IoT-Based Waste Collection System

AZZA IQBAL
Department of Electrical and Computer Engineering
University of Waterloo, Ontario, Canada
a63iqbal@uwaterloo.ca

OMKAR TINGRE
Department of Electrical and Computer Engineering
University of Waterloo, Ontario, Canada
ostingre@uwaterloo.ca

TEJAS LATNE
Department of Electrical and Computer Engineering
University of Waterloo, Ontario, Canada
tglatne@uwaterloo.ca

***Abstract*—Many countries, developing and developed alike, struggle with efficient garbage collection that benefits both residents and waste management companies. The traditional method of scheduling a fixed day of the week to collect waste is outdated, as residents and businesses cannot always commit to having their garbage pile up on that day due to several reasons which will be outlined in this paper. They often resort to dumping their waste into other bins not assigned to them which leads to unsightly overflowing of garbage bins all around the cities. This paper offers a prototype for an IoT-based smart garbage collection system that takes readings from ultrasonic sensors placed in the bin lids and dynamically decides the varying days and time of the week when collection should proceed, depending on the fullness of bins in a particular region enroute of each collection truck. All readings from bins in our prototype will be sent to an online user interface to be read and analyzed via Wi-Fi modules.**

Keywords—Waste Management, Wi-Fi Module, Ultrasonic Sensors, IoT

I. INTRODUCTION

The proliferation of IoT smart systems rapidly replacing traditional practices in our day-to-day lives requires us to rethink many outdated inefficient systems. This includes the typical weekly scheduled garbage collection which is practiced by disposal establishments around the world for the past several decades which has many disadvantages.

Residencial homeowners, governments overseeing public bins, condo managers, schools, and business owners, among many others, face these issues. When their assigned bins are full before the day of scheduled pick up, they may resort to disposing of their waste in other bins. Or they may keep piling up their trash well above the maximum limit of a bin. Overflowing bins house bacteria, insects, and vermin, as well as cause air pollution and respiratory diseases. Not only is this an unsightly view, but it is also dangerous in many cases. For example, a survey conducted in Naples, Italy in 2016, confirmed a high rate of death and cancer among residents in and around Naples, thanks to decades of overflowing toxic waste dumped by local mafias [1].

Changes in the season also play a big part in the inefficiency of the traditional fixed once-a-week garbage collection. In the summertime, for example, public areas such as parks are visited more often, and bins overflow a lot more than in the wintertime. Wintertime also calls for harsh weather conditions with strong winds that will cause a mess of piled-up trash in overflowing bins. Cleanup of these extra messes requires more cost of labor and time spent, which could have been used to implement a smarter waste management system. If a smarter IoT solution was implemented in these cases, the waste management systems would have the knowledge of when which garbage bins should be emptied sooner or earlier than otherwise planned.

This paper will outline the steps of such a system, including the main components and their requirements. It will also explore the construction of a prototype done on a small scale, which includes the process of data collection, analysis and display of data on an HTML page as an example of practical and useful implementation.

II. LITERATURE REVIEW

The idea of an IoT-based smart system to revolutionize waste collection and management systems has certainly been discussed in the past. Several ideas have been proposed, and few have been implemented. Most of the existing ideas are only commercial for consumers to buy and install for their personal use or only targeted to certain users or in particular areas.

A. SWAM System

In the city of Luxembourg, the SWAM system was proposed for businesses and large entities like malls and restaurants which uses ultrasonic sensors in bins in order to advise collection drivers of the best time to collect garbage [2]. The paper that was written on this system also proposes an extensive optimization of the route to be taken to minimize environmental impacts while maximizing quality of service. They also intended to integrate the intended disposal of large amounts of waste, such as inventory disposal by large companies to anticipate potential overflows of bins.

B. SENSONEO

There are also many companies focused on the development of smart waste collection technology, such as Slovakian-based Sensoneo. Among their products, they have developed sensors that can be placed inside bins to detect ultrasonic ranges from 3 cm up to 12 m and can be connected to several IoT networks or GPRS[3]. Although this was not implemented country-wide, it is a commercial product for homeowners to track their garbage level and collection. However, it is not being used to optimize routes or schedules from the waste management systems themselves.

C. City of Edinburgh Smart Waste Sensors Deployment

A city that has already put forward a plan for similar technology is Edinburgh, Scotland where 11,000 sensors smart bins are to be installed in bins around the city as of July 2022 [4]. The overflowing rubbish in bins in this tourist city was proving to be a big issue and this smart system was planned to be implemented in public areas where ultrasonic sensors would detect waste levels to optimize collections and manage routes more efficiently and predict trends in usage. The sensors also include heat sensors to send an alert in case of a fire. In 2016, a trial to test this system was deployed in more than 300 bins which led to an increase in collections in relevant areas by an average of 24% and even stepped up the frequency fourfold in some places.

III. System Requirements

A. Components

The IoT-Based Smart Waste Collection System we propose will comprise of the following main components which are to be installed on the inside of each garbage bin lid:

1) Ultrasonic Sensor

We require long-lasting, reliable, and robust ultrasonic sensors which will endure the conditions inside a garbage bin. These bins can typically get moist due to liquids thrown in them, so precautions like hermetic coating can help the sensor readings not be tampered by such external factors. We also need the sensors to have the adequate range to accurately detect up to the full height of the bin

2) Wi-Fi Module

In order to connect our sensors to a wireless network and have the readings sent to the main system manager, W-Fi Modules are used to send the readings to a cloud database for further processing. The Wi-Fi Module used should ideally be a low power consumption one that would require minimum replacement, and it should also be as robust and weatherproof as the sensor. Its should also have adequate range or extenders installed to account for the distance between bins.

3) User Interface

It is very important to construct useful user interfaces that will display the readings of the installed ultrasonic sensors. Most crucially, the main system manager on the waste management company's side should be able to view the fullness levels of all bins, as well as their locations updated in a timely manner. The owners of the bins can also have access to see when their bins were emptied and when they are overflowing. The user interface should be accessible anywhere so the data should be uploaded to a cloud database.

B. System Overview

Figure 1 shows the basic overview of this system in which the readings from the ultrasonic sensor are sent via the Wi-Fi Module to the cloud database for the data to be accumulated, stored and processed. After manipulating the data and extracted useful information from it, it is uploaded to a user interface for the user to view and send the decision to the waste management systems to schedule an optimized garbage pickup.

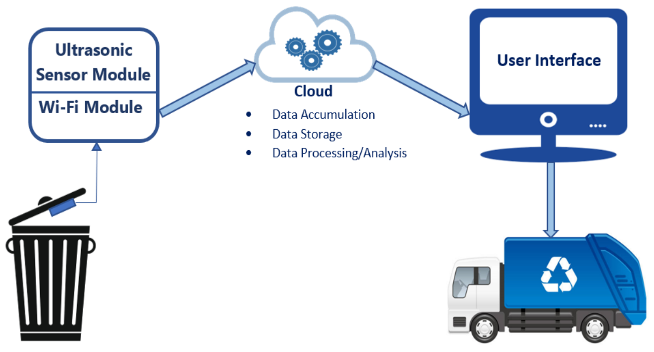


Fig. 1. Overview of IoT Based Smart Waste Collection

Any optimization that is implemented to improve the overall system will be carried out in the cloud where the data is stored and accumulated, so that a final result can be conveniently displayed for the user to see.

IV. Design and Implementation of Prototype

A majority of this paper will be discussing the construction of a prototype which was implemented as well as its readings. Keeping in mind the restricted scope of this prototype, very basic components were used and implemented at a small scale to simply mimic the ease and practicality of the functionality of such a system. Thus, proposed enhancements at each step will also be outlined in this section theoretically without including them in the prototype application.

A. Components

1) HC-SRC04 Ultrasonic Sensor

This component is an ultrasonic distance sensor with a range of 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. This is more than enough for our prototype where we assume an average Canadian bin of height 100 cm. The two ultrasonic transducers in the sensors include the transmitter, which sends ultrasonic sound pulses at 40,000 Hz (Trigger). If there is an object in the path, the wave will bounce back to the module and be heard by the receiver (Echo). The distance from the sensor, which will be mounted to the inside of the dustbin lid, to the closest bin entry is calculated with the following formula in equation 1:

$$distance = \frac{Echo\ Time \times Ultrasonic\ Speed}{2} \quad (1)$$

It is important to note that although we are taking one sensor to read the fullness of each bin in this prototype, due to the measuring angle of this sensor being only 30 degrees, larger bin dimensions would require more sensors in each bin. The bins could be split into vertical sections with a sensor reading in each section, and the mean of the readings taken to represent the fullness of that bin more accurately [5].

2) ESP8266 Node MCU

This component is a compact Wi-Fi enabled low power consumption module with a complete TCP/IP stack and flash memory. Due to its onboard processing capabilities, GPIO pins, and ability to communicate with other Wi-Fi devices, it is the only module needed to interface with the sensors. As a microprocessor, it can be programmed with the Arduino IDE to read the ultrasonic sensors that each is interfaced with.

B. Setup

In our prototype, we will use three ultrasonic sensors interfaced with 3 Wi-Fi modules. There will also be a fourth Wi-Fi Module which will act as the receiving slave device, while the other 3 modules will act as the master devices that will send their sensor's readings to the slave via ESP-NOW Communication protocol (many to one configuration) [6]. The master/slave configuration is similar to that in figure 2.

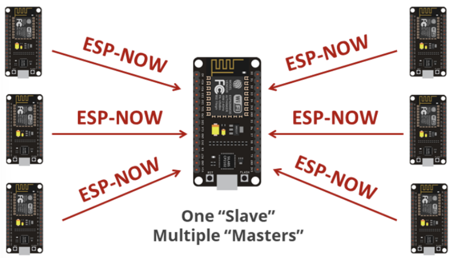


Fig. 2. Connection between Wi-Fi Modules [6]

Figure 3 shows an overview of the connections in our prototype, where each NodeMCU is powered by an external 5V source. The 5V is provided with a voltage divider circuit and a 9V battery (not shown in schematic below). The interfacing between each sensor and its Wi-Fi Module is done by connecting the Trig and Echo pin of the HC-SRC04 to any two GPIO Pins of the corresponding ESP8266

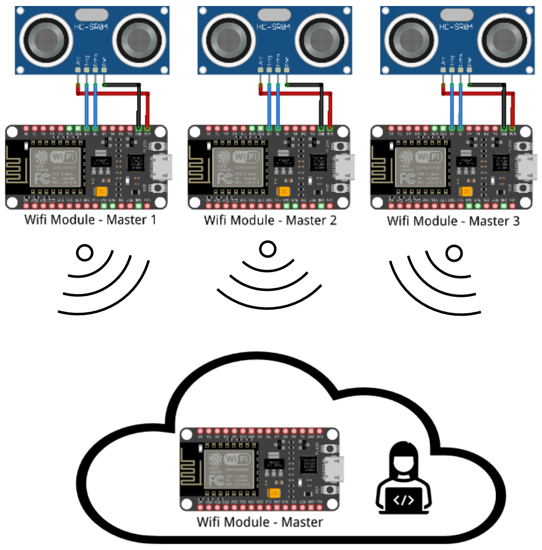


Fig. 3. Connections in the prototype

A photo of setup that was implemented is shown in figure 4

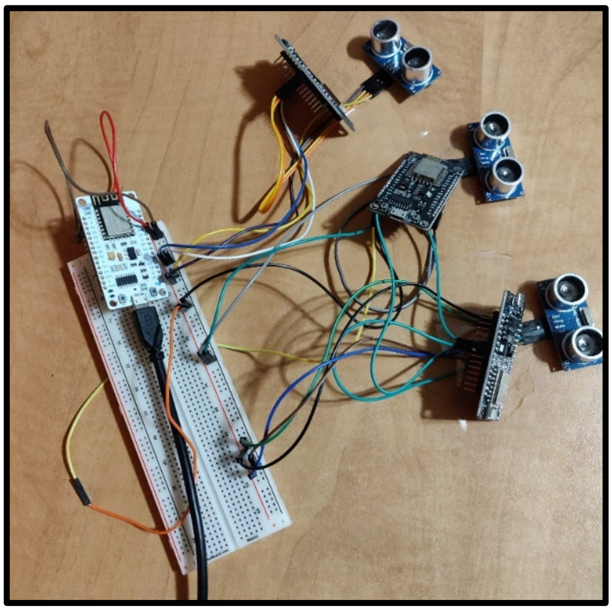


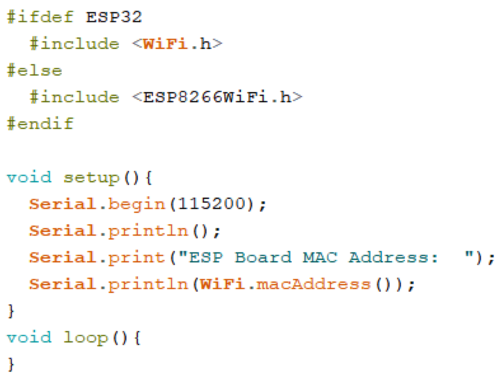
Fig. 4. Prototype Hardware Setup

This photo shows our setup with three HC-SR04 Ultrasonic distance sensors connected to three black ESP8266 Node MCU modules which are the masters. The white ESP8266 which is placed on top of the breadboard is the receiver module. All these modules and sensors are powered by an external 9V battery supply which is stepped down to 5V with a voltage divider circuit of resistors (not shown in image)

C. *Implementation of Code*

Once all the connections are set up, the four Wi-Fi Modules have to be programmed. We will write a different code for the senders and a unique one for the receiver on the Arduino IDE, and then upload the codes to the respective modules. Prior to those two codes, we will first write one to get the MAC Address of the receiver module. Next, we will write a Python code to take the readings from the serial port of the receiver Wi-Fi Module. This code will also include the deciding algorithm of what is deduced from the readings and will create a CSV file to be uploaded to a html page. Finally a HTML code is written to design a prototype of an online database which can be viewed by users. Detailed descriptions of all the codes are outlined below:

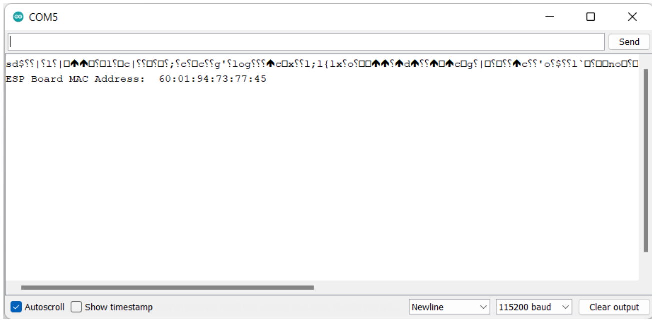
1) *Obtaining MAC Address of the reciever*



```
#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif

void setup(){
  Serial.begin(115200);
  Serial.println();
  Serial.print("ESP Board MAC Address:  ");
  Serial.println(WiFi.macAddress());
}
void loop(){
}
```

Fig. 5. Obtaining MAC Address of Reciever Wi-Fi Module

The code written on the Arduino IDE shown in figure 5 is a very simple one to obtain the receiver board's unique MAC Address. When running the code, the MAC Address is displayed on the serial monitor and noted for further use. Results from the serial monitor are shown in figure 6



```
sd$ʕʕ|ʕlʕ|□🡅🡅□ʕ□lʕ□c|ʕʕ□ʕ□ʕ;ʕcʕ□cʕʕg'ʕlogʕʕʕ🡅c□xʕʕl;l{lxʕoʕ□□🡅🡅ʕ🡅d🡅ʕʕ🡅□🡅c□gʕ|□ʕ□ʕʕ🡅cʕʕ'oʕ$ʕʕl`□ʕ□□no□ʕ□
ESP Board MAC Address:  60:01:94:73:77:45
```

Fig. 6. Serial Monitor Output for Obtaining MAC Address of Reciever

We will use this MAC address in the next sender codes so that each sender can successfully determine its receiver.

2) *Sender Codes*

The code shown in figure 7 is replicated for the three senders (masters) in this configuration, and each of them are labelled with a unique ID number 1, 2 and 3 for easy identification. The BOARD_ID variable is set to 1 in the code shown, but will be replaced with 2 and 3 before uploading to the second and third master sender boards.

The MAC address that was obtained from the code in figure 5 is placed in this code in hex format so each sender can successfully determine its receiver. The message that is being sent is split into two data packets: one is the board ID, and the second is the actual distance reading from the sensor connected to that particular Wi-Fi Module. The distance is obtained by the logic in Equation (1) described earlier in this paper.

To send the complete packet of this information, we will first it as a Wi-Fi Station, and then initialize the ESP-NOW protocol. The board role will be set to a master controller board, and a peer device (the receiver) will be paired so that data can be transferred between them. A callback function was also defined to be executed once the message is sent, and it indicates if the message was sent successfully or not.

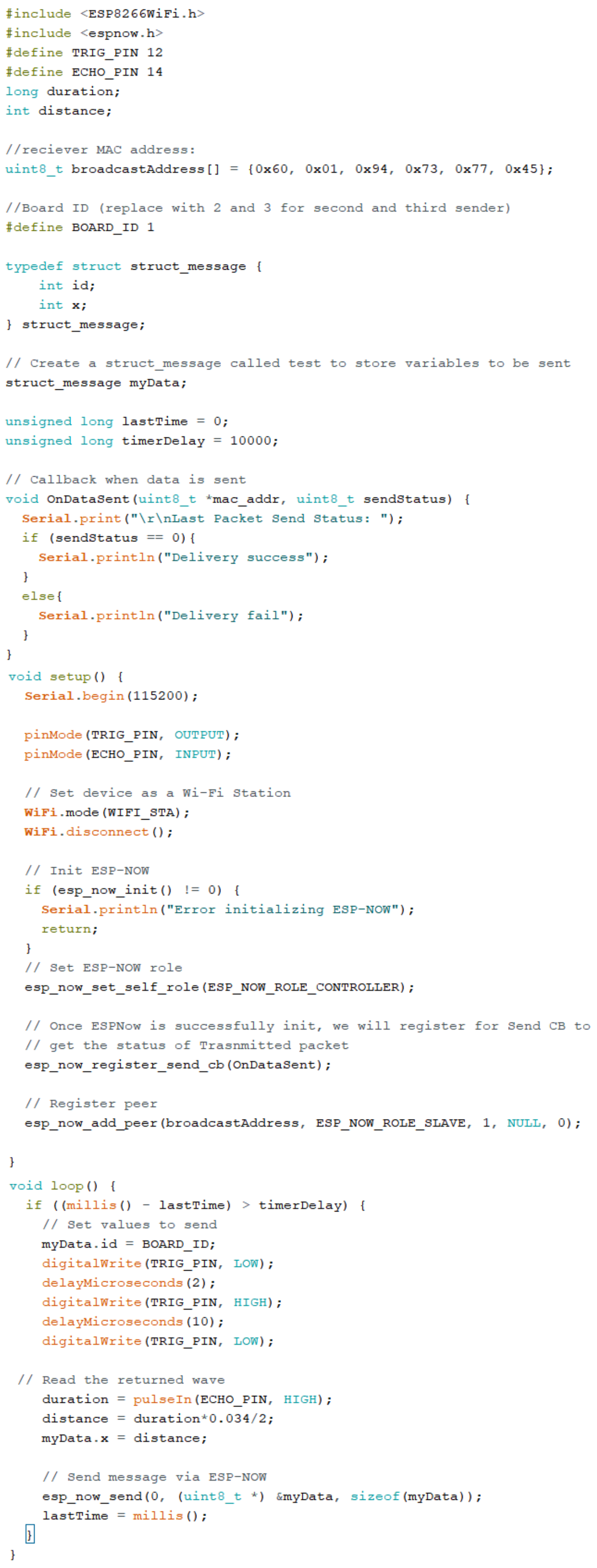
3) *Reciever Code*

Now, we will write the code that will be uploaded to our single receiver Wi-Fi Module as shown in figure 8. It allows our slave board to receive from all three sender master boards.

We start by creating a structure type which matches the sender structure exactly, where we had the ID of the sender and the distance read by the corresponding sensor. We will also define structures for each of our three boards then define an array structure to hold them all.

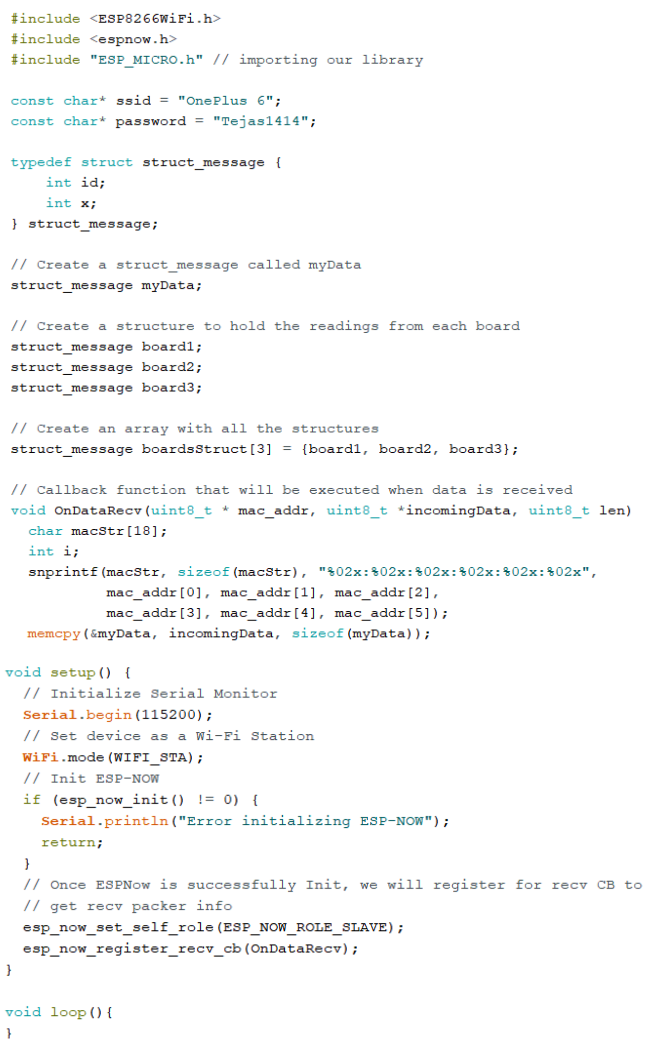
At the start of the code, we will define the SSID and password of the Wi-Fi hotspot that we used for this prototype. Next, in this code's callback message, we will get the sender board's MAC Address, and read the content of the incoming data. The data has values sent by all our sender boards, and we can identify which board sent a specific package using the ID variable in the main message structure.

Similar to the sender codes, we will set up the device as a Wi-Fi station then disconnect it, prior to initializing the ESP-NOW protocol. This time, we set the board role as receiver slave board. Readings will be sent every 10 seconds in this prototype. For debugging purposes, we have set up the serial monitor at 115200 baud rate to confirm we are getting the results that we want.



```
#include <ESP8266WiFi.h>
#include <espnow.h>
#define TRIG_PIN 12
#define ECHO_PIN 14
long duration;
int distance;

//reciever MAC address:
uint8_t broadcastAddress[] = {0x60, 0x01, 0x94, 0x73, 0x77, 0x45};

//Board ID (replace with 2 and 3 for second and third sender)
#define BOARD_ID 1

typedef struct struct_message {
    int id;
    int x;
} struct_message;

// Create a struct_message called test to store variables to be sent
struct_message myData;

unsigned long lastTime = 0;
unsigned long timerDelay = 10000;

// Callback when data is sent
void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("\r\nLast Packet Send Status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }
}
void setup() {
  Serial.begin(115200);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  // Set ESP-NOW role
  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);

  // Once ESPNow is successfully init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress, ESP_NOW_ROLE_SLAVE, 1, NULL, 0);

}
void loop() {
  if ((millis() - lastTime) > timerDelay) {
    // Set values to send
    myData.id = BOARD_ID;
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

 // Read the returned wave
    duration = pulseIn(ECHO_PIN, HIGH);
    distance = duration*0.034/2;
    myData.x = distance;

    // Send message via ESP-NOW
    esp_now_send(0, (uint8_t *) &myData, sizeof(myData));
    lastTime = millis();
  }
}
```

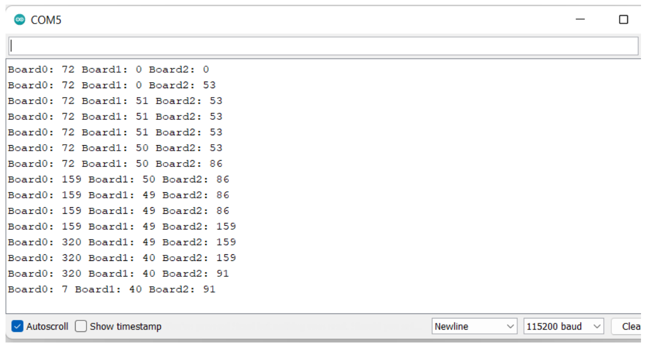
Fig. 7. Sender Arduino Code



```
#include <ESP8266WiFi.h>
#include <espnow.h>
#include "ESP_MICRO.h" // importing our library

const char* ssid = "OnePlus 6";
const char* password = "Tejas1414";

typedef struct struct_message {
    int id;
    int x;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// Create a structure to hold the readings from each board
struct_message board1;
struct_message board2;
struct_message board3;

// Create an array with all the structures
struct_message boardsStruct[3] = {board1, board2, board3};

// Callback function that will be executed when data is received
void OnDataRecv(uint8_t * mac_addr, uint8_t *incomingData, uint8_t len)
  char macStr[18];
  int i;
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2],
           mac_addr[3], mac_addr[4], mac_addr[5]);
  memcpy(&myData, incomingData, sizeof(myData));

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(OnDataRecv);
}

void loop(){
}
```

Fig. 8. Reciever Arduino Code

Upon running the receiver code, we see the results as shown in figure 9 where we have printed the ID of the board and the corresponding sensor's distance readings. These readings were obtained as we set up our equipment as shown in figure 4 and tested the sensors by varying the distance between an obstruction and the sensor. We can see the readings for Board0, Board1, Board2 and their distance readings. When the reading is 0, that indicates the object is so close to the sensor that it is almost touching, and a larger value indicates the object is that many centimeters away from the sensor. For our application, we do not expect to see a distance reading above 100 cm



```
Board0: 72 Board1: 0 Board2: 0
Board0: 72 Board1: 0 Board2: 53
Board0: 72 Board1: 51 Board2: 53
Board0: 72 Board1: 51 Board2: 53
Board0: 72 Board1: 51 Board2: 53
Board0: 72 Board1: 50 Board2: 53
Board0: 72 Board1: 50 Board2: 86
Board0: 159 Board1: 50 Board2: 86
Board0: 159 Board1: 49 Board2: 86
Board0: 159 Board1: 49 Board2: 86
Board0: 159 Board1: 49 Board2: 159
Board0: 320 Board1: 49 Board2: 159
Board0: 320 Board1: 40 Board2: 159
Board0: 320 Board1: 40 Board2: 91
Board0: 7 Board1: 40 Board2: 91
```

[✓ Autoscroll] Show timestamp Newline 115200 baud Clea

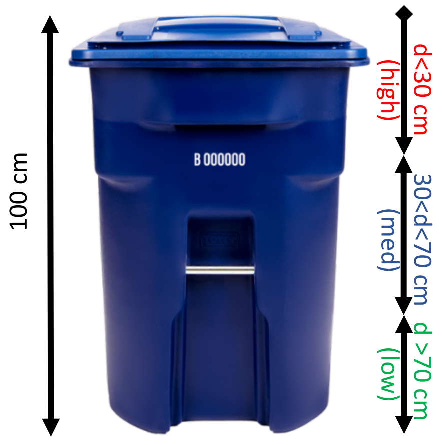
Fig. 9. Reciever Code Serial Monitor Output

4) Python Code

Now that all boards have been uploaded with their respective codes, we write a script on Python to take all these readings from the slave board, use them to make a decision, and create a CSV file of the required information to be sent to the waste collection services manager for further action. The code is shown in figure 11.

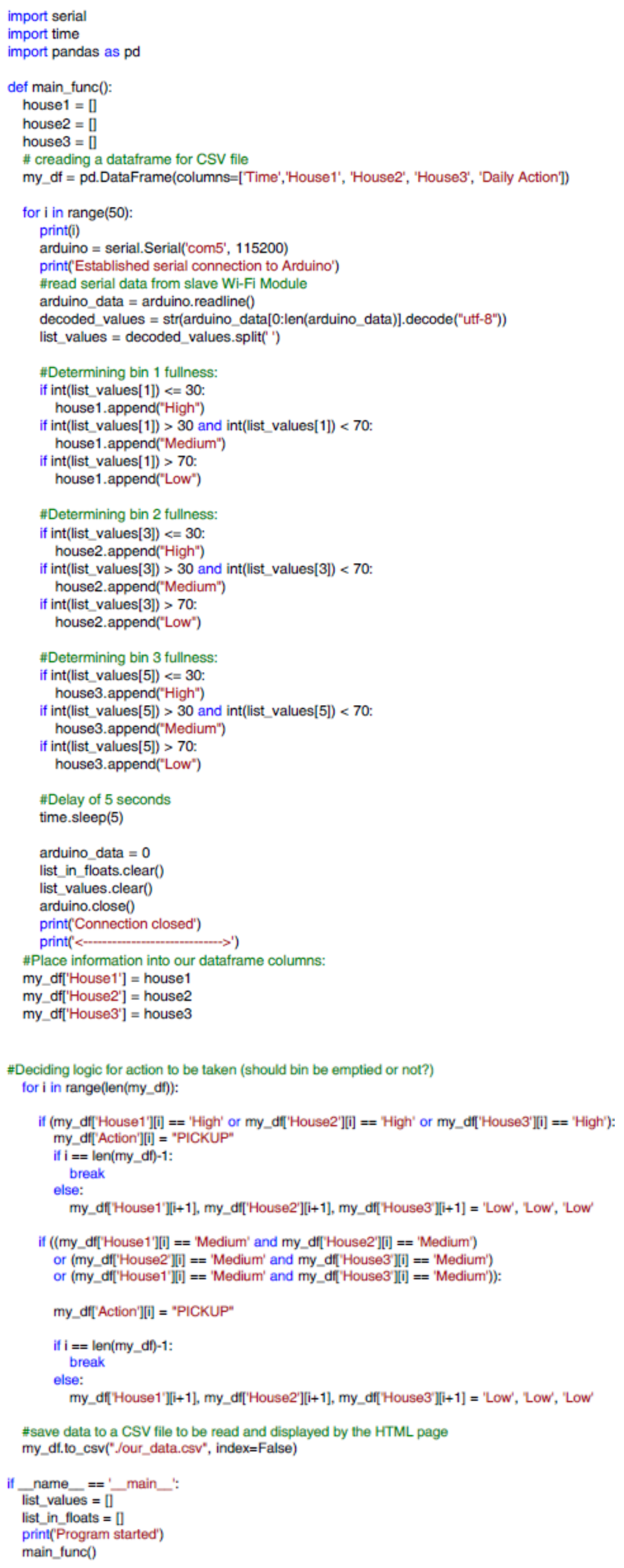
Since the outcome of this script will ultimately be a CSV file, we start by defining a Pandas Dataframe with 5 columns. This includes the time, and action to be taken based on the results, as well as our 3 'houses'. This imitates the 3 houses in a small fictional neighborhood whose bins are being monitored to decide when the waste management system should send disposal trucks to make a round in that vicinity to empty their bins. We expect the functionality to be easily expanded to a larger scope than just three houses as the design is completely modular.

We proceed to take readings from the serial port of the slave module and the values are decoded and split into an array where we have [House1 ID, House1 distance readings, House2 ID, House2 distance readings, House3 ID, House3 distance readings]. Next, we take the distance readings and determine if they fall in the range of 'high', 'medium' or 'low'. To set logical ranges for this decision, we considered the one-meter height of an average Canadian house garbage pin which is emptied by the regional garbage pickup service. We took an empty bin to be anything with a distance reading of greater than 70 cm i.e., when the distance from the latest thrown waste is more than 70 meters away from the sensor mounted on the inside of the lid. Similarly, a full bin is considered with a distance reading of less than 30 cm, and a medium reading is given to anything between 30 and 70, as shown in figure 10. The entries of 'low', 'medium' and 'high' are then placed in the corresponding columns of the dataframe.



Fig, 10. Reciever Code Serial Monitor Output

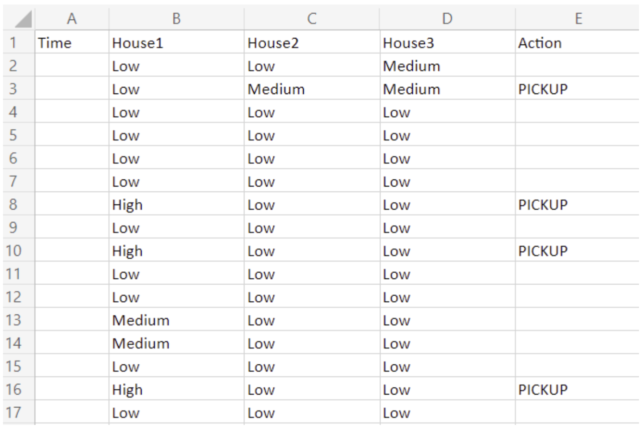
Next, we must use that information to make a decision. In this simple case of our prototype, we decided to inform the user to empty out the trash if at least one of the three bins' fullness level is 'high' or if at least two are 'medium'. The logic used to determine if pickup should be carried out at that time interval is shown further in this paper in figure 18



```
import serial
import time
import pandas as pd

def main_func():
   house1 = []
   house2 = []
   house3 = []
   # creading a dataframe for CSV file
   my_df = pd.DataFrame(columns=['Time','House1', 'House2', 'House3', 'Daily Action'])

   for i in range(50):
      print(i)
      arduino = serial.Serial('com5', 115200)
      print('Established serial connection to Arduino')
      #read serial data from slave Wi-Fi Module
      arduino_data = arduino.readline()
      decoded_values = str(arduino_data[0:len(arduino_data)].decode("utf-8"))
      list_values = decoded_values.split(' ')

      #Determining bin 1 fullness:
      if int(list_values[1]) <= 30:
         house1.append("High")
      if int(list_values[1]) > 30 and int(list_values[1]) < 70:
         house1.append("Medium")
      if int(list_values[1]) > 70:
         house1.append("Low")

      #Determining bin 2 fullness:
      if int(list_values[3]) <= 30:
         house2.append("High")
      if int(list_values[3]) > 30 and int(list_values[3]) < 70:
         house2.append("Medium")
      if int(list_values[3]) > 70:
         house2.append("Low")

      #Determining bin 3 fullness:
      if int(list_values[5]) <= 30:
         house3.append("High")
      if int(list_values[5]) > 30 and int(list_values[5]) < 70:
         house3.append("Medium")
      if int(list_values[5]) > 70:
         house3.append("Low")

      #Delay of 5 seconds
      time.sleep(5)

      arduino_data = 0
      list_in_floats.clear()
      list_values.clear()
      arduino.close()
      print('Connection closed')
      print('<---------------------------------->')
   #Place information into our dataframe columns:
   my_df['House1'] = house1
   my_df['House2'] = house2
   my_df['House3'] = house3


#Deciding logic for action to be taken (should bin be emptied or not?)
   for i in range(len(my_df)):

      if (my_df['House1'][i] == 'High' or my_df['House2'][i] == 'High' or my_df['House3'][i] == 'High'):
         my_df['Action'][i] = "PICKUP"
         if i == len(my_df)-1:
            break
         else:
            my_df['House1'][i+1], my_df['House2'][i+1], my_df['House3'][i+1] = 'Low', 'Low', 'Low'

      if ((my_df['House1'][i] == 'Medium' and my_df['House2'][i] == 'Medium')
         or (my_df['House2'][i] == 'Medium' and my_df['House3'][i] == 'Medium')
         or (my_df['House1'][i] == 'Medium' and my_df['House3'][i] == 'Medium')):

         my_df['Action'][i] = "PICKUP"

         if i == len(my_df)-1:
            break
         else:
            my_df['House1'][i+1], my_df['House2'][i+1], my_df['House3'][i+1] = 'Low', 'Low', 'Low'

   #save data to a CSV file to be read and displayed by the HTML page
   my_df.to_csv("./our_data.csv", index=False)

if __name__ == '__main__':
   list_values = []
   list_in_floats = []
   print('Program started')
   main_func()
```

Fig. 11. Deciding Algorithm and CSV File Creation

Finally a CSV file of the high/low/medium level for each house, as well as final decision are created, and a snippet of the file is shown in figure 12. We keep the column of Time empty for now to be filled with theoretical values in the following steps. In practical application, we can easily extract the time and date of when the reading was taken to export it as well into the CSV file.

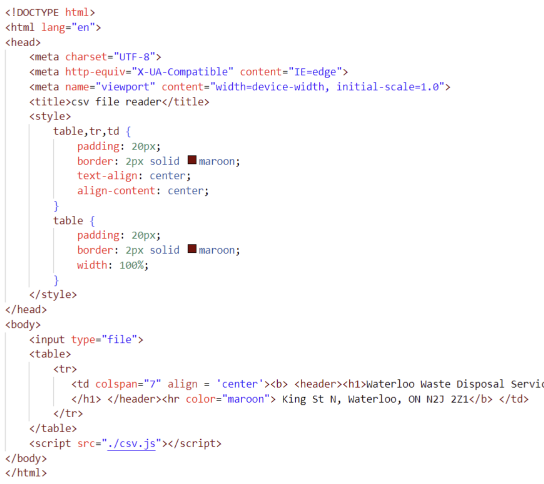


	A	B	C	D	E
1	Time	House1	House2	House3	Action
2		Low	Low	Medium	
3		Low	Medium	Medium	PICKUP
4		Low	Low	Low	
5		Low	Low	Low	
6		Low	Low	Low	
7		Low	Low	Low	
8		High	Low	Low	PICKUP
9		Low	Low	Low	
10		High	Low	Low	PICKUP
11		Low	Low	Low	
12		Low	Low	Low	
13		Medium	Low	Low	
14		Medium	Low	Low	
15		Low	Low	Low	
16		High	Low	Low	PICKUP
17		Low	Low	Low	

Fig. 12. CSV File created by Python Code

5) *HTML Code*

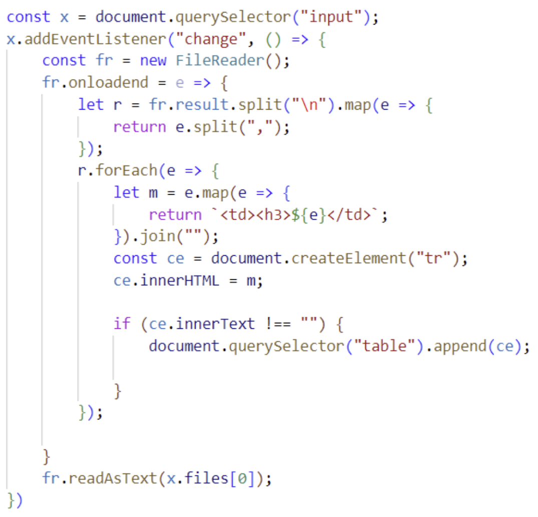
For the Data Visualization that can be access anywhere with Wi-Fi, an HTML code was written to read the CSV file and display the required information clearly to the user, as shown in figure 13.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>csv file reader</title>
    <style>
        table,tr,td {
            padding: 20px;
            border: 2px solid maroon;
            text-align: center;
            align-content: center;
        }
        table {
            padding: 20px;
            border: 2px solid maroon;
            width: 100%;
        }
    </style>
</head>
<body>
    <input type="file">
    <table>
        <tr>
           <td colspan="7" align = 'center'><b> <header><h1>Waterloo Waste Disposal Services
           </h1> </header><hr color="maroon"> King St N, Waterloo, ON N2J 2Z1</b> </td>
        </tr>
    </table>
    <script src="./csv.js"></script>
</body>
</html>
```

Fig. 13. HTML Code for User Interface

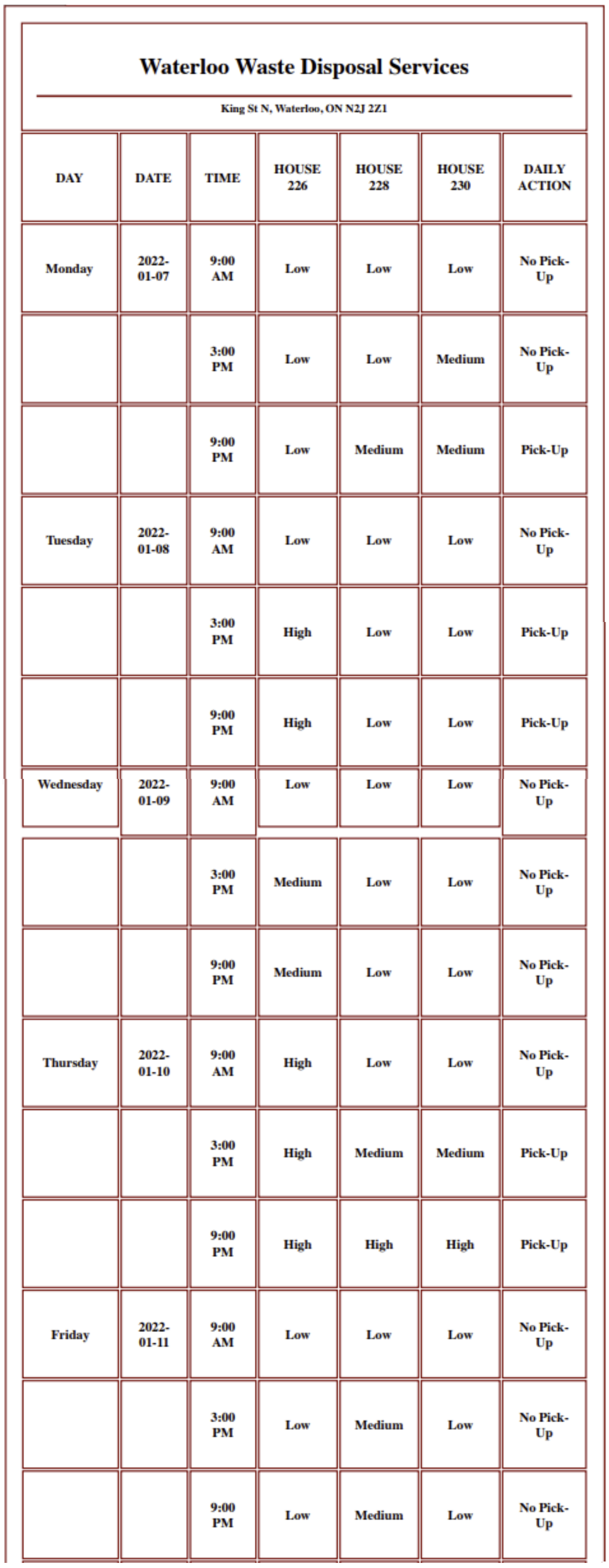
The supporting Javascript file is also shown in figure 14.



```
const x = document.querySelector("input");
x.addEventListener("change", () => {
    const fr = new FileReader();
    fr.onloadend = e => {
        let r = fr.result.split("\n").map(e => {
            return e.split(",");
        });
        r.forEach(e => {
            let m = e.map(e => {
                return `<td><h3>${e}</td>`;
            }).join("");
            const ce = document.createElement("tr");
            ce.innerHTML = m;

            if (ce.innerText !== "") {
                document.querySelector("table").append(ce);

            }
        });

    }
    fr.readAsText(x.files[0]);
})
```

Fig. 14. Javascript Code for User Interface

The resulting user interface is shown in figure 15.



Waterloo Waste Disposal Services

King St N, Waterloo, ON N2J 2Z1

DAY	DATE	TIME	HOUSE 226	HOUSE 228	HOUSE 230	DAILY ACTION
Monday	2022-01-07	9:00 AM	Low	Low	Low	No Pick-Up
		3:00 PM	Low	Low	Medium	No Pick-Up
		9:00 PM	Low	Medium	Medium	Pick-Up
Tuesday	2022-01-08	9:00 AM	Low	Low	Low	No Pick-Up
		3:00 PM	High	Low	Low	Pick-Up
		9:00 PM	High	Low	Low	Pick-Up
Wednesday	2022-01-09	9:00 AM	Low	Low	Low	No Pick-Up
		3:00 PM	Medium	Low	Low	No Pick-Up
		9:00 PM	Medium	Low	Low	No Pick-Up
Thursday	2022-01-10	9:00 AM	High	Low	Low	No Pick-Up
		3:00 PM	High	Medium	Medium	Pick-Up
		9:00 PM	High	High	High	Pick-Up
Friday	2022-01-11	9:00 AM	Low	Low	Low	No Pick-Up
		3:00 PM	Low	Medium	Low	No Pick-Up
		9:00 PM	Low	Medium	Low	No Pick-Up

Fig. 15. User Interface HTML Page

D. *Results Analysis*

Once all the codes are run, a resulting interface is displayed for the waste disposal managers to view. For our prototype, we have simulated 5 days of readings. The user interface shows us the day of the week, the date, and the time the reading was taken. We recorded readings for three times of the day, 9 AM, 3 PM, and 9 PM. These times were chosen when taking into account the practical times the bins should be checked. At all these times, the fullness level of the bins are displayed as low/high/medium, and the final decision is also displayed as to if the garbage truck should make a round in that specific neighborhood at that time using the logic shown in figure 16.

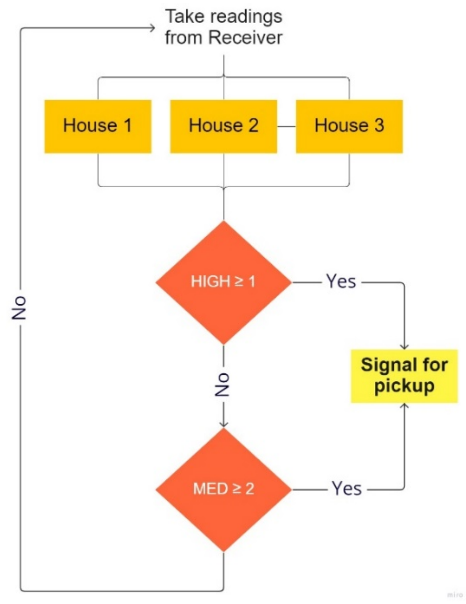


Fig. 16. User Interface HTML Page

The reason why the fullness levels are also mentioned despite already displaying their resulting decision is so that the manager has the ability to make a manual executive decision based on some emergency lack of resources. For example: Imagine a first scenario where a pickup was requested at a time when 2 out of 3 bins were medium, and a second scenario where a all 3 bins were high. Clearly one of the two scenarios is more critical than the other where the operator could manually override the system's decision to send for an immediate pickup.

The results in our interface are showing us all scenarios where at least two bins are medium or at least one high is high that the resulting action is to go ahead and pickup. In all other cases, the resulting action says 'No Pick-Up'.

V. Conclusion

This project is an implementation of the practical use of IoT to simplify redundant and repetitive day-to-day tasks to save on manpower and time. Our design includes a ultrasonic distance sensor mounted on the inside of garbage bins which are placed outside houses, corporations, restaurants, malls and so on. Traditionally, these bins are emptied once a week on a certain day and people are expected to be prepared to accumulate all the trash to be picked up on that day only. This approach raises many issues which were discussed in this paper and requires an alternate solution of a IoT based monitoring system for dynamic pickup scheduling.

In our design, the ultrasonic sensor reads the fullness level of these bins and sends the information via their paired Wi-Fi Module to the cloud database 3 times a day. The database must be divided in terms of the neighborhood where the network of sensors is placed. Once the readings are uploaded, a deciding algorithm establishes whether or not a pickup should be conducted. The original system of routes typically used by a truck can be used here to establish which houses should be grouped as one network of sensors. Thus, each regional waste management can have a convenient interface to monitor all the bins in their area and decide which trucks to deploy.

The major advantage is that users are not restricted to waiting for a certain time and date to have their bins emptied. Not only is the pickup date dynamic, but there is even the option of 3 different times of the day when pickups can happen. This is an easily expandable project which can be implemented in existing bins all across the country with minimal cost, considering the inexpensive nature of the devices used in this prototype.

Upon using this system, countries can expect less air pollution from piled up trash. We can also hope to see that people and businesses do not have to resort to dumping their trash in public areas or in bins not designated to them. Areas would remain to look and smell cleaner without the pileup of unsightly trash

VI. Future work

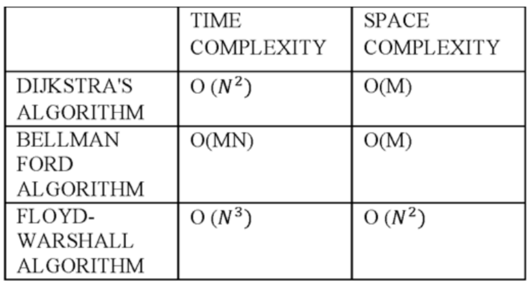
1) *Expansion of scope of current design*

Due to restrictions on this prototype, only four Wi-Fi Modules were used, where one was a receiver and the other three were senders with ultrasonic sensors. To expand on this, we may add more sensors, and make the necessary changes to the codes to link all of them together. Currently, with the equipment we are using, the bins must be within 4-5m from each other due to the range of the ESP8266 Node MCU Module. This can be extended with the help of a portable Wi-Fi Repeater, as in practical usage of a larger scope, houses will be further than that distance from each other. There are also methods to use the ESP8266 itself as a Wi-Fi repeater.

2) *Optimization of Path*

Upon expansion, we suggest adding a GPS module within the sensor unit to have the most accurate tracking of the bins, and place them all on a map, which could be displayed on the user interface for the waste management system to view. A shortest path algorithm can be developed to dynamically establish the best route to take when it was determined a pickup need to be established.

Some examples of applicable routing algorithms to find the shortest path are Dijkstra's Algorithm, Bellman Ford Algorithm and Floyd-Warshall Algorithm, and a comparison of their time and space complexity are show in figure 17 [9]



	TIME COMPLEXITY	SPACE COMPLEXITY
DIJKSTRA'S ALGORITHM	O (N^2)	O(M)
BELLMAN FORD ALGORITHM	O(MN)	O(M)
FLOYD-WARSHALL ALGORITHM	O (N^3)	O (N^2)

Fig. 17. Deciding Flowchart on Pickup or Not

As per the analysis completed by Medehal et al. [9], the Floyd-Warshall was deemed the best fit algorithm for this purpose as it computes the shortest path between all pairs of nodes. And that feature is relevant for this application where we might have multi-stop trips made by the garbage truck, so an optimized route can be found from any location.

3) *Machine Learning and Data Analysis*

The data mined from sensors in a system like this are very useful for a lot of data analysis in the scope of waste produced by houses in a specific region. This data will also help to give an insight of statistics into the production of waste from different companies, businesses and establishments, which can be useful analysis for certain studies.

In further enhancement, predictive algorithms can be developed to determine when pickup should be carried out, rather than only relying on the sensor readings. Patterns can be extracted between different neighborhoods, different times of the year, different days of the week and so on.

4) *Data Fusion*

As most successful and useful IoT networks do, we can incorporate Complementary Sensor Data Fusion by placing up to three ultrasonic sensors in each bin as suggested in the System Requirements section of this paper. Due to the measuring angle of this sensor being only 30 degrees, larger bin dimensions would require more sensors in each bin. The bins could be split into vertical sections with a sensor reading in each section and a cumulative reading could be extracted like in the flowchart shown in figure 18 [7]

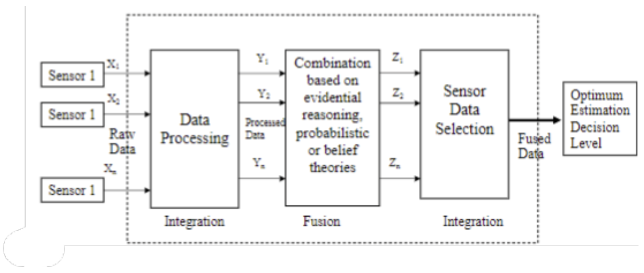


Fig. 18. Data Fusion Source Selection and Processes

5) *Addition of Different Sensors*

To reach the maximum potential of this project, one could install more sensors within the same unit which could be of significant convenience depending on the country or region, not necessarily related to garbage collection. For example, in many places it would be useful to have a fire detector in the same unit because some trash items can pose a fire hazard, including rechargeable batteries, flammable liquids, household chemicals and ashes [8]. In some countries, like Canada, raccoons invading dustbins is a common problem, and installing a motion detector in the unit could alert the homeowner or authorities to take care of the issue.

6) *User Interface*

The user interface can be expanded to a second interface for the bin owners themselves: business owners, school building managers, homeowners etc. They can have their bin information as well as details of the dynamic pickup time and date on a simple mobile application which can be viewed at their convenience. This may seem unnecessary for small families monitoring a bin when they would already be well aware of bin levels themselves. But for business owners and larger establishments, it is very useful information to have on hand without manually checking. This interface can also have additional features as per the addition of more sensors if required i.e. fire detection system, motion sensor etc.

VII. References

[1] "Mafia toxic waste dumping 'causes higher cancer and death rates in Naples'," The Independent, 02-Jan-2016. [Online]. Available: https://www.independent.co.uk/news/world/europe/mafia-toxic-waste-dumping-causes-higher-cancer-and-death-rates-in-naples-a6794236.html. [Accessed: 15-Jul-2022].

[2] S. Faye, F. Melakessou and W. Mtalaa, "Swam: A novel smart waste management approach for businesses using iot", Proceedings of the 1st ACM International Workshop on Technology Enablers and Innovative Applications for Smart Cities and Communities, pp. 38-45, 2019.

[3] "Ultrasonic bin sensors," Sensoneo, 10-Jun-2022. [Online]. Available: https://sensoneo.com/product/ultrasonic-bin-sensors/. [Accessed: 15-Jul-2022].

[4] D. VanReenen, "Edinburgh getting new 'smart bins' that will tell council when they are ready to be emptied," edinburghlive, 13-Jul-2022. [Online]. Available: https://www.edinburghlive.co.uk/news/edinburgh-news/edinburgh-getting-new-smart-bins-24481925. [Accessed: 15-Jul-2022].

[5] A. Medehal, A. Annaluru, S. Bandyopadhyay and T. S. Chandar, "Automated Smart Garbage Monitoring System with Optimal Route Generation for Collection," 2020 IEEE International Smart Cities Conference (ISC2), 2020, pp. 1-7, doi: 10.1109/ISC251055.2020.9239002.

[6] RayB, XENgg, S. Santos, A. Ward, G. PP, F. D. Giust, R. Fernandez-Rey, Josman, R. R., ShaivalD, Itzel, G. Grøtte, Dragoș, Dragos, W. J. Burris, F. T, and Swati, "ESP-now: Receive data from multiple ESP8266 boards (many-to-one)," Random Nerd Tutorials, 12-Jun-2020. [Online]. Available: https://randomnerdtutorials.com/esp-now-many-to-one-esp8266-nodemcu/. [Accessed: 15-Jul-2022].

[7] 659 Lecture Slides: SENSOR DATA FUSION pg 16, O. A. Basir 2022, University of Waterloo

[8] "Help prevent trash fires: Keep these 5 materials out of your garbage," *Granger*. [Online]. Available: https://www.grangerwasteservices.com/help-prevent-trash-fires-keep-5-materials-trash/. [Accessed: 17-Jul-2022].

[9] A. Medehal, A. Annaluru, S. Bandyopadhyay and T. S. Chandar, "Automated Smart Garbage Monitoring System with Optimal Route Generation for Collection," 2020 IEEE International Smart Cities Conference (ISC2), 2020, pp. 1-7, doi: 10.1109/ISC251055.2020.9239002.