A Mini-Project Report

on

**GARBAGE MONITORING AND COLLECTION ASSISTANCE**

Submitted for partial fulfillment of the requirements for the award of the degree

of

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

**Varaganti Manikanta (2451-17-733-123)**

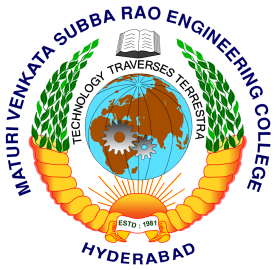
Under the guidance of

**B.Saritha**

Assistant Professor

Department of CSE

M.V.S.R.E.C., Hyderabad.

****

Department of Computer Science and Engineering

M.V.S.R. ENGINEERING COLLEGE

(Affiliated to Osmania University & Recognized by AICTE)

Nadergul, Saroor Nagar Mandal, Hyderabad – 501 510

2018-19.

**M.V.S.R. ENGINEERING COLLEGE**

(Affiliated to Osmania University, Hyderabad)

Nadergul(P.O.), Hyderabad-501510



**Certificate**

This is to certify that the mini-project work entitled “Garbage Monitoring and Collection Assistance**”** is a bonafide work carried out by Mr. V.Manikanta(2451-17-733-123) in partial fulfillment of the requirements for the award of degree of **BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING** from M.V.S.R. Engineering College, affiliated to OSMANIA UNIVERSITY, Hyderabad, under our guidance and supervision.

The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide Faculty Coordinator

**B.Saritha D.Sirisha**

Assistant Professor Assistant Professor

Department of CSE Department of CSE

MVSREC, Hyderabad. MVSREC, Hyderabad.

**DECLARATION**

This is to certify that the work reported in the present mini-project entitled **“Garbage Monitoring And Collection Assistance”** is a record of bonafide work done by me in the Department of Computer Science and Engineering, M.V.S.R. Engineering College, Osmania University. The reports are based on the mini-project work done entirely by us and not copied from any other source.

The results embodied in this mini-project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of my knowledge and belief.

Varaganti Manikanta

(2451-17-733-123)

**ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude and indebtedness to my mini-project guide **B.Saritha** for her valuable suggestions and interest throughout the course of this mini-project.

I am also thankful to our principal **Dr.G.KANAKA DURGA** and **Dr.AKHIL KHARE**, Professor and Head, Department of Computer Science and Engineering, MVSR Engineering College, Hyderabad for providing excellent infrastructure for completing this mini-project successfully as a part of our B.E. Degree (CSE). I would like to thank my mini-project coordinator **D**.**Sirisha ,B.Janaiah** for their constant monitoring, guidance and support.

I convey my heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Finally, I would like to take this opportunity to thank my family for their support through the work. I sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Varganti Manikanta (2451-17-733-123)

**ABSTRACT**

Many times, in our city we see that the garbage bins or dustbins placed at Public places are overloaded. It creates unhygienic conditions for people as well as ugliness to that place leaving bad odour. To avoid all such situations we are going to implement a project called Garbage Monitoring and Collection Assistance. These dustbins are interfaced with Ultrasonic sensor along with central system showing current status of garbage, on Thingspeak server and displays the GPS coordinates on Blynk app. Whenever the garbage reaches a threshold value it send the notification to concerned person using IFTTT. The main aim of this project is to reduce human resources and efforts along with the enhancement of a SMART CITY vision.

**TABLE OF CONTENTS**

PAGE NOS.

Certificate…………………………………………………………………………………………………… i

Declaration……………………………………………………………………………………………….. ii

Acknowledgements…………………………………………………………………………………… iii

Abstract............................................................................................................ iv

Table of contents............................................................................................... v

List of Figures ................................................................................................. vi

List of Tables .....................................................................................................vii

CONTENTS INSIDE THE DOCUMENT

CHAPTER I

1. INTRODUCTION 09– 11
   1. PROBLEM STATEMENT 09

1.2 PROPOSED SYSTEM 10

1.3 SCOPE OF THE MINI-PROJECT 11

CHAPTER II

2. TOOLS AND TECHNOLOGIES 12 – 27

* 1. LITERATURE SURVEY 12
  2. HARDWARE REQUIREMENTS 13
  3. SOFTWARE REQUIREMENTS 23

CHAPTER III 28-35

3. SYSTEM DESIGN 28

* 1. FLOW CHARTS 34
  2. SYSTEM ARCHITECTURE

CHAPTER IV

4. SYSTEM IMPLEMENTATION & METHODOLOGIES 36-51

4.1 SYSTEM IMPLEMENTATION 36 4.2 SOURCE CODE 46

CHAPTER V

5. SYSTEM TESTING 51-53

5.1 TEST APPROACH 51

5.2 FEATURES TO BE TESTED 51

5.3 EXPECTED OUTPUT 52

CHAPTER VI

6. CONCLUSION & REFERENCES 54- 56

6.1 CONCLUSION 54

6.2 REFERENCES 55

APPENDIX

STEPS TO VERIFIED SOFTWARE

REMAINING CODE

**LIST OF FIGURES**

|  |
| --- |
| **FIGURE NO. NAME OF FIGURE PAGE NO.**  Figure 1.2 Use of present technology 10  Figure 2.2.1.1 Ultrasonic sensor 13  Figure 2.2.1.2 Ultrasonic working 14  Figure 2.2.1.3 Distance calculation using ultrasonic 15  Figure 2.2.2.1 NodeMCU 16  Figure 2.2.2.2 Pin diagram of NodeMCU 17  Figure 2.2.3.1 Neo 6M GPS Module 19  Figure 2.2.3.2 GPS block diagram 20  Figure 2.3.1 Arduino IDE 23  Figure 2.3.2 IoT 24  Figure 2.3.3 Blynk Network 25  Figure 2.3.4 IFTTT 27  Figure 3.1 Flowchart 28  Figure 3.2 System Architecture 29  Figure 4.1.1 Circuit Diagram 30  Figure 4.1.2.3.1 IFTTT logo 34  Figure 4.1.2.3.2 Applet in IFTTT 34  Figure 4.1.2.3.3 Creating trigger in IFTTT 34  Figure 4.1.2.3.4 Action service in IFTTT 35  Figure 4.1.2.3.5 Android SMS action 35  Figure 4.1.2.3.6 Creating action 35  Figure 4.1.2.3.7 VoIP in IFTTT 36  Figure 4.1.2.3.8 Creating call action 36  Figure 4.1.2.4.1 Creating blynk account 37  Figure 4.1.2.4.2 Creating new project in blynk 37  Figure 4.1.2.4.3 Selecting board in blynk 38  Figure 4.1.2.4.4 Selecting widgets in blynk 39  Figure 4.1.2.4.5 Output shown in blynk 39  Figure 5.3.1 Output shown in ThingSpeak 46  Figure 5.3.2 Notifying through call/message 47  Figure 5.3.3 Location shown in blynk 47 |
|  |

**LIST OF TABLES**

|  |
| --- |
| **Table Name Page no**  Table 2.2.1 pin configuration of ultrasonic sensor 13  Table 2.2.3.1 Gps performance 20  Table 2.2.3.4 Boot-time configuration 21  Table 2.2.4 protocols and Interfaces 21  Table 2.2.5 pin configuration of gps module 22 |

**CHAPTER I**

**INTRODUCTION**

We are living in an age where tasks and systems are fusing together with the power of IOT to have a more efficient system of working and to execute jobs quickly! With all the power at our fingertips this is what we have come up with.The Internet of Things (IoT) shall be able to incorporate transparently and seamlessly a large number of different systems, while providing data for millions of people to use and capitalize. Building a general architecture for the IoT is hence a very complex task, mainly because of the extremely large variety of devices, link layer technologies, and services that may be involved in such a system. One of the main concerns with our environment has been solid waste management which impacts the health and environment of our society. The detection, monitoring and management of wastes is one of the primary problems of the present era. The traditional way of manually monitoring the wastes in waste bins is a cumbersome process and utilizes more human effort, time and cost which can easily be avoided with our present technologies. This is our solution, a method in which waste management is automated. This is our Garbage Monitoring and Collection Assistance system, an innovative way that will help to keep the cities clean and healthy.

**1.1 The Problem**

India generates 62 million tonnes of waste every year, of which less than 60% is collected and around 15% processed. With landfills ranking third in terms of greenhouse gas emissions in India, and increasing pressure from the public, the Government of India revised the Solid Waste Management after 16 years.Nowadays, there are tons of flats and apartments which have been built in the rapid urbanization area. This is due to high housing demands which have been drastically risen as a result of migration from villages to cities to find work. In order to accommodate the growing population in the urban area, the government has also constructed more apartment complexes. There are several issues faced by the residents of the flats. One of them is disposal of solid waste. Unlike private houses, the residents of all the apartments use a common dustbin, which tends to fill up very quickly. This overflowing of garbage is a sanitary issue which might cause diseases like cholera and dengue. Moreover it is a waste of fuel to travel around a complex or an area to find that some of the garbage are filled and some are not. Also, on rare days, problems might arise that there is so much garbage that the truck doesn’t have enough capacity. The idea struck us when we observed that the garbage truck use to go around the town to collect solid waste twice a day. Although this system was thorough it was very inefficient. For example let's say street A is a busy street and we see that the garbage fills up really fast whereas maybe street B even after two days the bin isn't even half full.

**1.2 Proposed System**

Garbage is a waste generated due to the various activities, such as industry waste, wet waste like vegetable waste, dry waste, commercial waste, household wastes etc. While carrying the garbage, it must be carried and disposed by following various types of protocols.The processing of garbage involves the carrying the garbage from one place to other through trucks and cranes.Nowadays the waste collection is inefficiently performed using static routes and schedules. A hardware or any other means must be installed in the place of garbage. The sensors are going to deploy in the garbage. The sensors must be low cost, less power consumption sensors. With those sensors, sense the level of garbage and send the same information to server and it must be stored in the cloud. Also information must be displayed in the web server accordingly. An SMS has to be sent to the authorized person and also the GPS coordinates of the trash can.



Fig 1.2 use of present technology

**1.3 Future Scope**

The main objective is to maintain the level of cleanliness in the city and form an environment which is better for living. By using this system we can constantly check the level of the garbage in the dustbins which are placed in various parts of the city. If a particular dustbin has reached the maximum level then the employees can be informed and they can immediately take certain actions to empty it as soon as possible. The employees can check the status of these bins anytime on their mobile phones. This can prove to be a very useful system if used properly. The system can be used as a benchmark by the people who are willing to take one step further for increasing the cleanliness in their respected areas. Ultrasonic sensor is being used in this system to check the level of garbage in the dustbins but in future various other types of sensors can be used with the ultrasonic sensor to get more precise output and to take this system to another level. Now this system can be used in certain areas but as soon as it proves its credibility it can be used in all the big areas. As this system also reduces manual work certain changes can be done in the system to take it to another level and make it more useful for the employees and people who are using it. In future, a team can be made which will be in charge for handling and maintaining this system and also to take care of its maintenances.Moreover this system can be used to segregate the dry and wet waste in future by modifying the design.

**CHAPTER II**

**TOOLS AND TECHNOLOGIES**

**2.1 Literature Survey**

Maher arebey [1] proposed an integrated system Solid waste management is a big challenge in urban areas for most of the countries throughout the world. An efficient waste management is a prerequisite for maintaining a safe and green environment as there are increasing all kinds of waste disposal. In his paper an integrated system is introduced which comprises of Radio Frequency Identification (RFID), Global Position System (GPS),

General Packet Radio Service (GPRS), Geographic Information System (GIS) and web camera.The RFID is used for the purpose of verification, the GSM module is used for the purpose of communication. This system uses web cameras for taking the snapshots of the garbage bins at regular intervals of time.

Waikhom reshmi [2] proposed an electronic system, in order to provide a solution to irregular waste disposal system. The designed system makes use of biosensor sensor, weight sensor and height sensor to detect overflow of the waste in the dust bin. The technology which are suggested in this paper, achieved effective waste management system where sensors unit are used for sensing, microcontroller for controlling and for communication they have used GSM module. The LCD display is used to display the current condition of the waste.

B. Vijayalaxmi [3] proposed a smart alert system for garbage clearance by giving an alert signal to the municipal web server for instant cleaning of dustbin with proper verification based on level of garbage filling. This process is aided by the ultrasonic sensor which is interfaced with Arduino UNO to check the level of garbage filled in the dustbin and sends the alert to the municipal web server once the bin is filled. After cleaning the dustbin, the driver confirms the task of emptying the garbage with the aid of RFID Tag. RFID is a computing technology that is used for verification process and in addition, it also enhances the smart garbage alert system by providing automatic identification of garbage filled in the dustbin and sends the status of cleanup to the server affirming that the work is done. The whole process is upheld by an embedded module integrated with RFID and IOT Facilitation.

Andrei Borozdukhin [4] proposed a solution of the optimization garbage removal problem in the large cities is suggested. In this paper, system architecture is defined to find time-optimal dynamic route for garbage trucks within “Smart Clean City” project. This system makes use of infrared and ultrasound sensor for inspection of fullness. It makes use of GSM module for communication. After 70% fill level, the system decides to remove garbage from it. In this system the sensor is located at the side of the container.

**2.2 Hardware Requirements**

**2.2.1 Ultrasonic Sensor HC SR04**

### 

### Fig 2.2.1.1 ultrasonic sensor

### Ultrasonic Sensor Pin Configuration

|  |  |  |
| --- | --- | --- |
| Pin Number | Pin Name | Description |
| 1 1 | Vcc Vcc | Th The Vcc pin powers the sensor, typically with +5V |
| 2 2 | Tri Trigger | Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave. |
| 3 3 | Echo Echo | Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor. |
| 4 4 | Gnd | This pin is connected to the Ground of the system |

### 

### Table 2.2.1 pin configuration of ultrasonic sensor

### HC-SR04 Sensor Features

* Operating voltage: +5V
* Theoretical Measuring Distance: 2cm to 450cm
* Practical Measuring Distance: 2cm to 80cm
* Accuracy: 3mm
* Measuring angle covered: <15°
* Operating Current: <15mA
* Operating Frequency: 40Hz

### HC-SR04 Ultrasonic Sensor - Working

As shown above the HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below



Fig 2.2.1.2 ultrasonic working

Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

### How to use the HC-SR04 Ultrasonic Sensor

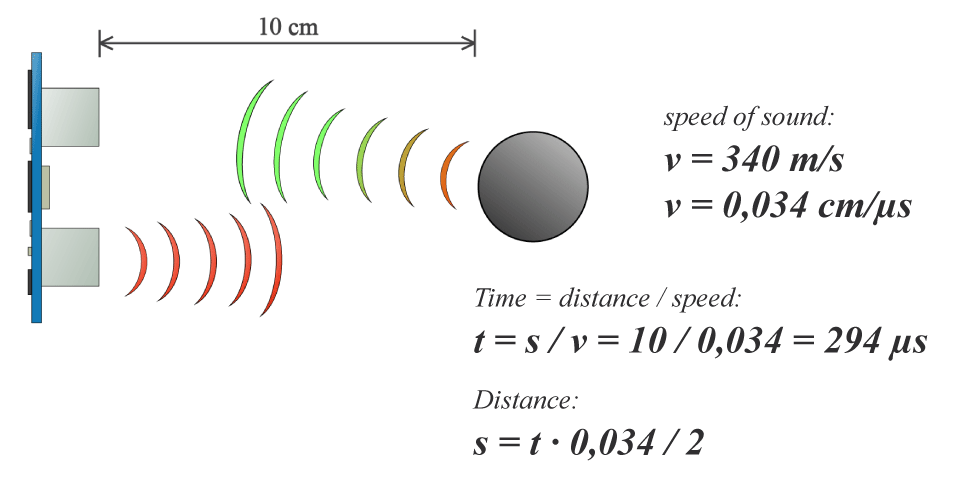


Fig 2.2.1.3 Distance calculation using ultrasonic data

HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used.

Power the Sensor using a regulated +5V through the Vcc and Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.

The amount of time during which the Echo pin stays high is measured by the MCU/MPU as it gives the information about the time taken for the wave to return back to the Sensor. Using this information the distance is measured as explained in the above heading.

### Applications

* Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
* Used to measure the distance within a wide range of 2cm to 400cm
* Can be used to map the objects surrounding the sensor by rotating it
* Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water

**2.2.2 NodeMCU**

NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board. Since NodeMCU is open source platform, their hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The **ESP8266** is a low-cost [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer [ESP8266 WiFi Module](http://www.electronicwings.com/sensors-modules/esp8266-wifi-module). There is Version2 (V2) available for NodeMCU Dev Kit i.e. **NodeMCU Development Board v1.0 (Version2)**, which usually comes in black colored PCB.



Fig 2.2.2.1 NodeMCU

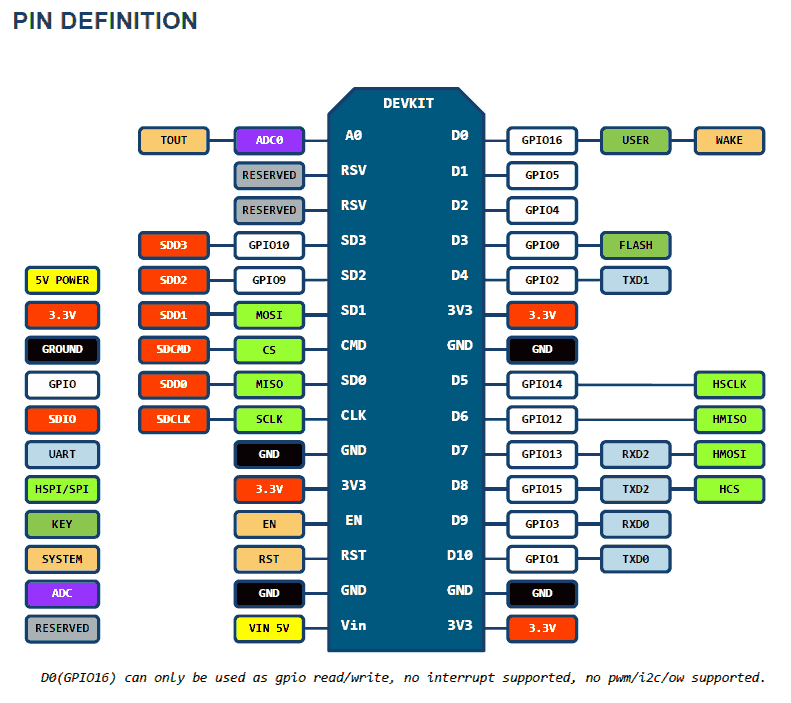
NodeMCU Dev Kit has Arduino like Analog (i.e. A0) and Digital (D0-D8) pins on its board. It supports serial communication protocols i.e. UART, SPI, I2C etc. Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc

Fig 2.2.2.2 Pin diagram of NodeMCU

# NodeMCU Dev Kit v1.0 pin descriptions

GPIO (General Purpose Input Output) Pins:

NodeMCU has general purpose input output pins on its board as shown in above pinout diagram. We can make it digital high/low and control things like LED or switch on it. Also, we can generate PWM signal on these GPIO pins.

ADC (Analog to Digital Converter) channel (A0) :

NodeMCU has one ADC channel/pin on its board.

SPI (Serial Peripheral Interface) Pins :

NodeMCU based ESP8266 has Hardware SPI (HSPI) with four pins available for SPI communication. It also has SPI pins for Quad-SPI communication. With this SPI interface, we can connect any SPI enabled device with NodeMCU and make communication possible with it.

I2C (Inter-Integrated Circuit) Pins :

NodeMCU has I2C functionality support on ESP8266 GPIO pins. Due to internal functionality on ESP-12E we cannot use all its GPIOs for I2C functionality. So, do tests before using any GPIO for I2C applications.

UART (Universal Asynchronous Receiver Transmitter) Pins :

NodeMCU based ESP8266 has two UART interfaces, UART0 and UART1. Since UART0 (RXD0 & TXD0) is used to upload firmware/codes to board, we can’t use them in applications while uploading firmware/codes.

**2.2.3. NEO 6M GPS module**

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints. The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments.



Fig 2.2.3.1 Neo 6M GPS Module

**2.2.3.1 GPS Performance**

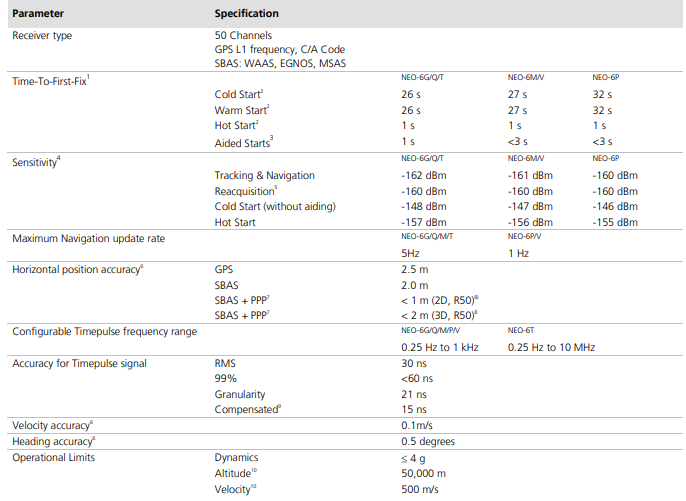
****

Table 2.2.3.1 Gps performance

**2.2.3.2 Block Diagram**

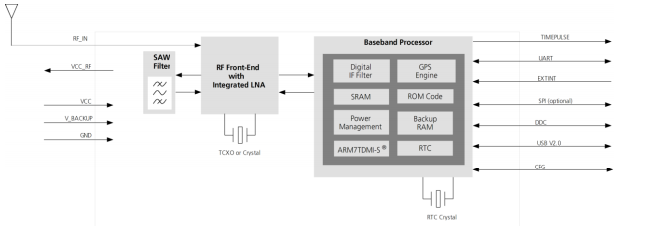
****

Fig 2.2.3.2 GPS block diagram

**2.2.3.3 Assisted GPS(A-GPS) :**

Supply of aiding information like ephemeris, almanac, rough last position and time and satellite status and an optional time synchronization signal will reduce time to first fix significantly and improve the acquisition sensitivity. All NEO-6 modules support the u-blox AssistNow Online and AssistNow Offline A-GPS services and are OMA SUPL compliant.

**2.2.3.4 Boot-time configuration** :

NEO-6 modules provide configuration pins for boot-time configuration. These become effective immediately after start-up. Once the module has started, the configuration settings can be modified with UBX configuration messages. The modified settings remain effective until power-down or reset. If these settings have been stored in battery-backup RAM, then the modified configuration will be retained, as long as the backup battery supply is not interrupted. NEO-6 modules include both CFG\_COM0 and CFG\_COM1 pins and can be configured as seen in Table 6.

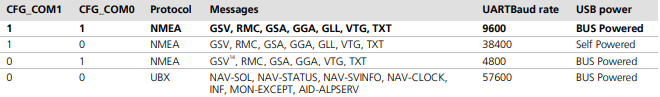


Table 2.2.3.4 Boot-time configuration

**2.2.3.5 Precise Point positioning :**

u-blox’ industry proven PPP algorithm provides extremely high levels of position accuracy in static and slow moving applications, and makes the NEO-6P an ideal solution for a variety of high precision applications such as surveying, mapping, marine, agriculture or leisure activities. Ionospheric corrections such as those received from local SBAS12 geostationary satellites (WAAS, EGNOS, MSAS) or from GPS enable the highest positioning accuracy with the PPP algorithm. The maximum improvement of positioning accuracy is reached with PPP+SBAS and can only be expected in an environment with unobstructed sky view during a period in the order of minutes.

**2.2.3.6 Oscillators :**

NEO-6 GPS modules are available in Crystal and TCXO versions. The TCXO allows accelerated weak signal acquisition, enabling faster start and reacquisition times.

**2.2.4 Protocols and Interfaces :**

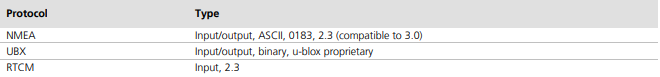


Table 2.2.4 protocols and Interfaces

**2.2.4.1 UART :**

NEO-6 modules include one configurable UART interface for serial communication.

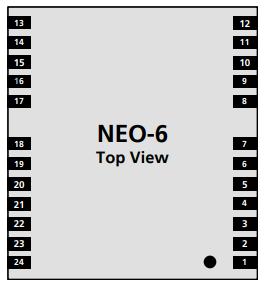
**2.2.4.2 USB:**

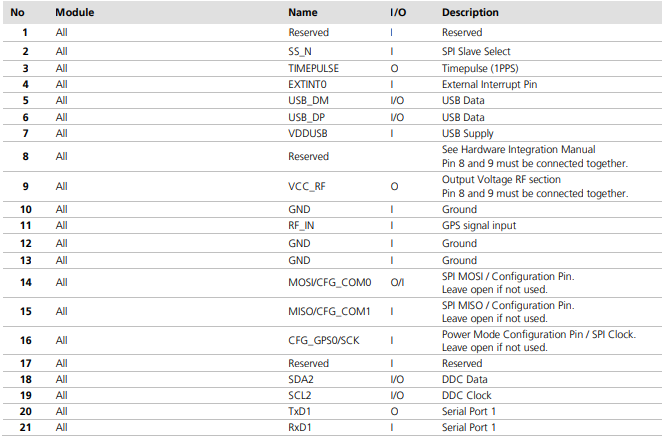
NEO-6 modules provide a USB version 2.0 FS (Full Speed, 12Mbit/s) interface as an alternative to the UART. The pull-up resistor on USB\_DP is integrated to signal a full-speed device to the host. The VDDUSB pin supplies the USB interface. u-blox provides a Microsoft® certified USB driver for Windows XP, Windows Vista and Windows 7 operating systems.

**2.2.4.3 Serial Peripheral Interface (SPI) :**

The SPI interface allows for the connection of external devices with a serial interface, e.g. serial flash to save configuration and AssistNow Offline A-GPS data or to interface to a host CPU. The interface can be operated in master or slave mode. In master mode, one chip select signal is available to select external slaves. In slave mode a single chip select signal enables communication with the host.

**2.2.5 Pin Assignment :**

****

****

****

Table 2.2.5 pin configuration of gps module

**2.3 Software Requirements**

**2.3.1 Arduino IDE**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.Software written using Arduino are called sketches. These sketches are written in the text editor. Sketches are saved with the file extension .ino. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom right-hand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

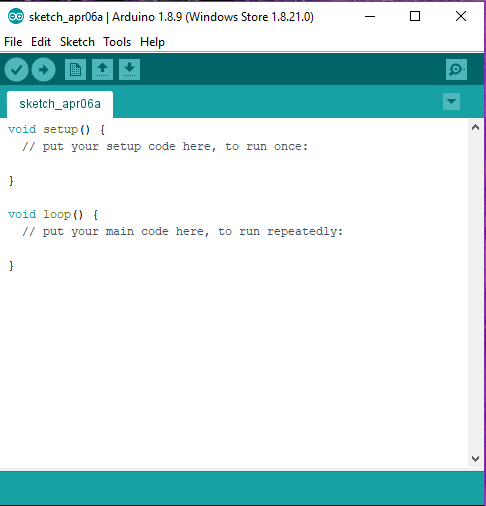
****

Fig 2.3.1 Arduino IDE

**2.3.2 ThingSpeak**

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak™ from your devices, create instant visualizations of live data, and send alerts using web services like Twitter® and Twilio®. With MATLAB® analytics inside ThingSpeak, you can write and execute MATLAB code to perform preprocessing, visualizations, and analyses. ThingSpeak enables engineers and scientists to prototype and build IoT systems without setting up servers or developing web

software.

**What is IoT ?**

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

At a high level, many IoT systems can be described using the diagram below:

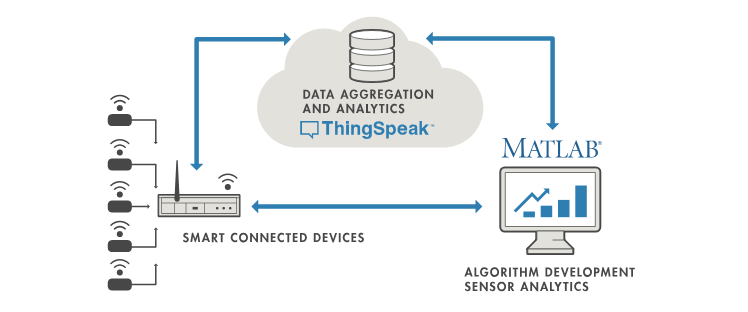
****

Fig 2.3.2 Iot

On the left, we have the smart devices (the “things” in IoT) that live at the edge of the network. These devices collect data and include things like wearable devices, wireless temperatures sensors, heart rate monitors, and hydraulic pressure sensors, and machines on the factory floor.

In the middle, we have the cloud where data from many sources is aggregated and analyzed in real time, often by an IoT analytics platform designed for this purpose.

The right side of the diagram depicts the algorithm development associated with the IoT application. Here an engineer or data scientist tries to gain insight into the collected data by performing historical analysis on the data. In this case, the data is pulled from the IoT platform into a desktop software environment to enable the engineer or scientist to prototype algorithms that may eventually execute in the cloud or on the smart device itself.

An IoT system includes all these elements. ThingSpeak fits in the cloud part of the diagram and provides a platform to quickly collect and analyze data from internet connected sensors.

ThingSpeak Key Features

ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:

* Easily configure devices to send data to ThingSpeak using popular IoT protocols.
* Visualize your sensor data in real-time.
* Aggregate data on-demand from third-party sources.
* Use the power of MATLAB to make sense of your IoT data.
* Run your IoT analytics automatically based on schedules or events.
* Prototype and build IoT systems without setting up servers or developing web software.
* Automatically act on your data and communicate using third-party services like Twilio® or Twitter®.

**2.3.3 Blynk app**

Blynk is a Platform with iOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet.It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets.Blynk is not tied to some specific board or shield. Instead, it's supporting hardware of your choice. Whether your Arduino or Raspberry Pi is linked to the Internet over Wifi, Ethernet or this new ESP8266 chip, Blynk will get you online and ready for the Internet Of Your Things. Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other cool things.

There are three major components in the platform:

Blynk App - allows to you create amazing interfaces for your projects using various

widgets we provide.

Blynk Server - responsible for all the communications between the smart phone and

hardware. You can use our Blynk Cloud or run your private Blynk server locally. It’s

open-source, could easily handle thousands of devices and can even be launched on a

Raspberry Pi.

Blynk Libraries - for all the popular hardware platforms - enable communication with

the server and process all the incoming and out coming commands.

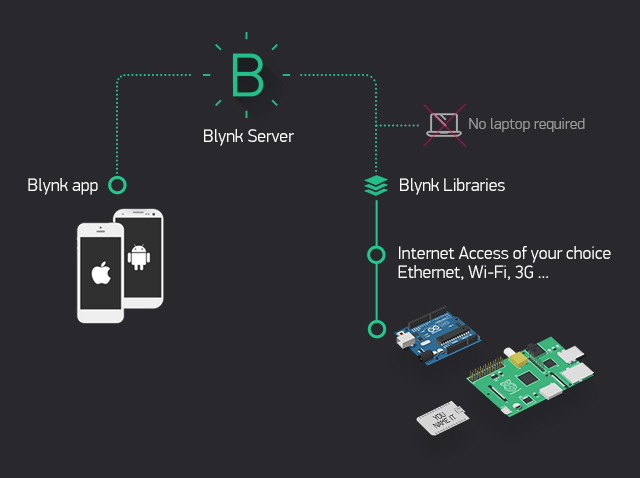


Fig 2.3.3 Blynk network

Now imagine: every time you press a Button in the Blynk app, the message travels to space the Blynk Cloud, where it magically finds its way to your hardware. It works the same in the opposite direction and everything happens in a blynk of an eye.

**2.3.4 IFTTT**

If This Then That, also known as IFTTT (/ɪft/), is a free web-based service to create chains of simple conditional statements, called applets. An applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, or Pinterest. For example, an applet may send an e-mail message if the user tweets using a hashtag, or copy a photo on Facebook to a user's archive if someone tags a user in a photo. In addition to the web-based application, the service runs on iOS and Android. In February 2015, IFTTT renamed its original application to IF, and released a new suite of apps called Do, which lets users create shortcut applications and actions. As of 2015, IFTTT users created about 20 million recipes each day. All of the functionalities of the Do suite of apps have since been integrated into a redesigned IFTTT app. IFTTT is an initialism for "If This Then That.”[.](https://en.wikipedia.org/wiki/IFTTT#cite_note-ifttt_wtf-9).

IFTTT employs the following concepts:

* Services (formerly known as channels) are the basic building blocks of IFTTT. They mainly describe a series of data from a certain web service such as YouTube or eBay. Services can also describe actions controlled with certain APIs, like SMS. Sometimes, they can represent information in terms of weather or stocks. Each service has a particular set of triggers and actions.
* Triggers are the "this" part of an applet. They are the items that trigger the action. For example, from an RSS feed, you can receive a notification based on a keyword or phrase.
* Actions are the "that" part of an applet. They are the output that results from the input of the trigger.
* Applets (formerly known as recipes) are the predicates made from Triggers and Actions. For example, if you like a picture on Instagram (trigger), an IFTTT app can send the photo to your Dropbox account (action).
* Ingredients are basic data available from a trigger—from the email trigger, for example; subject, body, attachment, received date, and sender’s address.

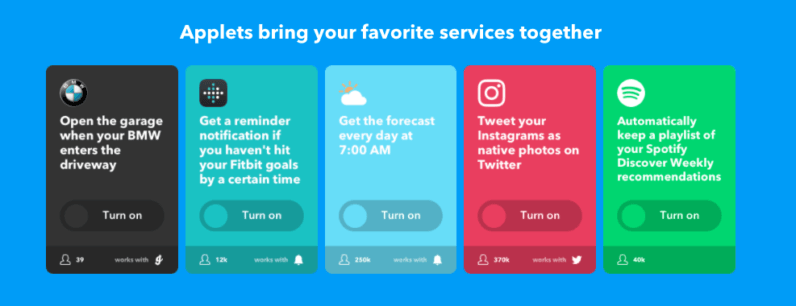
****

Fig 2.3.4 IFTTT

**CHAPTER III**

**SYSTEM DESIGN**

In this chapter we design the system that helps visualization. The design is done with the help of UML(Unified Modeling Language).

**3.1 Flowchart**

What our system does is it gives a real time indicator of the garbage level in a trash can at any given time and also provides the location of the bin.The below chart explains how this project works and how it’s control flows.

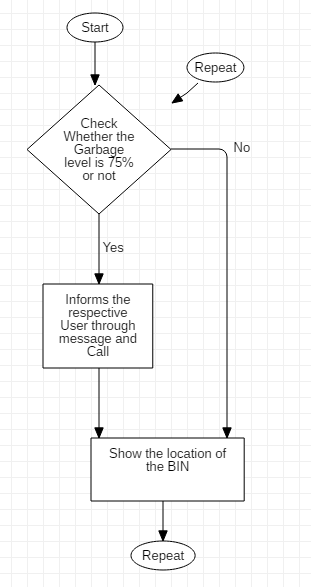


Fig 3.1 Flowchart

**3.1.1 Description**

An Ultrasonic Sensor is used for detecting whether the trash can is filled with garbage or not. Here Ultrasonic Sensor is installed at the top of Trash Can and will measure the distance of garbage from the top of Trash can and we set a threshold value according to the size of trash can. If the distance is more than this threshold value, means that the Trash can is full of garbage and we will be notified accordingly.

The data collected from the sensor is transferred to our Server,in our case it is Thingspeak. We are using this server because it is open source and user-friendly.But it has the limitation that the data can be sent to the server with a minimum time delay of 15 seconds.The location of the bin is displayed in the Blynk app.

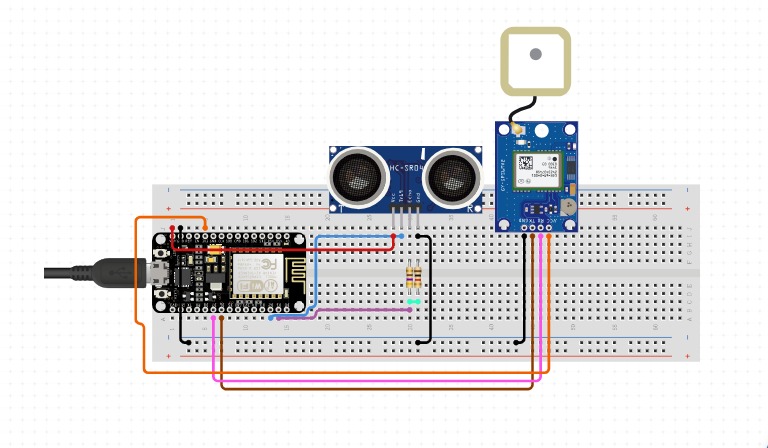
**3.2 System Architecture **

Fig 3.2 System Architecture

**CHAPTER IV**

**SYSTEM IMPLEMENTATION AND METHODOLOGIES**

**4.1 System implementation**

**4.1.1 Hardware implementation**

Connections of the ultrasonic sensor with the NodeMCU are very simple. Connect the VCC and the ground of the ultrasonic sensor to the 5V and the ground of the NodeMCU respectively . Then connect the TRIG and ECHO pin of ultrasonic sensor to the digital pin 4(D4) and digital pin 5(D5) of the NodeMCU respectively(you can use any other pin as well).Connect the D2 pin of the NodeMCU with the TX pin of the GPS module and the D1 pin of the NodeMCU with the RX pin of the GPS module. Connect the GND of the GPS module to the ground of the NodeMCU and connect the VCC of the GPS module to the 3.3V pin of the NodeMCU .

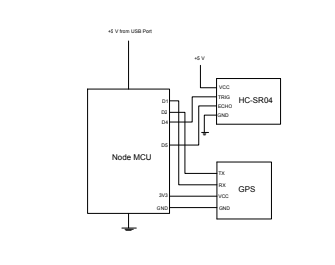


Fig 4.1.1 Circuit diagram

**4.1.2 Software implementation**

**4.1.2.1 Setting up Arduino IDE**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board.

Visit the website <https://arduino.cc> to install the arduino IDE for Mac OS and Windows.

**4.1.2.1.1 Adding libraries**

The following libraries will be used for our Arduino code:

ESP8266 Library:

Navigate to File -> Preferences (Ctrl +, on Windows OS);Add the following URL to Additional Boards Manager textbox (the one on the bottom of the Preferences window):

*<http://arduino.esp8266.com/stable/package_esp8266com_index.json>*

Navigate for *Tools -> Board -> Boards Manager* for adding our ESP8266 board.  
Hit "Ok" button and close the Preferences Window.  
Type "ESP8266" on the search text box, select "esp8266 by ESP8266 Community" and install it.Now our Arduino IDE will be ready to work with a lot of ESP8266 based development boards, like the generic ESP8266, NodeMCU 1.0, etc.

select “NodeMCU 1.0(ESP-12E Module)” for the Project involving NodeMCU board.

TinyGPS++ Library:

TinyGPS++ is a new Arduino library for parsing NMEA data streams provided by GPS modules.Like its predecessor, TinyGPS, this library provides compact and easy-to-use methods for extracting position, date, time, altitude, speed, and course from consumer GPS devices. However, TinyGPS++’s programmer interface is considerably simpler to use than TinyGPS, and the new library can extract arbitrary data from any of the myriad NMEA sentences out there, even proprietary ones.

Visit the link to install TinyGPS++ library:

*https://github.com/mikalhart/TinyGPSPlus/releases/tag/v1.0.2*

Blynk Library:

### Blynk is the most popular Internet of Things platform for connecting any hardware to the cloud, designing apps to control them, and managing your deployed products at scale.

* With Blynk Library you can connect over 400 hardware models (including ESP8266, ESP32, NodeMCU, all Arduinos, Raspberry Pi, Particle, Texas Instruments, etc.)to the Blynk Cloud.
* With Blynk apps for iOS and Android apps you can easily drag-n-drop graphic interfaces for any DIY or commercial project. No coding on iOS or Android required.
* Hardware can connect to Blynk Cloud (open-source server) over the Internet using hardware connectivity available on your board (like ESP32), or with the use of various shields (Ethernet, WiFi, GSM, LTE, etc). Blynk Cloud is available for every user of Blynk for free. Direct connection over Bluetooth is also possible.

Visit the link to install Blynk Library:

*https://github.com/blynkkk/blynk-library.git*

Onewire Library:

Visit the below link to add the “Onewire” library in Arduino IDE

[*https://github.com/PaulStoffregen/OneWire.git*](https://github.com/PaulStoffregen/OneWire.git)

PubSubClient Library:

PubSubClient Library is used to implement the HTTP protocol.

Visit the below link to add the “PubSubClient” library in Arduino IDE.

<https://github.com/knolleary/pubsubclient.git>

**4.1.2.2 Setting up ThingSpeak**

[ThingSpeak](https://thingspeak.com/) is an [IoT analytics](https://www.mathworks.com/solutions/internet-of-things.html) platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualizations of live data, and send alerts using web services like Twitter and Twilio. With MATLAB® analytics inside ThingSpeak, you can write and execute MATLAB code to perform preprocessing, visualizations, and analyses. ThingSpeak enables engineers and scientists to prototype and build IoT systems without setting up servers or developing web software.

Configuring Accounts and Channels:

To read and write to a ThingSpeak channel, your application sends requests to the ThingSpeak server by issuing HTTP requests, publishing MQTT messages, or using MATLAB® functions. Each ThingSpeak channel can have up to eight fields of 255 characters of data, in either numeric or alphanumeric format. A channel also has location information and a status update field. Each channel data entry is stored with a date and timestamp. You can retrieve stored data by time or by entry ID.Use the ThingSpeak API to process numeric data, which includes timescaling, averaging, median, summing, and rounding. You can create and update a ThingSpeak channel by posting a feed with your API key and data using HTTP POST. The channel feeds support JSON, XML, and CSV formats for integration into applications.

Create MathWorks account in the website:

[*https://in.mathworks.com*](https://in.mathworks.com)

Using mathworks Credentials sign up for the ThingSpeak account.

Login with your credentials and create a new channel.

Provide the channel details like Name of channel,Fields name,etc and save the channel.

Now copy the write API key in API keys and paste it in the code.

Act on the Data:

Now go to the Apps and open React

React works with ThingHTTP, ThingTweet, and MATLAB Analysis apps to perform actions when channel data meets a certain condition. For example, you can have a mobile app report your latitude and longitude to a ThingSpeak channel. When your position is within a certain distance of your house, have ThingHTTP turn on your living room lights.

To create a new reaction, click New React.

React Settings

* React Name: Enter a unique name for your React.
* Condition Type: Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
* Test Frequency: Choose whether to test your condition every time data enters the channel or on a periodic basis.
* Condition: Select a channel, a field and the condition for your React.
* Action: Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
* Options: Select when the React runs.

After Creating the React, save the React.

ThingHTTP enables communication among devices, websites, and web services without having to implement the protocol on the device level. You specify actions in ThingHTTP, which you trigger using other ThingSpeak apps such as [TweetControl](https://thingspeak.com/apps/tweetcontrol),[TimeControl](https://thingspeak.com/apps/timecontrols), and [React](https://thingspeak.com/apps/reacts).

Click New ThingHTTP to create a new HTTP request.

In ThingHTTP URL, paste the URL of IFTTT Webhooks Documentation link and save the ThingHTTP.

**4.1.2.3 Setting Up IFTTT**

IFTTT is a free platform that helps you connects apps and devices. You can use it to connect your smartphone with other gadgets, or to share data between your favourite web services like Google, Facebook, Twitter, Instagram, etc. and other physical devices, for instance. And the best part is that it’s really easy to use IFTTT uses a “if this then that” logic, where “this” represents a service that will trigger a given action given by “that”. This way you create small applets connecting web services and devices.

First you’ll have to sign in at: [*https://ifttt.com/*](https://ifttt.com/)



Fig 4.1.2.3.1 IFTTT Logo

On the website, navigate to New Applet (click the arrow button next to our login to access the menu).

Now click on the *+this* and select the service webhook

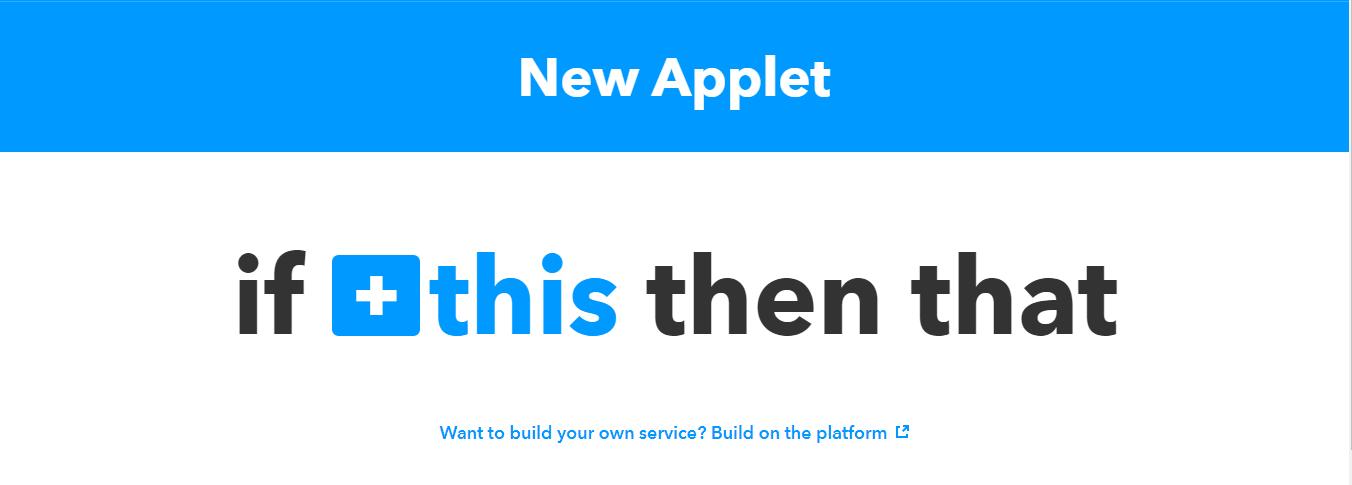


Fig 4.1.2.3.2 Applet in IFTTT

Create a webrequest and give the name of the event as Garbage and create Trigger.

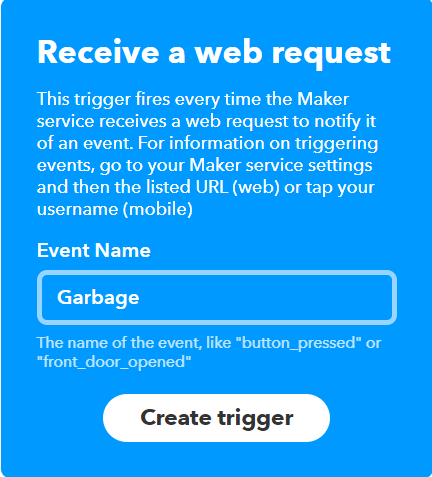


Fig 4.1.2.3.3 Creating trigger in IFTTT

Now click on the *+that*  and select the Android sms



Fig 4.1.2.3.4 Action service in IFTTT

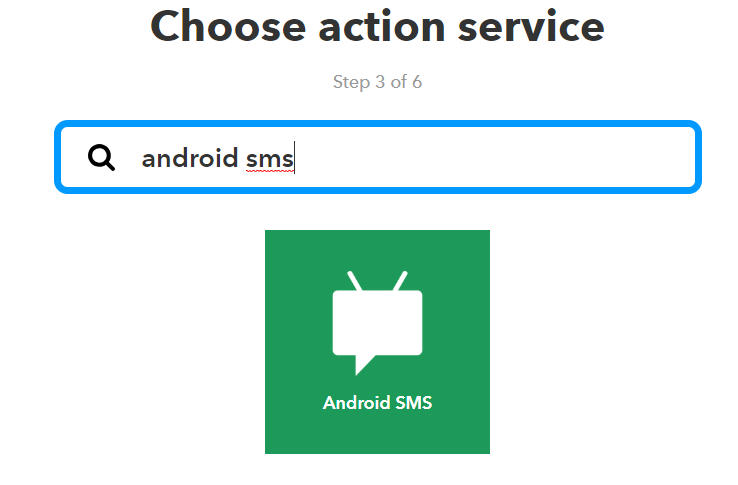


Fig 4.1.2.3.5 Android sms action

Provide details as shown in the figure and create action.

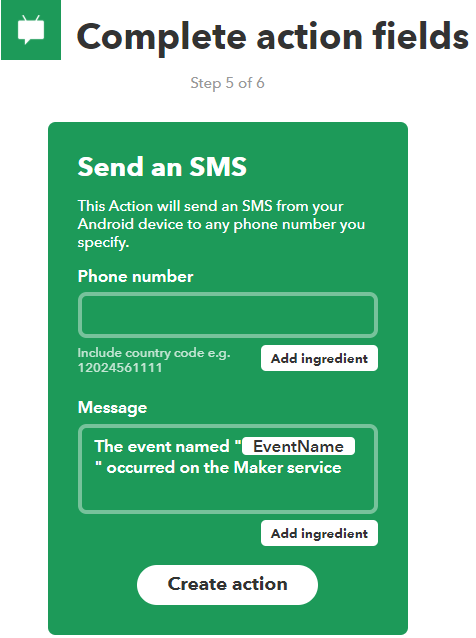


Fig 4.1.2.3.6 Creating action

click on the Finish button and applet will be created.

For making calls use VoIP Voice Over Internet Provider as the *+that*

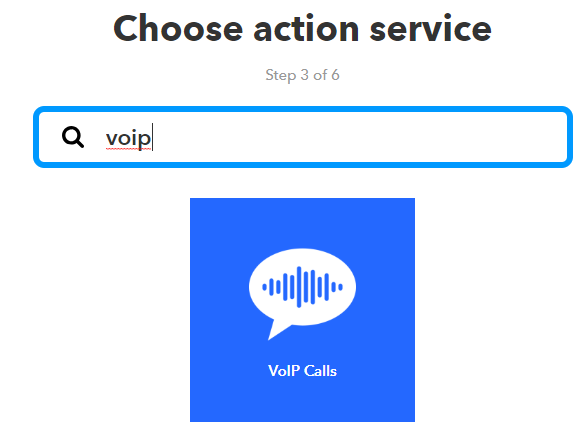


Fig 4.1.2.3.7 VOIP in IFTTT

Now provide the message which will be played on attending the call and create action.

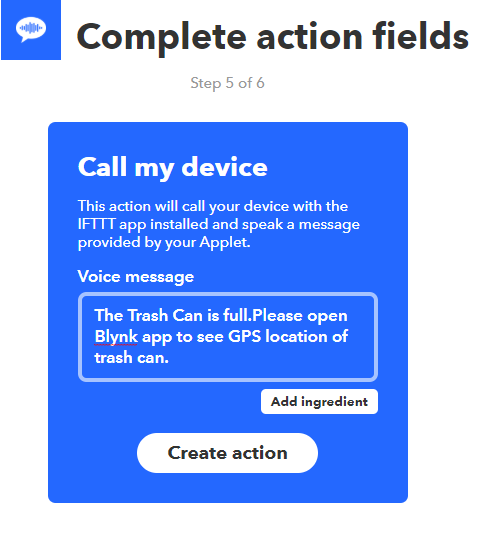


Fig 4.1.2.3.8 Creating call action

**4.1.2.4 Setting up Blynk app**

* Create a Blynk Account.

After you download the Blynk App, you'll need to create a New Blynk account.

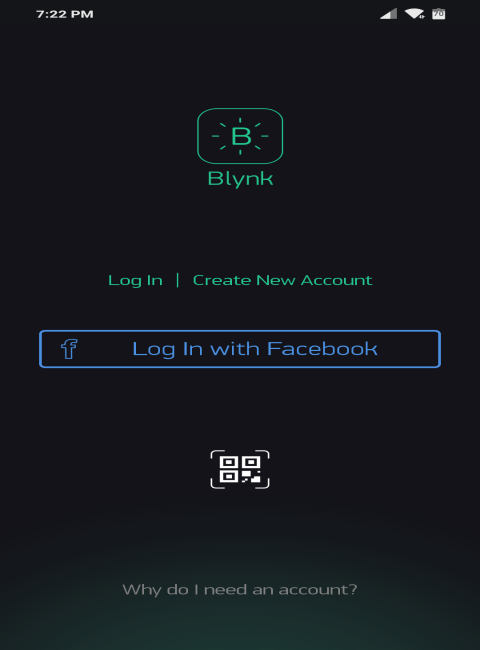


Fig 4.1.2.4.1 Creating Blynk account

* Create a New Project.

After creating an account log into your account and create a new project

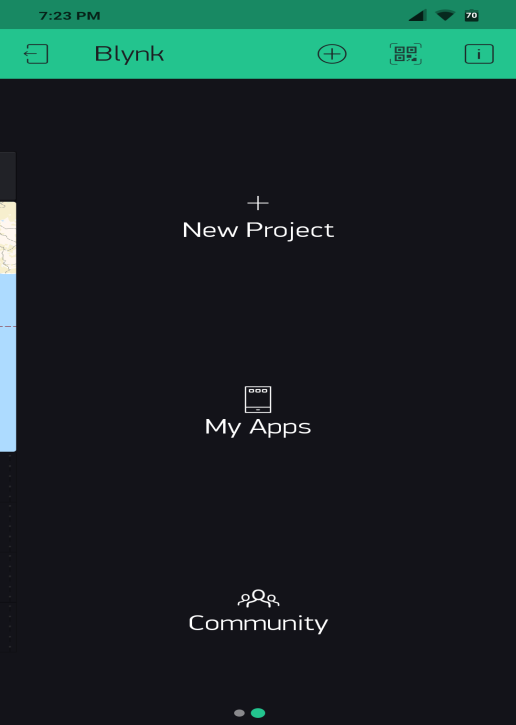


Fig 4.1.2.4.2 Creating new project in blynk

* Choose Your Hardware

Choose the board with which you’re working

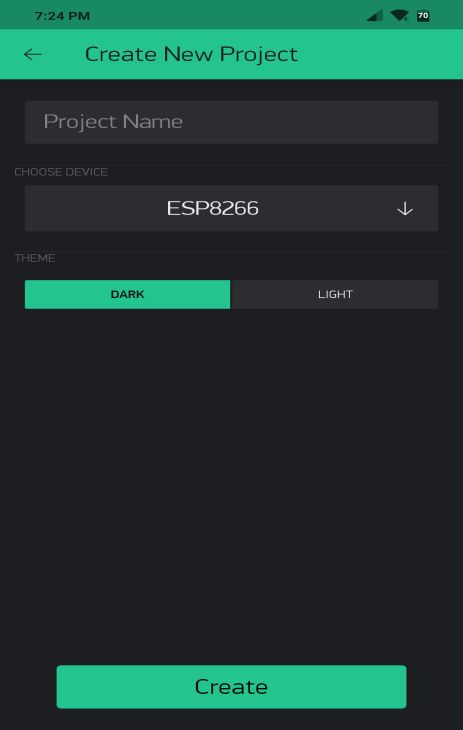


Fig 4.1.2.4.3 Selecting board in blynk

* Authentication Token

After choosing your hardware you will get a authentication token to your mail Include this token your code

* Add a Widget

After creating new project add desired widgets to your project like

* virtual pin V0 is assigned to MAP widget
* virtual pin V1,V2 are assigned to LABEL widgets
* virtual pin V3,V4 are assigned to VALUE widgets

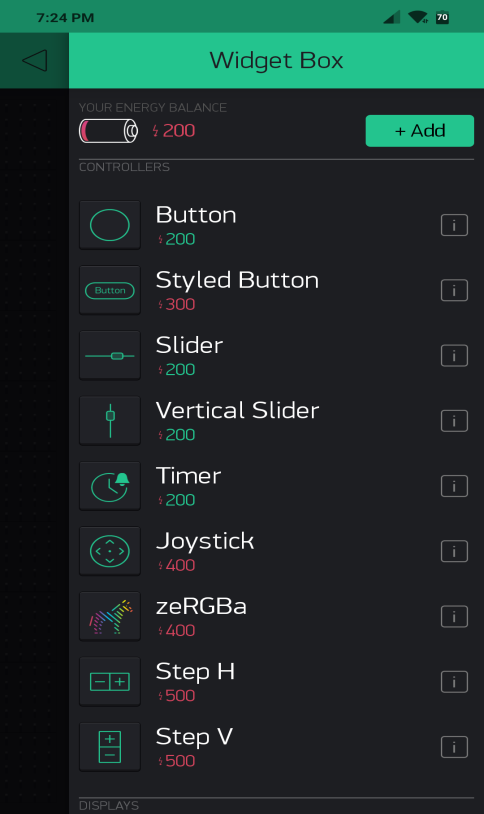


Fig 4.1.2.4.4 Selecting widgets in blynk

Now use these virtual pins in your code to upload the data to blynk server and to visualize in blynk app

* Run The Project.

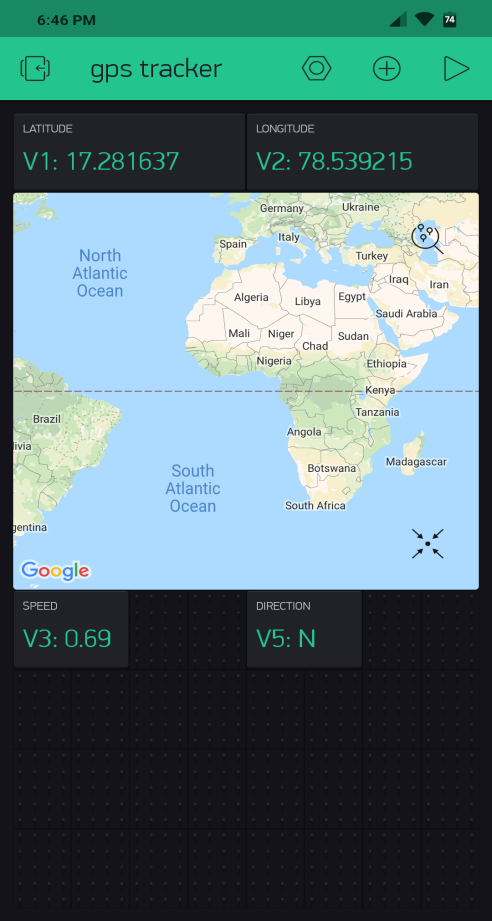


Fig 4.1.2.4.5 Output shown in blynk

**4.2 Source code**

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#define BLYNK\_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

#include <ESP8266WiFi.h>

#include <OneWire.h>

#include <PubSubClient.h>

static const int RXPin = 4, TXPin = 5; // GPIO 4=D2(conneect Tx of GPS) and GPIO 5=D1(Connect Rx of GPS

static const uint32\_t GPSBaud = 9600; //if Baud rate 9600 didn't work in your case then use 4800

const char \*ssid = "ACT101014564385"; //Your Access Point or Personal Hotspot, cannot be longer than 32 characters!

const char \*pass = "44017881"; //Your Access Point or Personal Hotspot password

const char\* serverTS = "api.thingspeak.com";

char auth[] = "aa26238b0db54c69ae84a952d46d981c";

String apiKey = "HWP2ST3URKAB3QQE"; //Insert your Channel API Key here

//const int pingPin = 2; //Ultrasonic connected to GPIO0

int TRIGGER = 2; //Pin D4 = TRIGGER

int ECHO = 14; //Pin D5 = ECHO

static const float bin=25;

TinyGPSPlus gps; // The TinyGPS++ object

WidgetMap myMap(V0); // V0 for virtual pin of Map Widget

SoftwareSerial ss(RXPin, TXPin); // The serial connection to the GPS device

BlynkTimer timer;

float spd; //Variable to store the speed

float sats; //Variable to store no. of satellites response

String bearing; //Variable to store orientation or direction of GPS

//unsigned int move\_index; // moving index, to be used later

unsigned int move\_index = 1; // fixed location for now

void setup()

{

Serial.begin(115200);

Serial.println();

ss.begin(GPSBaud);

Blynk.begin(auth, ssid, pass);

timer.setInterval(5000L, checkGPS); // every 5s check if GPS is connected, only really needs to be done once

pinMode(0,OUTPUT); //LED connected to GPIO2

//Serial.begin(115200); //Recommended speed is 115200

pinMode(TRIGGER,OUTPUT);

pinMode(ECHO,INPUT);

}

void checkGPS(){

if (gps.charsProcessed() < 10)

{

Serial.println(F("No GPS detected: check wiring."));

Blynk.virtualWrite(V4, "GPS ERROR"); // Value Display widget on V4 if GPS not detected

}

}

void loop()

{

while (ss.available() > 0)

{

// sketch displays information every time a new sentence is correctly encoded.

if (gps.encode(ss.read()))

displayInfo();

}

Blynk.run();

timer.run();

// establish variables for duration of the ping,

// and the distance result in inches and centimeters:

long duration, inches, cm,percentage;

// The PING))) is triggered by a HIGH pulse of 2 or more microseconds.

// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

digitalWrite(TRIGGER, LOW);

delayMicroseconds(2);

digitalWrite(TRIGGER, HIGH);

delayMicroseconds(10);

digitalWrite(TRIGGER, LOW);

// The same pin is used to read the signal from the PING))): a HIGH

// pulse whose duration is the time (in microseconds) from the sending

// of the ping to the reception of its echo off of an object.

duration = pulseIn(ECHO, HIGH);

// convert the time into a distance

//inches = microsecondsToInches(duration);

cm = microsecondsToCentimeters(duration);

percentage=(1-(cm/bin))\*100;

//Serial.print(inches);

//Serial.print("in, ");

Serial.print(cm);

Serial.print("cm");

Serial.println();

delay(100);

digitalWrite(2, HIGH); // turn the LED on (HIGH is the voltage level)

delay(1000); // wait for a second

digitalWrite(2, LOW); // turn the LED off by making the voltage LOW

delay(1000); // wait for a second

//sendHeight(cm);

sendHeight(percentage);

}

void displayInfo()

{

if (gps.location.isValid() )

{

float latitude = (gps.location.lat()); //Storing the Lat. and Lon.

float longitude = (gps.location.lng());

Serial.print("LAT: ");

Serial.println(latitude, 6); // float to x decimal places

Serial.print("LONG: ");

Serial.println(longitude, 6);

Blynk.virtualWrite(V1, String(latitude, 6));

Blynk.virtualWrite(V2, String(longitude, 6));

myMap.location(move\_index, latitude, longitude, "GPS\_Location");

spd = gps.speed.kmph(); //get speed

Blynk.virtualWrite(V3, spd);

sats = gps.satellites.value(); //get number of satellites

Blynk.virtualWrite(V4, sats);

bearing = TinyGPSPlus::cardinal(gps.course.value()); // get the direction

Blynk.virtualWrite(V5, bearing);

}

Serial.println();

}

long microsecondsToCentimeters(long microseconds) {

// The speed of sound is 340 m/s or 29 microseconds per centimeter.

// The ping travels out and back, so to find the distance of the

// object we take half of the distance travelled.

return microseconds / 29 / 2;

}

void sendHeight(float cm)

{

WiFiClient tclient;//not to be confused with "client" in PubSub{}, and wclient for mqtt

if (tclient.connect(serverTS, 80)) { // use ip 184.106.153.149 or api.thingspeak.com

//Serial.println("WiFi Client connected ");

String postStr = apiKey;

postStr += "&field1=";

postStr += String(cm);

postStr += "\r\n\r\n";

tclient.print("POST /update HTTP/1.1\n");

tclient.print("Host: api.thingspeak.com\n");

tclient.print("Connection: close\n");

tclient.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");

tclient.print("Content-Type: application/x-www-form-urlencoded\n");

tclient.print("Content-Length: ");

tclient.print(postStr.length());

tclient.print("\n\n");

tclient.print(postStr);

delay(1000);

}//end if

tclient.stop();

}//end send to ts

**CHAPTER V**

**SYSTEM TESTING**

**5.1 Test Approach:**

The purpose of testing is to discover errors.Testing is the process of trying to discover every conceivable fault or weakness in a work product.We will test the project in two stages: Software and Hardware. The software part is to be tested via the Arduino IDE, whereas the hardware part has to be tested physically. It is necessary to check whether the system is working properly or not. To check whether the readings are accurate, we will check the distance pointed out by the sensor by a meter tape.

**5.2 Features to be tested**

After building the whole circuit we test it with the testing procedure has given in 5.1. This project should satisfy some features. Features to be tested as follows:

* The ultrasonic sensor should give proper output. To check whether the output is accurate or not, the output of the sensor will be checked against a meter tape.
* Check GPS module for proper working(Accurate location).
* The NodeMCU must be connected to Internet .
* Check on the respective server whether the data is being received or not, and respective action to be performed .
* Check whether the GPS data is sent to the Blynk Server or Not.

**5.2.1 Testing tools and environment**

For testing of the project we require some tools, like to test Arduino program we require a software called Arduino IDE. Using this we can check the program that program is working properly or not. For hardware checking we require power supply and proper range of measurements and a meter tape. The garbage dump should have only solid waste.The NodeMCU should connect to the Blynk app and the app should show the output. For this, the NodeMCU must be connected to the wifi hotspot.

**5.3 Expected output**

The expected outputs of this project are:

* **Real-time Garbage Monitoring**

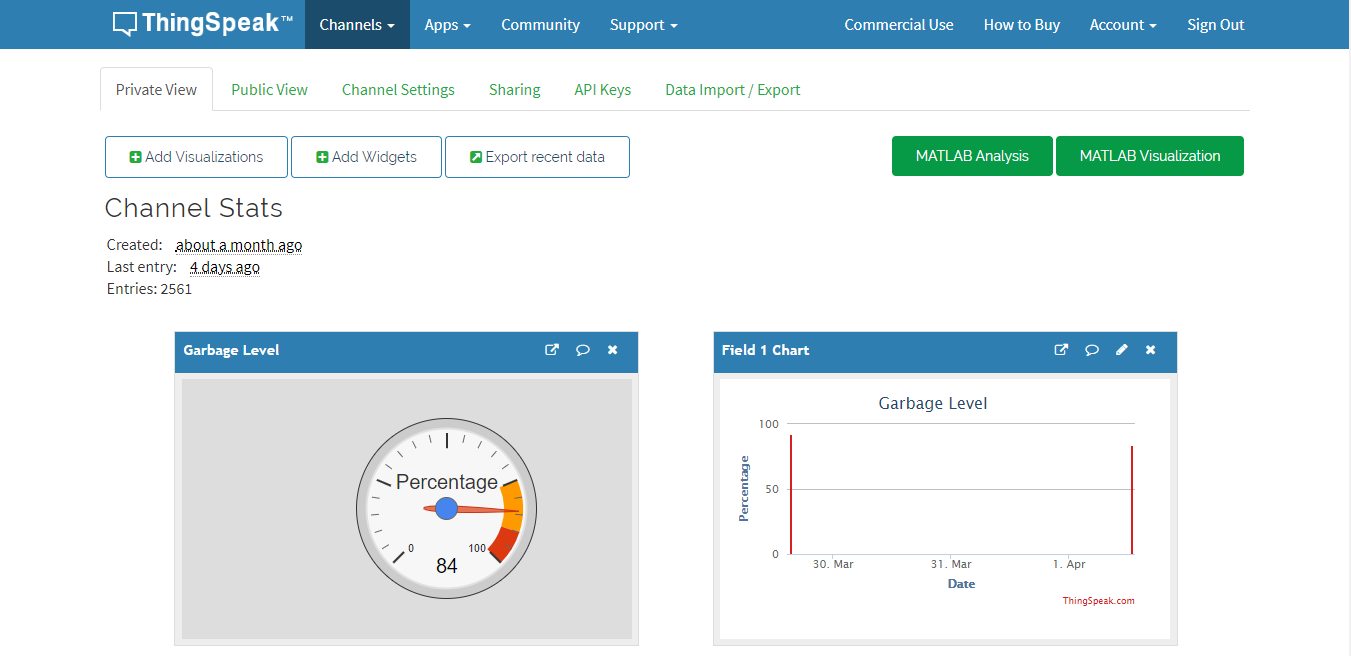


Fig 5.3.1 Output shown in ThingSpeak

As shown in the above figure,we can know the percentage of the garbage present in the bin,and also this data is graphically as well as digitally represented

* **Notification in terms of Voice Call and Text Message**

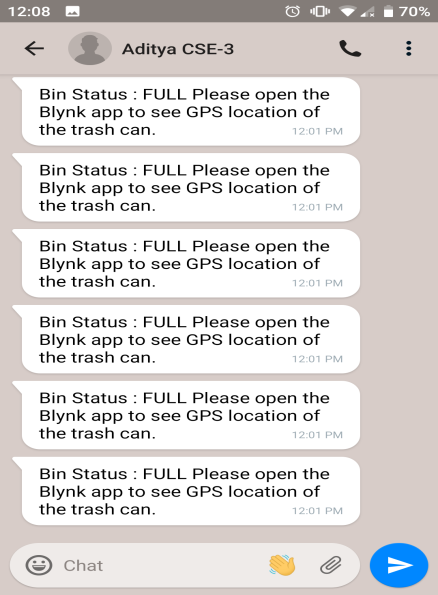
** **

Fig 5.3.2 Notification through call or message

* **Get the location of the Bin**

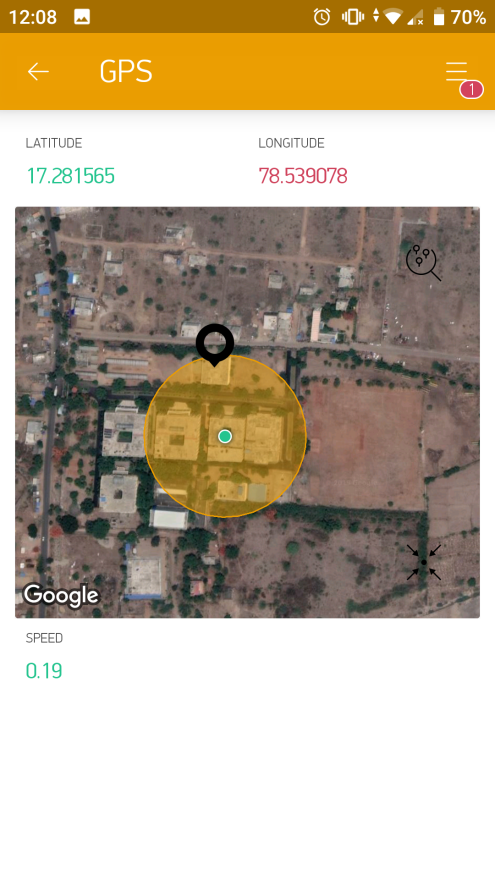
****

Fig 5.3.3 Location shown in blynk

**CHAPTER VI**

**CONCLUSION AND REFERENCES**

**6.1 Conclusion**

In this project, an integrated system of Wi-Fi modem, IoT, GPS and Ultrasonic Sensor is introduced for efficient and economic garbage collection. The developed system provides improved database for garbage collection time and waste amount at each location. We analysed the solutions currently available for the implementation of IoT. By implementing this project we will avoid overflowing of garbage from the container in residential area which is previously either loaded manually or with the help of loaders in traditional trucks. It can automatically monitor the garbage level & send the information to collection truck. The technologies which are used in the proposed system are good enough to ensure the practical and perfect for solid garbage collection process monitoring and management for green environment. Thus,we built an efficient garbage monitoring system which can be used to monitor the level of garbage in the dump. This data can be further used to plan garbage collection trips more efficiently, ultimately reducing overflowing bins and helping have better public sanitation.

**Advantages**:

* Very simple circuit.
* Helps monitor garbage levels. Uses very small amount of electricity.
* Ultimately helps in better planning of garbage pickups.
* Saves fuel and time by optimizing collections routes and schedules.
* Can help in reducing overflowing bins.
* Reduces trips to areas where the bins still have a lot of capacity.

**Disadvantages:**

* Cannot detect liquid waste.
* Only detects the top of the garbage level. It wouldn’t realize if there is space left.
* NodeMCU module always needs to be connected to internet a source

**6.2 References**

* Maher arebey, Killedar M S and Rohokale D V 2016 IoT Based Smart Garbage and waste collection, International Journal of Advanced Research in Electronics And Communication.
* Waikhom reshmi,Rao N, Prapulla S B and Shobha G 2016 Smart Dustbin-An Efficient Garbage Monitoring System International Journal of Engineering Science and Computing 6 7113-16.
* B. Vijayalaxmi, Fedchenkov P, Zaslavsky A,Anagnostopoulos T,2015 Waste management as an IoT-enabled service in smart cities In Conference on Smart Spaces Springer International Publishing 104-15.
* <https://arduino.cc.com>
* <https://github.com>
* <https://www.youtube.com/watch?v=WZesl6az8L8&t=608s>