TechTrunk Ventures Pvt. Ltd

# Project Name:

**SMART WASTE MANGEMENT SYSTEM**

## GUIDED BY:

Ashish Pandey

## SUBMITTED BY:

* Sriram Setty Sanmukha(17ece121)
* Subudhi Anvesh(17ece123)
* V.Kartik Kumar Patro(17it062)
* K.Likhit Kumar(17cse183)
* K.Abhishek

## INTRODUCTION:

Things that are connected to Internet and sometimes these devices can be controlled from the internet is commonly called as Internet of Things. The Internet of Things (IoT) is a concept in which surrounding objects are connected through wired and wireless networks without user intervention. In the field of IoT, the objects communicate and exchange information to provide advanced intelligent services for users. Owing to the recent advances in mobile devices equipped with various sensors and communication modules, together with communication network technologies such as Wi-Fi and LTE, the IoT has gained considerable academic interests.

Owing to the characteristics and merits of IoT services, waste management has also become a significant issue in academia, industry, and government as major IoT application fields. An indiscriminate and illegal discharge of waste, an absence of waste disposal and management systems, and inefficient waste management policies have caused serious environmental problems and have incurred considerable costs for waste disposal.

In our system, the Smart dust bins are connected to the internet to get the real time information of the smart dustbins. In the recent years, there was a rapid growth in population which leads to more waste disposal. So a proper waste management system is necessary to avoid spreading some deadly diseases. Managing the smart bins by monitoring the status of it and accordingly taking the decision. This waste is further picked up by the municipal corporations to finally dump it in dumping areas and landfills.

## OBJECTIVES:

In this proposed system, waste is managed in two different waste accordingly i.e. dry and wet. In this system the speed of the network does not matter as MQTT is a very light weight protocol which will work with low internet speed also with a high speed transfer of data. Automation i.e. machine to machine communication without any involvement of the human.

The dustbin indication is done by 3 different LED’s.

If the dustbin is full then LED will be on and this information is directly sent to described office’s.

## COMPONENTS USED:

1.Raspberry Pi 3 B+Model

2.Ultrasonic Sensor

3.NodeMCU

4.LED’s

## COMPONENTS DESCRIPTION ALONG WITH IMAGES

Ultrasonic Sensor:

**ultrasonic sensors** are a type of acoustic sensor divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signal into ultrasound , receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.

In a similar way to radar and sonar , ultrasonic transducer are used in systems which evaluate targets by interpreting the reflected signals. For example, by measuring the time between sending a signal and receiving an echo the distance of an object can be calculated. Passive ultrasonic sensors are basically microphones that detect ultrasonic noise that is present under certain conditions.

It consists of 4 different pins: VCC, ground, Trigger pin, Echo pin.

Ultrasonic sensor trig signals and when it find any object it reflect back the signal and receives through the echo pin.

The distance is measured by the formula:

distance = (duration\*.0343)/2;



Raspberry pi 3 B+ model:

The latest Raspberry Pi 3 Model B+ has a faster 64-bit 1.4GHz quad core processor, 1GB of RAM, faster dual-band 802.11 b/g/n/ac wireless LAN, Bluetooth 4.2, and significantly faster 300Mbit/s ethernet.

* 1.4GHz 64-bit quad-core ARM Cortex-A53 CPU (BCM2837)
* 1GB RAM (LPDDR2 SDRAM)
* On-board wireless LAN - dual-band 802.11 b/g/n/ac (CYW43455)
* On-board Bluetooth 4.2 HS low-energy (BLE) (CYW43455)
* 4 x USB 2.0 ports
* 300Mbit/s ethernet
* 40 GPIO pins
* Full size HDMI 1.3a port
* Combined 3.5mm analog audio and composite video jack
* Camera interface (CSI)
* Display interface (DSI)
* microSD slot
* VideoCore IV multimedia/3D graphics core @ 400MHz/300MHz
* It consists of 40 GPIO pins.
* Improved thermals on the Pi 3 B+ means that the CPU on the BCM2837 SoC can now run at 1.4GHz, a 17% increase on the previous Pi 3 model (which ran at 1.2GHz)
* The Pi 3 B+ has significantly faster wired networking, thanks to an upgraded USB/LAN chip, and you should see speeds that are 3-5x faster than on previous models of the Pi, at least 300Mbit/s.



NodeMCU:

NodeMCU is an open-source firmware and development kit that helps you to prototype or build IoT product. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266.

• Voltage:3.3V.

• Wi-Fi Direct (P2P), soft-AP.

• Current consumption: 10uA~170mA.

• Flash memory attachable: 16MB max (512K normal).

• Integrated TCP/IP protocol stack.

• Processor: Tensilica L106 32-bit.

• Processor speed: 80~160MHz.

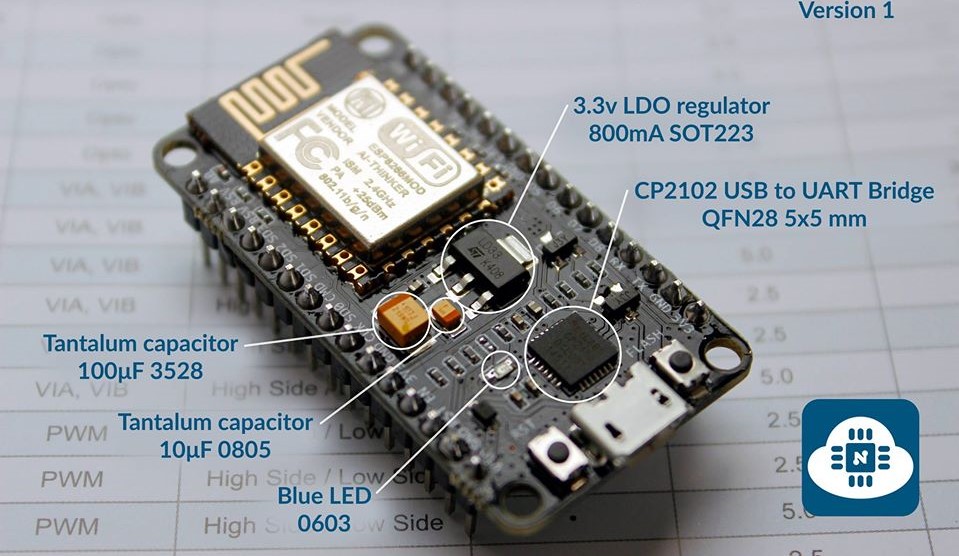
• RAM: 32K + 80K.

• GPIOs: 17 (multiplexed with other functions).

• Analog to Digital: 1 input with 1024 step resolution.

• +19.5dBm output power in 802.11b mode

We usually use Arduino IDE to execute the nodeMCU programs. This makes it very convenient to use the ESP8266 chip as we will be using the well-known Arduino IDE.



## LED’S:

A light-emitting diode is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

Here we use 3 different LED’s.They are RED,GREEN,BLUE.

RED led indicates that the dustbin is full.

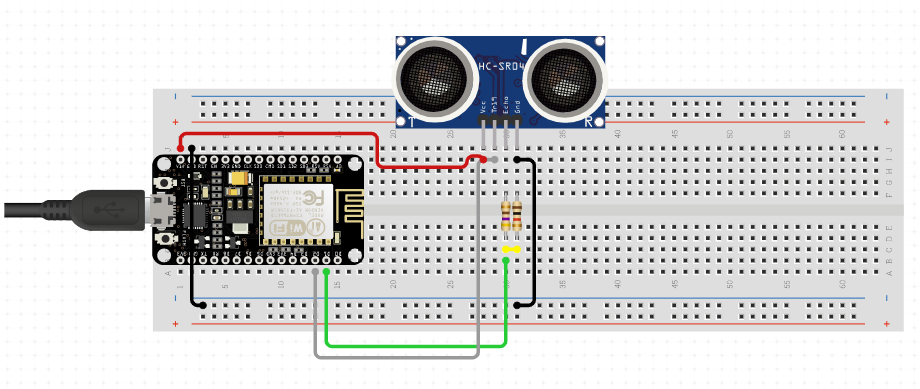
GREEN led indicates that the dustbin is half filled.

BLUE led indicates that the dustbin is empty.

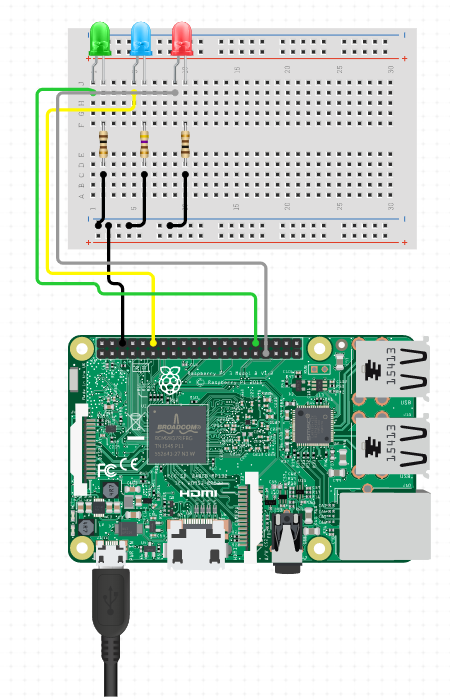
 

## Features

Publisher part



Subsriber part



Setup:



Coding of NodeMCU:

#include <UltrasonicSensor.h>

#include <ESP8266WiFi.h>

#include "Adafruit\_MQTT.h"

#include "Adafruit\_MQTT\_Client.h"

/\*\*\*\* WiFi Access Point \*\*\*\*\*/

#define WLAN\_SSID "2nd floor"

#define WLAN\_PASS "8500351132"

/\*\*\*\*Adafruit.io Setup \*\*\*\*/

#define AIO\_SERVER "io.adafruit.com"

#define AIO\_SERVERPORT 1883 // use 8883 for SSL

#define AIO\_USERNAME "Pawan\_Raj"

#define AIO\_KEY "f7e59e61d15940a6823170821eb45bb2"

/\*\*\*\*\*\*\*\*\*\* Global State (you don't need to change this!) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.

WiFiClient client;

// or... use WiFiFlientSecure for SSL

//WiFiClientSecure client;

const int trigPin = D4;

const int echoPin = D3;

long duration;

int distance;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.

Adafruit\_MQTT\_Client mqtt(&client, AIO\_SERVER, AIO\_SERVERPORT, AIO\_USERNAME, AIO\_KEY);

// Setup a feed called 'potValue' for publishing.

// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>

Adafruit\_MQTT\_Publish potValue = Adafruit\_MQTT\_Publish(&mqtt, AIO\_USERNAME "/feeds/potValue");

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration

// for some reason (only affects ESP8266, likely an arduino-builder bug).

void MQTT\_connect();

uint16\_t potAdcValue = 0;

uint16\_t ledBrightValue = 0;

void setup() {

Serial.begin(9600);

delay(10);

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

Serial.println(F("Adafruit MQTT demo"));

Serial.print("Connecting to ");

Serial.println(WLAN\_SSID);

WiFi.begin(WLAN\_SSID, WLAN\_PASS);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println();

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

// Setup MQTT subscription for ledBrightness feed.

}

void loop() {

digitalWrite(trigPin, LOW); // Clears the trigPin

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);// Sets the trigPin on HIGH state for 10 micro seconds

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

long duration = pulseIn(echoPin, HIGH);

float distance= duration\*0.034/2;// Calculating the distance

delay(1500);

Serial.print("Distance: ");// Prints the distance on the Serial Monitor

delay(50);

Serial.println(distance);

// Ensure the connection to the MQTT server is alive (this will make the first

// connection and automatically reconnect when disconnected). See the MQTT\_connect

// function definition further below.

MQTT\_connect();

float d=distance;

if(d<7)

{

potValue.publish("FULL"); //dustbin is full

}

if(d>7 && d<18)

{

potValue.publish("HALF FILLED"); //dustbin is hallfilled

}

if(d>18)

{

potValue.publish("EMPTY"); //dustbin is empty

}

}

// Function to connect and reconnect as necessary to the MQTT server.

// Should be called in the loop function and it will take care if connecting.

void MQTT\_connect() {

int8\_t ret;

// Stop if already connected.

if (mqtt.connected()) {

return;

}

Serial.print("Connecting to MQTT... ");

uint8\_t retries = 3;

while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected

Serial.println(mqtt.connectErrorString(ret));

Serial.println("Retrying MQTT connection in 5 seconds...");

mqtt.disconnect();

delay(5000); // wait 5 seconds

retries--;

if (retries == 0) {

// basically die and wait for WDT to reset me

while (1);

}

}

Serial.println("MQTT Connected!");

}

Coding of Raspberry Pi:

import sys

import RPi.GPIO as a

import ssl

import time

# Import standard python modules.

# Import Adafruit IO MQTT client

from Adafruit\_IO import MQTTClient

a.setmode(a.BOARD);

a.setwarnings(False);

a.setup(8,a.OUT);

a.setup(10,a.OUT);

a.setup(12,a.OUT);

# Set to your Adafruit IO key.

# Remember, your key is a secret,

# so make sure not to publish it when you publish this code!

ADAFRUIT\_IO\_KEY = 'f7e59e61d15940a6823170821eb45bb2'

# Set to your Adafruit IO username.

# (go to https://accounts.adafruit.com to find your username)

ADAFRUIT\_IO\_USERNAME = 'Pawan\_Raj'

# Set to the ID of the feed to subscribe to for updates.

FEED\_ID = 'potValue'

# Define callback functions which will be called when certain events happen.

def connected(client):

# Connected function will be called when the client is connected to Adafruit IO.

# This is a good place to subscribe to feed changes. The client parameter

# passed to this function is the Adafruit IO MQTT client so you can make

# calls against it easily.

#print('Connected to Adafruit IO! Listening for {0} changes...'.format(FEED\_ID))

# Subscribe to changes on a feed named DemoFeed.

client.subscribe(FEED\_ID)

def disconnected(client):

# Disconnected function will be called when the client disconnects.

print('Disconnected from Adafruit IO!')

sys.exit(1)

def message(client, feed\_id,payload):

#print('Feed {0} received new value: {1}'.format(feed\_id, payload))

print(payload)

if payload=="FULL":

a.output(8,a.HIGH)

a.output(10,a.LOW)

a.output(12,a.LOW)

elif payload=="HALF FILLED":

a.output(10,a.HIGH)

a.output(12,a.LOW)

a.output(8,a.LOW)

elif payload=="EMPTY":

a.output(12,a.HIGH)

a.output(10,a.LOW)

a.output(8,a.LOW)

# Create an MQTT client instance.

client = MQTTClient(ADAFRUIT\_IO\_USERNAME, ADAFRUIT\_IO\_KEY)

# Setup the callback functions defined above.

client.on\_connect = connected

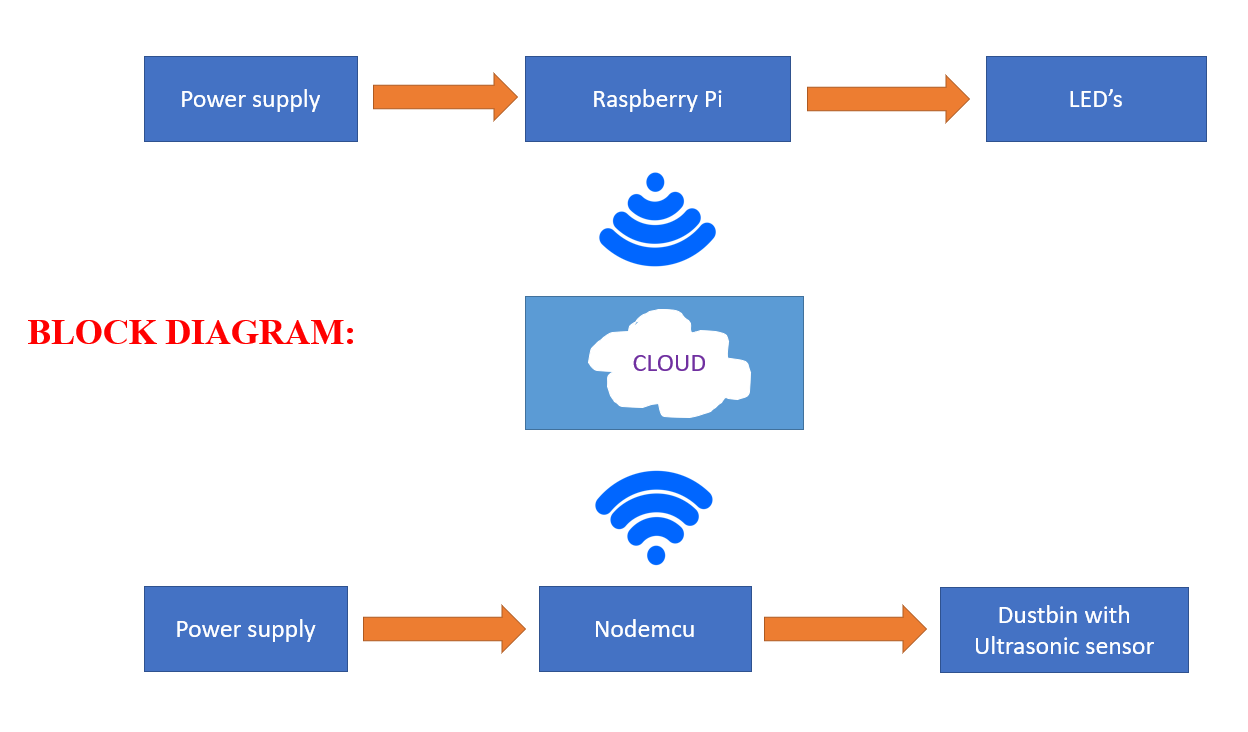
client.on\_disconnect = disconnected

client.on\_message = message

# Connect to the Adafruit IO server.

client.connect()

client.loop\_blocking()



Conculsion:

We have implemented real time waste management system by using dustbin to check the fill level of smart dustbin whether the dustbin are filled or not.In this system the information off all smart dustbin can be accesed from anywhere and anytime by the concern person and he can take decision accordingly.By implementing the proposed system the reductiin,resource optimisation,effective usage of smart dustbin can be done.This system indirectly reducing traffic in the city.In major cities the bgarbage collection vehicles visit the everydaytwice or thrice depends on the population of the particular area and sometimes these dustbinmay not be full.Our sustem will inform the status of the dustbinin real time so that the concern authority can send the garbage collection vehicles onlywhen the dustbin is full.The scope for the future work is the system can be implemented with the time stamp in which real time clock shown to the concern personat what time dustbin is full and at what time the waste is collected from the smart dustbin.