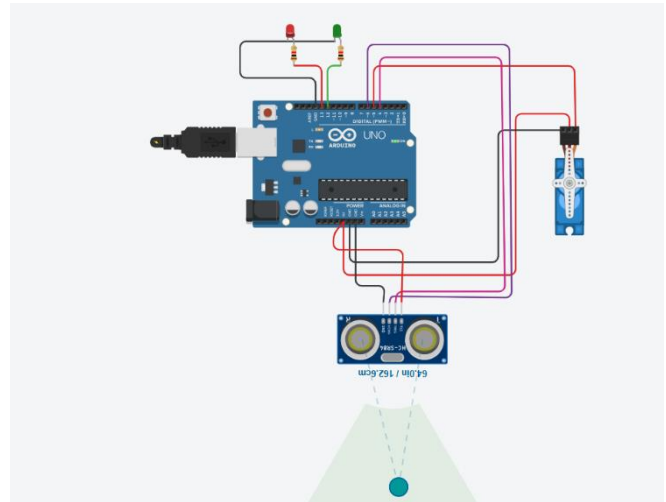
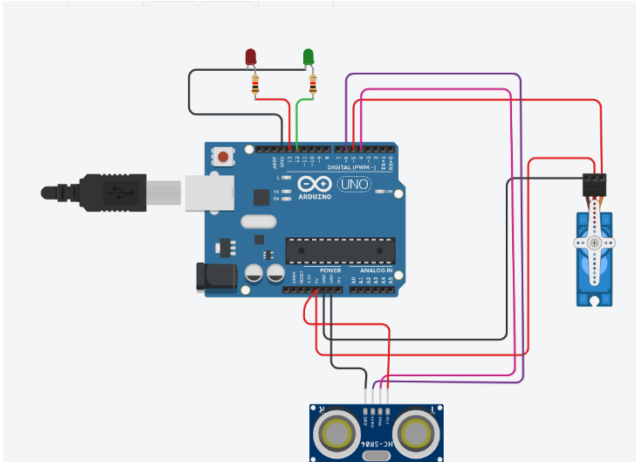
### **7. Connecting Wires:**

- Used to connect the components to the Arduino board. Different colors are often used to distinguish between different types of connections (e.g., power, ground, signal).

### **8. USB Power Supply Cable:**

- Provides power to the Arduino Uno board and allows for programming and communication with a computer

## Simulation:



## Coding:

```
#include <Servo.h>
```

```
long readUltrasonicDistance(int triggerPin, int echoPin)
```

```
{
```

```
  pinMode(triggerPin, OUTPUT); // Clear the trigger
```

```
  digitalWrite(triggerPin, LOW);
```

```
  delayMicroseconds(2);
```

```
  // Sets the trigger pin to HIGH state for 10 microseconds
```

```
  digitalWrite(triggerPin, HIGH);
```

```
  delayMicroseconds(10);
```

```
digitalWrite(triggerPin, LOW);  
  
pinMode(echoPin, INPUT);  
  
// Reads the echo pin, and returns the sound wave travel time in microseconds  
  
return pulseIn(echoPin, HIGH);  
  
}
```

```
Servo servo_5;
```

```
void setup()  
{  
  
  pinMode(12, OUTPUT);  
  
  pinMode(13, OUTPUT);  
  
  servo_5.attach(5, 500, 2500);  
  
}
```

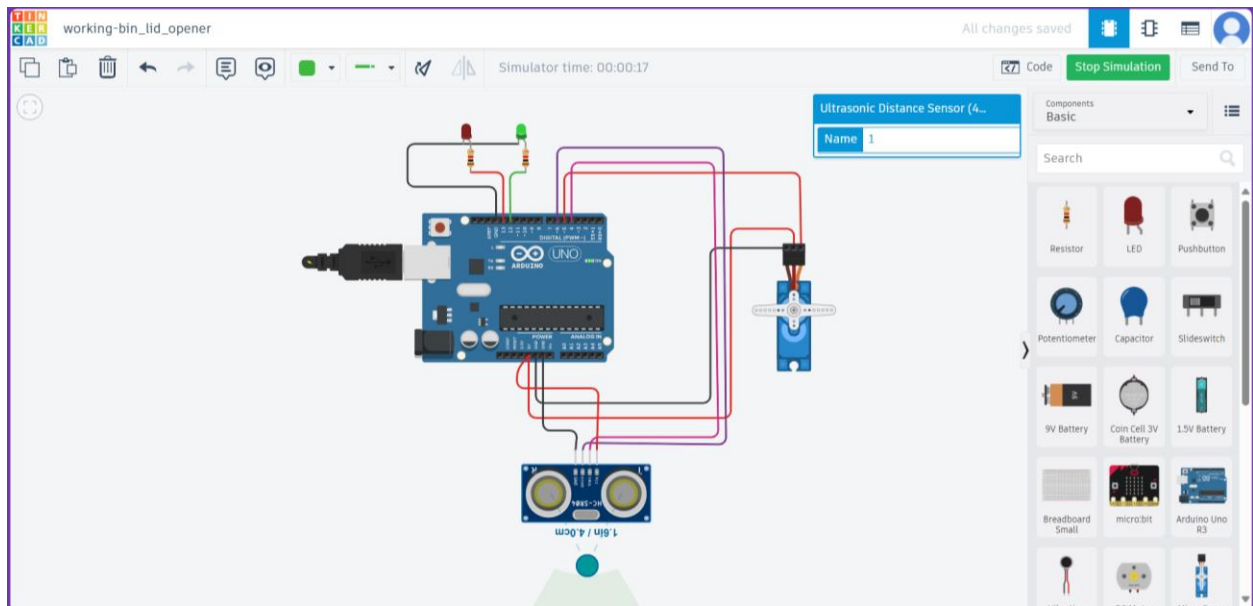
```
void loop()  
{  
  
  if (0.01723 * readUltrasonicDistance(4, 6) < 5) {  
  
    digitalWrite(12, HIGH);  
  
    digitalWrite(13, LOW);  
  
    servo_5.write(90);  
  
    delay(7000); // Wait for 7000 millisecond(s)  
  
  } else {
```

```
digitalWrite(12, LOW);
```

```
digitalWrite(13, HIGH);
```

```
servo_5.write(0);}}
```

OUTPUT:



## Artifacts used in waste level detector:

The following artifacts were utilized throughout the project:

- **Hardware Assembly and Integration:** The hardware components are assembled in line with the layout of the circuit. During this stage, all of the hardware components (Ultrasonic Sensor(HC-SR04), Weight Sensor(Load cell with HX711), Gas Sensor(MQ-135), Temperature and humidity sensor(dht22 or dht11), RFID/NFC Reader, GPS-Module(NEO – 6M), Accelerometer(ADXL345), Camera Module(ESP32-CAM) Relay components must be physically linked.
- **Development of software:** The software stack is created, Tinkercad, a web-based platform, enables the development of software for simulating and designing electronic circuits. Its user-friendly interface facilitates the creation of custom applications for interfacing with sensors in the smart waste management system.
- **With Tinkercad,** users can create 3D models of objects, buildings, and more, and then export them for 3D printing or use in other projects. Tinkercad also has a range of features that make it ideal for educational use, such as lesson plans, tutorials, and project ideas.

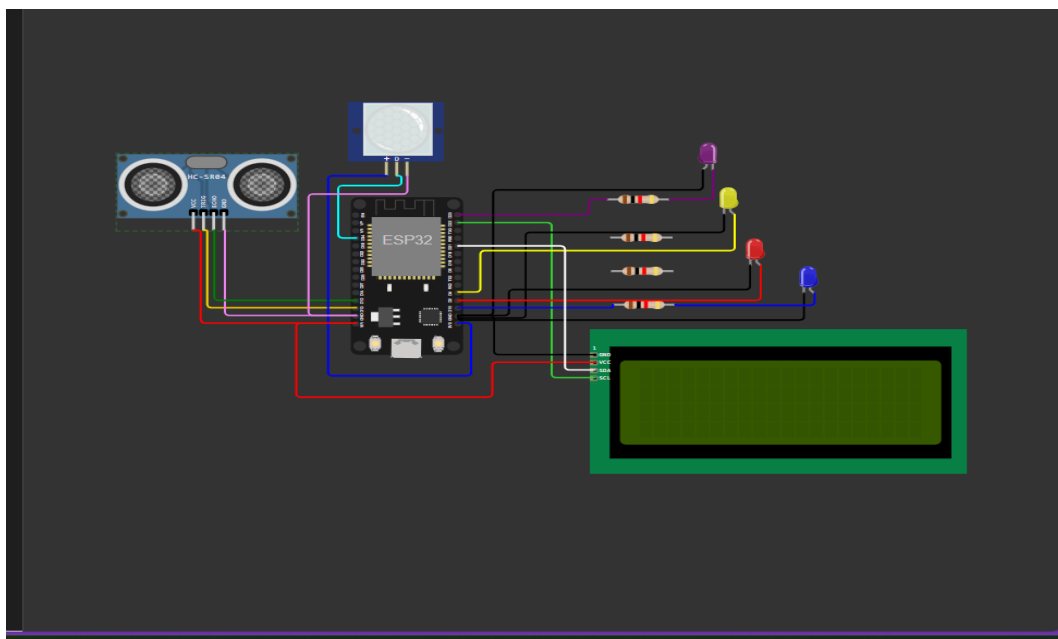
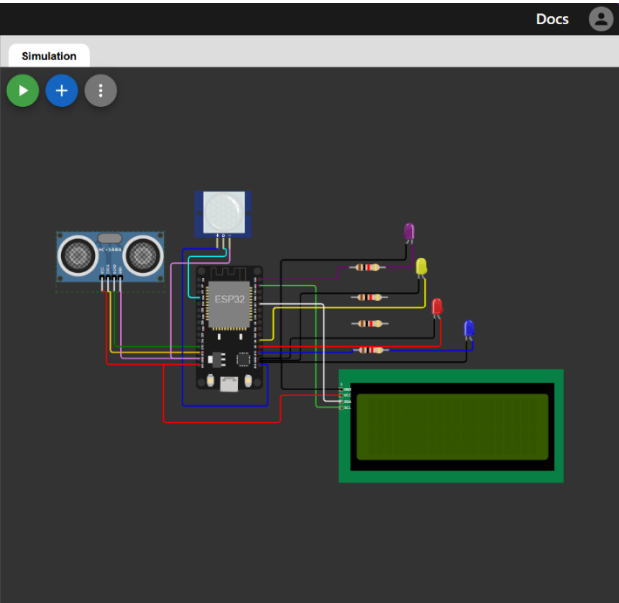
## Simulation:

WOKWI SAVE SHARE level detector Docs

sketch.ino diagram.json libraries.txt Library Manager

```
1
2 #include <LiquidCrystal_I2C.h>
3
4 LiquidCrystal_I2C lcd(0x27, 20, 4);
5
6 #define ECHO_PIN 12
7 #define TRIG_PIN 13
8
9 float dist;
10
11 void setup() {
12   Serial.begin(115200);
13
14   lcd.init();
15   lcd.backlight();
16   lcd.setCursor(1, 0);
17   lcd.print("Initializing...");
18
19   pinMode(LED_BUILTIN, OUTPUT);
20   pinMode(TRIG_PIN, OUTPUT);
21   pinMode(ECHO_PIN, INPUT);
22 }
23
24 float readcmCM() {
25   // simulate distance reading
26   return random(0, 400); // Random value between 0 and 400 (simulating distance in cm)
27 }
28
29 void loop() {
30   float cm = readcmCM();
31
32   // Simulate motion detection
33   bool motionDetected = random(0, 2);
34
35   if (motionDetected) {
```

Simulation





## Coding

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

#define ECHO_PIN 12

#define TRIG_PIN 13

float dist;

void setup() {

  Serial.begin(115200);

  lcd.init();

  lcd.backlight();

  lcd.setCursor(1, 0);

  lcd.print("Initializing...");

  pinMode(LED_BUILTIN, OUTPUT);

  pinMode(TRIG_PIN, OUTPUT);

  pinMode(ECHO_PIN, INPUT);

}

float readcmCM() {

  return random(0, 400); (simulating distance in cm)

}

void loop() {

  float cm = readcmCM();
```

```

bool motionDetected = random(0, 2);

if (motionDetected) {

  Serial.println("Motion Detected");

  Serial.println("Lid Opened");

  digitalWrite(LED_BUILTIN, HIGH);

} else {

  digitalWrite(LED_BUILTIN, LOW);

}

if (cm <= 100) {

  Serial.println("High Alert!!!, Trash bin is about to be full");

  Serial.println("Lid Closed");

  lcd.clear();

  lcd.print("Full! Don't use");

  delay(2000);

  lcd.clear();

} else if (cm > 150 && cm < 250) {

  Serial.println("Warning!!, Trash is about to cross 50% of bin level");

} else if (cm > 250 && cm <= 400) {

  Serial.println("Bin is available");

}

float inches = (cm / 2.54);

lcd.setCursor(0, 0);

```

```

    lcd.print("Inches");

    lcd.setCursor(4, 0);

    lcd.print("cm");

    lcd.setCursor(1, 1);

    lcd.print(inches, 1);

    lcd.setCursor(11, 1);

    lcd.print(cm, 1);

    lcd.setCursor(14, 1);

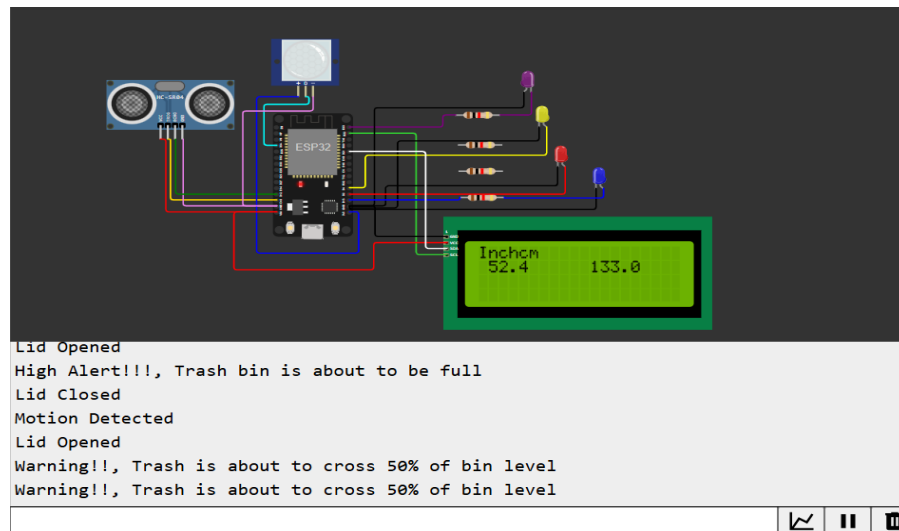
    delay(1000);

    lcd.clear();

    delay(1000);
}

```

## Output:



## **Artifacts used in dry and wet waste segregator:**

### **1. Arduino Uno Board:**

- A microcontroller board based on the ATmega328P, used for processing and controlling the system.

### **2. Ultrasonic Sensor (HC-SR04):**

- A sensor used for measuring distance by emitting ultrasonic waves and measuring their reflection.

### **3. Servo Motor:**

- A motor used for precise control of angular position, typically to open or close a lid.

### **4. LCD Display:**

- A 16x2 character LCD used to display various information such as sensor readings or system status.

### **5. Soil Moisture Sensor:**

- A sensor used to measure the moisture content in soil.

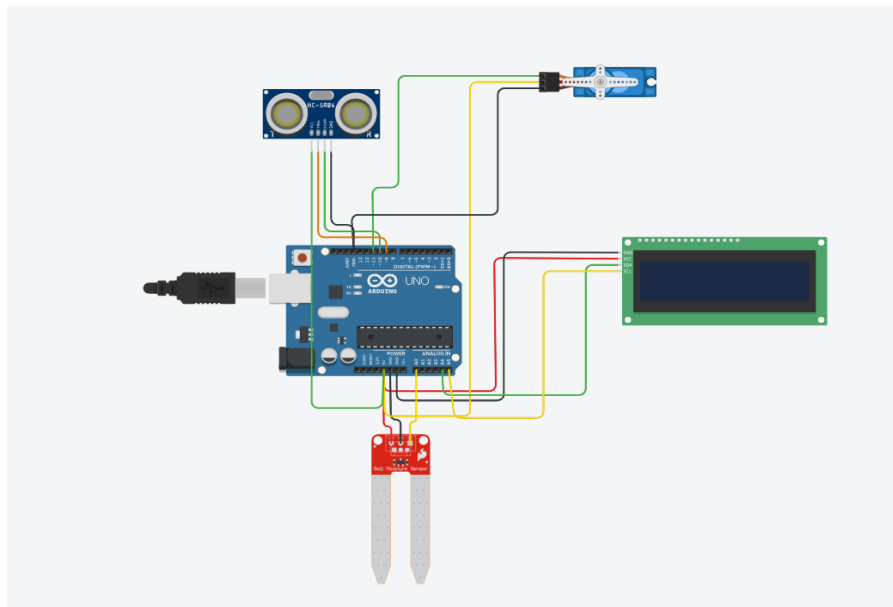
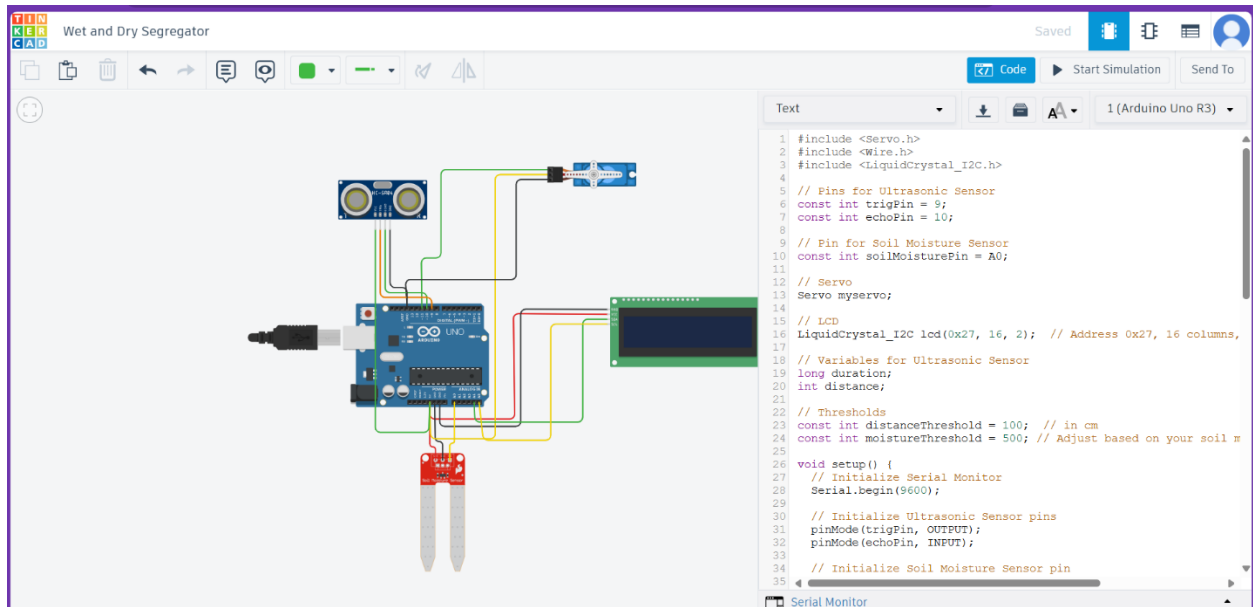
### **6. Connecting Wires:**

- Various wires used to connect components to the Arduino board.

### **7. USB Power Supply Cable:**

- A cable used to provide power to the Arduino Uno and facilitate programming and communication with a computer.

## Simulation:



## CODE:

```
#include <Servo.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>

// Pins for Ultrasonic Sensor

const int trigPin = 9;

const int echoPin = 10;

// Pin for Soil Moisture Sensor

const int soilMoisturePin = A0;

// Servo

Servo myservo;

// LCD

LiquidCrystal_I2C lcd(0x27, 16, 2); // Address 0x27, 16 columns, 2 rows

// Variables for Ultrasonic Sensor

long duration;

int distance;

// Thresholds

const int distanceThreshold = 100; // in cm

const int moistureThreshold = 500; // Adjust based on your soil moisture sensor's readings

void setup() {

    // Initialize Serial Monitor

    Serial.begin(9600);

    // Initialize Ultrasonic Sensor pins
```

```

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

// Initialize Soil Moisture Sensor pin
pinMode(soilMoisturePin, INPUT);

// Initialize Servo
myservo.attach(11);

// Initialize LCD
lcd.init();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print("Wet/Dry Segregator");

delay(2000); // Display welcome message for 2 seconds
}

void loop() {

    // Read distance from Ultrasonic Sensor

    digitalWrite(trigPin, LOW);

    delayMicroseconds(2);

    digitalWrite(trigPin, HIGH);

    delayMicroseconds(10);

    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH, 30000); // Adding a timeout of 30ms (30000
microseconds)

```

```

distance = duration * 0.034 / 2;

// Read value from Soil Moisture Sensor

int soilMoistureValue = analogRead(soilMoisturePin);


// Display values on LCD

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Dist: ");

lcd.print(distance);

lcd.print(" cm");

lcd.setCursor(0, 1);

lcd.print("Moist: ");

lcd.print(soilMoistureValue);

// Determine if the object is wet or dry based on soil moisture value

if (distance > 0 && distance <= distanceThreshold) {

    if (soilMoistureValue > moistureThreshold) {

        // Wet object

        myservo.write(90); // Adjust servo position for wet object

        lcd.setCursor(10, 1);

        lcd.print("Wet ");

        Serial.println("Wet object detected");

    } else {

        // Dry object

```



```

myservo.write(0); // Adjust servo position for dry object

lcd.setCursor(10, 1);

lcd.print("Dry ");

Serial.println("Dry object detected");

}

} else {

    Serial.println("No object detected within threshold");

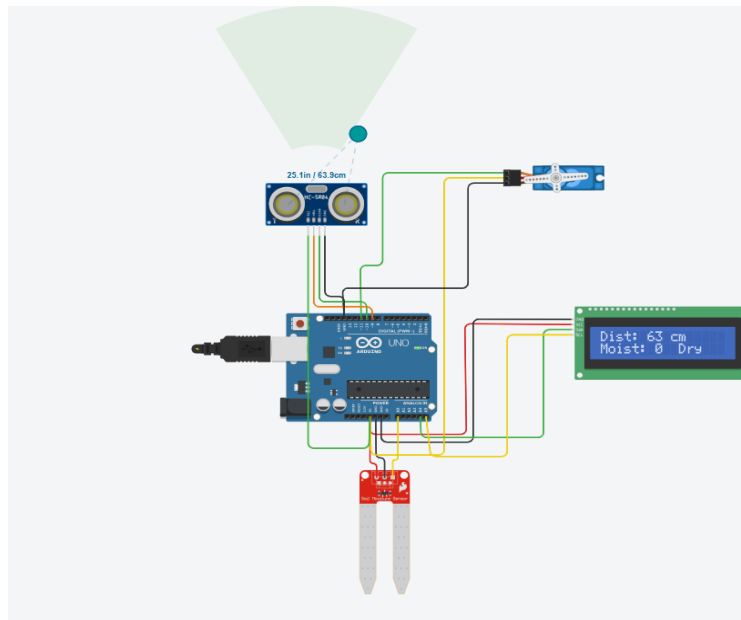
}

delay(1000); // Wait for a second before the next reading

}

```

## OUTPUT



## Technical coverage

The IoT-based smart waste management system for a university campus involves a comprehensive range of technical components and considerations. The technical coverage includes the following key elements:

### Sensor Integration

- **Ultrasonic Sensors (HC-SR04):** Measure the fill levels of waste bins using ultrasonic waves.
- **Weight Sensors (Load Cell with HX711):** Determine the weight of the waste collected in bins.
- **Gas Sensors (MQ-135):** Detect harmful gases such as ammonia and methane.
- **Temperature and Humidity Sensors (DHT22 or DHT11):** Monitor environmental conditions around the waste bins.
- **RFID/NFC Readers:** Enable identification and tracking of individual waste bins.
- **GPS Modules (NEO-6M):** Provide real-time location tracking of waste bins.
- **Accelerometers (ADXL345):** Detect movements and potential tampering of waste bins.
- **Camera Modules (ESP32-CAM):** Offer visual monitoring of the waste bins' status.

### Microcontroller and Communication

- **Arduino or Raspberry Pi:** Serve as the central processing units for the system.
- **Wireless Communication Modules (Wi-Fi, GSM, or LoRa):** Facilitate real-time data transmission from waste bins to the central server.

### Software and Interface

- **Data Acquisition and Processing:** Collect and process sensor data to extract meaningful information.
- **Node-Red Web Interface:** Provide a user-friendly web interface for accessing real-time data, visualization, alerts, and reporting.
- **Cloud Integration:** Store and process data on cloud platforms for scalability and accessibility.
- **Security Protocols:** Implement robust security measures for data integrity and privacy.

## System Implementation and Optimization

- **Power Management:** Employ energy-efficient components and power-saving techniques to extend battery life.
- **System Testing and Validation:** Conduct extensive testing to validate system performance.
- **Data Analytics and Optimization:** Use advanced analytics and machine learning for predictive analysis and optimization of waste collection routes.
- **Documentation and Support:** Provide comprehensive documentation and support mechanisms for smooth operation and system longevity.

This technical coverage ensures a holistic approach to efficient waste management, contributing to a cleaner and more sustainable university campus environment.

## Results

The implementation of the smart waste management system using IoT technology on the university campus has led to several significant outcomes:

1. **Enhanced Efficiency:** Real-time monitoring of fill levels and waste weight has optimized waste collection schedules, reducing unnecessary collections and operational costs.
2. **Environmental Impact:** Improved segregation and timely collection have increased recycling rates and reduced landfill waste, promoting sustainability.
3. **Data-Driven Decisions:** The centralized data processing system provides actionable insights, enabling better resource allocation and strategic planning.
4. **User Engagement:** Mobile applications and dashboards have increased student and staff participation in waste management efforts, fostering a culture of sustainability.
5. **System Reliability:** Continuous monitoring and maintenance protocols have ensured the reliability and scalability of the system, allowing for future expansion as needed.

## Challenges and Resolutions

### Challenges:

1. High Initial Costs: Significant investment is required for sensors, microcontrollers, and integration.
2. Technical Complexity: Integrating various sensors and ensuring seamless communication can be challenging.
3. Maintenance: Regular upkeep is needed to ensure sensor functionality and system reliability.
4. Data Privacy: Handling large amounts of data raises privacy and security concerns.
5. User Adoption: Resistance to new technologies and processes may slow adoption.

### Solutions:

1. Funding and Grants: Seek funding from environmental grants and university budgets.
2. Expert Collaboration: Partner with technical experts and researchers for system design and integration.
3. Maintenance Plans: Implement scheduled maintenance and remote diagnostics.
4. Data Security Protocols: Use encryption and secure data handling practices.
5. Training Programs: Conduct training sessions and awareness campaigns to encourage user adoption.

## Advantages

### 1. Enhanced Efficiency:

- Optimized collection schedules reduce operational costs and resource usage by avoiding unnecessary collections.
- Real-time monitoring ensures timely waste bin servicing, preventing overflows and associated issues.

### 2. Environmental Benefits:

- Increased recycling rates and reduced landfill waste promote sustainability.
- Better waste segregation minimizes contamination and enhances resource recovery.

### 3. Data-Driven Decision Making:

- Centralized data processing provides actionable insights for strategic planning and resource allocation.
- Predictive analytics help forecast waste generation patterns, allowing proactive management.

### 4. User Engagement:

- Mobile applications and dashboards raise awareness and encourage participation in waste management practices.
- Incentives and notifications promote responsible waste disposal behavior among students and staff.

### 5. System Reliability and Scalability:

- Continuous monitoring and maintenance ensure system reliability.
- Scalable design allows for future expansion to accommodate growing needs.

## Disadvantages

### 1. High Initial Costs:

- Significant investment is required for sensors, microcontrollers, and other hardware.
- Developing custom software and integrating systems can be expensive.

### 2. Technical Complexity:

- Integration of various sensors and modules requires advanced technical expertise.
- Ensuring seamless communication between components can be challenging.

### 3. Maintenance Requirements:

- Regular maintenance is necessary to keep sensors and systems functioning correctly.
- Any sensor malfunctions or system failures need prompt attention, which can be resource-intensive.

### 4. Data Privacy Concerns:

- Collecting and processing large amounts of data raises privacy and security issues.
- Ensuring data protection and compliance with regulations is essential.

### 5. User Adoption:

- Effective user training is required to ensure proper use of the system.
- Initial resistance to new technologies and processes can slow adoption.

### 6. Reliability of Wireless Communication:

- Dependence on wireless communication can lead to issues with data transmission reliability, especially in areas with poor connectivity.

## Conclusion

The implementation of a smart waste management system using IoT technology on university campuses offers substantial benefits in terms of efficiency, environmental sustainability, and data-driven decision-making. By integrating sensors for real-time monitoring and employing advanced data processing techniques, universities can optimize waste collection schedules, reduce operational costs, and enhance recycling efforts. This system fosters greater user engagement and promotes a culture of sustainability among students and staff.

However, the deployment of such systems involves significant initial costs, technical complexity, and ongoing maintenance requirements. Addressing data privacy concerns and ensuring reliable wireless communication are also critical for the system's success. Despite these challenges, the long-term advantages of improved waste management, environmental stewardship, and cost savings make smart waste management systems a valuable investment for universities. With careful planning, continuous monitoring, and regular updates, these systems can significantly contribute to creating cleaner, greener, and more sustainable campus environments.

## References

Here are some references that can provide valuable insights and support for your project on smart waste management using IoT technology:

### 1. Books and Articles:

- "Smart Waste Management: A Systematic Literature Review" by Divya R. Santhosh and Dr. Shilpa G. Karvannan
- "Waste Management Practices: Municipal, Hazardous, and Industrial" by John Pichtel

### 2. Academic Papers:

- Ghosh, S., & Roy, S. (2018). "Smart Bin Implementation for Smart Cities." International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 3(1), 245-249.
- Al Mamun, M. A., Hannan, M. A., Hussain, A., & Basri, H. (2016). "Real Time Solid Waste Bin Monitoring System Framework Using Wireless Sensor Network." In Proceedings of the 4th International Conference on Electronics, Communications and Networks (CECNet), 162-167.

### 3. Websites:

- IoT For All: [Smart Waste Management](<https://www.iotforall.com/smart-waste-management>)
- Smart Cities Dive: [How Smart Cities Are Using IoT to Improve Waste Management](<https://www.smartcitiesdive.com/news/how-smart-cities-are-using-iot-to-improve-waste-management/570171/>)

### 4. Case Studies:

- Enevo: "How Enevo Transformed Waste Collection in Smart Cities" – Case studies available on their [website](<https://www.enevo.com/case-studies>).