Nalaiya Thiran

Batch No: B1-1M3E

SRI SAIRAM ENGINEERING COLLEGE

Department of Electronics and communication

Smart Waste Management System For Metropolitan Cities

Team ID: project ID PNT2022TMID04058

Team Members:

412519106189	Yukkendran D P
412519106071	Kishore kumar R
412519106307	Santhosh kumar P
412519106305	Manojkumar V

Project Guide:

Industry mentor: Mr. Dinesh

Faculty Mentor: Ms. Rajalakshmi B

ABSTRACT

One issue that most cities and municipalities are dealing with currently, is the degradation of environmental cleanliness with reference to waste management. This is a result of improper garbage collection management. Dumping garbage onto the streets and in public areas is a common synopsis found in all developing countries and this mainly ends up affecting the environment and creating several unhygienic conditions. To avoid improper garbage management and to create a hygienic environment, the concept of automation is used in waste management system. Any city being referred to as a "smart city" is because of its orderly and tidy surroundings. But currently, many issues including those related to smart grids, smart environments, and smart living are faced. Today, cities and metropolitan areas' top priority is proper garbage management.

Traditional waste management techniques are too simplistic to create an effective and reliable waste management. The ideology put forward includes hardware and software technologies i.e. connecting Wi-Fi system to the normal dustbin in order to provide free internet facilities to the user for a particular period of time. The technology awards the user for keeping the surrounding clean and thus work hand in hand for the proper waste management in a locality. The smart bin uses multiple technologies - firstly the technology for measuring the amount of trash dumped and secondly the movement of the waste and lastly sending necessary signals and connecting the user to the WiFi system. The proposed system will function on client server model, a cause that will assure clean environment, good health, and pollution free society.

INDEX

CHAPTER 1: INTRODUCTION

1.1 Project Overview

Smart waste management is an innovative approach to handling and collecting waste. Based on IoT (Internet of Things) technology, smart waste management provides data on waste generation patterns and behaviour.

Our Smart waste management solution uses sensors placed in garbage bins to measure fill levels and notifies city collection services when bins are ready to be emptied. There are load and ultrasonic sensors placed to continuously monitor the bins. This data is sent to the cloud (via a microcontroller that is connected to Wi-Fi) where it is stored after which it is processed further. When the levels exceed a certain limit, a notification is sent to the garbage collector via a web application.

Over time, historical data collected by sensors can be used to identify fill patterns, optimize driver routes and schedules, and reduce operational costs. The cost of these sensors is steadily decreasing, making IoT waste bins more feasible to implement and more attractive.

1.2 Purpose

Around 2.1 billion tonnes of municipal solid waste is generated annually around the globe. Population growth and rapid urbanization lead to a huge increase in waste generation, so the traditional methods of waste collection have become inefficient and costly. This system cannot measure the fullness levels of containers, and as a result, half-full containers can be emptied, and in contrast, pre-filled ones need to wait until the next collection period comes. Moreover, since drivers collect empty bins, predefined collection routes of the system cause waste of time, an increase in fuel consumption, and excessive use of resources.

In today’s ever-technological world, an innovative and data-driven approach is the only way forward, the waste sector needs a solution that empowers event-driven waste collection. The most efficient way this extraordinary amount of waste can be solved is through smart waste management without obsolete methods of waste collection. This empowers municipalities, cities, and waste collectors to optimize their waste operations, become more sustainable, and make more intelligent business decisions.

CHAPTER 2: LITERATURE SURVEY

2.1 Existing Problem

Around 80% of waste collections happen at the wrong time. Late waste collections lead to overflowing bins, unsanitary environments, citizen complaints, illegal dumping, and increased cleaning and collection costs. Early waste collections mean unnecessary carbon emissions, more traffic congestion, and higher running costs. The old way of doing waste management is highly inefficient. And in today's ever-technological world, an innovative and data-driven approach is the only way forward. Traditionally, municipalities and waste management companies would operate on a fixed collection route and schedule. This means that waste collection trucks would drive the same collection route and empty every single waste container – even if the waste container did not need emptying. This means high labour and fuel costs – which residents ultimately foot the bill for.

2.2 References

Paper Title	Author	Outcome
IOT based Smart Garbage System	1) T.Sinha 2) R.M. Sahuother	IoT Based Smart Garbage System which indicates directly that the dustbin is filled to a certain level by the garbage and cleaning or emptying them is a matter of immediate concern. This prevents lumping of garbage in the roadside dustbin which ends up giving foul smell and illness to people. The design of the smart dustbin includes a single by ultrasonic sensor which configured with Arduino Uno with this research, it is sending SMS to the Municipal Council that dustbin is to overflow.

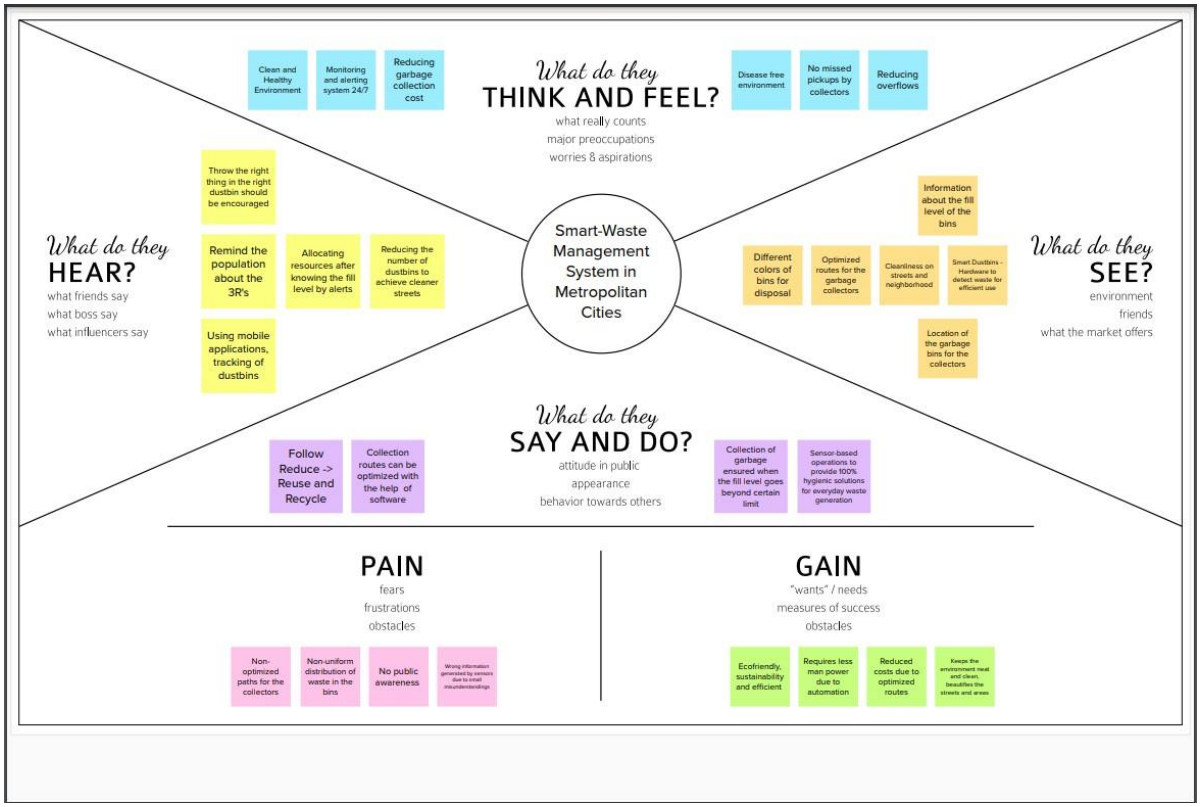
Raspberry pi-based smart waste management system using Internet of Things.	1)Shaik Vaseem Akram 2)Rajesh Singh	Nowadays it is becoming a difficult task to distinguish wet and dry waste. The new waste management system covers several levels of enormous workforce. Every time, laborers must visit the garbage bins in the city area to check whether they are filled or not. The data communicates to the cloud server for real-time monitoring of the system. With the real-time fill level information collected via the monitoring platform, the system reduces garbage overflow by informing about such instances before they arrive
Smart Waste Management System.	1) Sanjiban Charkraborty	This Waste management is one of the serious challenges of the cities, the system now used in cities, we continue to use an old and outmoded paradigm that no longer serves the entail of municipalities, Still find over spilled waste containers giving off irritating smells causing serious health issues and atmosphere impairment.
Smart Solid Waste Management.	1) Mohd Helmy Abd Wahab	At the time of trash disposal, the material to be recycled could be identified using RFID technology.
Analysis of Load cell.	1) Ranjeet Kumar 2) Sandeep Chhabra	Load Cells 4.1 General Load Cell related information A load cell is meant to measure the size of a mass but actually is a force sensor which transforms force into an electrical signal. The load cell needs the earth gravity to work. Every mass is attracted by the earth gravimetric field, that force is named “load”.

Smart Waste Management using Wireless Sensor Network	1) Tarandeep Singh 2) Rita Mahajan 3) Deepak Bagai	In most of the places, garbage bins are not cleaned at periodic intervals, giving a hygienic issue. Thus, a system to manage bins, by using intelligent bins, gateway and remote base station is created. But this system is prone to attacks from hackers and complexity to build it is very high.
Smart Waste Management for Green Environment	1) T. P. Fei	The system is based on Bootstrap platform. This system works on the waterfall methodology which has 4 crucial phases: planning and analysis, system design, system implementation and system testing. Using this system, operators can get the information regarding collection from trash bins. The limitations of this approach are that the resultant product has a short life and uniformity is lost after a certain period.

2.3 Problem Statement Definition

CHAPTER 3: IDEATION & PROPOSED SOLUTION

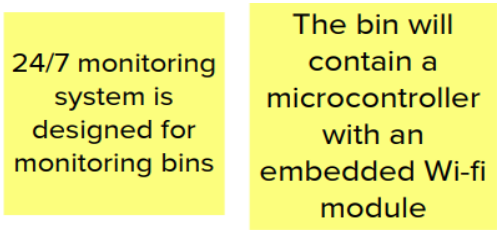
3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

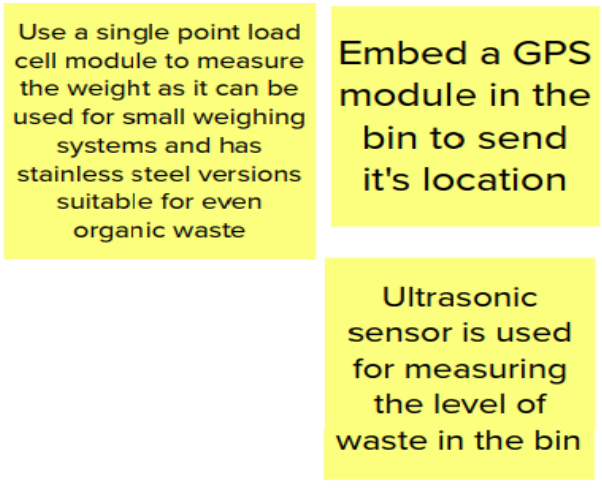
3.2.1 Brainstorm by team members

Yukkendran DP:

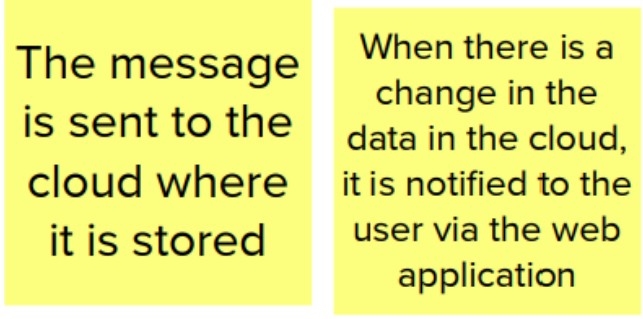


Kishore kumar R:

Santhosh kumar P:

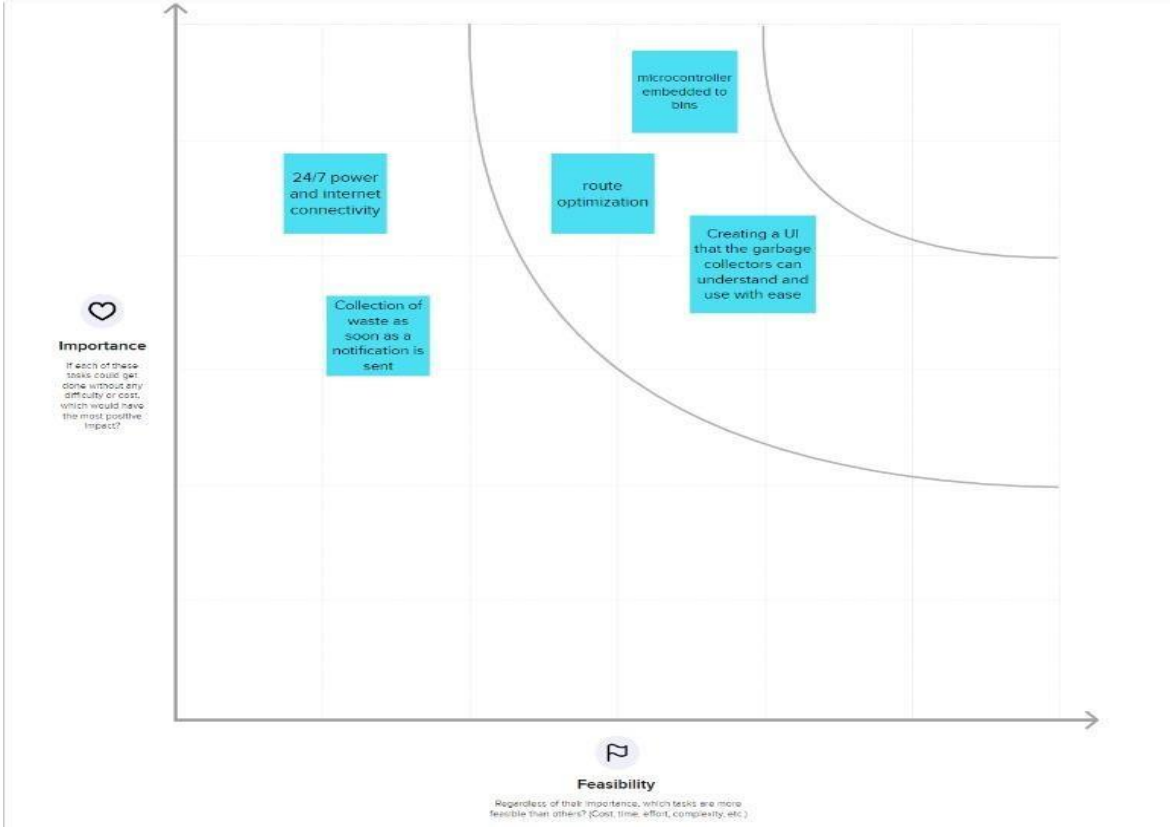


Manojkumar V:



3.2.2 Group ideas

3.2.3 Prioritize

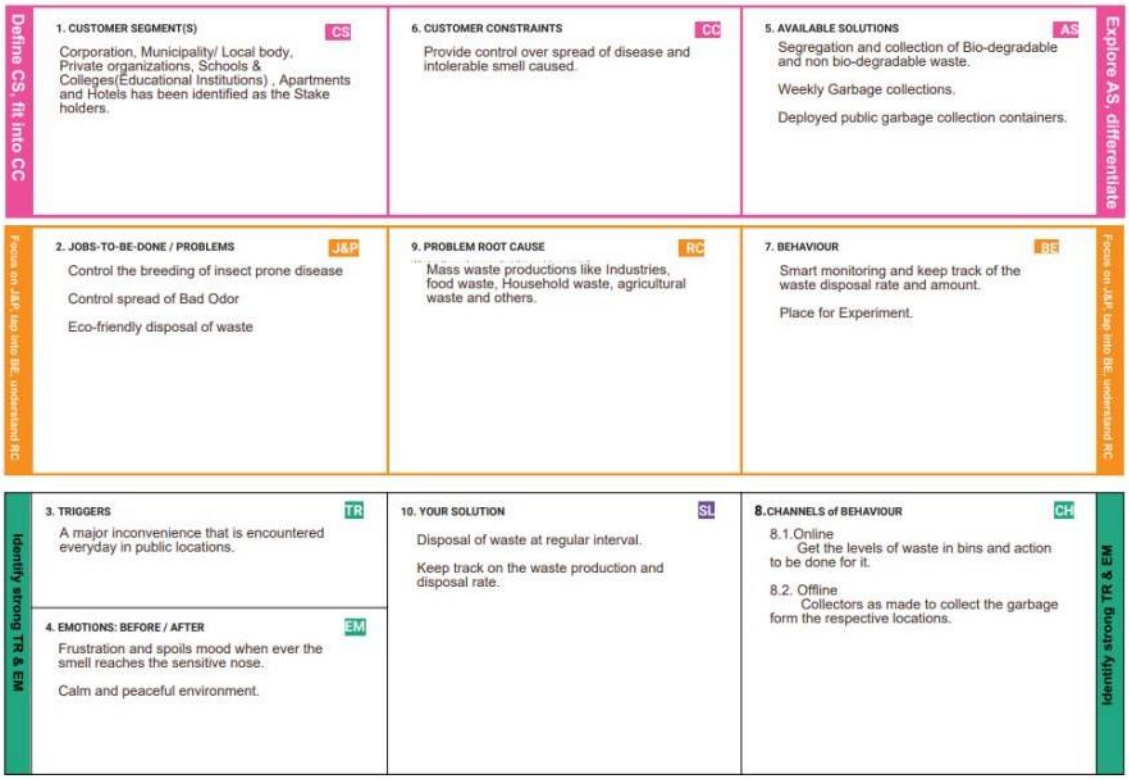


3.3 Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Solid waste disposal is a significant problem in metropolitan areas of most developing nations, and it seriously jeopardizes the residents' ability to live a healthy lifestyle. Both the local authorities and the public will benefit from having access to trustworthy information on the state of solid trash at various sites throughout the city for managing the threat.
2.	Idea / Solution description	A 24/7 monitoring system is designed for the bins consisting of a microcontroller and embedded Wi-Fi module and has sensors for detecting the weight and level of the waste. When the bin is full, information is sent to the cloud which is forwarded to the users via the web application. Once the message is sent, garbage collection is initiated for all the bins whose level has risen more than the threshold value.
3.	Novelty / Uniqueness	Garbage collection is made simple and efficient with the help of this web application, which can be used to monitor the bins throughout the city. The fill level of the bin is also displayed which makes it easier for garbage collectors. Additionally, it alerts the location of the bin to the garbage collectors.
4.	Social Impact / Customer Satisfaction	Large overflowing bins are a potential threat because they not only pollute the air nearby but also serve as a breeding ground for contagious diseases. Also, the waste collection process is more effective for the garbage collector. Normally, they might see a bin that is overflowing and is difficult for them to collect or one that is only partially filled, but with the help of this application, this problem is resolved.

5.	Business Model (Revenue Model)	Without any financial advantages, the primary goal of this solution is to assist locals and government employees. Recycling dry waste and composting wet waste, which could then be sold, are two ways to generate income if it is necessary.
6.	Scalability of the Solution	The waste collection for an entire major metropolitan area should be supported by the platform. The implementation of this will consider various implementation-related factors, including the storage and security of data in the cloud.

3.4 Proposed Solution fit



CHAPTER 4: REQUIRMENT ANALYSIS

4.1 Functional Requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	Detailed bin inventory.	All monitored bins can be seen on the map, such that the route can be optimized for the garbage collectors. Bins or stands are visible on the map as green, orange, or red circles. You can see bin details such as – capacity, last measurement, etc.
2	Real time bin monitoring.	The amount of fill is displayed in %, based on the garbage level and the tool predicts when the bin will become full, which is one the functionalities not included in the best waste management software. Sensors recognize picks as well; so, we can check when the bin was last collected. With real-time data and predictions, you can eliminate the overflowing bins and stop collecting half-empty ones.
3	Plan waste collection routes.	The tool semi-automates waste collection route planning. Based on current bin fill-levels and predictions of reaching full capacity, you are ready to respond and schedule waste collection. You can compare planned vs. executed routes to identify any inconsistencies.
4	Adjust bin distribution.	Ensure the most optimal distribution of bins. Identify areas with either dense or sparse bin distribution. Make sure all trash types are represented within a stand.
5	Eliminate inefficient picks.	Eliminate the collection of half-empty bins. The sensors recognize picks. By using real-time data on fill-levels and pick recognition, we can show you how full the bins you collect are.

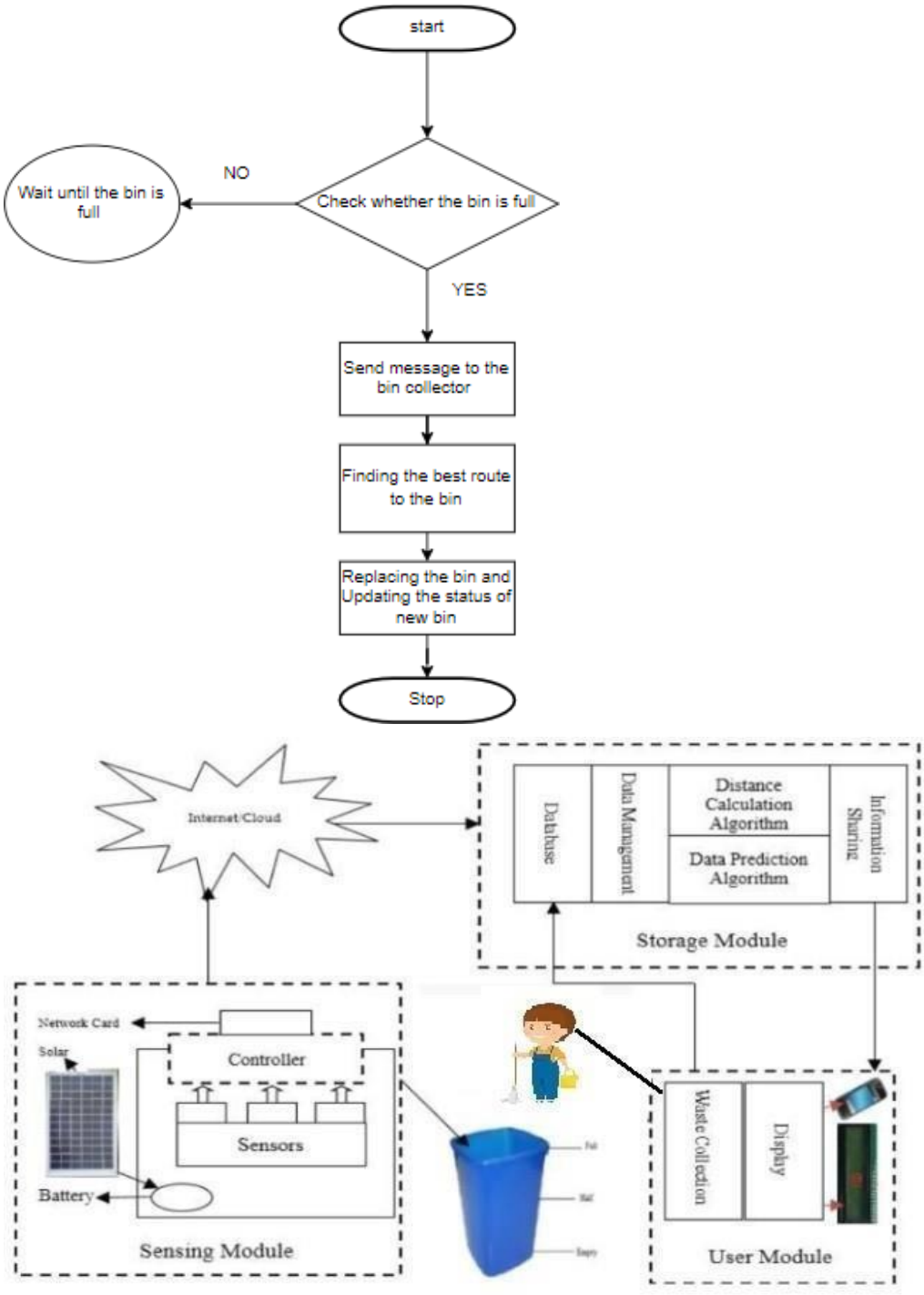
4.2 Non-functional Requirements

NFR No.	Non-Functional Requirement	Description
1	Usability	IoT device verifies that usability is a special and important perspective to analyze user requirements, which can further improve the design quality. In the design process with user experience as the core, the analysis of users' product usability can indeed help designers better understand users' potential needs in waste management, behavior and experience.
2	Security	• Use of reusable bottles • Use of reusable grocery bags • Purchase wisely and recycle • Avoid single use food and drink containers
3	Reliability	Smart waste management is also about creating better working conditions for waste collectors and drivers. Instead of driving the same collection routes and servicing empty bins, waste collectors will spend their time more efficiently, taking care of bins that need servicing.
4	Performance	The Smart Sensors use ultrasound technology to measure the fill levels (along with other data) in bins several times a day. Using a variety of IoT networks (NB-IoT, GPRS), the sensors send the data to IBM Watson, that contains all the devices. Customers are hence provided data-driven decision making, and optimization of waste collection routes, frequencies, and vehicle loads resulting in route reduction by at least 30%

CHAPTER 5: PROJECT DESIGN

5.1 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



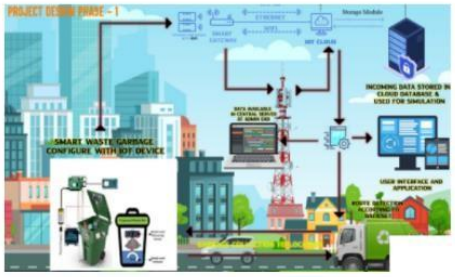
5.2 Solution and Technical Architecture

Design:

To segregate different types of waste and aiming for a clean environment by collectin_ the waste on a daily basis using modern technologies like notifications to notify the garbage collectors when the bin levels are high.

Usability:

easy to segregate
reduces pollution levels
route optimization is done
saves dumping area space



Approach - Functional Elements:

Our main aim is to make the environment pollution free, optimize the routes for garbage collectors by using the sensors and sending out notifications when the height level of the bin has reached a threshold value.

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Customer (Mobile user)	Login	USN-1	Can access his or her account to access site with the given credentials.	Admin can access the account / dashboard
Admin	Admin	USN-2	Monitor the user/customer and other participants in the process of garbage disposal/collection.	Authorized User
Garbage Collector	Login	USN – 3	Updates the status of the garbage bins once visited.	Registered by admin and authorized
Truck Driver	Login	USN-4	As user, they are directed to the work assigned to them in site and take the given route.	Admin can register and route can be dynamically re-routed
Organization Head	Login	USN-5	Has the privilege to monitor over the sectors and customers under their control and division.	Admin verified and authorized user from the organization

Project Planning Phase
Sprint Delivery Plan

Date	21October 2022
Team ID	PNT2022TMID04058
Project Name	SmartWaste Management For Metropolitan Cities
Maximum Marks	8 Marks

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Santhoshkumar P
Sprint-1		USN-2	As a user, I will receive confirmation email oncel have registered for the application	1	High	KishoreKumar R
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	ManojKumar
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Yukken dran D.P
Sprint-1	Login	USN-5	As a user, I can log into the application by Entering email & password	1	High	Santhos hKumar P

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	30	30 OCT 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	49	6 NOV 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	50	7 NOV 2022

Velocity:

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

CHAPTER 7: CODING AND SOLUTIONING

7.1 Feature 1:

The main and first feature of the smart waste management is to get the live location of anyone who access the website for putting out a request for garbage collection in their locality. The live location is obtained as a result of the below code.

Web Application to get the Live location:

index.html:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width">
   <title>Smart Waste Management System</title>
   <link rel="icon" type="image/x-icon" href="/imgs/DUMPSTER.png">
   <link href="style.css" rel="stylesheet" type="text/css" />
   <script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-app.js"></script>
   <script  src="https://www.gstatic.com/firebasejs/9.14.0/firebase-analytics.js"></script>
   <script  src="https://www.gstatic.com/firebasejs/9.14.0/firebase-database.js"></script>

   <script>
      var firebaseConfig =
      {
         apiKey: "AIzaSyCcZk7b1CLOGviwUpthRDLotrmFX0MFuTs",
         authDomain: "swms-3840.firebaseapp.com",
         projectId: "swms-3840",
         storageBucket: "swms-3840.appspot.com",
         messagingSenderId: "479902726304",
         appId: "1:479902726304:web:3d822880d1275ee57a71c5",
         measurementId: "G-MHP4N77MTP"
      };
      firebase.initializeApp(firebaseConfig)
   </script>
   <script defer src="db.js"></script>
</head>

<body style="background-color:#1F1B24;">
   <script src="maps.js"></script>
   <div id="map_container">
```

```
        <h1 id="live_location_heading" >LIVE LOCATION</h1>
        <div id="map"></div>
        <div id="alert_msg">ALERT MESSAGE!</div>
    </div>
    </div>
    <center>
        <a href="https://goo.gl/maps/G9XET5mzSw1ynHQ18" type="button" class="btn
btn-dark">
            DUMPSTER
        </a>
    </center>
    <script
        src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBBLyWj-
3FWtCbCXGW3ysEiI2fDfrv2v0Q&callback=myMap"></script></div>
</body>
</html>
```

db.js:

```
const cap_status = document.getElementById("cap_status");
const alert_msg = document.getElementById("alert_msg");

var ref = firebase.database().ref();

ref.on(
  "value",
  function (snapshot) {
    snapshot.forEach(function (childSnapshot) {
    var value = childSnapshot.val();

      const alert_msg_val = value.alert;
      const cap_status_val = value.distance_status;

      alert_msg.innerHTML = `${alert_msg_val}`;
    });
  },
  function (error) {
    console.log("Error: " + error.code);
  }
);
```

maps.js:

```
const database = firebase.database();

function myMap() {
  var ref1 = firebase.database().ref();

  ref1.on(
    "value",
```

```
    function (snapshot) {
      snapshot.forEach(function (childSnapshot) {
      var value = childSnapshot.val();
        const latitude = value.latitude;
        const longitude = value.longitude;

        var latlong = { lat: latitude, lng: longitude };
        var mapProp = {
          center: new google.maps.LatLng(latlong),
          zoom: 10,
        };
        var map = new google.maps.Map(document.getElementById("map"), mapProp);
        var marker = new google.maps.Marker({ position: latlong });
        marker.setMap(map);
      });
    },
    function (error) {
      console.log("Error: " + error.code);
    }
  );
}
```

7.2 Feature 2:

In this part, the filled level of the bin is measured with the help of IBM IOT Watson platform devices, IBM Cloud interface and Node-RED is used for creating the dashboard nodes that helps us create a UI to display the distance, that is, the fill level of the bin. It also intimates the location of the bin with the fill level and alerts the collection authority if the fill level goes beyond a threshold value.

Code to evaluate the level of the garbage in bin:

bin1.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
organization = "73ffyv"
devicType = "BIN1"
deviceId = "BIN1ID"
authMethod= "token"
authToken= "123456789"
```

```
#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()
#connect and send a datapoint "temp" with value integer value into the cloud as a type of event
for every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}


    if loadcell < 13 and loadcell > 15:
        load = "90 %"
    elif loadcell < 8 and loadcell > 12:
        load = "60 %"
    elif loadcell < 4 and loadcell > 7:
        load = "40 %"
    else:
        load = "0 %"

    if distance < 15:
        dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
        dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
        dist = 'Risk warning:' '40 %'
    else:
        dist = 'Risk warning:' '17 %'

    if load == "90 %" or distance == "90 %":
        warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
        warn = 'alert :' 'garbage is above 60%'
    else :
        warn = 'alert :' 'Levels are low, collection not needed '
```

```
    def myOnPublishCallback(lat=11.035081,long=77.014616):
        print("Peelamedu, Coimbatore")
        print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s "
%long,"lat = %s" %lat)
        print(load)
        print(dist)
        print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
        print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

bin2.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
organization = "73ffyv"
devicType = "BIN2"
deviceId = "BIN2ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
```

```
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event
for every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}


    if loadcell < 13 and loadcell > 15:
        load = "90 %"
    elif loadcell < 8 and loadcell > 12:
            load = "60 %"
    elif loadcell < 4 and loadcell > 7:
            load = "40 %"
    else:
            load = "0 %"


    if distance < 15:
            dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
            dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
            dist = 'Risk warning:' '40 %'
    else:
            dist = 'Risk warning:' '17 %'


    if load == "90 %" or distance == "90 %":
            warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
            warn = 'alert :' 'garbage is above 60%'
    else :
            warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.068774,long=77.092978):
        print("PSG iTech, Coimbatore")
        print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s "
%long,"lat = %s" %lat)
        print(load)
        print(dist)
        print(warn)

    time.sleep(10)
```

```
    success=deviceCli.publishEvent                ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent                 ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
        print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

bin3.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
organization = "73ffyv"
devicType = "BIN3"
deviceId = "BIN3ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event
for every 10 seconds
deviceCli.connect()

while True:
```

```
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}

    if loadcell < 13 and loadcell > 15:
       load = "90 %"
    elif loadcell < 8 and loadcell > 12:
         load = "60 %"
    elif loadcell < 4 and loadcell > 7:
         load = "40 %"
    else:
         load = "0 %"


    if distance < 15:
         dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
         dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
         dist = 'Risk warning:' '40 %'
    else:
         dist = 'Risk warning:' '17 %'


    if load == "90 %" or distance == "90 %":
         warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
         warn = 'alert :' 'garbage is above 60%'
    else :
         warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.007403,long=76.963439):
       print("Kattoor, Coimbatore")
       print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s "
%long,"lat = %s" %lat)
       print(load)
       print(dist)
       print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent                ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent                 ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
       print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback
```

```
#disconnect the device
deviceCli.disconnect()
```

bin4.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
organization = "73ffyv"
devicType = "BIN4"
deviceId = "BIN4ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event
for every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}


    if loadcell < 13 and loadcell > 15:
        load = "90 %"
    elif loadcell < 8 and loadcell > 12:
         load = "60 %"
    elif loadcell < 4 and loadcell > 7:
```

```
        load = "40 %"
    else:
        load = "0 %"


    if distance < 15:
        dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
        dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
        dist = 'Risk warning:' '40 %'
    else:
        dist = 'Risk warning:' '17 %'


    if load == "90 %" or distance == "90 %":
        warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
        warn = 'alert :' 'garbage is above 60%'
    else :
        warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.453306,long=77.426024):
       print("Seethammal Colony, Gobichittipalayam")
       print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s "
%long,"lat = %s" %lat)
       print(load)
       print(dist)
       print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent                ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent                 ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
       print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

7.3 Feature 3:

An additional feature added to the smart waste management system is to measure the weight of the bin using hx711 load cell. The weight of the bin is the output of the below code.

Measuring the weight of the garbage bin:

main.py:

```
from hx711 import HX711
hx = HX711(5,4,64)
print(1)
while True:
    hx.tare()
    read = hx.read()
    value=hx.read_average()
    print(value,"#")
```

hx711.py:

```
from machine import Pin, enable_irq, disable_irq, idle

class HX711:
    def__init__(self, dout, pd_sck, gain=128):
        self.pSCK = Pin(pd_sck , mode=Pin.OUT)
        self.pOUT = Pin(dout, mode=Pin.IN, pull=Pin.PULL_DOWN)
        self.pSCK.value(False)

        self.GAIN = 0
        self.OFFSET = 0
        self.SCALE = 1
        self.time_constant = 0.1
        self.filtered = 0
        self.set_gain(gain);

    def set_gain(self, gain):
        if gain is 128:
            self.GAIN = 1
        elif gain is 64:
            self.GAIN = 3
        elif gain is 32:
            self.GAIN = 2
        self.read()
        self.filtered = self.read()
        print('Gain & initial value set')

    def is_ready(self):
        return self.pOUT() == 0

    def read(self):
        # wait for the device being ready
```

```
    while self.pOUT() == 1:
        idle()

    # shift in data, and gain & channel info
    result = 0
    for j in range(24 + self.GAIN):
        state = disable_irq()
        self.pSCK(True)
        self.pSCK(False)
        enable_irq(state)
        result = (result << 1) | self.pOUT()

    # shift back the extra bits
    result >>= self.GAIN

    # check sign
    if result > 0x7fffff:
        result -= 0x1000000

    return result

def read_average(self, times=3):
    s = 0
    for i in range(times):
        s += self.read()
    ss=(s/times)/210
    return '%.1f' %(ss)

def read_lowpass(self):
    self.filtered += self.time_constant * (self.read() - self.filtered)
    return self.filtered

def get_value(self, times=3):
    return self.read_average(times) - self.OFFSET

def get_units(self, times=3):
    return self.get_value(times) / self.SCALE

def tare(self, times=15):
    s = self.read_average(times)
    self.set_offset(s)

def set_scale(self, scale):
    self.SCALE = scale

def set_offset(self, offset):
    self.OFFSET = offset
```

```
def set_time_constant(self, time_constant = None):
    if time_constant is None:
        return self.time_constant
    elif 0 < time_constant < 1.0:
        self.time_constant = time_constant

def power_down(self):
    self.pSCK.value(False)
    self.pSCK.value(True)

def power_up(self):
    self.pSCK.value(False)
```

CHAPTER 8: TESTING

8.1 Test cases:

Unit testing

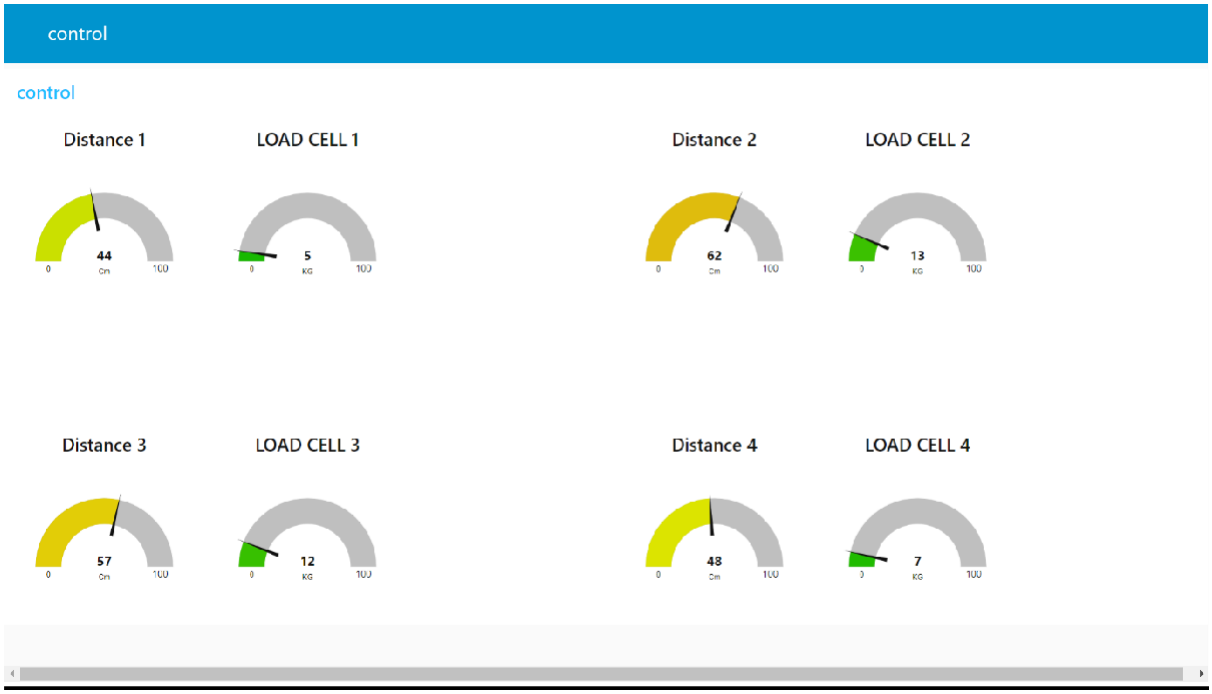
Test case no.	Sensor/Stage	Input	Expected output	Obtained output	Status
1.	Ultrasonic	Garbage level in bin i)Null ii)Full iii)Range in %	Correct level or distance	As expected	Pass
2.	ESP – 32	Microcontroller to process the input data	To collect the data from sensor	As expected	Pass
3.	Load cell	To measure mechanical force	Calculate the force due to the bin weight	As expected	Pass
4.	Gauge	To display the tares	Display the level for collection	As expected	Pass
5.	HX710	Weight of the bin (in kg)	Measure the weight	As expected	Pass

8.2 User Acceptance testing

Acceptance testing - is the final phase of product testing prior to public launch. A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

CHAPTER 9: RESULTS

Sample output:



Before implementation of Smart waste management system

		Ground truth	
		Emptying	Nonemptying
Predicted	Emptying	911	68
	Nonemptying	990	6041

After implementation of smart waste management system

		Ground truth	
		Emptying	Nonemptying
Predicted	Emptying	1720	90
	Nonemptying	181	6091

CHAPTER 10: ADVANTAGES AND DISADVANTAGES

10.1 Advantages:

- Intelligent compaction of waste by monitoring fill level in real-time using sensors.
- It keeps our surroundings clean and keeps free from bad odour.
- Reduces manpower requirement to handle the garbage collection
- Emphasizes of healthy environment and keep the cities cleaner and more beautiful.
- It reduces infrastructure, operating and maintenance costs by upto 30%.
- Increases recycling rate of waste.

10.2 Disadvantages:

- Initial large-scale implementation takes cost.
- System requires more number waste bins for separate waste collection.
- Wireless technologies used should have proper connections as they have shorter range and lower data speed
- Training programs should be provided to people involving in the ecosystem of smart waste management.
- Sensors may encounter damage so it should be kept under protective ambience to prevent the damage.
- Replacement of sensors require knowledgeable people and thus acknowledgement of malfunction of sensor.

CHAPTER 11: CONCLUSION

Improper disposal and improper maintenance of domestic waste create issues in public health and environment pollution thus this paper attempts to provide practical solution towards managing the waste collaborating it with the use of IOT. by using the smart waste management system, we can manage waste properly we are also able to sort the Bio-degradable and non-Biodegradable waste properly which reduces the pollution in the environment. Various waste management initiatives taken for human well-being and to improve the TWM practices were broadly discussed in this chapter. The parameters that influence the technology and economic aspects of waste management were also discussed clearly. Different types of barriers in TWM, such as economic hitches, political issues, legislative disputes, informative and managerial as well as solutions and success factors for implementing an effective management of toxic organic waste within a globular context, were also discussed giving some real examples. The effect of urbanization on the environmental degradation and economic growth was also discussed. The proposed system will help to overcome all the serious issues related to waste and keep the environment clean.

CHAPTER 12: FUTURE WORK

Based on the real-time and historical data collected and stored in the cloud waste collection schedules and routes can be optimized. Predictive analytics could be used to make decisions ahead of time and offers insight into waste bin locations. Graph theory optimization algorithms can be used to manage waste collection strategies dynamically and efficiently. Every day, the workers can receive the newly calculated routes in their navigation devices. The system can be designed to learn from experience and to make decisions not only on the daily waste level status but also on future state forecast, traffic congestion, balanced cost-efficiency functions, and other affecting factors that a priori humans cannot foresee.

Garbage collectors could access the application on their mobile phone/tablets using the internet. Real-time GPS assistance can be used to direct them to the pre-decided route. As they go collecting the garbage from the containers, the management is also aware of the progress as the vehicle, as well as the garbage containers, are traced in real-time. The management staff gets their own personalized administration panel over a computer/tablet which gives them a bird eye view over the entire operations.

An alternative solution using image processing and camera as a passive sensor could be used. But, the cost of those image processing cameras is higher as compared to the ultrasonic sensors, which leads to high solution implementation cost.

CHAPTER 13: APPENDIX

13.1 Source Code:

Web Application to get the Live location:

index.html:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width">
   <title>Smart Waste Management System</title>
   <link rel="icon" type="image/x-icon" href="/imgs/DUMPSTER.png">
   <link href="style.css" rel="stylesheet" type="text/css" />
   <script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-app.js"></script>
   <script  src="https://www.gstatic.com/firebasejs/9.14.0/firebase-analytics.js"></script>
   <script  src="https://www.gstatic.com/firebasejs/9.14.0/firebase-database.js"></script>

   <script>
      var firebaseConfig =
      {
         apiKey: "AIzaSyCcZk7b1CLOGviwUpthRDLotrmFX0MFuTs",
         authDomain: "swms-3840.firebaseapp.com",
         projectId: "swms-3840",
         storageBucket: "swms-3840.appspot.com",
         messagingSenderId: "479902726304",
         appId: "1:479902726304:web:3d822880d1275ee57a71c5",
         measurementId: "G-MHP4N77MTP"
      };
      firebase.initializeApp(firebaseConfig)
   </script>
   <script defer src="db.js"></script>
</head>

<body style="background-color:#1F1B24;">
   <script src="maps.js"></script>
   <div id="map_container">
      <h1 id="live_location_heading" >LIVE LOCATION</h1>
      <div id="map"></div>
      <div id="alert_msg">ALERT MESSAGE!</div>
   </div>
   </div>
   <center>
```

```
        <a href="https://goo.gl/maps/G9XET5mzSw1ynHQ18" type="button" class="btn btn-
    dark">
            DUMPSTER
        </a>
    </center>
    <script
        src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBBLyWj-
    3FWtCbCXGW3ysEiI2fDfrv2v0Q&callback=myMap"></script></div>
    </body>
    </html>
```

db.js:

```
    const cap_status = document.getElementById("cap_status");
    const alert_msg = document.getElementById("alert_msg");

    var ref = firebase.database().ref();

    ref.on(
      "value",
      function (snapshot) {
        snapshot.forEach(function (childSnapshot) {
        var value = childSnapshot.val();

          const alert_msg_val = value.alert;
          const cap_status_val = value.distance_status;

          alert_msg.innerHTML = `${alert_msg_val}`;
        });
      },
      function (error) {
        console.log("Error: " + error.code);
      }
    );
```

maps.js:

```
    const database = firebase.database();

    function myMap() {
      var ref1 = firebase.database().ref();

      ref1.on(
        "value",
        function (snapshot) {
          snapshot.forEach(function (childSnapshot) {
          var value = childSnapshot.val();
            const latitude = value.latitude;
            const longitude = value.longitude;
```

```
        var latlong = { lat: latitude, lng: longitude };
        var mapProp = {
          center: new google.maps.LatLng(latlong),
          zoom: 10,
        };
        var map = new google.maps.Map(document.getElementById("map"), mapProp);
        var marker = new google.maps.Marker({ position: latlong });
        marker.setMap(map);
       });
     },
     function (error) {
      console.log("Error: " + error.code);
     }
   );
  }
```

style.css:

```
  html,
  body {
    height: 100%;
    margin: 0px;
    padding: 0px;
  }
  #container {
    display: flex;
    flex-direction: row;
    height: 100%;
    width: 100%;
    position: relative;
  }
  #logo_container {
    height: 100%;
    width: 12%;
    background-color: #c5c6d0;
    display: flex;
    flex-direction: column;
    vertical-align: text-bottom;
  }
  .logo  {
    width: 70%;
    margin: 5% 15%;

    /*  border-radius: 50%; */
  }
  #logo_3 {
    vertical-align: text-bottom;
  }
  #data_container {
```

```
  height: 100%;
  width: 20%;
  margin-left: 1%;
  margin-right: 1%;
  display: flex;
  flex-direction: column;
}
#data_status {
  height: 60%;
  width: 8%;
  margin: 7%;
  background-color: #691f6e;
  display: flex;
  flex-direction: column;
  border-radius: 20px;
}
#load_status {
  background-image: url("/imgs/KG.png");
  background-repeat: no-repeat;
  background-size: 170px;
  background-position: left center;
}
#cap_status {
  background-image: url("/imgs/dust.png");
  background-repeat: no-repeat;
  background-size: 150px;
  background-position: left center;
}
.status {
  width: 80%;
  height: 40%;
  margin: 5% 10%;
  background-color: #185adc;
  border-radius: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  color: white;
  font-size: 60px;
}
.datas {
  width: 86%;
  margin: 2.5% 7%;
  height: 10%;
  background: url(water.png);
  background-repeat: repeat-x;
  animation: datas 10s linear infinite;
```

```
  box-shadow: 0 0 0 6px #98d7eb, 0 20px 35px rgba(0, 0, 0, 1);
}
#map_container {
  height: 100%;
  width: 100%;
  display: flex;
  flex-direction: column;
}
#live_location_heading {
  margin-top: 10%;
  text-align: center;
  color: GREY;
}
#map {
  height: 70%;
  width: 90%;
  margin-left: 4%;
  margin-right: 4%;
  border: 10px solid white;
  border-radius: 25px;
}
#alert_msg {
  width: 92%;
  height: 20%;
  margin: 4%;
  background-color: grey;
  border-radius: 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  color: #41af7f;
  font-size: 25px;
  font-weight: bold;
}
.lat {
  margin: 0px;
  font-size: 0px;
}

@keyframes datas {
  0% {
    background-position: -500px 100px;
  }
  40% {
    background-position: 1000px -10px;
  }

  80% {
```

```
    background-position: 2000px 40px;
   }
   100% {
    background-position: 2700px 95px;
   }
  }
```

Code to evaluate the level of the garbage in bin and intimate the collection authority with the location of the bin:

bin1.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
organization = "73ffyv"
devicType = "BIN1"
deviceId = "BIN1ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()
#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for
every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}
```

```
    if loadcell < 13 and loadcell > 15:
       load = "90 %"
    elif loadcell < 8 and loadcell > 12:
         load = "60 %"
    elif loadcell < 4 and loadcell > 7:
         load = "40 %"
    else:
         load = "0 %"

    if distance < 15:
         dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
         dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
         dist = 'Risk warning:' '40 %'
    else:
         dist = 'Risk warning:' '17 %'

    if load == "90 %" or distance == "90 %":
         warn = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
         warn = 'alert :' 'garbage is above 60%'
    else :
         warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.035081,long=77.014616):
       print("Peelamedu, Coimbatore")
       print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat
= %s" %lat)
       print(load)
       print(dist)
       print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent                    ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent                     ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
       print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

bin2.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
organization = "73ffyv"
devicType = "BIN2"
deviceId = "BIN2ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for
every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}


    if loadcell < 13 and loadcell > 15:
        load = "90 %"
    elif loadcell < 8 and loadcell > 12:
          load = "60 %"
    elif loadcell < 4 and loadcell > 7:
          load = "40 %"
    else:
          load = "0 %"
```

```
    if distance < 15:
        dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
        dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
        dist = 'Risk warning:' '40 %'
    else:
        dist = 'Risk warning:' '17 %'


    if load == "90 %" or distance == "90 %":
        warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
        warn = 'alert :' 'garbage is above 60%'
    else :
        warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.068774,long=77.092978):
       print("PSG iTech, Coimbatore")
       print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat
= %s" %lat)
       print(load)
       print(dist)
       print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent                    ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent                     ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
       print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

bin3.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
```

```
organization = "73ffyv"
devicType = "BIN3"
deviceId = "BIN3ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for
every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}


    if loadcell < 13 and loadcell > 15:
       load = "90 %"
    elif loadcell < 8 and loadcell > 12:
        load = "60 %"
    elif loadcell < 4 and loadcell > 7:
        load = "40 %"
    else:
        load = "0 %"


    if distance < 15:
        dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
        dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
        dist = 'Risk warning:' '40 %'
    else:
        dist = 'Risk warning:' '17 %'
```

```
    if load == "90 %" or distance == "90 %":
        warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
        warn = 'alert :' 'garbage is above 60%'
    else :
        warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.007403,long=76.963439):
        print("Kattoor, Coimbatore")
        print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat
= %s" %lat)
        print(load)
        print(dist)
        print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent                    ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent                     ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
        print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

bin4.py:

```
import requests
import json
import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details
organization = "73ffyv"
devicType = "BIN4"
deviceId = "BIN4ID"
authMethod= "token"
authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)
def myCommandCallback(cmd):
    global a
```

```
    print("command recieved is:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-
method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Exception while connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for
every 10 seconds
deviceCli.connect()

while True:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}


    if loadcell < 13 and loadcell > 15:
        load = "90 %"
    elif loadcell < 8 and loadcell > 12:
        load = "60 %"
    elif loadcell < 4 and loadcell > 7:
        load = "40 %"
    else:
        load = "0 %"


    if distance < 15:
        dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'
    elif distance < 40 and distance >16:
        dist = 'Risk warning:' 'garbage is above 60%'
    elif distance < 60 and distance > 41:
        dist = 'Risk warning:' '40 %'
    else:
        dist = 'Risk warning:' '17 %'


    if load == "90 %" or distance == "90 %":
        warn  = 'alert :' ' Garbage level is high, collection time :)'
    elif load == "60 %" or distance == "60 %":
        warn = 'alert :' 'garbage is above 60%'
    else :
        warn = 'alert :' 'Levels are low, collection not needed '

    def myOnPublishCallback(lat=11.453306,long=77.426024):
```

```
        print("Seethammal Colony, Gobichittipalayam")
        print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat
= %s" %lat)
        print(load)
        print(dist)
        print(warn)

    time.sleep(10)
    success=deviceCli.publishEvent                    ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)
    success=deviceCli.publishEvent                     ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

    if not success:
        print("not connected to ibmiot")

    time.sleep(30)
    deviceCli.commandCallback=myCommandCallback

#disconnect the device
deviceCli.disconnect()
```

Measuring the weight of the garbage bin:

main.py:

```
from hx711 import HX711
hx = HX711(5,4,64)
print(1)
while True:
    hx.tare()
    read = hx.read()
    #average=hx.read_average()
    value=hx.read_average()
    print(value,"#")
```

hx711.py:

```
from machine import Pin, enable_irq, disable_irq, idle

class HX711:
    def__init__(self, dout, pd_sck, gain=128):

        self.pSCK = Pin(pd_sck , mode=Pin.OUT)
        self.pOUT = Pin(dout, mode=Pin.IN, pull=Pin.PULL_DOWN)
        self.pSCK.value(False)

        self.GAIN = 0
        self.OFFSET = 0
        self.SCALE = 1
```

```
    self.time_constant = 0.1
    self.filtered = 0

    self.set_gain(gain);

def set_gain(self, gain):
    if gain is 128:
        self.GAIN = 1
    elif gain is 64:
        self.GAIN = 3
    elif gain is 32:
        self.GAIN = 2

    self.read()
    self.filtered = self.read()
    print('Gain & initial value set')

def is_ready(self):
    return self.pOUT() == 0

def read(self):
    # wait for the device being ready
    while self.pOUT() == 1:
        idle()

    # shift in data, and gain & channel info
    result = 0
    for j in range(24 + self.GAIN):
        state = disable_irq()
        self.pSCK(True)
        self.pSCK(False)
        enable_irq(state)
        result = (result << 1) | self.pOUT()

    # shift back the extra bits
    result >>= self.GAIN

    # check sign
    if result > 0x7fffff:
        result -= 0x1000000

    return result

def read_average(self, times=3):
    s = 0
    for i in range(times):
        s += self.read()
    ss=(s/times)/210
```

```
        return '%.1f' %(ss)

    def read_lowpass(self):
        self.filtered += self.time_constant * (self.read() - self.filtered)
        return self.filtered

    def get_value(self, times=3):
        return self.read_average(times) - self.OFFSET

    def get_units(self, times=3):
        return self.get_value(times) / self.SCALE

    def tare(self, times=15):
        s = self.read_average(times)
        self.set_offset(s)

    def set_scale(self, scale):
        self.SCALE = scale

    def set_offset(self, offset):
        self.OFFSET = offset

    def set_time_constant(self, time_constant = None):
        if time_constant is None:
            return self.time_constant
        elif 0 < time_constant < 1.0:
            self.time_constant = time_constant

    def power_down(self):
        self.pSCK.value(False)
        self.pSCK.value(True)

    def power_up(self):
        self.pSCK.value(False)
```

13.2 Project Links:

https://github.com/IBM-EPBL/IBM-Project-39260-1660403380