



**NAALAIYA THIRAN PROJECT - 2022  
19ECI01-PROFESSIONAL READINESS FOR  
INNOVATION, EMPLOYABILITY AND  
ENTREPRENEURSHIP**



**IT - ITes SSC  
NASSCOM**



**SMART WASTE MANAGEMENT SYSTEM FOR  
METROPOLITAN CITIES**

**A PROJECT REPORT**

*Submitted by*

<b>AISHWARYA B</b>	<b>1904065</b>
<b>ANUJA V</b>	<b>1904067</b>
<b>BALA DHARANI S</b>	<b>1904071</b>
<b>DHARANI K</b>	<b>1904073</b>

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**

**COIMBATORE INSTITUTE OF TECHNOLOGY, COIMBATORE – 641 014**

**(Government Aided Autonomous Institution affiliated to Anna University)**

**ANNA UNIVERSITY: CHENNAI 600025**

**NOVEMBER 2022**

# **COIMBATORE INSTITUTE OF TECHNOLOGY**

**(Government aided Autonomous Institution Affiliated to Anna University)**

**COIMBATORE – 641014**

**ANNA UNIVERSITY: CHENNAI 600 025**

## **BONAFIDE CERTIFICATE**

Certified that this report “**SMART WASTE MANAGEMENT SYSTEM FOR METROPOLITAN CITIES**” is the Bonafide work of **AISHWARYA B (1904065), ANUJA V (1904067), BALA DHARANI S (1904071), DHARANI K (1904073)** who carried out **19ECI01 Professional Readiness for Innovation, Employability and Entrepreneurship** project offered by IBM and Anna University, Chennai.

---

**SIGNATURE**

**Dr.T.LOGANAYAKI**

**FACULTY MENTOR**

Professor

Department of Electronics

and Communication

Engineering

---

**SIGNATURE**

**Dr.S.Uma, M.E. Ph.D.,**

**FACULTY EVALUATOR**

Associate Professor

Department of Electronics

and Communication

Engineering

---

**SIGNATURE**

**Dr. A. RAJESWARI, M.E. Ph.D.,**

**PRINCIPAL AND HEAD**

Professor

Department of Electronics

and Communication

Engineering

## **ABSTRACT**

Indiscriminate disposal of solid waste is a major issue in urban centers of most developing countries and it poses a serious threat to healthy living of the citizens. Access to reliable data on the state of solid waste at different locations within the city will help both the local authorities and the citizens to effectively manage the menace. In this paper, an intelligent solid waste monitoring system is developed using Internet of Things (IoT) and cloud computing technologies. The fill level of solid waste in each of the containers, which are strategically situated across the communities, is detected using ultrasonic sensors. Depending on the fill level, the system sends appropriate notification message (in form of tweet) to alert relevant authorities and concerned citizen(s) for necessary action. The system performance shows that the proposed solution may be found useful for efficient waste management in smart and connected communities.

## PROJECT CALENDAR

Phase	Phase Description	Week	Dates	Activity Details
1	Preparation Phase (Pre- requisites, Registrations, Environment Set-up, etc.)	2	22 - 27 Aug 2022	Creation GitHub account & collaborate with Project repository in project workspace
2	Ideation Phase (Literature Survey, Empathize, Defining Problem Statement, Ideation)	2	29 Aug – 3rd Sept 2022	Literature survey (Aim, objective, problem statement and need for the project)
		3	5 - 10th Sept 2022	Preparing Empathy Map Canvas to capture the user Pains & Gains
		4	12 - 17 Sept 2022	Listing of the ideas using brainstorming session
3	Project Design Phase -I (Proposed Solution, Problem- Solution Fit, Solution Architecture)	5	19 - 24 Sept 2022	Preparing the proposed solution document
		6	26 Sept - 01 Oct 2022	Preparing problem - solution fit document & Solution Architecture
4	Project Design Phase -II (Requirement Analysis, Customer Journey, Data Flow Diagrams, Technology Architecture)	7	3 - 8 Oct 2022	Preparing the customer journey maps
		8	10 - 15 Oct 2022	Preparing the Functional Requirement Document & Data- Flow Diagrams and Technology Architecture
5	Project Planning Phase (Milestones & Tasks, Sprint Schedules )	9	17 - 22 Oct 2022	Preparing Milestone & Activity List, Sprint Delivery Plan
6	Project Development Phase (Coding & Solutioning, acceptance Testing, Performance Testing)	10	24 - 29 Oct 2022	Preparing Project Development - Delivery of Sprint-1
		11	31 Oct - 5 Nov 2022	Preparing Project Development - Delivery of Sprint-2
		12	7 - 12 Nov 2022	Preparing Project Development - Delivery of Sprint-3
		13	14 - 19 Nov 2022	Preparing Project Development - Delivery of Sprint-4

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>iii</b>
	<b>LIST OF TABLE</b>	<b>xvi</b>
	<b>LIST OF FIGURES</b>	<b>xviii</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 PROJECT OVERVIEW	1
	1.2 PURPOSE	1
<b>2.</b>	<b>LITERATURE REVIEW</b>	<b>2</b>
	2.1 EXISTING PROBLEM	2
	2.2 PROBLEM STATEMENT DEFINITION	2
	2.3 SUMMARY	2
<b>3.</b>	<b>IDEATION &amp; PROPOSED SOLUTION</b>	<b>4</b>
	3.1 EMPATHY MAP CANVAS	4
	3.2 IDEATION & BRAINSTORMING	5
	3.3 PROPOSED SOLUTION	6
	3.4 PROBLEM SOLUTION FIT	7
<b>4.</b>	<b>REQUIREMENT ANALYSIS</b>	<b>8</b>
	4.1 FUNCTIONAL REQUIREMENT	8
	4.2 NON-FUNCTIONAL REQUIREMENT	9
<b>5.</b>	<b>PROJECT DESIGN</b>	<b>11</b>
	5.1 DATA FLOW DIAGRAMS	11
	5.2 SOLUTION & TECHNICAL ARCHITETURE	14
	5.3 USER STORIES	15
<b>6.</b>	<b>PROJECT PLANNING &amp; TESTING</b>	<b>17</b>

	6.1 SPRINT PLANNING & ESTIMATION	17
	6.2 SPRINT DELIVERY SCHEDULE	
<b>7.</b>	<b>ADVANTAGES AND DISADVANTAGES</b>	<b>35</b>
	7.1 ADVANTAGES	35
	7.2 DISADVANTAGES	35
<b>8.</b>	<b>CONCLUSION</b>	<b>36</b>
<b>9.</b>	<b>FUTURE SCOPE</b>	<b>37</b>

## **APPENDIX**

### **SOURCE CODE**

### **GITHUB AND PROJECT DEMO LINK**

## LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	EMAPATHY MAP	4
3.2	BRAINSTROMING	5
5.1	DATA FLOW DIAGRAM	13
5.2	SOLUTION & TECHNICAL ARCHITECTURE	14
6.1	BURNDOWN CHART	20
6.2	WOKWI SIMULATION	29
6.3	WATSON PLATFORM	30
6.4	NODE RED EXECUTION	30
6.5	WOKWI SIMULATION FOR VARIOUS CONDITION	31
6.6	WATSON RESULT FOR VARIOUS CONDITIONS	31
6.7	DATABASE OUTPUT	32
6.8	METADATA ANALYSIS	32
6.9	CLOUD RESULT	33
6.10	NODE RED OUTPUT	33
6.11	STATUS VIEWING IN NODE RED	34
6.12	BIN STATUS IN NODE RED	34

## **LIST OF TABLES**

<b>TABLE NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
3.3	PROPOSED SOLUTION	6
4.1	FUNCTIONAL REQUIREMENT	9
4.2	NON- FUNCTIONAL REQUIREMENT	10
5.1	USER STORIES	16
6.1	SPRINT SCHEDULE	19
6.2	BURNDOWN TABLE	20



# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PROJECT OVERVIEW**

At present solid waste management is a major concern in the metropolitan cities of the developing and developed countries. As the population is growing, the garbage is also increasing. This huge unmanaged accumulation of garbage is polluting the environment, spoiling the beauty of the area and also leading to the health hazard. In this era of Internet, IOT (Internet of Things) can be used effectively to manage this solid waste. In this paper, we have discussed the definition of Internet of Things and its elements, testing and prototyping tool cooja simulator and finally the study of various literatures available on smart waste management system using IOT.

### **1.2 PURPOSE**

Today big cities around the world are facing a common problem, managing the city waste effectively without making city unclean. Today's waste management systems involve a large number of employees being appointed to attend a certain number of dumpsters this is done every day periodically. This leads to a very inefficient and unclean system in which some dumpsters will be overflowing some dumpsters might not be even half full. This is caused by variation in population density in the city or some other random factor this makes it impossible to determine which part needs immediate attention. Here a waste management system is introduced in which each dumpster is embedded in a monitoring system that will notify the corresponding personal if the dumpster is full. In this system, it is also possible to separate wet and dry waste into two separate containers. This system provides an effective solution to the waste management problem.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 EXISTING SOLUTION**

Truck based waste management system, that transforms any garbage truck into a smart truck by measuring weight and capacity for each lifted can, while monitoring pickup time and location. Measure bulk waste and monitor the collection by mounting GreenQ's system on your grapple truck. The smart grapple solution helps you gather data and turn into useful insights for your collection process. GreenQ's system is installed on the compactor, and can tell you when are your stationary compactors are full and when is the right time to pick them up. Stop collecting half empty containers. Fill level sensors designed for underground containers. Sensors are activated via Narrow Band LTE, which allows for low energy consumption, minimal maintenance, and increased process efficiency.

#### **2.2 PROBLEM STATEMENT DEFINITION**

The current process of waste management starts with the waste being created by people in the cities and disposed in trash bins near its creation point. The disposed trash is collected by municipality or private company trucks at the predefined times and transferred to temporary collection centers. The trash at the collection centers is then sent for recycling. This process in current city setting solves the waste problem partially while it creates other problems such as;

- Some trash bins are overfilled while others are underfilled by the trash collection time,
- Overfilled trash bins create unhygienic conditions,
- Unoptimized truck routes result in excessive fuel usage and environmental pollution and
- All collected trash is combined which complicates sorting at the recycling facility. Some of these problems can be mitigated by implementing smart waste management systems.

#### **2.3 SUMMARY**

On the other hand, smart solid waste management system (SSWMS) is a smart system that links smart waste bins (as smart objects) to web-based and/or mobile-based application through cloud servers using Internet-of-Things (IoT) technologies. IoT allows traditional, physical objects to communicate among each other by transforming them into “smart objects” using

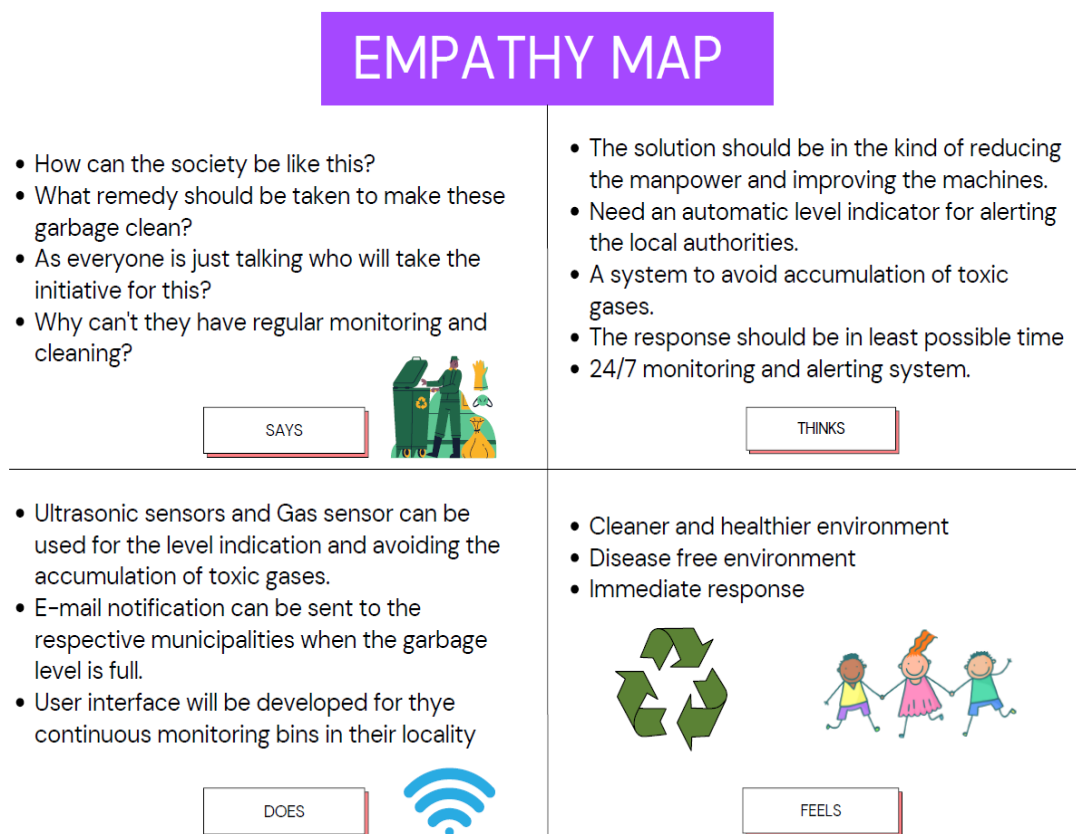
several essential technologies such as embedded devices, sensor networks, and Internet protocols. The overall concept of IoT is depicted in Figure 1 which shows an example of domains suitable for IoT services

## CHAPTER 3

### IDEATION & PROPOSED SOLUTION

#### 3.1 EMPATHY MAP CANVAS

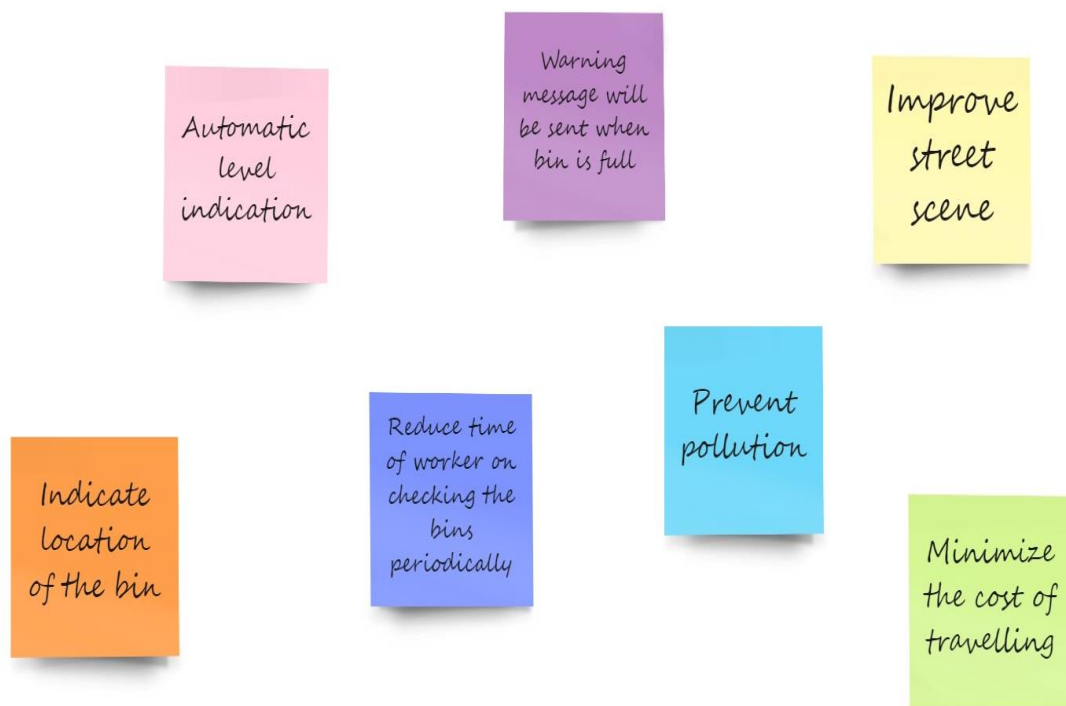
An empathy map is a collaborative visualization used to express clearly what one knows about a particular type of user. It externalizes knowledge about users in order to create a shared understanding of user needs, and aid in decision making. Empathy maps are split into 4 quadrants (Says, Thinks, Does, and Feels), with the user in the middle. Empathy maps provide a glance into who a user is as a whole. The Says quadrant contains what the user says or what he needs. The Thinks quadrant captures what the user is thinking throughout the experience. The Does quadrant encloses the actions the user takes. The Feels quadrant is the user's emotional state.



**FIGURE 3.1 EMPATHY MAP**

### **3.2 IDEATION & BRAINSTORMING**

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. Brainstorming is usually conducted by getting a group of people together to come up with either general new ideas or ideas for solving a specific problem or dealing with a specific situation. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity. Both brainstorming and ideation are processes invented to create new valuable ideas, perspectives, concepts and insights, and both are methods for envisioning new frameworks and systemic problem solving.



**FIGURE 3.2 BRAINSTORMING**

### **3.3 PROPOSED SOLUTION**

<b>S. No.</b>	<b>Parameter</b>	<b>Description</b>
	Problem Statement (Problem to be solved)	Due to lack of proper systems for disposal and collections, wastes and garbage's end up in the roads and surrounding. With the existing methods of collecting and disposal it is near impossible to manage such amount of waste.
	Idea / Solution description	The solution approached is to create a device for monitoring the garbage levels in the bin and alert the sector members when it is full.
	Novelty / Uniqueness	Determining the weight of the garbage to provide accuracy in detecting the levels. In case if there is more amount of light weight waste, the system will indicate that the garbage is full (as it will only detect only the level according to the existing solution).
	Social Impact / Customer Satisfaction	The response will be faster which in turn provides Customer Satisfaction.
	Business Model (Revenue Model)	We will create a model regarding customer segments, channels (website), customer relationship, value propositions.
	Scalability of the Solution	This will be more scalable as many projects were developed regarding this problem statement.

**TABLE 3.3 PROPOSED SOLUTION**

### **3.4 PROBLEM SOLUTION FIT**

The Problem solution fit simply means that one have found a problem with the customer and that the solution one have realised for it actually solves the customers problem. The problem solution fit is an important step towards the Product-Market Fit. The structure of problem solution fit is given below.

**Customer state fit:** To make sure one understand the target group, their limitations and their currently available solutions, against which one is going to compete.

**Problem-Behavior fit:** To help one to identify the most urgent and frequent problems, understand the real reasons behind them and see which behavior supports it.

**Communication-Channel fit:** To help one to sharpen the communication with strong triggers, emotional messaging and reaching customers via the right channels.

**Solution guess:** Translate all the validated data one have gathered into a solution that fits the customer state and his/her limitations, solves a real problem and taps into the common behavior of the target group.

## **CHAPTER 4**

### **REQUIREMENT ANALYSIS**

Requirements analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and Non-functional requirements.

#### **4.1 FUNCTIONAL REQUIREMENT**

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

<b>FR No.</b>	<b>Functional Requirement (Epic)</b>	<b>Sub Requirement (Story / Sub-Task)</b>
FR-1	Detailed bin inventory	<ul style="list-style-type: none"><li>• We can see bin details in the Dashboard – capacity, waste type, last measurement, GPS location and collection schedule or pi</li></ul>
FR-2	Real time bin monitoring	<ul style="list-style-type: none"><li>• Ultrasonic sensor is used to indicate the level of garbage</li><li>• MQ-136-Hydrogen Sulfide Gas Sensor is used to detect the foul smell of the garbage</li><li>• With real-time data and predictions, you can eliminate the overflowing bins and stop collecting half-empty ones.</li><li>• Sensors recognize picks as well; so you can check when the bin was last collected.</li></ul>
FR-3	Expensive bins	<ul style="list-style-type: none"><li>• We help you identify bins that drive up your collection costs. The tool calculates a rating for each bin in terms of collection costs</li></ul>



FR-4	Adjustment of bins	<ul style="list-style-type: none"> <li>Based on the historical data, you can adjust bin capacity or location where necessary</li> </ul>
------	--------------------	---

**TABLE 4.1 FUNCTIONAL REQUIREMENT**

## **4.2 NON FUNCTIONAL REQUIREMENT**

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements. They basically deal with issues like Portability, Security, Maintainability, Reliability, Scalability, Performance, Reusability, Flexibility

<b>FR No.</b>	<b>Non-Functional Requirement</b>	<b>Description</b>
NFR-1	<b>Usability</b>	With user experience as the core, the analysis of users' product usability can indeed help designers better understand users' potential needs in waste management, behaviour and experience.
NFR-2	<b>Security</b>	Use a reusable bottles Use reusable grocery bags Purchase wisely and recycle Avoid single use food and drink containers.
NFR-3	<b>Reliability</b>	Smart waste management is also about creating better working conditions for waste collectors and drivers by reducing the collection of empty bins.
NFR-4	<b>Performance</b>	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%.
NFR-5	<b>Availability</b>	By developing & deploying resilient hardware and beautiful software we empower cities,

		businesses, and countries to manage waste smarter.
NFR-6	<b>Scalability</b>	In most of the villages , garbages are not cleaned for longer time .so by smart bins will help them to make their places clean and also it is more cost effect and scalability when we moves to smarter

**TABLE 4.2 NON- FUNCTIONAL REQUIREMENT**

## **CHAPTER 5**

### **PROJECT DESIGN**

#### **5.1 DATA FLOW DIAGRAMS**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

There are four main elements of a DFD — external entity, process, data store, and data flow.

##### **External entity**

An external entity, which are also known as terminators, sources, sinks, or actors, are an outside system or process that sends or receives data to and from the diagrammed system. They’re either the sources or destinations of information, so they’re usually placed on the diagram’s edges. External entity symbols are similar across models except for Unified, which uses a stick-figure drawing instead of a rectangle, circle, or square.

##### **Process**

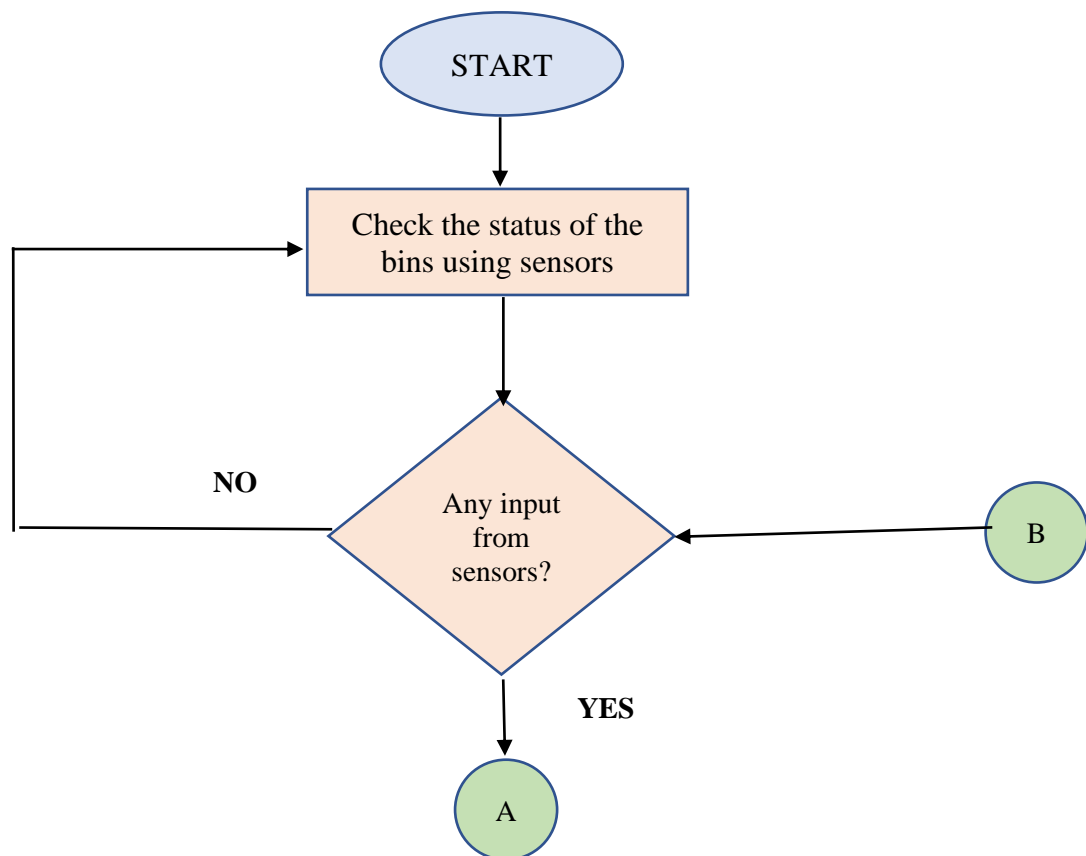
Process is a procedure that manipulates the data and its flow by taking incoming data, changing it, and producing an output with it. A process can do this by performing computations and using logic to sort the data, or change its flow of direction. Processes usually start from the top left of the DFD and finish on the bottom right of the diagram.

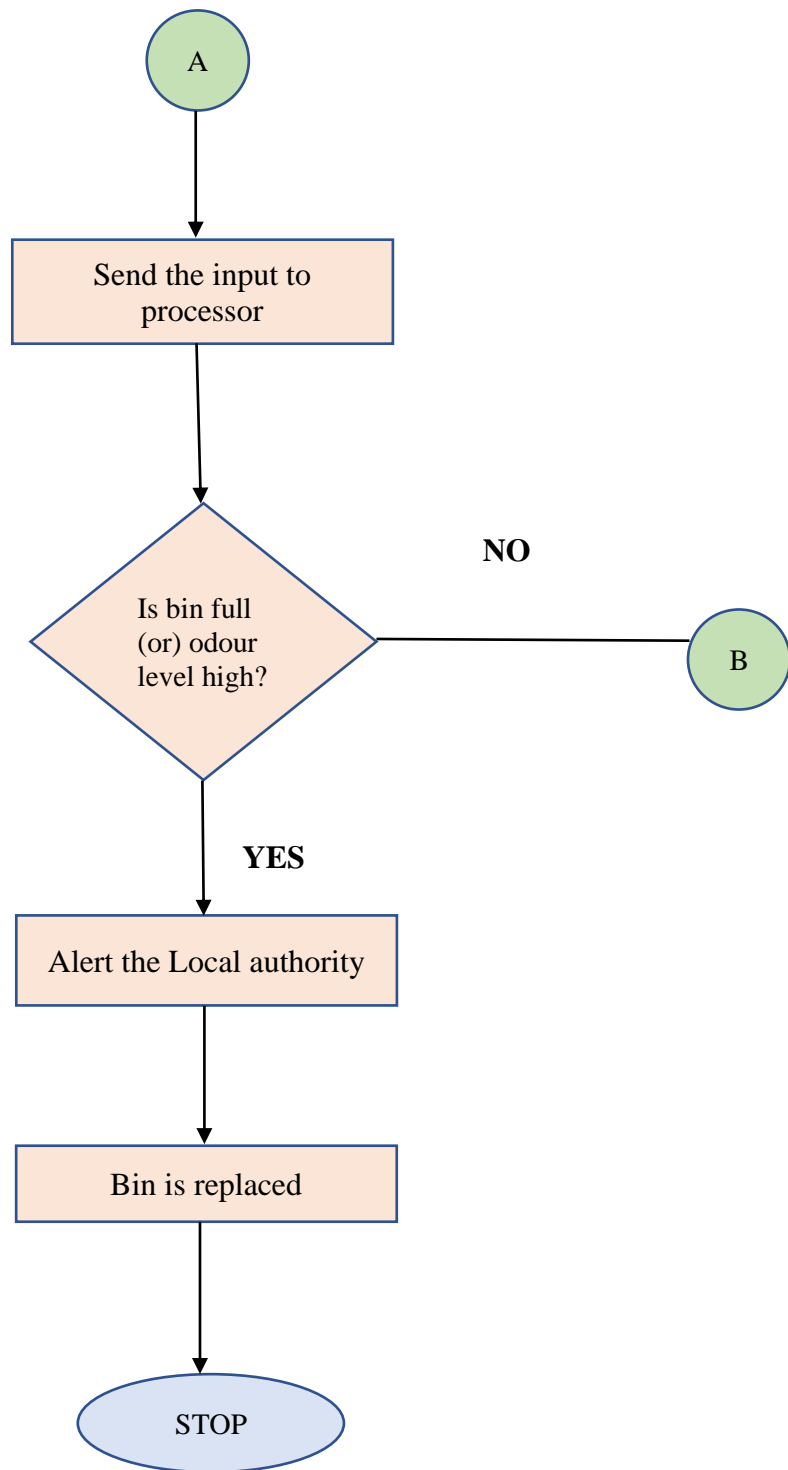
## Data store

Data stores hold information for later use, like a file of documents that's waiting to be processed. Data inputs flow through a process and then through a data store while data outputs flow out of a data store and then through a process.

## Data flow

Data flow is the path the system's information takes from external entities through processes and data stores. With arrows and succinct labels, the DFD can show the direction of the data flow



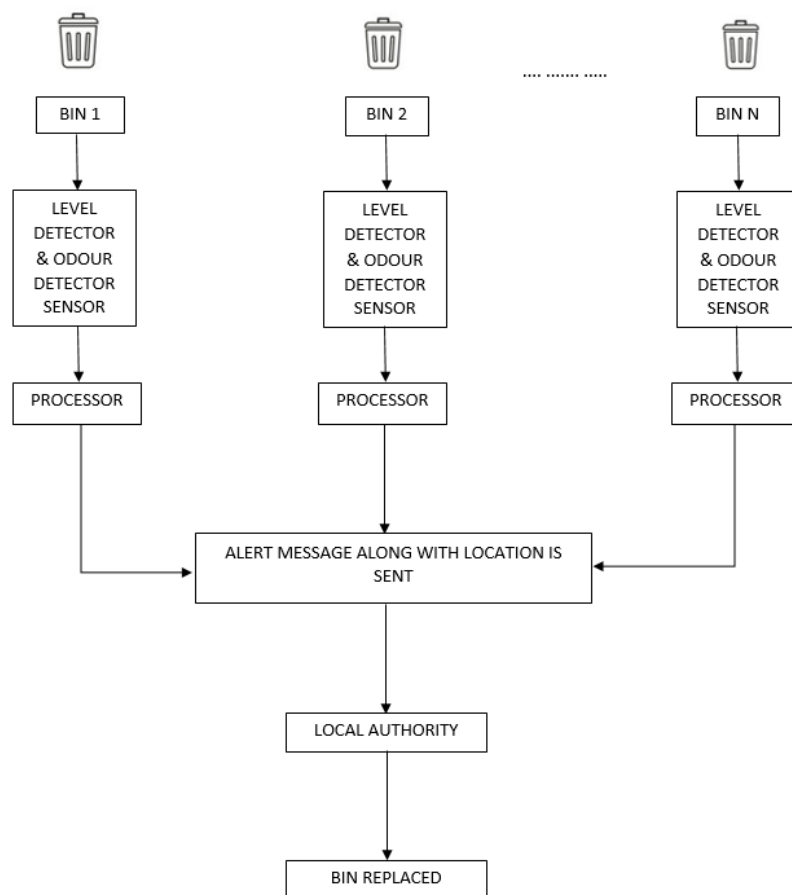


**FIGURE 5.1 DATA FLOW DIAGRAM**

## **5.2 SOLUTION & TECHNICAL ARCHITECTURE**

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



**FIGURE 5.2 SOLUTION & TECHNICAL ARCHITECTURE**

### **5.3USERSTORIES**

<b>TITLE</b>	<b>DESCRIPTION</b>	<b>DATE</b>
<b>Literature Survey &amp; Information Gathering</b>	Literature Survey	4 <sup>th</sup> OCTOBER 2022
<b>Prepare Empathy Map</b>	Prepare Empathy Map for the problem statement	4 <sup>th</sup> OCTOBER 2022
<b>Ideation</b>	Preparing Brainstorming session, ideas to implement etc	6 <sup>th</sup> OCTOBER 2022
<b>Proposed Solution</b>	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	10 <sup>th</sup> OCTOBER 2022
<b>Problem Solution Fit</b>	Prepare problem - solution fit document.	14 OCTOBER 2022
<b>Solution Architecture</b>	Prepare solution architecture document.	14 OCTOBER 2022

<b>Customer Journey</b>	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	28 OCTOBER 2022
<b>Functional Requirement</b>	Block diagram representation	21 OCTOBER 2022
<b>Data Flow Diagrams</b>	Draw the data flow diagrams and submit for review.	28 OCTOBER 2022
<b>Technology Architecture</b>	Prepare the technology architecture diagram.	28 OCTOBER 2022
<b>Prepare Milestone &amp; Activity List</b>	Prepare the milestones & activity list of the project.	4 <sup>th</sup> NOVEMBER 2022
<b>Project Development Delivery of Sprint-1, 2, 3 &amp; 4</b>	Develop & submit the developed code by testing it.	IN PROGRESS..

**TABLE 5.1 USER STORIES**



**CHAPTER 6**  
**PROJECT PLANNING & TESTING**

**6.1 SPRINT PLANNING AND ESTIMATION**

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team. The sprint is a set period of time where all the work is done. However, before leap intoaction it is necessary to set up the sprint. It need to decide on how long the time box is going to be, the sprint goal, and where it is going to start. The sprint planning session kicks off the sprint by setting the agenda and focus. If done correctly, it also creates an environment wherethe team is motivated, challenged, and can be successful.

**6.2 SPRINT DELIVERY SCHEDULE**

The sprint delivery plan is scheduled accordingly as shown in the below table 6.2 which consists of the sprints with respective to their duration, sprint start and end date and the releasing data.

**SPRINT 1:**

**Product Backlog, Sprint Schedule, and Estimation (4 Marks)**

<b>Sprint</b>	<b>Functional Requirement (Epic)</b>	<b>User story Number</b>	<b>User Story / Task</b>	<b>Story Points</b>	<b>Priority</b>	<b>Team Members</b>
Sprint-1	Registration	USN-1	User can signup using their email and password and confirm the details	20	High	Bala dharani
Sprint-1	Login	USN-2	A confirmation mail is sent to the user.	10	High	Aishwarya

Sprint-2	Dashboard	USN-3	User can login using login credentials and is authenticated	10	Medium	Anuja
Sprint-3	Work space	USN-4	User can view the previous login activities of the account and updates.	20	High	Aishwarya
Sprint-4	Exit	USN-5	User can search for the bins available around the location	10	Medium	Dharani

**TABLE 6.1 USER STORIES**

**Project Tracker, Velocity & Burndown Chart: (4 Marks)**

<b>Sprint</b>	<b>Total Story Point</b>	<b>Duration</b>	<b>Sprint Start Date</b>	<b>Sprint End Date (Planned)</b>	<b>Story Points Completed (as on Planned End Date)</b>	<b>Sprint Release Date(Actual)</b>
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	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

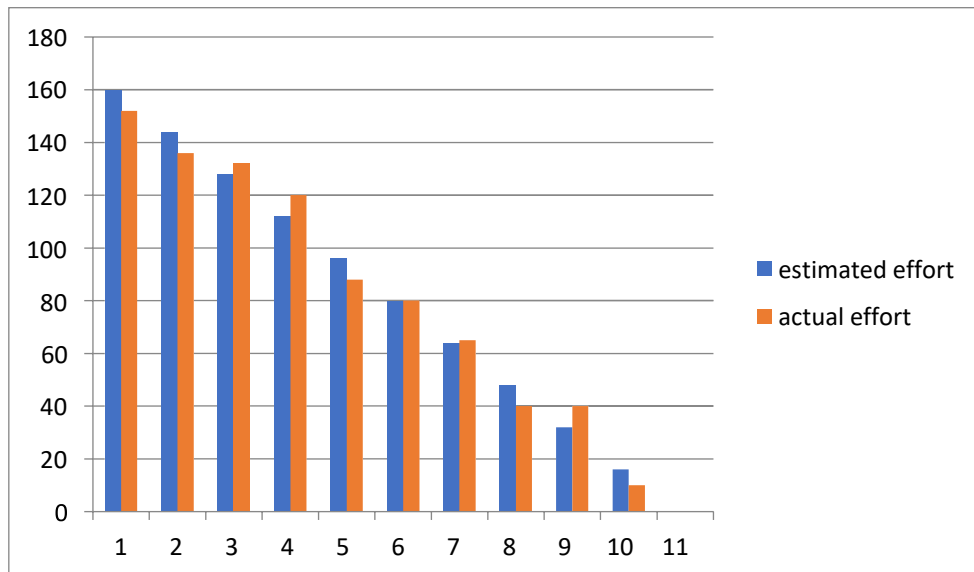
**TABLE 6.2 SPRINT SCHEDULE**

**Burndown Chart:**

The Burndown chart for 10 days is given below:

DAY	ESTIMATED WORK	ACTUAL WORK
1	160	152
2	144	136
3	128	132
4	112	120
5	96	88
6	80	80
7	64	65
8	48	40
9	32	40
10	16	10

**TABLE 6.3 BURNDOWN TABLE**



**FIGURE 6.1 BURNDOWN CHART**

SPRINT 2:

### Code for Data Transfer from Sensors

```
#include <WiFi.h>                // library for wifi
#include <PubSubClient.h>         // library for MQTT
#include <LiquidCrystal_I2C.h>
#include <mjson.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);

//----- credentials of IBM Accounts -----

#define ORG "gw0bk3"             // IBM organisation id
#define DEVICE_TYPE "NodeMCU"    // Device type mentioned in ibm watson iot
platform
#define DEVICE_ID "Anu"          // Device ID mentioned in ibm watson iot platform
#define TOKEN "123456789"        // Token

//----- customise above values -----
```

```

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";           // server name
char publishTopic[] = "iot-2/evt/data/fmt/json";                         // topic name and type of
event perform and format in which data to be send
char topic[] = "iot-2/cmd/led/fmt/String";                               // cmd Represent type and
command is test format of strings
char authMethod[] = "use-token-auth";                                    // authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;               //Client id

//-----

WiFiClient wifiClient;                                                  // creating instance for wificlient
PubSubClient client(server, 1883, wifiClient);

#define ECHO_PIN 12
#define TRIG_PIN 13
float dist;
String data3;
bool SealBin = true;
void setup()
{
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  //pir pin
  pinMode(34, INPUT);

  //ledpins
  pinMode(23, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);

```

```
pinMode(15, OUTPUT);
```

```
lcd.init();  
lcd.backlight();  
lcd.setCursor(1, 0);  
lcd.print("");  
wifiConnect();  
mqttConnect();  
}
```

```
float readcmCM()  
{  
  digitalWrite(TRIG_PIN, LOW);  
  delayMicroseconds(2);  
  digitalWrite(TRIG_PIN, HIGH);  
  delayMicroseconds(10);  
  digitalWrite(TRIG_PIN, LOW);  
  int duration = pulseIn(ECHO_PIN, HIGH);  
  return duration * 0.034 / 2;  
}
```

```
void loop()  
{  
  
  lcd.clear();  
  
  publishData();  
  delay(500);  
  if (!client.loop())  
  {  
    mqttConnect();          // function call to connect to IBM  
  }  
}
```

```
/* -----retrieving to cloud-----*/
```

```
void wifiConnect()
{
  Serial.print("Connecting to ");
  Serial.print("Wifi");
  WiFi.begin("Wokwi-GUEST", "", 6);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.print("WiFi connected, IP address: ");
  Serial.println(WiFi.localIP());
}

void mqttConnect()
{
  if (!client.connected())
  {
    Serial.print("Reconnecting MQTT client to ");
    Serial.println(server);
    while (!client.connect(clientId, authMethod, token))
    {
      Serial.print(".");
      delay(500);
    }
    initManagedDevice();
    Serial.println();
  }
}

void initManagedDevice()
{
```

```

if (client.subscribe(topic))
{
    Serial.println("IBM subscribe to cmd OK");
}
else
{
    Serial.println("subscribe to cmd FAILED");
}
}

void publishData()
{
    float cm = readcmCM();

    if(digitalRead(34))                //pir motion detection
    {
        Serial.println("Motion Detected");
        Serial.println("Lid Opened");
        digitalWrite(15, HIGH);

    }

    if(digitalRead(34)== true)
    {
        if(cm <= 60)                //Bin level detection
        {
            digitalWrite(2, HIGH);
            Serial.println("High Alert!!!,Trash bin is about to be full");
            Serial.println("Lid Closed");
            lcd.print("Full! Don't use");
            delay(2000);
            lcd.clear();
            digitalWrite(4, LOW);
            digitalWrite(23, LOW);
        }
    }
}

```



```

else if(cm > 60 && cm < 120)
{
    digitalWrite(4, HIGH);
    Serial.println("Warning!!,Trash is about to cross 50% of bin level");
    digitalWrite(2, LOW);
    digitalWrite(23, LOW);

}
else if(cm > 120)
{
    digitalWrite(23, HIGH);
    Serial.println("Bin is available");
    digitalWrite(2,LOW);
    digitalWrite(4, LOW);

}
    delay(10000);
    Serial.println("Lid Closed");
}
else
{
    Serial.println("No motion detected");
    digitalWrite(2, LOW);
    digitalWrite(15, LOW);
    digitalWrite(4, LOW);
    digitalWrite(23, LOW);
}

}
else
{
    digitalWrite(15, LOW);

```

```

    }

    if(cm <= 60)
    {
        digitalWrite(21,HIGH);
        String payload = "{\"High_Alert\":\"";
        payload += cm;
        payload += " }";
        Serial.print("\n");
        Serial.print("Sending payload: ");
        Serial.println(payload);

        if (client.publish(publishTopic, (char*) payload.c_str())) // if data is uploaded to cloud
            successfully,prints publish ok else prints publish failed
        {
            Serial.println("Publish OK");
        }
    }
    else if(cm <= 120)
    {
        digitalWrite(22,HIGH);
        String payload = "{\"Warning\":\"";
        payload += cm ;
        payload += " }";
        Serial.print("\n");
        Serial.print("Sending payload: ");
        Serial.println(payload);
        if(client.publish(publishTopic, (char*) payload.c_str()))
        {
            Serial.println("Publish OK");
        }
    }
    else
    {

```

```

Serial.println("Publish FAILED");
}
}
else if(cm > 120)
{
digitalWrite(23,HIGH);
String payload = "{";
payload += cm;
payload += " }";
Serial.print("\n");
Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) // if data is uploaded to cloud
successfully,prints publish ok else prints publish failed
{
Serial.println("Publish OK");
}

}

float inches = (cm / 2.54); //print on lcd
lcd.setCursor(0,0);
lcd.print("Inches");
lcd.setCursor(4,0);
lcd.setCursor(12,0);
lcd.print("cm");
lcd.setCursor(1,1);
lcd.print(inches, 1);
lcd.setCursor(11,1);
lcd.print(cm, 1);
lcd.setCursor(14,1);
delay(1000);
lcd.clear();

```

```
}
```

```
//handles commands from user side
```

```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
```

```
{
```

```
    Serial.print("callback invoked for topic: ");
```

```
    Serial.println(subscribetopic);
```

```
    for (int i = 0; i < payloadLength; i++) {
```

```
        data3 += (char)payload[i];
```

```
    }
```

```
    Serial.println("data: "+ data3);
```

```
    const char *s =(char*) data3.c_str();
```

```
    double pincode = 0;
```

```
        const char *buf;
```

```
        int len;
```

```
        if (mjson_find(s, strlen(s), "$.command", &buf, &len)) // And print it
```

```
        {
```

```
            String command(buf,len);
```

```
            if(command=="SealBin")
```

```
            {
```

```
                SealBin = true;
```

```
            }
```

}

data3="";

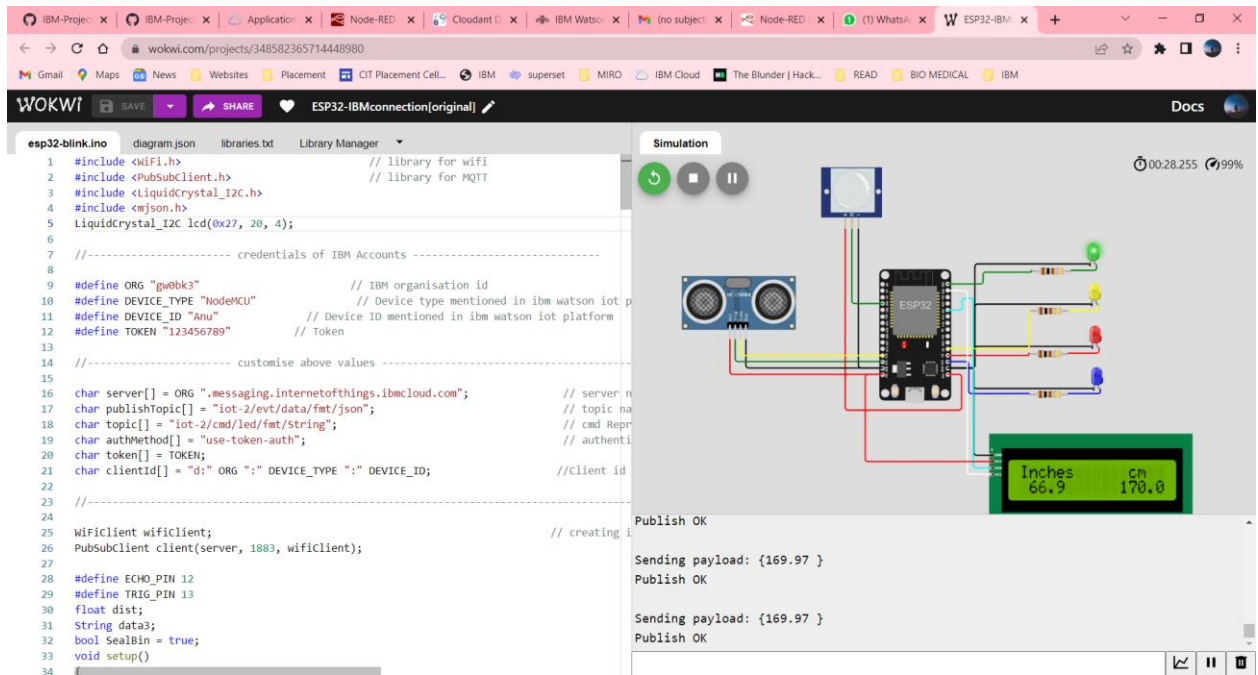
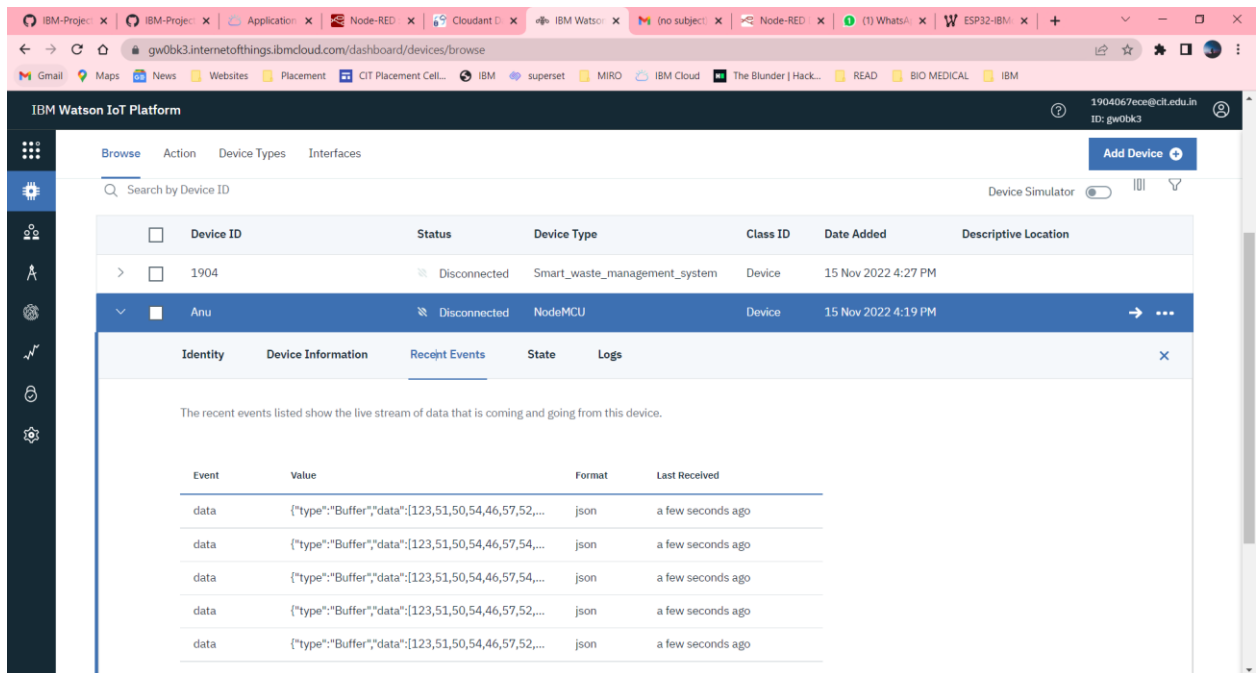
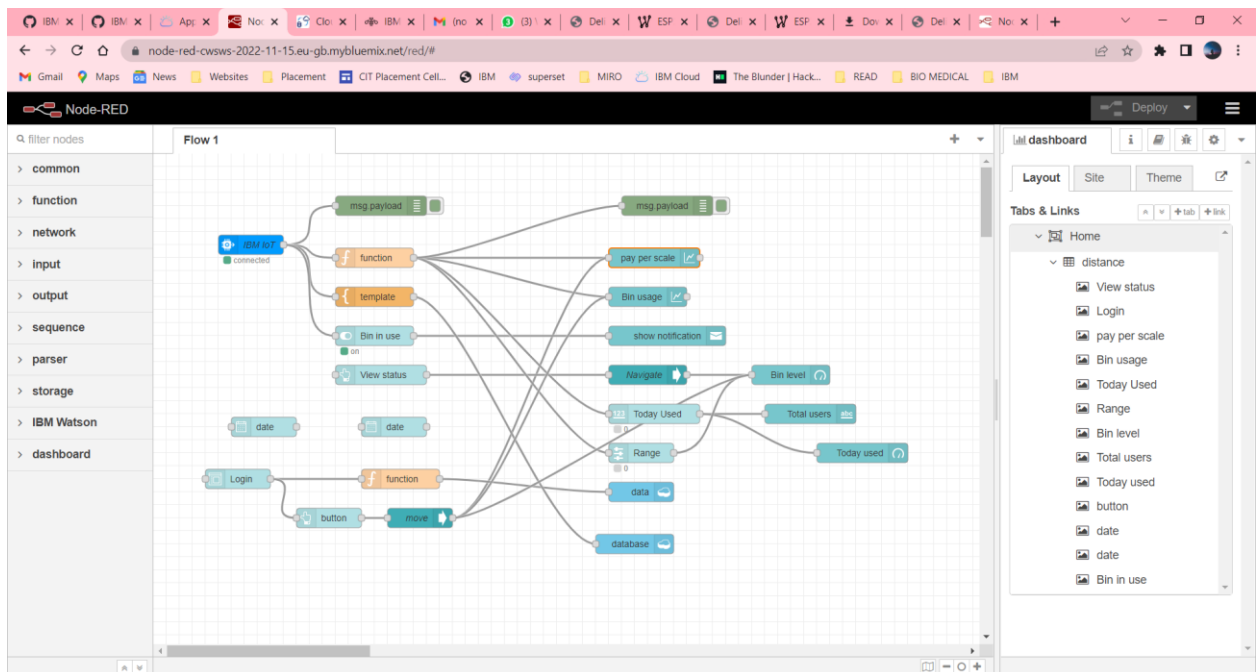


FIGURE 6.2 WOKWI SIMULATION

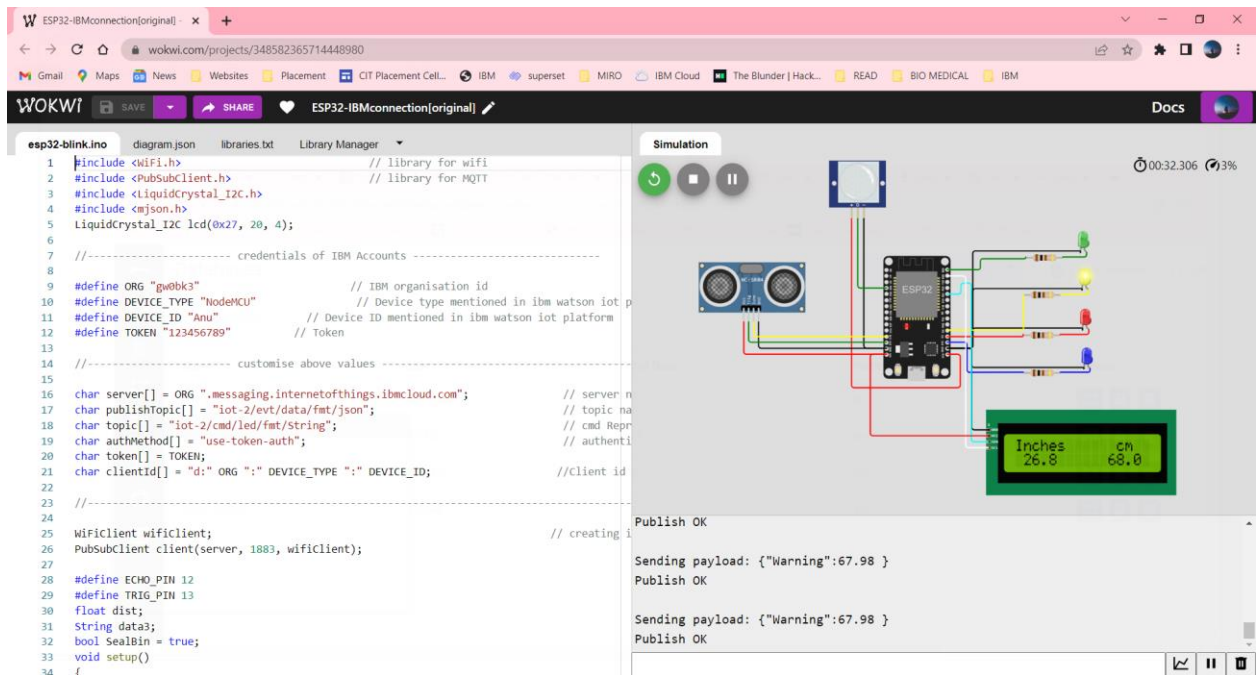


**FIGURE 6.3 WATSON PLATFORM**

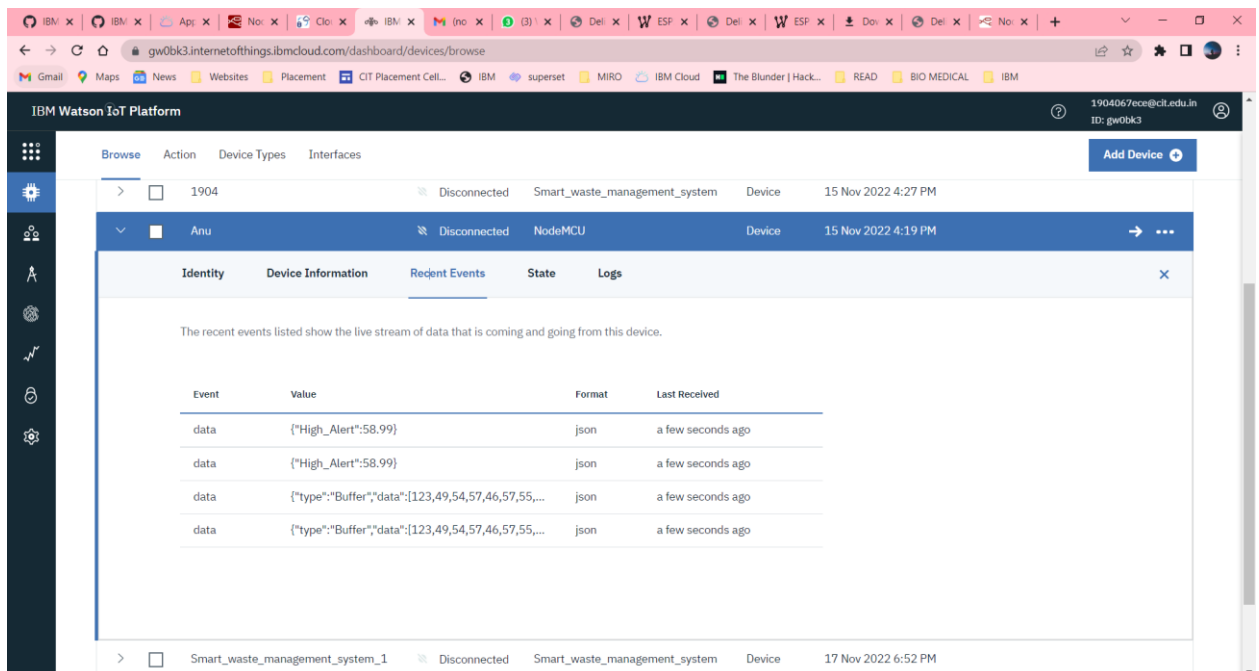
### SPRINT 3:



**FIGURE 6.4 NODE RED EXECUTION**



**FIGURE 6.5 WOKWI SIMULATION FOR VARIOUS CONDITION**



**FIGURE 6.6 WATSON RESULT FOR VARIOUS CONDITIONS**

The screenshot shows the IBM Cloudant Databases dashboard. The left sidebar contains navigation icons for Home, Databases, Documents, Query, Permissions, Changes, and Design Documents. The main area is titled 'Databases' and shows a table of existing databases. The table has columns for Name, Size, # of Docs, Partitioned, and Actions. There are four databases listed: 'data', 'database', 'noderdcws20221115', and 'smart\_waste'. Each database has a set of action icons (add, edit, delete, etc.). At the bottom right, it says 'Showing 1-4 of 4 databases. Databases per page: 20'.

Name	Size	# of Docs	Partitioned	Actions
data	202 bytes	2	No	[Icons]
database	3.7 KB	21	No	[Icons]
noderdcws20221115	30.3 KB	4	No	[Icons]
smart_waste	0 bytes	0	Yes	[Icons]

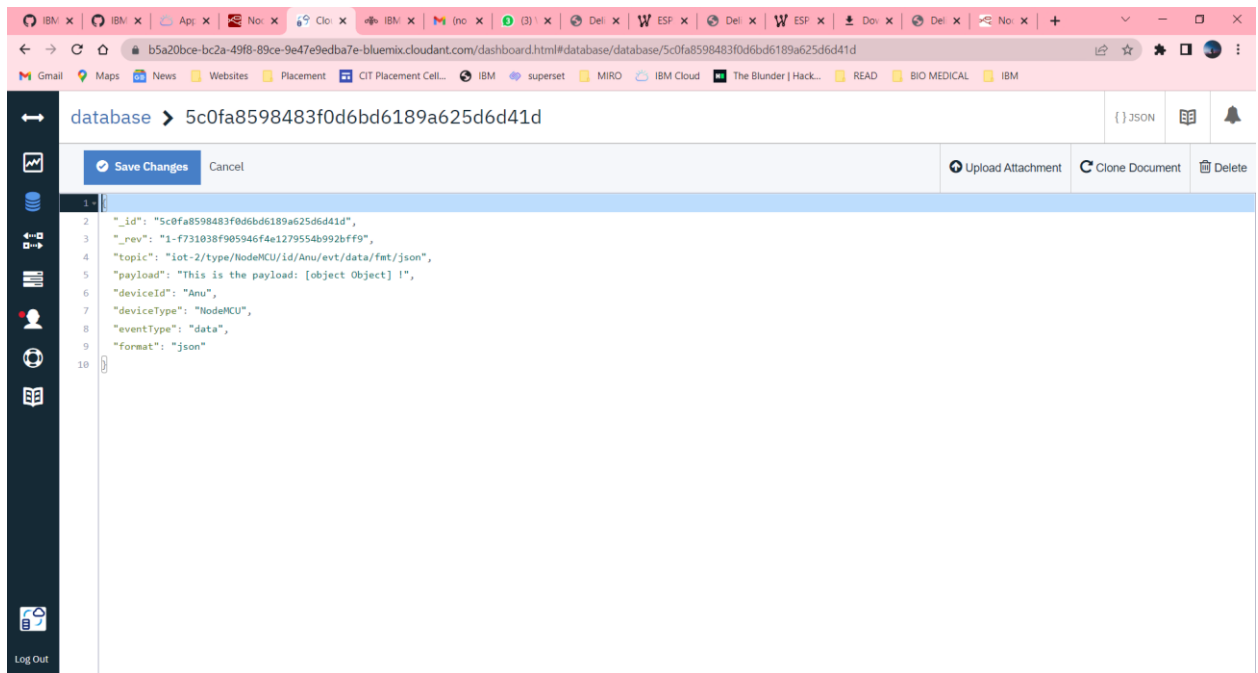
FIGURE 6.7 DATABASE OUTPUT

The screenshot shows the IBM Cloudant Documents dashboard for the 'database' database. The left sidebar has a 'database' breadcrumb and links to All Documents, Query, Permissions, Changes, and Design Documents. The main area is titled 'database' and shows a table of documents. The table has columns for id, key, and value. There are 11 documents listed. Each document has a set of action icons. At the bottom right, it says 'Showing document 1 - 11. Documents per page: 20'.

id	key	value
10d69c10317178f83d7d178a4f573fde	10d69c10317178f83d7d178a4f573fde	{ "rev": "1-f731038f9059464e1279554b992b..." }
1ec8f131638ba727ee3f8c05bd9ee868	1ec8f131638ba727ee3f8c05bd9ee868	{ "rev": "1-f731038f9059464e1279554b992b..." }
5b72d602b60a1186c691608712c0ec0b	5b72d602b60a1186c691608712c0ec0b	{ "rev": "1-f731038f9059464e1279554b992b..." }
5c0fa8598483f0d6bd6189a625d6d41d	5c0fa8598483f0d6bd6189a625d6d41d	{ "rev": "1-f731038f9059464e1279554b992b..." }
653efc7317cc0d26dee8e004fc5170c6	653efc7317cc0d26dee8e004fc5170c6	{ "rev": "1-f731038f9059464e1279554b992b..." }
653efc7317cc0d26dee8e004fc5a7fde	653efc7317cc0d26dee8e004fc5a7fde	{ "rev": "1-f731038f9059464e1279554b992b..." }
8c81919ec2bdcab5780d84055a683ba5	8c81919ec2bdcab5780d84055a683ba5	{ "rev": "1-f731038f9059464e1279554b992b..." }
ba321918c25b88721a4e6ff1825012e6	ba321918c25b88721a4e6ff1825012e6	{ "rev": "1-f731038f9059464e1279554b992b..." }
d1d0bb48285ac000788ef17d22b91fa	d1d0bb48285ac000788ef17d22b91fa	{ "rev": "1-f731038f9059464e1279554b992b..." }
d77314922a6712855fda32b3a92d7923	d77314922a6712855fda32b3a92d7923	{ "rev": "1-f731038f9059464e1279554b992b..." }
f9ab5f71a978f09f3029bec00c1a4bcc	f9ab5f71a978f09f3029bec00c1a4bcc	{ "rev": "1-f731038f9059464e1279554b992b..." }

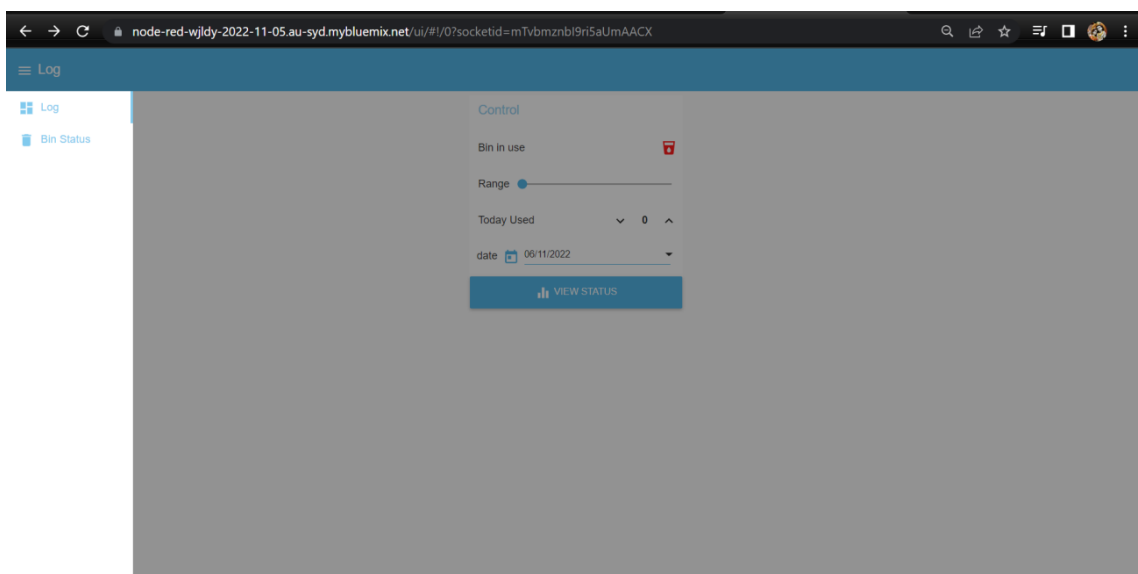
FIGURE 6.8 METADATA ANALYSIS



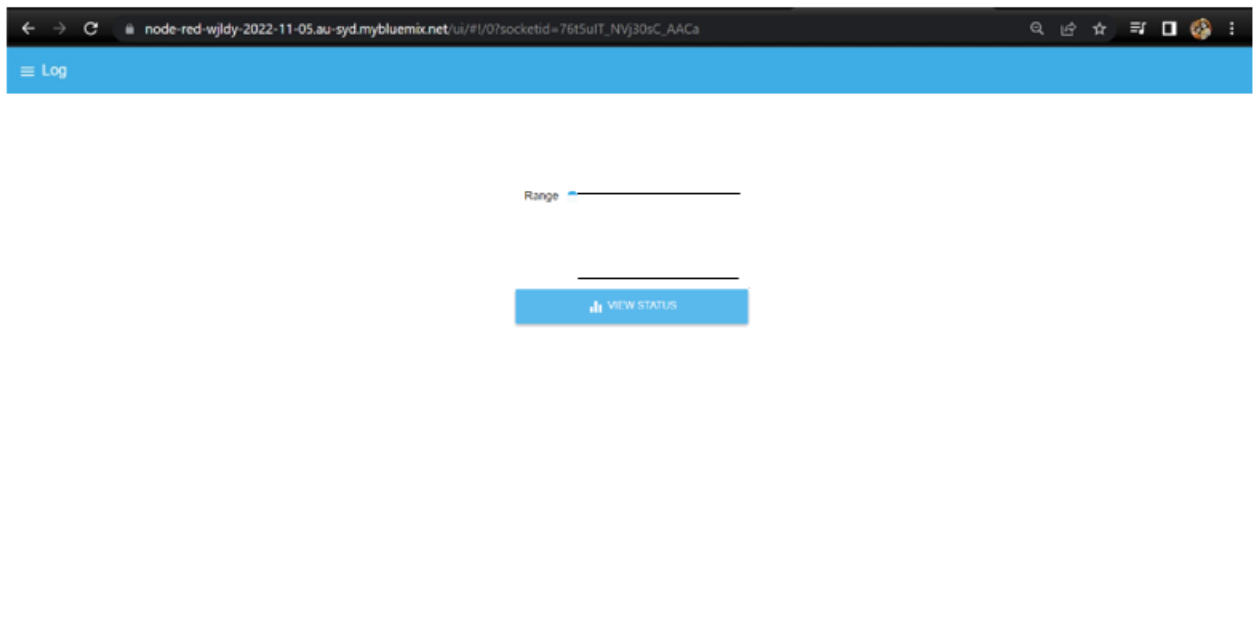


**FIGURE 6.9 CLOUD RESULT**

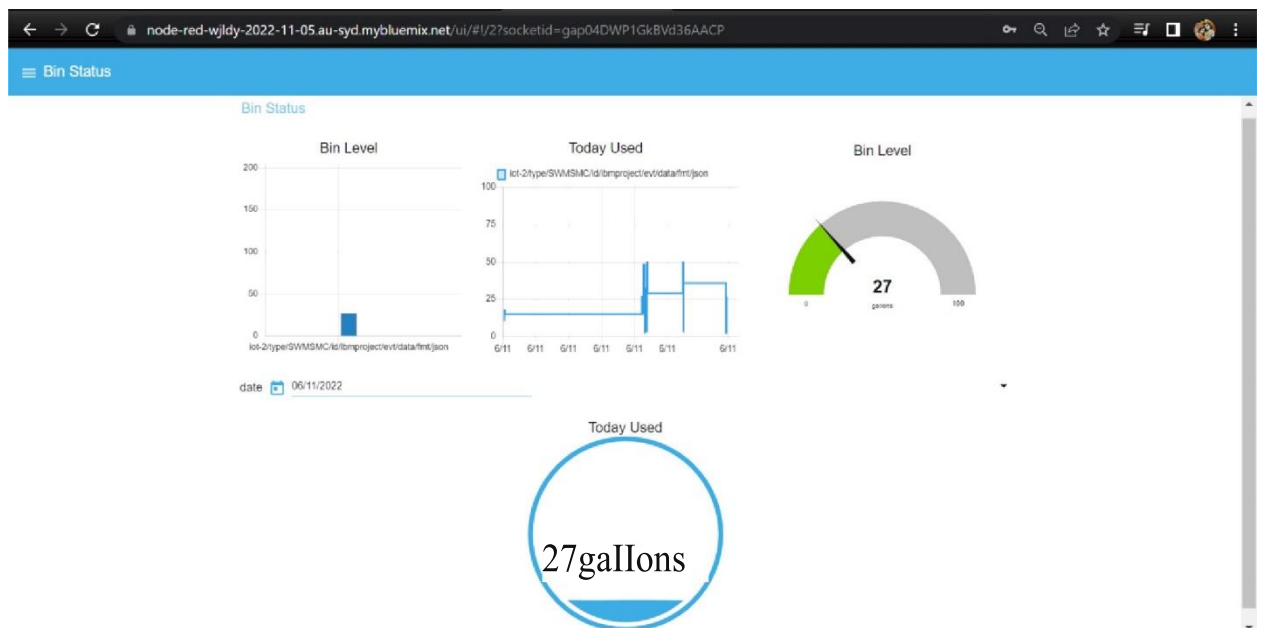
## **SPRINT 4:**



**FIGURE 6.10 NODE RED OUTPUT**



**FIGURE 6.11 STATUS VIEWING IN NODE RED**



**FIGURE 6.12 BIN STATUS IN NODE RED**

## **CHAPTER 7**

### **ADVANTAGES AND DISADVANTAGES**

#### **7.1 ADVANTAGES**

- It reduces infrastructure, operating and maintenance costs by upto 30%.
- It decreases traffic flow and consecutively noise due to less air pollution as result of less waste collection vehicles on the roads. This has become possible due to two way communication between smart dustbins and service operators.
- It keeps our surroundings clean and green and free from bad odour of wastes, emphasizes on healthy environment and keep cities more beautiful.
- It further reduces manpower requirements to handle the garbage collection process.
- Applying smart waste management process to the city optimizes management, resources and costs which makes it a "smart city".
- It helps administration to generate extra revenue by advertisements on smart devices.
- It saves time and money by using smart waste collection bins and systems equipped with fill level sensors. As smart transport vehicles go only to the filled containers or bins.

#### **7.2 DISADVANTAGES**

- System requires more number of waste bins for separate waste collection as per population in the city. This results into high initial cost due to expensive smart dustbins compare to other methods.
- Sensor nodes used in the dustbins have limited memory size.
- Wireless technologies used in the system such as zigbee and wifi have shorter range and lower data speed. In RFID based systems, RFID tags are affected by surrounding metal objects (if any).
- It reduces man power requirements which results into increase in unemployments for unskilled people.
- The training has to be provided to the people involved in the smart waste management system.

## **CHAPTER 8**

### **CONCLUSION**

Solid waste management is faced with a number of issues which include lack of throughput, inadequate solid waste data, efficiency problem, delays in collection and resistance to new technologies. Presently, waste management is a major problem for authorities who are responsible for such task because it's a costly service and it hugely impacts the environment as a whole. This study introduced a smart waste monitoring system that uses several sensors and communication technologies to achieve the set task. The proposed system was achieved through the development of theoretical models, layout and decision-making algorithms in the course of the project. There is an enormous amount of room for the development of this project in order for it to meet commercial standards. One of my many recommendations would be that of the addition of other sensors e.g. accelerometer. The accelerometer will make the system save more energy by turning on the system to measure the bin level only when the lid is opened to dispose waste. The system would then update its current state on ThingSpeak and turn off, preventing unnecessary measurement when the bin's level has not been altered due to dormancy. Another recommendation is the use of solar panel for power generation making its power supply autonomous

## **CHAPTER 9**

### **FUTURE SCOPE**

In future CCTV systems with IP based cameras can be used for monitoring the visual videos captured of the bin. It will also be helpful for the people. GPS can also be used to detect exact location of bin fault area; IP cameras can also be used to show fault with the help of video. Locations on Google maps with the help of sensors can be used to detect in which area the bin is overloaded.

## APPENDIX

### SOURCE CODE

```
import cv2
import numpy as np
import time
import pyzbar.pyzbar as pyzbar
from ibmcloudant.cloudant_v1 import CloudantV1
from ibmcloudant import CouchDbSessionAuthenticator
from ibm_cloud_sdk_core.authenticators import BasicAuthenticator

authenticator = BasicAuthenticator('apikey-v2-
16u3crmdpkgghxefdikvpssoh5fwezrmuup5fv5g3ubz', 'b0ab119f45d3e6255eabb978e7e2f0')
service = CloudantV1(authenticator=authenticator)
service.set_service_url('https://apikey-v2-
16u3crmdpkgghxefdikvpssoh5fwezrmuup5fv5g3ubz:b0ab119f45d3e6255eabb978e7e2f0')

cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_PLAIN

while True:
    _, frame = cap.read()
    decodedObjects = pyzbar.decode(frame)
    for obj in decodedObjects:
        a = obj.data.decode('UTF-8')
        cv2.putText(frame, "Ticket", (50,50), font, 2,
                    (255, 0, 0), 3)

    try:
        response = service.get_document(
            db = 'booking',
            doc_id = a
        ).get_result()
```

```

        print(response)
        time.sleep(5)
    except Exception as e:
        print("Not a Valid Ticket")
        time.sleep(5)

cv2.imshow("Frame", frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
client.disconnect().
var m = global.get('m')
var d = new Date()
var utc = d.getTime() + (d.getTimezoneoffset() * 60000);
var offset = 5.5;
newDate = new Date(utc + (3600000*offset));
var n = newDate.toISOString()
var date = n.slice(0,10)
var time = n.slice(11,19)
var d1 = date + ',' + time
msg.payload = {
    "_id" : d1,
    "Name" : m.name,
    "Age" : m.age,
    "Mobile" : global.get('b'),
    "Destination" : global.get('d'),
    "Seat" : global.get('s')
}
return msg;
global.set('s1',0)
global.set('s2',0)
global.set('s3',0)
global.set('s4',0)

```

```

var a1 = [1,2,3,4,5]
global.set('a',a1)
msg.payload = global.get('a')
return msg;
var a = global.get('a')
var s = []
for (let i = 0; i<a.length;i++){
    s.push(a[i])
}
if (s.length == 0){
    msg.options = [{"No seats available":0}]
}
else{
    msg.options = s
}
msg.payload = s
return msg;

```

```

import wiotp.sdk.device
import time
#import random
myConfig = {
    "identity": {
        "orgId": "4k7d8l",
        "typeId": "IBM",
        "deviceId": "63697477"
    },
    "auth" : {
        "token": "2(zMmQG_bjWs3d+-7b"
    }
}

```

```

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])

```



```

m = cmd.data['command']

client = wiotp.sdk.device.DeviceClient(config = myConfig, logHandlers = None)
client.connect()

def pub(data):
    client.publishEvent(eventId = "status", msgFormat = "json", data = myData, qos = 0)
    print("Published data Successfully: %s", myData)

while True:
    myData = {'name':'Train1','lat':17.6387448,'lon':78.4754336}
    pub(myData)
    time.sleep(3)
    myData = {'name':'Train1','lat':17.6341908,'lon':78.4744722}
    pub(myData)
    time.sleep(3)
    myData = {'name':'Train1','lat':17.6340889,'lon':78.4745052}
    pub(myData)
    time.sleep(3)
    myData = {'name':'Train1','lat': 17.6248626,'lon':78.4720259}
    pub(myData)
    time.sleep(3)
    myData = {'name':'Train1','lat':17.6188577,'lon':78.4698726}
    pub(myData)
    myData = {'name':'Train1','lat':17.6132382,'lon':78.4707318}
    pub(myData)
    time.sleep(3)
    client.commandCallback = myCommandCallback
client.disconnect()

```

## APPENDIX 2

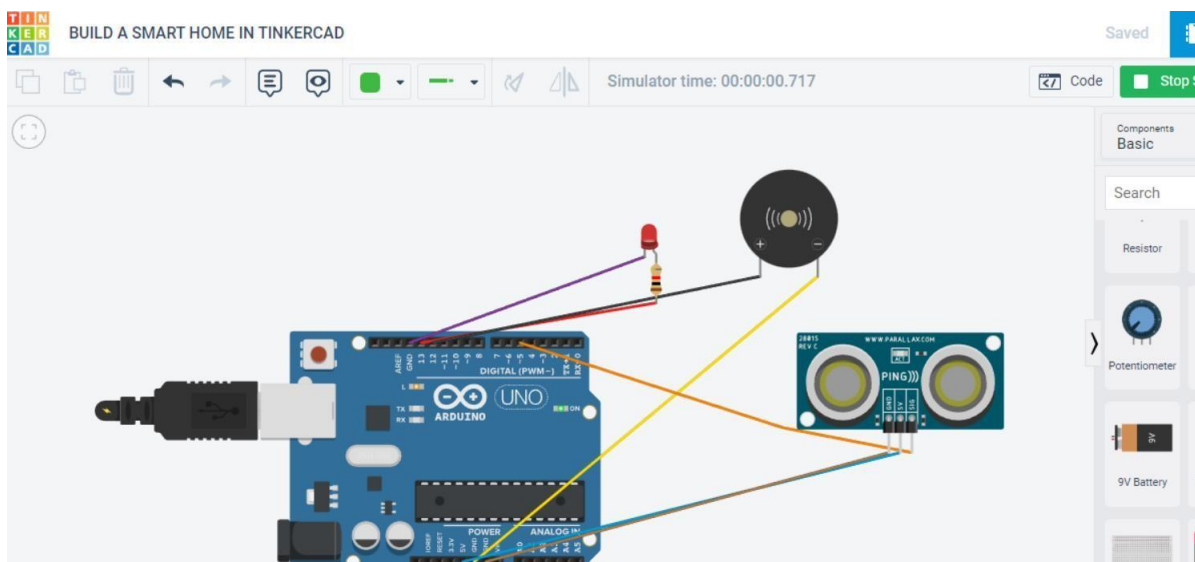
### ASSIGNMENTS

#### ASSIGNMENT 1

```
// C++ code
//
void setup()
{
  pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);
  Serial.begin(7000);
}

void loop()
{
  digitalWrite(13, HIGH);
  Serial.println("13LED ON");
  digitalWrite(12,OUTPUT);
  Serial.println("12LED ON");
}
```

#### SIMULATION:



## ASSIGNMENT 2

### CODE:

```
int t=2;
int e=3;
void setup()
{
  Serial.begin(9600);
  pinMode(t,OUTPUT);
  pinMode(e,INPUT);
  pinMode(12,OUTPUT);
}

void loop()
{
  //ultrasonic sensor
  digitalWrite(t,LOW);
  digitalWrite(t,HIGH);
  delayMicroseconds(10);
  digitalWrite(t,LOW);
  float dur=pulseIn(e,HIGH);
  float dis=(dur*0.0343)/2;
  Serial.print("Distance is: ");
  Serial.println(dis);

  //LED ON
  if(dis>=100)
  //LED ON
  if(dis>=100)
  {
    digitalWrite(8,HIGH);
    digitalWrite(7,HIGH);
  }

  //Buzzer For ultrasonic Sensor
  if(dis>=100)
  {
    for(int i=0; i<=30000; i=i+10)
    {
      tone(12,i);
      delay(1000);
      noTone(12);
      delay(1000);
    }
  }

  //Temperate Sensor
  double a= analogRead(A0);
```

```

//Temperate Sensor
double a= analogRead(A0);
double t=((a/1024)*5)-0.5)*100;
Serial.print("Temp Value: ");
Serial.println(t);
delay(1000);

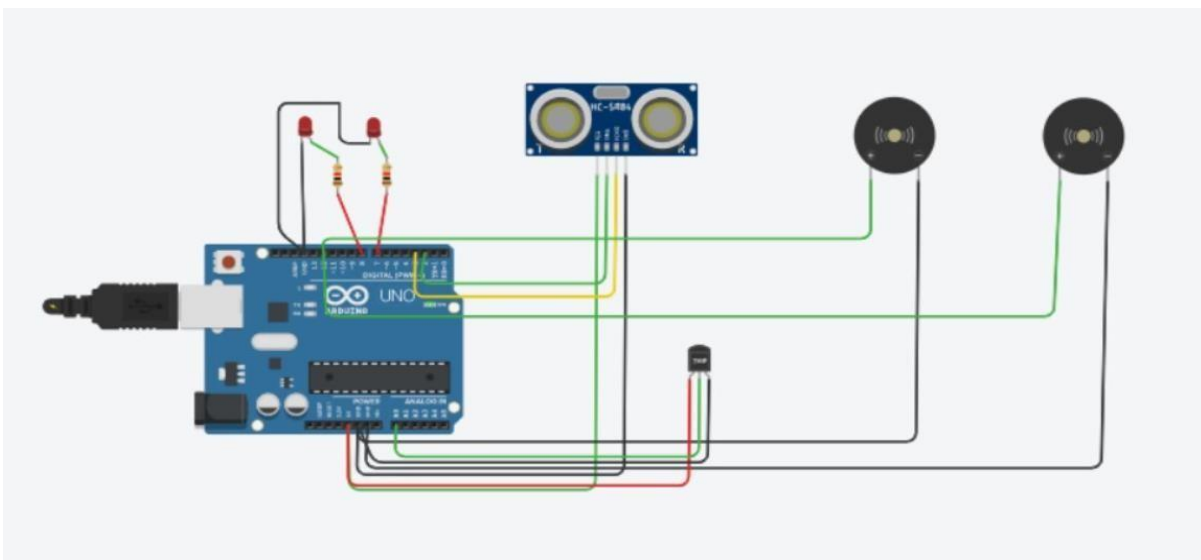
//LED ON
if(t>=100)
{
    digitalWrite(8,HIGH);
    digitalWrite(7,HIGH);
}

//Buzler for Temperature Sensor
if(t>=100)
{
    for(int i=0; i<=30000; i=i+10)
    {
        tone(12,i);
        delay(1000);
        noTone(12);
        delay(1000);
    }
}

//LED OFF
if(t<100)
{
    digitalWrite(8,LOW);
    digitalWrite(7,LOW);
}
}

```

## OUTPUT



### ASSIGNMENT 3

```
from gpiozero import
Button button =
Button(21)

while True:
    print(button.is_pressed)
while True:
    if
        button.is_pressed
    d: print("Hello")
```

Add an LED

```
from gpiozero import Button,
LED led = LED(25)
while True:
    button.wait_for_press()
    led.on()
    button.wait_for_release()
    led.off()
while True:
    led.on()
    button.wait_for_press()
    led.off()
    button.wait_for_release()
while True:
```

Traffic lights

```
from gpiozero import Button,
TrafficLights lights = TrafficLights(25,
8, 7)
while True:
    button.wait_for_press()
    lights.on()
    button.wait_for_release()
    lights.off()
while True:
```

## Traffic lights sequence

```
from time import
sleep while True:
    lights.green.on()
    sleep(1)
```

```
lights.red.on()
sleep(1) lights.off()
while True:
    button.wait_for_press()
    lights.green.on() sleep(1)
    lights.amber.on() sleep(1)
    lights.red.on() sleep(1)
    lights.off()
```

## ASSIGNMENT 4

Write code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100cms send an "alert" to the IBMcloud and display in the device recent events. Upload document withwokwi share link and images of IBM cloud.

**Wowki Link:** <https://wokwi.com/projects/348409044347650644>

### Source Code:

```
#include <WiFi.h>//library for wifi
#include <WiFiClient.h>
#include <PubSubClient.h>//library for MQTT
```

```

#include <ArduinoJson.h>

// creating the instance by passing pin and typr of dht connected

float distance;

#define sound_speed 0.034

int trigpin=18;

int echopin=19;

int led=5;

int LED=9;

long duration;

String message;// creating the instance by passing pin and typr of dht connected


void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);


//-----credentials of IBM Accounts-----


#define ORG "gw0bk3"//IBM ORGANITION ID

#define DEVICE_TYPE "Smart_waste_management_system"//Device type mentioned in ibm
watson IOT Platform

#define DEVICE_ID "1904"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "123456789"//Token

String data3;

float h, t;


//----- Customise the above values -----

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name

char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and
format in which data to be send

char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type
AND COMMAND IS TEST OF FORMAT STRING

```

```

char authMethod[] = "use-token-auth";// authentication method

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----

WiFiClient wifiClient; // creating the instance for wificlient

PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by
passing parameter like server id,portand wificredential

void setup()// configureing the ESP32

{
    Serial.begin(115200);
    pinMode(trigpin,OUTPUT);
    pinMode(echopin,INPUT);
    pinMode(led,OUTPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function

{

digitalWrite(trigpin,LOW);
digitalWrite(trigpin,HIGH);
delay(1000);
digitalWrite(trigpin,LOW);
duration=pulseIn(echopin,HIGH);
distance=duration*sound_speed/2;

```



```

Serial.println("distance"+String(distance)+"cm");

if(distance<100)
{
    message="Alert";
    digitalWrite(led,HIGH);
} else
{
    message="No problem";
    digitalWrite(led,LOW);
}

delay(1000);

PublishData(distance,message);

// if (!client.loop()) {
//   mqttconnect();
// }

}

/*.....retrieving to Cloud.....*/

void PublishData(float d, String a) {
    mqttconnect();//function call for connecting to ibm

    /*
        creating the String in in form JSon to update the data to ibm cloud
    */

    DynamicJsonDocument doc(1024);

    String payload;

    doc["Distance: "]=d;

    doc["Message: "]=a;

```

```
serializeJson(doc, payload);
```

```
Serial.print("Sending payload: ");
```

```
Serial.println(payload);
```

```
if (client.publish(publishTopic, (char*) payload.c_str())) {
```

```
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish  
ok in Serial monitor or else it will print publish failed
```

```
    } else {
```

```
        Serial.println("Publish failed");
```

```
    }
```

```
}
```

```
void mqttconnect() {
```

```
    if (!client.connected()) {
```

```
        Serial.print("Reconnecting client to ");
```

```
        Serial.println(server);
```

```
        while (!!!client.connect(clientId, authMethod, token)) {
```

```
            Serial.print(".");
```

```
            delay(500);
```

```
        }
```

```
        initManagedDevice();
```

```
        Serial.println();
```

```
    }
```

```
}
```

```
void wificonnect() //function defination for wificonnect
```

```

{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

```

```

void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

```

```

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{

```

```

  Serial.print("callback invoked for topic: ");

```

```

Serial.println(subscribetopic);

for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);

    data3 += (char)payload[i];
}

Serial.println("data: "+ data3);

if(data3=="lighton")
{
Serial.println(data3);
digitalWrite(LED,HIGH);
}

else
{
Serial.println(data3);
digitalWrite(LED,LOW);
}

data3="";
}

```

## OUTPUT:

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, showing an Arduino sketch for an ESP32 connected to an ultrasonic sensor. The code includes libraries for WiFi, WiFiClient, PubSubClient, and ArduinoJson. It defines variables for distance, sound speed, trigger pin, echo pin, LED, and duration. It also includes credentials for IBM Watson IoT Platform and a custom payload structure. The main loop publishes data to the IoT platform every 2 seconds.

```
1 #include <WiFi.h> //library for wifi
2 #include <WiFiClient.h>
3 #include <PubSubClient.h> //library for MQTT
4 #include <ArduinoJson.h>
5 // creating the instance by passing pin and type of dht connected
6 float distance;
7 #define sound_speed 0.034
8 int trigpin=18;
9 int echopin=19;
10 int led=5;
11 long duration;
12 String message; // creating the instance by passing pin and type of dht connected
13 void callback(char* subscribtopic, byte* payload, unsigned int payloadlength);
14 //-----credentials of IBM Accounts-----
15 #define ORG "gwobk3" //IBM ORGANIZATION ID
16 #define DEVICE_TYPE "Smart_waste_management_system" //Device type mentioned in ibm watson
17 #define DEVICE_ID "1904" //Device ID mentioned in ibm watson IOT Platform
18 #define TOKEN "123456789" //Token
19 String data3;
20 float h, t;
21 //----- Customise the above values -----
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
23 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event performed
24 char subscribtopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "ds:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
28 //-----
29 WiFiClient wifiClient; // creating the instance for wifiClient
```

On the right, the 'Simulation' window shows a 3D model of the ESP32 board connected to an ultrasonic sensor. The sensor is connected to the ESP32 via a breadboard. The simulation is running, and the console shows the following output:

```
Publish ok
distance256.95cm
Sending payload: {"Distance": ":256.9549866","Message": ":No problem"}
Publish ok
distance256.95cm
Sending payload: {"Distance": ":256.9549866","Message": ":No problem"}
Publish ok
```

Wokwi - Wokwi Arduino and x ASSIGNMENT 4 200420811.pdf x | +

https://wokwi.com/projects/348409044347650644

WOKWI SAVE SHARE

Docs SIGN IN

sketch.ino diagram.json libraries.txt Library Manager

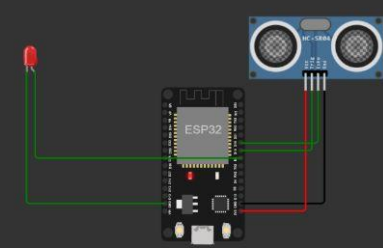
```

37 void setup()// configuring the ESP32
38 {
39   Serial.begin(115200);
40   pinMode(trigpin,OUTPUT);
41   pinMode(echopin,INPUT);
42   pinMode(led,OUTPUT);
43   delay(10);
44   Serial.println();
45   wificonnect();
46   mqttconnect();
47 }
48
49 void loop()// Recursive Function
50 {
51
52   digitalWrite(trigpin,LOW);
53   digitalWrite(trigpin,HIGH);
54   delay(1000);
55   digitalWrite(trigpin,LOW);
56   duration=pulseIn(echopin,HIGH);
57   distance=duration*sound_speed/2;
58   Serial.println("distance "+String(distance)+"cm");
59   if(distance<100)
60   {
61     message="Alert";
62     digitalWrite(led,HIGH);
63   } else
64   {
65     message="No problem";
66     digitalWrite(led,LOW);
67   }
68   delay(1000);
69   PublishData(distance,message);
70   // if (!client.loop()) {
71   //   mqttconnect();

```

Simulation

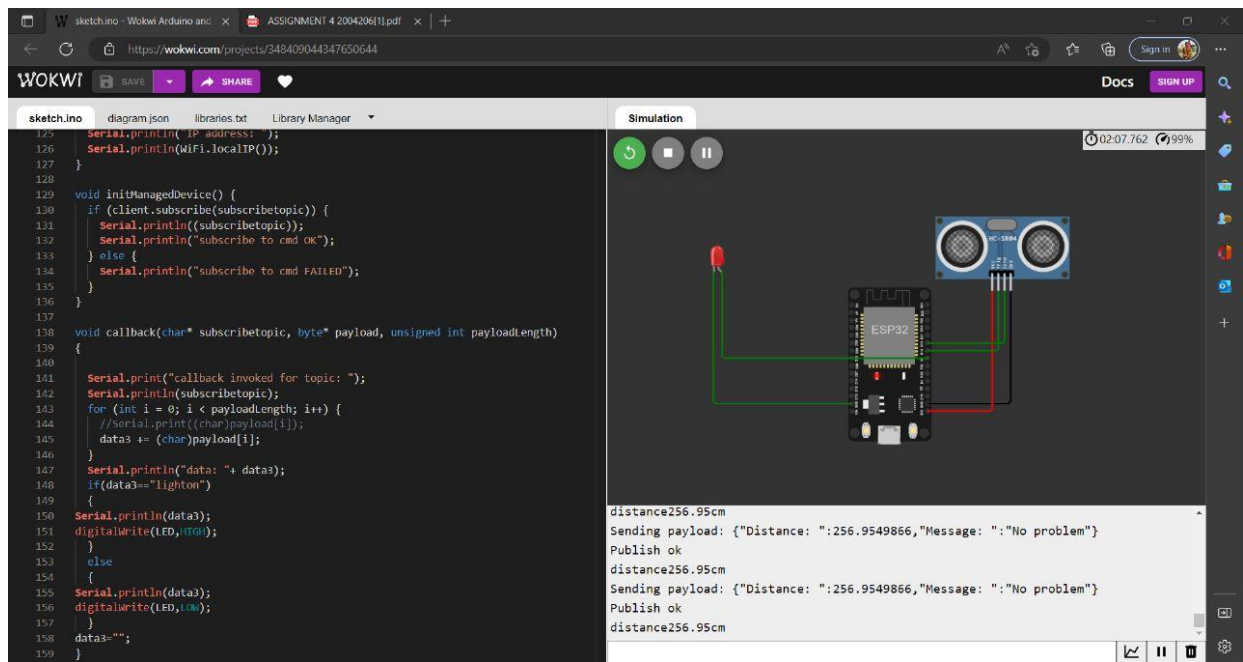
01:18.789 98%



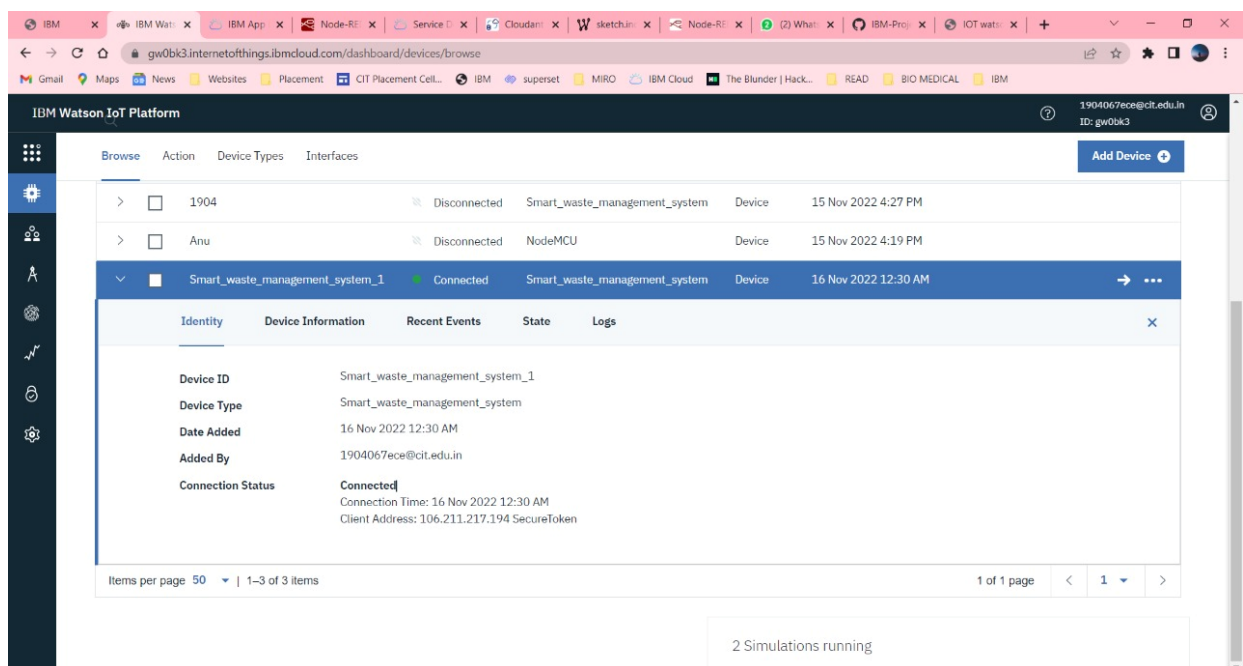
```

distance256.95cm
Sending payload: {"Distance": ":256.9549866","Message": ":No problem"}
Publish ok
distance256.94cm
Sending payload: {"Distance": ":256.9379883","Message": ":No problem"}
Publish ok
distance256.95cm

```



## OUTPUT IN WATSON:



IBM Watson IoT Platform

1904067ece@clt.edu.in  
ID: gw0bk3

Browse Action Device Types Interfaces

Add Device

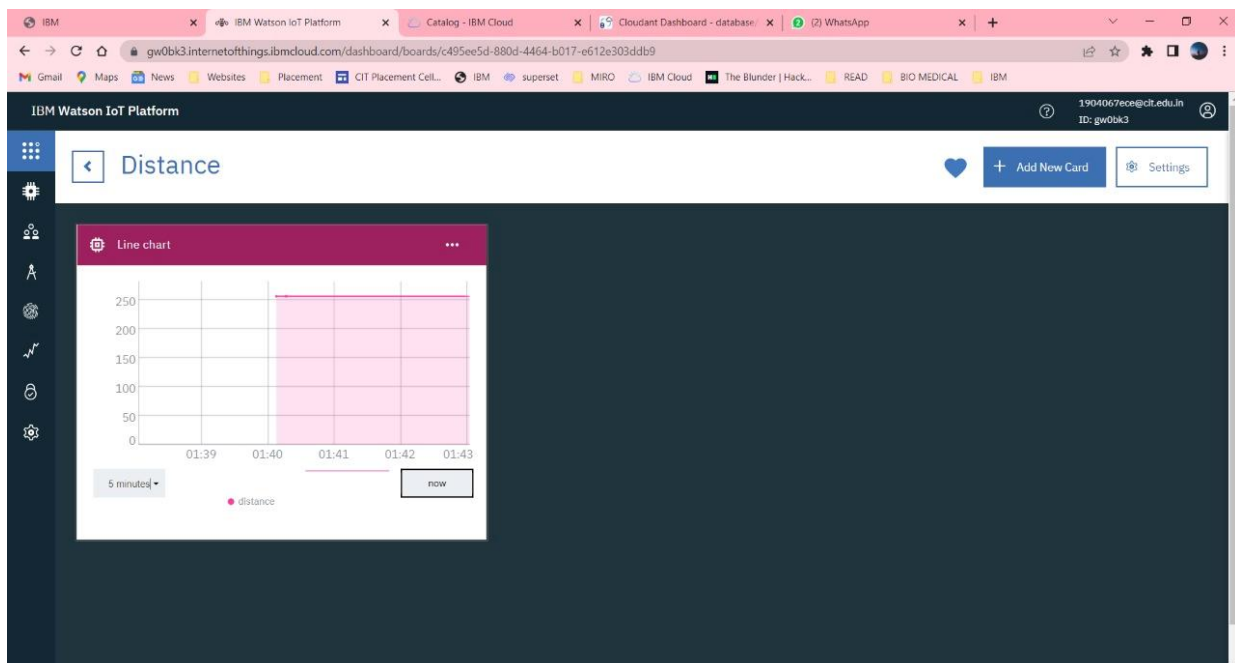
1904 Connected Smart\_waste\_management\_system Device 15 Nov 2022 4:27 PM

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"Distance": "256.9549866", "Message": "No pro..."}	json	a few seconds ago
Data	{"Distance": "256.9379883", "Message": "No pro..."}	json	a few seconds ago
Data	{"Distance": "256.9549866", "Message": "No pro..."}	json	a few seconds ago
Data	{"Distance": "256.9549866", "Message": "No pro..."}	json	a few seconds ago
Data	{"Distance": "256.9549866", "Message": "No pro..."}	json	a few seconds ago

> Anu Disconnected NodeMCU Device 15 Nov 2022 4:19 PM





## **REFERENCE**

1. Arindm Ghosh, Debajyoti Sarkar (2022), “Design and Fabrication of IoTbased Smart Dustbin” International Journal for Research in Applied Science & Engineering Tchnology
2. Akshayaa S, Evangiline R (2021), “Smart bin for Clean cities using IoT”, International Conference on Advanced Computing & Communication Systems
3. Upasana Sapra, Gayathri D (2019), “Efficient IoT based smart bin for waste management and disposal” ,Internatiosnal Journal of Informatics and Computer Science
4. Municipal Solid Waste Collection Problems: A Literature Review, Jeroen Beliën, Liesje De Boeck, Jonas Van Ackere
5. Nuortio, T., Kytojoki, J., Niska, H., Braysy, O.: Improved route planning and scheduling of waste collection and transport. Journal of Expert Systems with Applications 30(2), 223– 232 (2006) CrossRef
6. RFID and Integrated Technologies for Solid Waste Bin Monitoring System Maher Arebey, M A Hannan, Hassan Basri, R A Begum and Huda Abdullah
7. Concept, Design and Implementation of Automatic Waste Management System, Adil Bashir, Shoaib Amin Banday Ab. Rouf Khan, Mohammad Shafi
8. Smart Garbage Management System Vikrant Bhor<sup>1</sup>, Pankaj Morajkar<sup>2</sup>, Maheshwar Gurav<sup>3</sup>, Dishant Pandya<sup>4</sup>
9. Kanchan Mahajan, Chitode, J.S: Waste Bin Monitoring System Using Integrated Technologies

## **DEMO LINK:**

[https://drive.google.com/file/d/1Y4hSog\\_1jJ2YQtn1RsG71KZLQ\\_NoV8Ac/view?usp=drive\\_sdk](https://drive.google.com/file/d/1Y4hSog_1jJ2YQtn1RsG71KZLQ_NoV8Ac/view?usp=drive_sdk)

## **GIT HUB LINK:**

<https://github.com/IBM-EPBL/IBM-Project-10168-1659108242>