**ENVIRONMENTAL MANAGEMENT**

**INTRODUCTION :**

**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**

**EXISTING SYSTEM**

**Manual systems in which employees clear the dumpsters periodically**

**No systematic approach towards clearing the dumpsters**

**Unclear about the status of a particular location**

**Employees are unaware of the need for a particular location**

**Very less effective in cleaning city**

**PROPOSED SYSTEM**

**In this system, a 24×7 monitoring system is designed for monitoring dumpsters**

**Here a smart and organized system is designed for selective clearing**

**The ultrasonic sensor is used for measuring the level of waste in the dumpster**

**DC motor powered platform is used for segregating wet and dry waste**

**IR sensor and moisture sensor is used for separating wet and dry waste**

**If either of the containers is full then an alert message is sent from the dumpster**

**In turn, employees can clear the corresponding dumpster**

**All these sensors are connected to an Arduino Uno board**

**It can be used for controlling all mechanical setup based on current conditions**

**Ok**

**BLOCK DIAGRAM DESCRIPTION**

**Ultrasonic sensor Sensors measure distances by using ultrasonic waves. The sensor emits an ultrasonic wave and receives the reflected wave back from the target.**

**IR Sensor emits in order to sense some aspects of the surroundings.**

**Moisture Sensor measures the volumetric water content in the soil. … Reflected microwave radiation is affected by the soil moisture and is used for remote sensing hydrology and agriculture.**

**DC motor which is connected to the digital pins of Arduino**

**We are using the serial monitor for the display**

**HARDWARE REQUIREMENTS**

**Arduino Uno**

**Ultrasonic Sensor**

**IR Sensor**

**Moister Sensor**

**Dc Motor**

**SOFTWARE REQUIREMENTS**

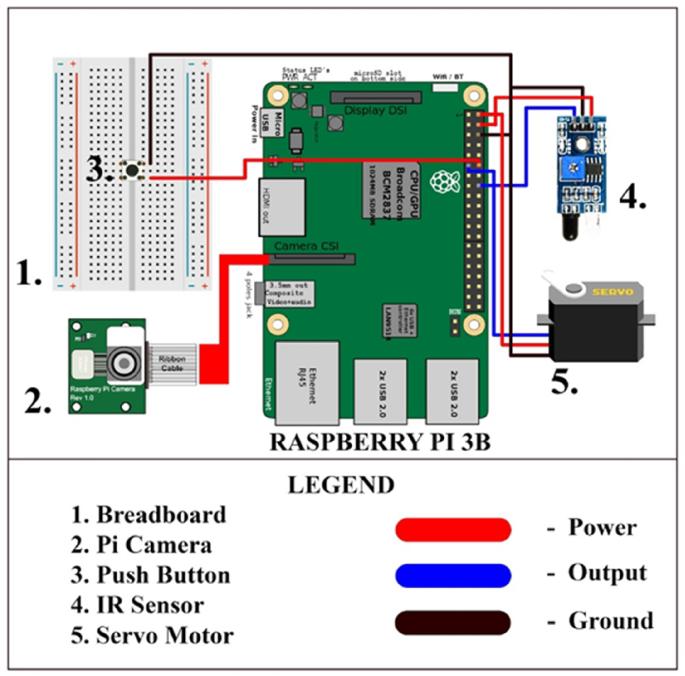
**Arduino IDE**

**The above figure shows the graphical representation of levels of waste in both containers as uploaded to the thingspeak cloud. This page can be accessed by any person who has the username and password of the account.**

**CONCLUSION**

**This project is very effective in managing waste in any big city. Rather than using conventional**

**CIRCUIT DESIGN:**

****

**CODING :**

**Python script**

**Import requests**

**Import json**

**Import ibmiotf.application**

**Import ibmiotf.device**

**Import time**

**Import random**

**Import sys**

**# watson device details**

**Organization = “4yi0vc”**

**devicType = “BIN1”**

**deviceId = “BIN1ID”**

**authMethod= “token”**

**authToken= “123456789”**

**#generate random values for randomo variables (temperature&humidity)**

**Def myCommandCallback(cmd):**

**Global a**

**Print(“command recieved:%s” %cmd.data[‘command’])**

**Control=cmd.data[‘command’]**

**Print(control)**

**Try:**

**deviceOptions={“org”: organization, “type”: devicType,”id”: deviceId,”auth-method”:authMethod,”authtoken”:authToken}**

**deviceCli = ibmiotf.device.Client(deviceOptions)**

**except Exception as e:**

**print(“caught exception 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:’ ‘Dumpster poundage getting high, Time to collect ☺ 90 %’**

**Elif distance < 40 and distance >16:**

**Dist = ‘Risk warning:’ ‘dumpster 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 :’ ‘ Dumpster poundage getting high, Time to collect ☺’**

**Elif load == “60 %” or distance == “60 %”:**

**Warn = ‘alert :’ ‘dumpster is above 60%’**

**Else :**

**Warn = ‘alert :’ ‘No need to collect right now ‘**

**Def myOnPublishCallback(lat=10.678991,long=78.177731):**

**Print(“Gandigramam, Karur”)**

**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**

**Python script**

**Import requests**

**Import json**

**Import ibmiotf.application**

**Import ibmiotf.device**

**Import time**

**Import random**

**Import sys**

**# watson device details**

**Organization = “4yi0vc”**

**devicType = “BIN1”**

**deviceId = “BIN1ID”**

**authMethod= “token”**

**authToken= “123456789”**

**#generate random values for randomo variables (temperature&humidity)**

**Def myCommandCallback(cmd):**

**Global a**

**Print(“command recieved:%s” %cmd.data[‘command’])**

**Control=cmd.data[‘command’]**

**Print(control)**

**Try:**

**deviceOptions={“org”: organization, “type”: devicType,”id”: deviceId,”auth-method”:authMethod,”authtoken”:authToken}**

**deviceCli = ibmiotf.device.Client(deviceOptions)**

**except Exception as e:**

**print(“caught exception 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:’ ‘Dumpster poundage getting high, Time to collect ☺ 90 %’**

**Elif distance < 40 and distance >16:**

**Dist = ‘Risk warning:’ ‘dumpster 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 :’ ‘ Dumpster poundage getting high, Time to collect ☺’**

**Elif load == “60 %” or distance == “60 %”:**

**Warn = ‘alert :’ ‘dumpster is above 60%’**

**Else :**

**Warn = ‘alert :’ ‘No need to collect right now ‘**

**Def myOnPublishCallback(lat=10.678991,long=78.177731):**

**Print(“Gandigramam, Karur”)**

**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”)**

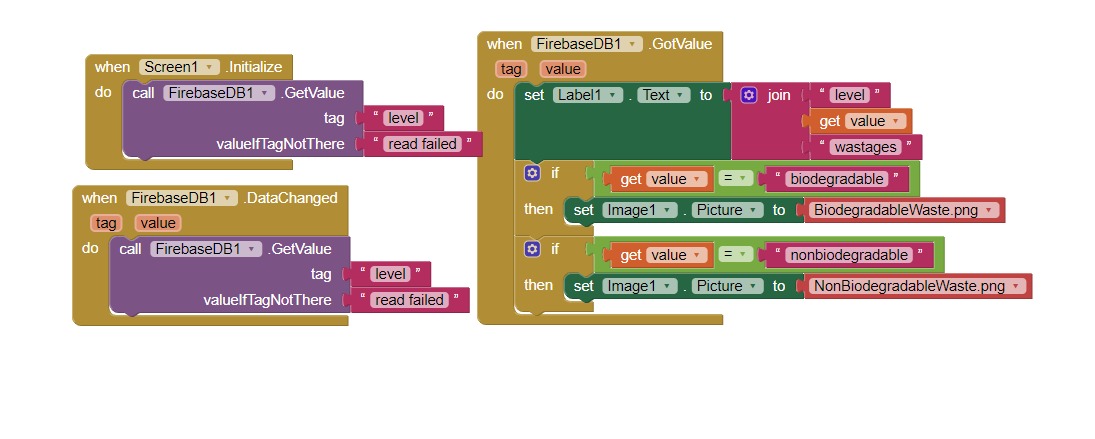
**Time.sleep(30)**

**deviceCli.commandCallback=myCommandCallback**

**#disconnect the device**

**deviceCli.disconnect**

**APP BLOCK DESIGN:**

****