GARBAGE SEGREGATION AND MONITORING SYSTEM

A PROJECT REPORT

by

SIDHARTH PRAKASH 17BEE1012

ROHAN CHANDRASEKAR 17BEE1099

CSE3009 – INTERNET OF THINGS

under the guidance of

Dr. RAJESH KUMAR

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRICAL ENGINEERING

VIT CHENNAI

MAY/JUNE 2020

ABSTRACT

IoT – Internet Of Things refers to any “thing” , be it home appliances, vehicles, portable devices and various other gadgets, that is connected to the Internet and have the capability to connect and exchange with each other. The future of IoT is very promising and it has already started to impact our lives, homes, the cities we live in , how we work and how we interact with the objects around us.

In this project, we aim to tackle a very pressing yet often sidelined issue in our society – the task of garbage segregation and monitoring. The objective of this project is to build a **Garbage Segregation and Monitoring System** that will segregate input garbage waste into namely three types – Metal, Dry and Wet waste and send the status of the dustbin (how full the dustbin is) to the Cloud and display the status there. If the amount of garbage in the dustbin exceeds 70%, a notification is sent to the users e-mail, alerting the user.

The segregation will be done by a conveyor belt setup, where the input garbage will pass by 2 sensors (Moisture sensor and Inductive metal proximity sensor. No sensor needed for Dry waste as it will fall of the conveyor belt at the end of the path) and upon detection by a sensor, a servo motor will block the path of the moving garbage, thus making it fall into its respective dustbin. Then the status of this Smart Dustbin (containing an Ultrasonic Sensor) will be monitored online. For this, we have used the API **ThingSpeak** to obtain and display the information from the sensor and with the help of **Webhooks** (a web service that allows various websites to connect with each other) have used **IFTTT** to send an e-mail notification to the user.

PROJECT DESCRIPTION

1. COMPONENTS USED:

Hardware:

S.No	Component	Quantity
1.	Arduino UNO	2
2.	Soil Moisture Sensor	1
3.	Inductive Metal Sensor (JA180PO)	1
4.	Servo (sg90)	2
5.	Ultrasonic Sensor (HC-SR04)	1
6.	ESP8266 WiFi Module	1
7.	DC Motor and Conveyor belt system	1
8.	Wires	-

Software:

S.No	Software / Service
1.	ThingSpeak
2.	IFTTT
3.	Arduino IDE
4.	Webhooks

2.CIRCUIT DIAGRAM:

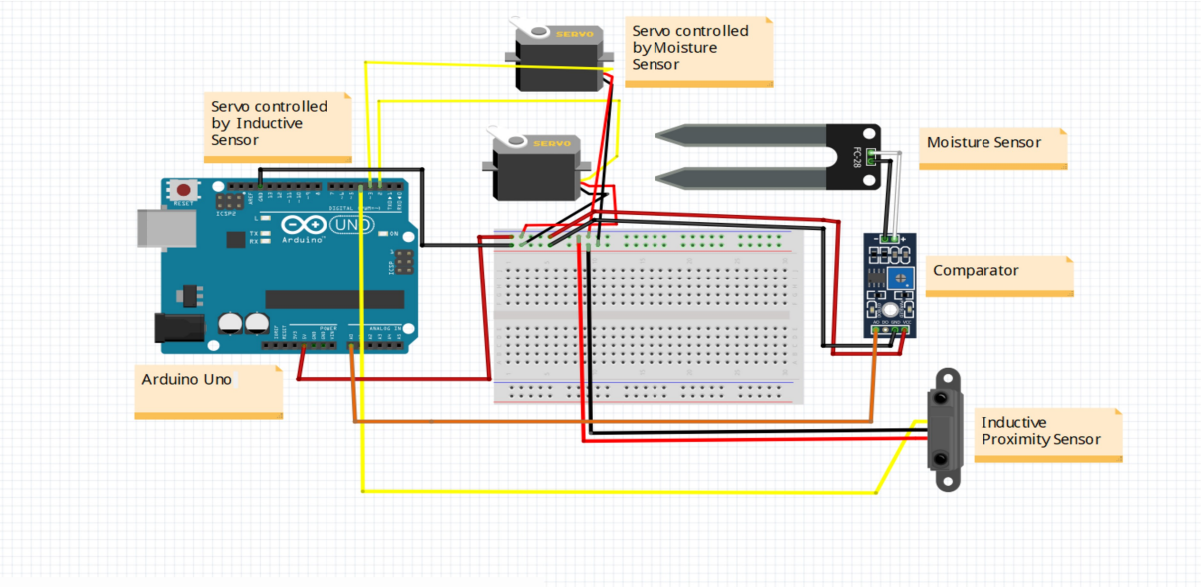

Figure. 1

Circuit diagram for the connections of the various sensors involved (to be mounted on the conveyor belt model)

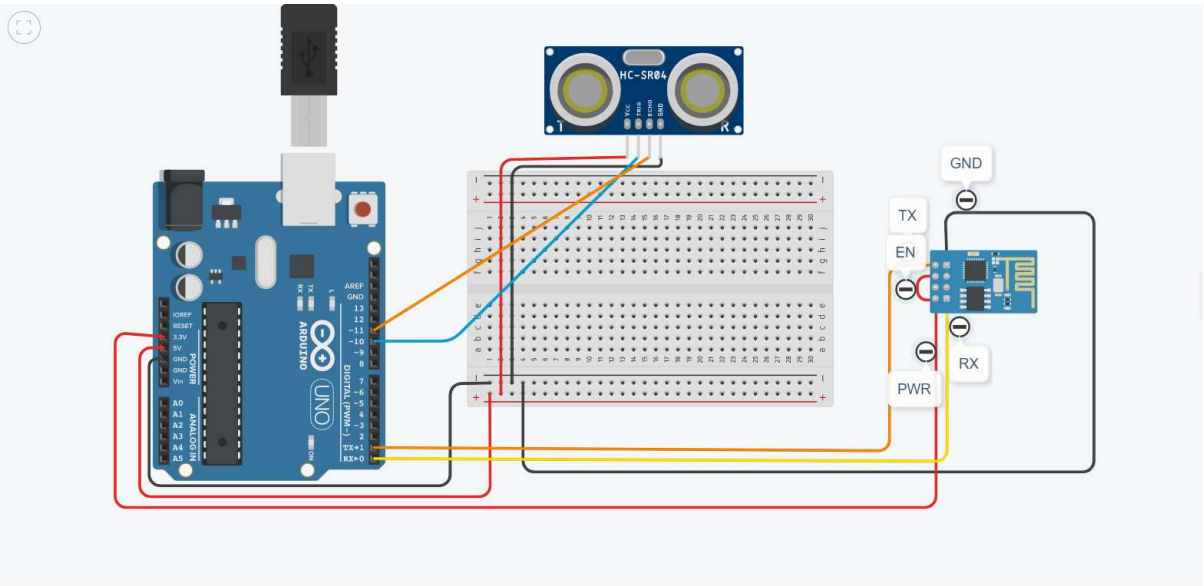

Figure 2

Circuit diagram of the connections for the Utrasonic Sensor and ESP8266 WiFi module. (To be placed under the lid of the Smart Dustbin)

3. HARDWARE SETUP :

1. Connect the various sensors and servos to the Arduino as shown in Figure-1.

2. Mount this connection on to the conveyor belt as per required design.

3. Connect ESP822 WiFi module and Ultrasonic Sensor to the Arduino UNO as shown in Figure 2.

4. Stick this setup under the lid of the dustbin such that the ultrasonic sensor faces the bottom of the dustbin.

5. Connect the conveyor belt setup to the motor.

3.1 Arduino Coding for ESP8266 Setup :

```
#include <SoftwareSerial.h>
const int trigger = 10;
const int echo = 11;
long T;
float distanceCM;
float x;
float y;
#define RX 2
#define TX 3
String AP = "Rohan";       // AP NAME
String PASS = "123456789"; // AP PASSWORD
String API = "XQG9IT4XES8TSMR4";   // Write API KEY
String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;
SoftwareSerial esp8266(RX,TX);


void setup() {
   pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(9600);
```

```
  esp8266.begin(115200);
  sendCommand("AT",5,"OK");
  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");
}

void loop() {
    digitalWrite(trigger, LOW);
  delay(1);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  T = pulseIn(echo, HIGH);
  distanceCM = T * 0.034;
  x=distanceCM;
 y=100-x;



  Serial.print("Percentage full: ");
  Serial.println(y);
  esp8266.print(y);
  delay(1000);
 valSensor = y;
 String getData = "GET /update?api_key="+ API +"&"+ field +"="+String(y);
sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+
PORT,15,"OK");
 sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");
 esp8266.println(getData);delay(1500);countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");
}

void sendCommand(String command, int maxTime, char readReplay[]) {
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
```

```
    found = true;
    break;
   }

  countTimeCommand++;
 }

 if(found == true)
 {
  Serial.println("OK");
  countTrueCommand++;
  countTimeCommand = 0;
 }

 if(found == false)
 {
  Serial.println("Fail");
  countTrueCommand = 0;
  countTimeCommand = 0;
 }

 found = false;
}
```

3.2 <u>Arduino Coding for Segregation sensors setup :</u>

```
#include <Servo.h>
#define sensorPin 3
Servo myservo;
Servo myservo2;
int output_value;
int sensor_pin=A0;
int pos=0;

void setup(){
 pinMode(sensorPin,INPUT_PULLUP);
 myservo.attach(2);
 myservo2.attach(0);
 myservo.writeMicroseconds(1000);
 myservo2.writeMicroseconds(2000);
 Serial.begin(9600);

}
```

```
void loop(){
  int sensed=digitalRead(sensorPin);
  if(sensed==HIGH)
  {
    Serial.println("Sensed");
    myservo2.write(45);
    delay(1000);
    myservo2.write(-20);
    delay(1000);

  }
  else
  {
    Serial.println("+++++");
    //myservo2.write(0);
  }
  output_value=analogRead(sensor_pin);
  output_value=map(output_value,1013,0,0,100);
  //Serial.print("Moisture : ");s
  //Serial.print(output_value);
  //Serial.println("%");
  delay(500);

  //myservo.write(0);
  if(output_value>20){
    myservo.write(45);
    delay(7000);
    myservo.write(-20);
    delay(1000);


  }

}
```

4. SOFTWARE / IoT SETUP :

1. After creating an account in ThingSpeak, under the Channels tab, fill up Name, Description, label the channel and the click SAVE.

2. Under API Keys, copy the key under 'Write API Key' and paste it in the Arduino code.

3. Update the WiFi name and password (to connect to the required WiFi) along with port number and ThingSpeak channel name in the Arduino coding.

4. Log in to your IFTTT account on ifttt.com and under services, choose the Webhooks card and select "Receive a Web request".

5. Complete the trigger fields, give a name and click Create Trigger.

6. Now to create an Action, click on the Notification card and specify to send notification by email to the required email address. An applet is created. (**Figure 3**).

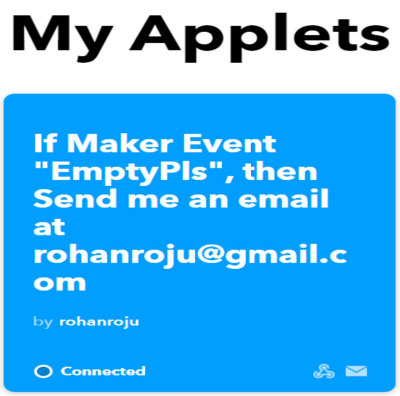

Figure. 3

7. Retrieve the Webhooks trigger information and copy the URL.

8. Go back to the ThingSpeak website and under Apps, choose ThingHTTP.

9. Paste the URL obtained in Step 7 in the field given and Save it (**Figure 4)**

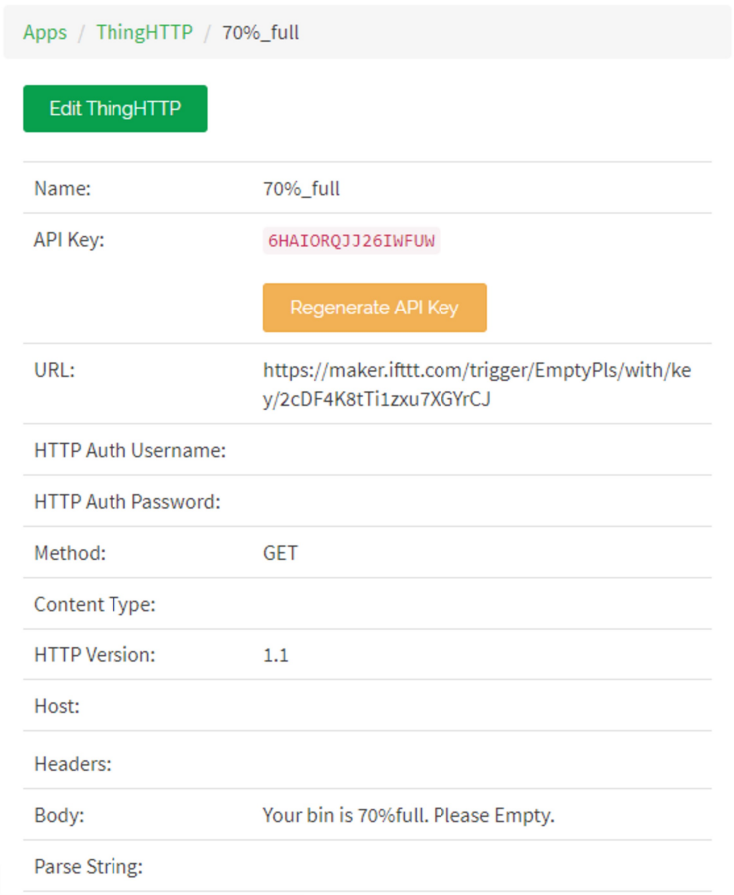

Figure 4

10. Create a React to trigger ThingHTTP based on you channel data. Here you can specify the condition to trigger ThingHTTP if the channel data >70%. This will cause ThingHTTP to communicate with IFTTT via Webhooks and send an email to the user immediately. (**Figure 5)**

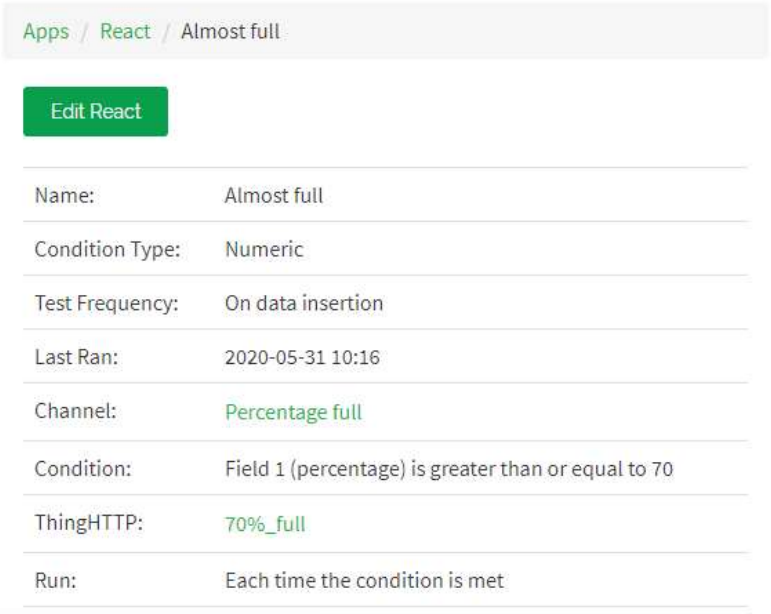

Figure 5

11. In order to calculate the mean, minimum, maximum and range of the data, go to the MATLAB Analysis option under Apps in ThingSpeak. Choose from an available set of mathematical functions and edit the code such that it caters to the data in your channel. Update the channel ID and API Key in the code. (See codes in **RESULT** section of this report)

5. IoT BLOCK DIAGRAM :

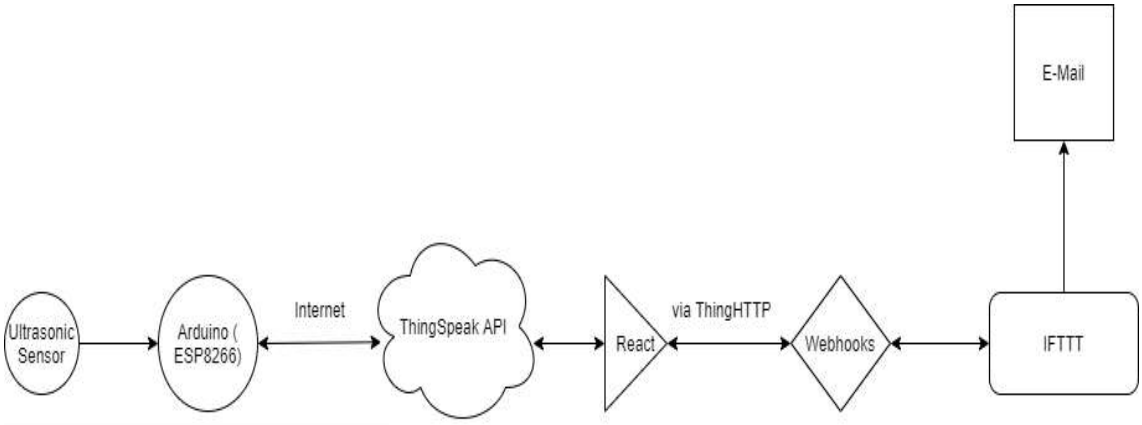

Figure 6 – Smart Dustbin

Thingspeak - ThingSpeak is an open-source Internet of Things application and API to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network.

REACT – An application existing within ThingSpeak which monitors the data obtained by the US Sensor and triggers ThingHTTP to communicate with other websites and services on the Internet.

ThingHTTP – An application existing within ThingSpeak that allows ThingSpeak to communicate with other websites and services on the Internet.

Webhooks – Webhooks is used to alter the beheviour of a webpage or web application with custom callbacks which may be maintained, modified and managed by 3rd party users and developers.

IFTTT – **If This Then That** – is responsible for sending the email notification to the user.

6. WORKING FLOW / PROCESS :

1. Ultrasonic Sensor sends data to ThingSpeak API via the ESP8266 WiFi module.

2. Data is visualized in the home screen of ThingSpeak website.

3. Data flowing into the channel is continuously monitored by **REACT**.

4. When the data >70%, **REACT** will trigger **ThingHTTP** to communicate with the web app responsible for notifying the user (i.e) **IFTTT** via **Webhooks.**

5. Furthermore, **Webhooks** is responsible for altering the behavior of **IFTTT** such that the latter sends an email to the user immediately.

6. An email notification is then sent to the user.

RESULT

Visualization of Dustbin Level on ThingSpeak :

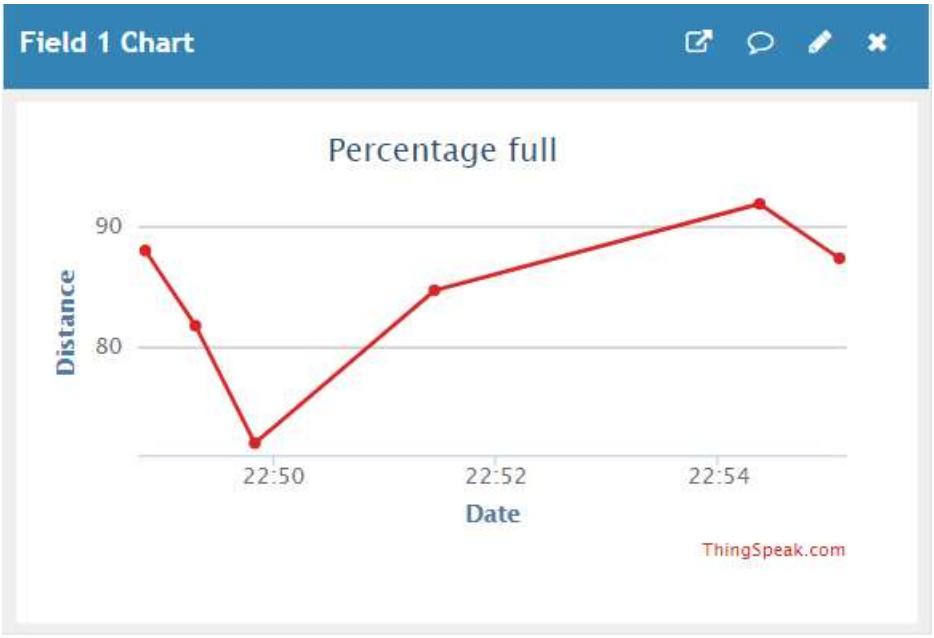

Figure 7

Finding Mean of the data in channel :

```
readChannelID = 1013125;
distanceFieldID = 1;
readAPIKey = 'N13WD4J9BX6QA7VU';
distance =
thingSpeakRead(readChannelID,'Fields',distanceFieldID,'Nu
mMinutes',60,'ReadKey',readAPIKey);
avgdistance = nanmean(distance);
display(avgdistance,'Average Percentage full(in %)');
writeChannelID =1013125;
writeAPIKey = 'XQG9IT4XES8TSMR4';
```

```
Average Percentage full(in %) =

   84.2533
```

Figure 8

Calculating highest, lowest and range of the data :

```
readChannelID = 1013125;
distanceFieldID = 1;
readAPIKey = 'N13WD4J9BX6QA7VU';
[distF,timeStamp] =
thingSpeakRead(readChannelID,'Fields',distanceFieldID, ..
.
'numDays',1,'ReadKey',readAPIKey);
[maxdistF,maxdistIndex] = max(distF);
[mindistF,mindistIndex] = min(distF);
timeMaxdist = timeStamp(maxdistIndex);
timeMindist = timeStamp(mindistIndex);
display(maxdistF,'Highest value of the %full is');
display(mindistF,'Lowest value of % full is');
display(maxdistF-mindistF,'Range of % full is');
writeChannelID = 1013125 ;
writeAPIKey = 'XQG9IT4XES8TSMR4';
```

```
Highest value of the %full is =

   91.9100


Lowest value of % full is =

   71.8800


Range of % full is =

   20.0300
```

Figure 9

E-Mail Notification sent to the user :

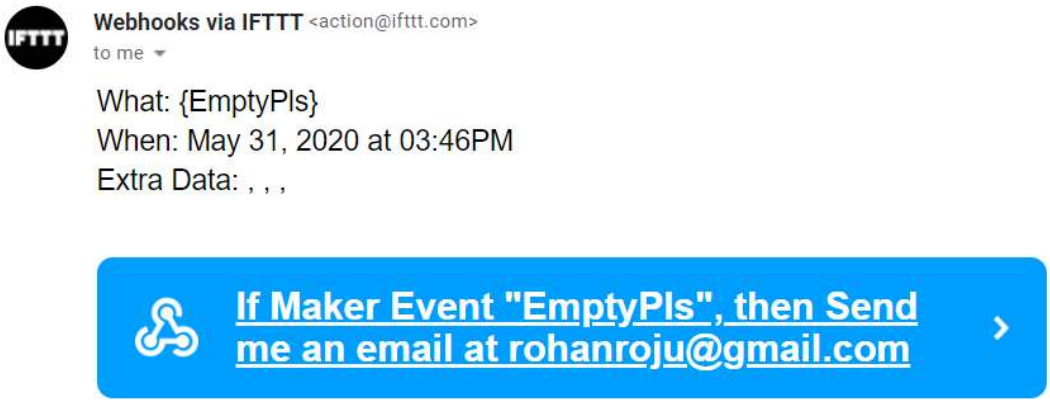

Figure 10

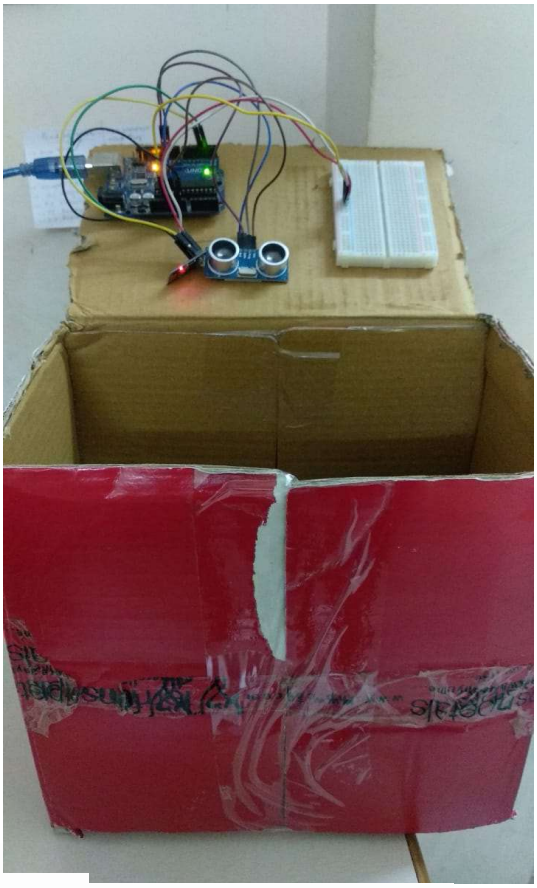

Figure 11 – Smart Dustbin