IOT ENABLED GARBAGE MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

ADHITHYAN S	**720421106001**
SHEBIN MATHEW	**720421106005**
ASHWINI VIJAYAN	**720421106301**
MUHAMMED SUHAIL E S	**720421106303**

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

CMS COLLEGE OF ENGINEERING AND TECHNOLOGY COIMBATORE

ANNA UNIVERSITY: CHENNAI - 600025

MAY 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"IOT ENABLED GARBAGE MANAGEMENT SYSTEM"** is the BONAFIDE work of **"ADHITHYAN S, SHEBIN MATHEW, ASWINI VIJAYAN, MUHAMMED SUHAIL E S"** who carried out the project work under my supervision.

SIGNATURE

Dr. Jayaprakash. M,

HEAD OF THE DEPARTMENT

Professor,

Department of Electronics and Communication Engineering

CMS College of Engineering And Technology
Coimbatore- 641 032

SIGNATURE

Mr. K. Sivaraj,

SUPERVISOR

Assistant Professor,

Department of Electronics and Communication Engineering

CMS College of Engineering and Technology
Coimbatore- 641 032

Submitted for the Anna University Project Viva Voce conducted on…….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Any organized and systematic word calls for the corporation of a team of people. Our project does not have any exception to this. Hence these pages find the space for thanking all those who have directly and indirectly contributed to the completion of this work in a successful manner.

We record our indebtedness to our Principal **Dr. N. Sudha, M.E, Ph.D.,** for her guidance and sustained encouragement for the successful completion of this project.

We express our heartiest thanks to **Dr. Jayaprakash M, M.E, Ph.D.,** Professor, Head of the Department, Department of Electronics and Communication Engineering, CMS College of Engineering and Technology, for his encouragement and valuable guidance in carrying out our project work.

We express our heartfelt thanks to Project Guide **Mr. K Sivaraj, M.E.,** Assistant Professor, Department, Department of Electronics and Communication Engineering, CMS College Of Engineering and Technology Coimbatore, for her valuable and timely support for our project work.

We express our heartfelt gratitude to Project Coordinator **Dr. S Prabakaran, MTech, Ph.D.,** Assistant Professor, Department of Electronics and Communication Engineering, CMS College of Engineering and Technology Coimbatore, for his valuable and timely support for our project work.

We also express thanks to our parents and friends for their encouragement and best wishes in the successful completion of this dissertation.

ABSTRACT

In recent years, the rapid pace of urbanization has significantly increased solid waste generation, creating major challenges for municipal waste management. Traditional waste collection systems often fail to provide efficient services, resulting in overflowing bins, unhygienic conditions, and delays in garbage collection. To address these issues, this project proposes a smart, automated solution using Internet of Things (IoT) technology, aimed at improving the efficiency, hygiene, and responsiveness of urban waste management. The proposed system utilizes a NodeMCU microcontroller integrated with an ultrasonic sensor to continuously monitor the garbage level inside bins. When the waste reaches 80% of the bin's capacity, a real-time alert is sent to the concerned authorities through a cloud-based IoT platform, ensuring timely waste collection and preventing overflow. Additionally, a gas/smoke sensor is incorporated to detect harmful gases and foul odors emitted by decomposing waste. Even if the bin is not full, the presence of these gases triggers a separate alert, highlighting potential health hazards and prompting quick action to mitigate environmental risks. To further enhance hygiene and reduce human contact, a servo motor is included to automate the opening and closing of the garbage bin lid. The system leverages Wi-Fi connectivity to transmit data to a mobile application or web-based dashboard, allowing real-time monitoring and control. By integrating multiple sensors with IoT and automation technologies, the system provides a smart, sustainable, and scalable solution to urban waste management, promoting cleaner environments and healthier communities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBEREVIATION

DWPT	Dynamic wireless power transmission
EV	Electric vehicles
SS	Series-series (SS) topology
CMPPT	Constant maximum power point tracking
MOSFET	Metal oxide semiconductor field effect transistor
IDE	Integrated development environment
TC	Charging the time of battery
WPDS	Wireless power distribution switch
VREF	Voltage reference
IC	Integrated circuit
AC	Alternating current
DC	Direct current
VGS	Gate threshold voltage

CHAPTER 1
INTRODUCTION

The exponential growth in urban population has intensified the demand for efficient public infrastructure and services, particularly in the area of waste management. Municipal bodies often struggle to keep up with the increasing volume of solid waste generated daily, leading to challenges such as delayed collection, overflowing bins, and the spread of diseases. Traditional garbage collection systems rely heavily on fixed schedules rather than actual waste levels, which results in inefficient resource utilization and public dissatisfaction. In recent years, the advent of smart technologies like the Internet of Things (IoT) has opened up new possibilities for transforming conventional systems into intelligent, real-time solutions. IoT enables interconnected devices to collect and transmit data, allowing for automation and informed decision-making. In the context of waste management, this means bins can be equipped with sensors and communication modules to monitor garbage levels, detect harmful emissions, and automatically alert authorities for timely intervention. This project introduces an IoT-enabled garbage management system that integrates ultrasonic and gas sensors with a NodeMCU microcontroller. The system monitors the waste level and the presence of hazardous gases, sending real-time alerts when thresholds are exceeded. Additionally, a servo motor automates the lid mechanism, reducing manual contact and enhancing hygiene. By enabling data-driven and responsive waste collection, this system aims to make urban environments cleaner, safer, and more sustainable.

The accelerated pace of urbanization in recent decades has brought about a considerable increase in the generation of solid waste. This rapid growth has imposed immense pressure on existing municipal waste management systems, which are often outdated and inefficient. Traditional waste collection methods typically operate on predetermined schedules and manual monitoring, failing to adapt to real-time conditions. As a result, cities frequently face overflowing garbage bins, unpleasant odors, unsanitary environments, and inefficient allocation of waste collection resources such as labor and fuel. These challenges not only degrade the quality of life in urban communities but also pose serious public health and environmental risks.

In response to these persistent issues, this project proposes a smart, automated waste management system based on Internet of Things (IoT) technology. The goal of the system is to enhance the efficiency, hygiene, and responsiveness of waste collection services through real-time monitoring, automated alerts, and intelligent decision-making support. At the core of this system lies the NodeMCU microcontroller, a low-cost, Wi-Fi-enabled device that facilitates seamless communication between sensors, actuators, and cloud platforms.

The system employs an **ultrasonic sensor** installed at the top of the garbage bin to monitor the waste level continuously. This sensor measures the distance from the sensor to the top of the waste pile and calculates the percentage of bin capacity utilized. Once the bin reaches 80% of its capacity, the system triggers a real-time alert that is transmitted via a cloud-based IoT platform to the designated municipal authorities. This proactive notification mechanism ensures timely waste collection and helps prevent bin overflow, contributing to cleaner urban environments.

CHAPTER 2

LITERATURE SURVEY

1. **Title**

   AI-Based Intelligent Garbage Management System

   Authors

   Siddhant Bansal et al.

   Journal

   Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 204–215, 2023

   Description

   The paper explores the integration of AI algorithms into garbage management systems to predict garbage overflow and optimize collection schedules. Machine learning techniques are applied to historical waste data, resulting in enhanced prediction accuracy and smarter waste disposal strategies for urban settings.

2. **Title**

   Self-Supervised Learning for Garbage Monitoring

   Authors

   Siti Sarah Md Ilyas et al.

   Journal

   IEEE Journal, May 2023

   Description

   This research introduces a self-supervised learning approach for monitoring garbage accumulation without requiring extensive labeled datasets. The model effectively detects garbage fill levels and abnormalities, significantly reducing system deployment costs and improving real-time decision-making in smart cities.

3. **Title**

Deep Learning-Based AI Assistants for Waste Level Monitoring

Authors

Md. Shahariar Nafiz et al.

Journal

IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 9, pp. 8092–8105, September 2022

Description

The study proposes deep learning-based AI assistants capable of monitoring waste bin levels accurately. Convolutional Neural Networks (CNNs) are utilized to process sensor data and camera images, enabling early detection of bin overflow and toxic gas emissions. The system enhances urban waste management and public health outcomes.

4. **Title**

Waste Classification at the Edge for Smart Bins

Authors

Gary White et al.

Journal

Proceedings of IEEE CVPR, pp. 3121–3130, 2022

Description

This work focuses on deploying edge computing techniques for waste classification in smart bins. Machine learning models are embedded directly into smart bins, enabling real-time waste type identification without relying on centralized servers. The approach reduces network load and ensures faster, more efficient waste management operations.

CHAPTER 3

EXISTING METHOD

3.1 EXISTING SYSTEM

Traditional waste management systems rely heavily on manual collection and fixed scheduling without considering the actual fill level of garbage bins. This often leads to inefficiencies such as bins being either collected too early (underfilled) or too late (overflowing), resulting in unhygienic conditions and public inconvenience. Furthermore, municipal workers have to physically inspect bins to determine their status, which is time-consuming, labor-intensive, and exposes them to health risks, especially when dealing with hazardous or decomposing waste.

Some existing smart systems have introduced basic sensor integration, such as ultrasonic sensors to measure bin levels. However, many of these setups lack real-time data transmission and advanced alert mechanisms. They also do not account for health hazards posed by harmful gases emitted from waste, nor do they offer hygienic features such as automated lid mechanisms. Additionally, the absence of centralized monitoring through IoT platforms limits the scalability and efficiency of such systems in handling large urban waste networks.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- Garbage collection is done on fixed schedules, not based on actual bin status.
- Overflowing bins create unhygienic and unsanitary conditions.
- Manual inspection of bins is time-consuming and labor-intensive.
- No real-time alerts or automated monitoring of bin fill levels.
- Fails to detect harmful gases or odors from decomposing waste.
- Lack of automation increases physical contact with garbage bins.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 PROPOSED METHOD

The proposed system introduces an IoT-enabled smart garbage management solution that monitors both the garbage level and the presence of harmful gases in real time. It utilizes a NodeMCU microcontroller integrated with an ultrasonic sensor to measure the fill level of the bin and a gas sensor to detect hazardous emissions from decomposing waste. When the garbage reaches 80% of the bin capacity or harmful gases are detected, the system automatically sends an alert to the concerned municipal authorities through a cloud-based IoT platform. This ensures timely waste collection, prevents bin overflow, and addresses health risks proactively.

To enhance hygiene and minimize physical contact, a servo motor is employed to automate the opening and closing of the bin lid. The entire system is connected via Wi-Fi, allowing real-time monitoring through a mobile app or web dashboard. This integration of multiple sensors and IoT technology improves the efficiency, responsiveness, and sustainability of urban waste management. By providing timely alerts and automating processes, the system promotes a cleaner, safer, and healthier environment.

4.2 ADVANTAGES

- Enables real-time monitoring of garbage levels and harmful gas emissions.
- Sends automatic alerts to authorities when bins are nearly full or gases are detected.
- Reduces overflow and maintains cleaner surroundings.
- Enhances hygiene through automated lid operation using a servo motor.

- Minimizes manual inspection and physical contact with waste.
- Uses IoT and Wi-Fi for efficient, remote monitoring via web or mobile dashboards.
- Promotes timely and optimized waste collection.

4.3 BLOCK DIAGRAM

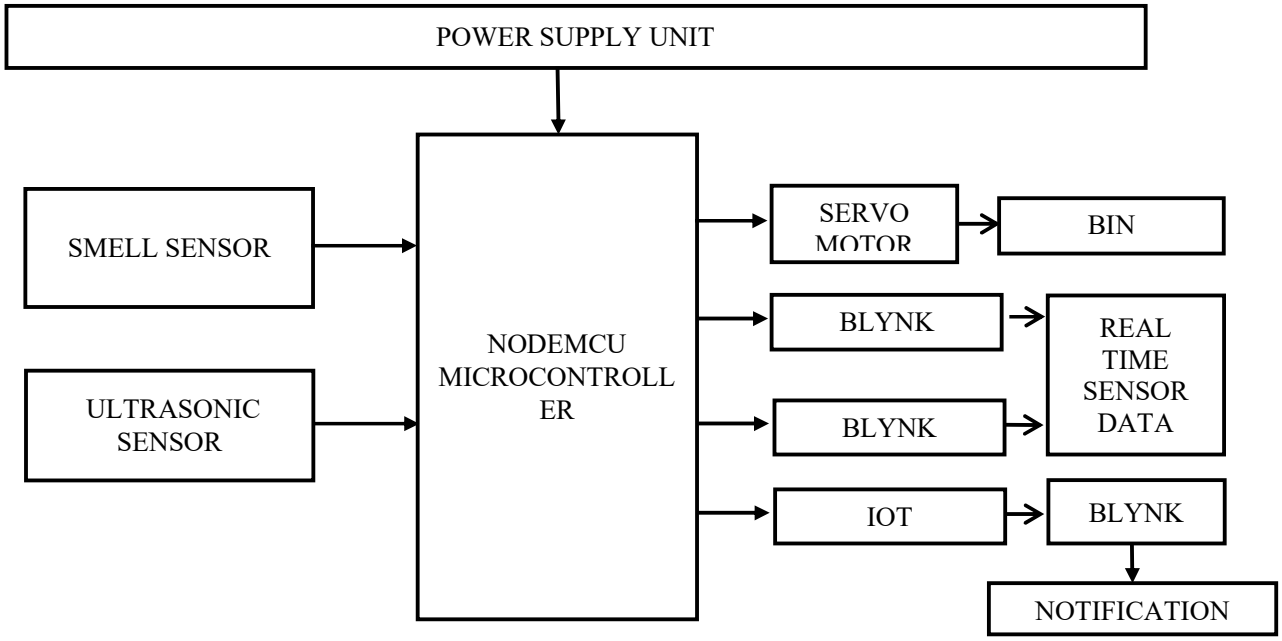


Fig 4.1 Block diagram of proposed system

CHAPTER 5

SYSTEM SPECIFICATION

5.1 HARDWARE REQUIREMENTS

5.1.1 POWER SUPPLY

Power supply is a reference to a source of electrical power. A device or system that supplies electrical or other types of energy to an output load or group of loads is called a power supply unit or PSU. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

Power supplies for electronic devices can be broadly divided into linear and switching power supplies. The linear supply is a relatively simple design that becomes increasingly bulky and heavy for high current devices; voltage regulation in a linear supply can result in low efficiency. A switched-mode supply of the same rating as a linear supply will be smaller, is usually more efficient, but will be more complex.

5.1.2 LINEAR POWER SUPPLY

An AC powered linear power supply usually uses a transformer to convert the voltage from the wall outlet (mains) to a different, usually a lower voltage. If it is used to produce DC, a rectifier is used. A capacitor is used to smooth the pulsating current from the rectifier. Some small periodic deviations from smooth direct current will remain, which is known as ripple. (for example, a multipleof50 or60 Hz).

The voltage produced by an unregulated power supply will vary depending on the load and on variations in the AC supply voltage. For critical electronics applications a linear regulator will be used to stabilize and adjust the voltage. This regulator will also greatly reduce the ripple and noise in the output direct current. Linear regulators often provide current limiting, protecting the power supply and attached circuit from over current.

Adjustable linear power supplies are common laboratory and service shop test equipment, allowing the output voltage to be set over a wide range. For example, a bench power supply used by circuit designers may be adjustable up to 30 volts and up to 5 amperes output. Some can be driven by an external signal, for example, for applications requiring a pulsed output.

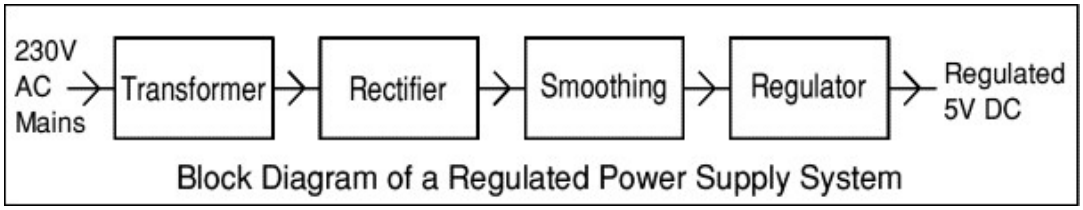


Fig 5.1.2.1 REGULATOR

5.1.3 TRANSFORMER

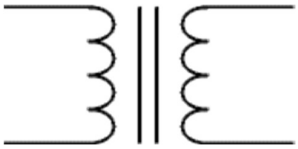


Fig 5.1.3.1 Transformer

Transformers convert AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC.

Step-up transformers increase voltage; step-down transformers reduce voltage. Most power supplies use a step-down transformer to reduce the dangerously high mains voltage (230V in UK) to as after low voltage.

The input coil is called the primary and the output coil is called the secondary.

There is no electrical connection between the two coils; instead, they are linked

by an alternating magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core.

Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down current is stepped up.

The ratio of the number of turns on each coil, called the turn's ratio, determines the ratio of the voltages. A step-down transformer has a large number of turns on its primary (input) coil which is connected to the high voltage mains supply, and a small number of turns on its secondary(output) coil to give a low output voltage.

Turns ratio= Vp/Vs=Nn/Ns and

Power out=Power in Vs*Is=Vp*Ip

Vp = primary(input)voltage Np = number of turns on primary coils	Vs = secondary(output)voltage Ns = number of turns on secondary coilIs = secondary(output)current

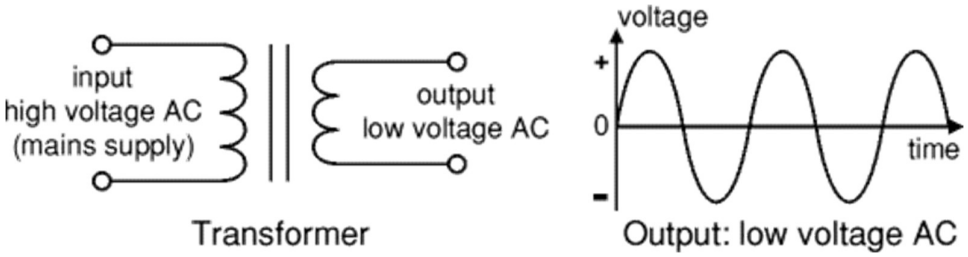


Fig 5.1.3.2 Transformer output

The low voltage AC output is suitable for lamps, heaters and special AC motors. It is not suitable for electronic circuits unless they include a rectifier and a smoothing capacitor.

5.1.4 RECTIFIER

There are several ways of connecting diodes to make a rectifier to convert AC to

DC. The bridge rectifier is the most important and it produces full-wave varying DC. A full-wave rectifier can also be made from just two diodes if a center-tap transformer is used, but this method is rarely used now that diodes are cheaper. A single diode can be used as a rectifier but it only uses the positive (+) parts of the AC wave to produce half-wave varying DC.

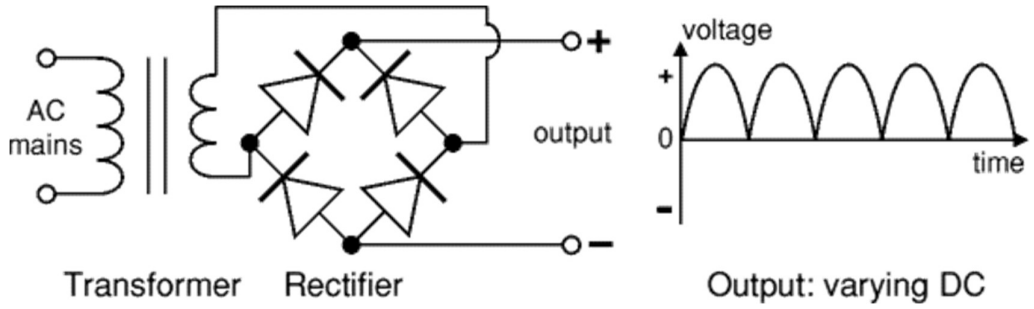


Fig 5.1.4.1 Rectifier Diagram

The varying DC output is suitable for lamps, heaters and standard motors. It is not suitable for electronic circuits unless they include a smoothing capacitor.

5.1.5 BRIDGE RECTIFIER

A bridge rectifier can be made using four individual diodes, but it is also available in special packages containing the four diodes required. It is called a full-wave rectifier because it uses the entire AC wave (both positive and negative sections). 1.4V is used up in the bridge rectifier because each diode uses 0.7V when conducting and there are always two diodes conducting, as shown in the diagram below. Bridge rectifiers are rated by the maximum current they can pass and the maximum reverse voltage they can withstand (this must be at least three times the supply RMS voltage so the rectifier can withstand and the peak voltages). Please see the Diodes page for more details, including pictures of ridge rectifiers.

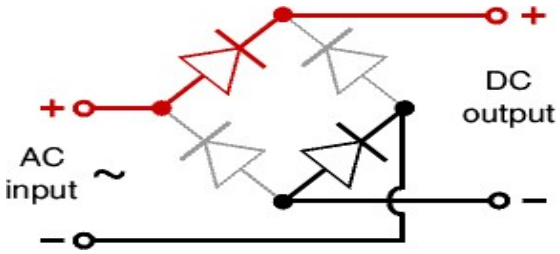
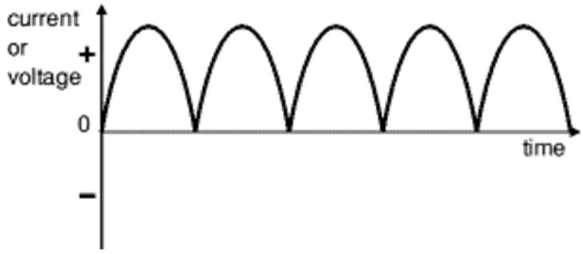


fig 5.1.5.1 Bridge Rectifier

Alternate pairs of diodes conduct, changing over the connections so the alternating directions of AC are converted to the one direction of DC.

Output: full-wave varying DC:(using the entire AC wave):



5.1.6 SINGLE DIODE RECTIFIER

A single diode can be used as a rectifier but this produces **half-wave** varying DC which has gaps when the AC is negative. It is hard to smooth this sufficiently well to supply electronic circuits unless they require a very small current so the smoothing capacitor does not significantly discharge during the gaps. Please see the Diodes page for some examples of rectifier diodes.

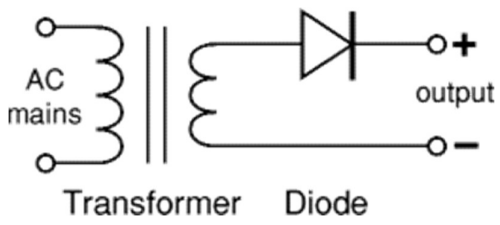


Fig 5.1.6.1 Single Wave Diode Rectifier

Output: half-wave varying DC (using only half the AC wave):

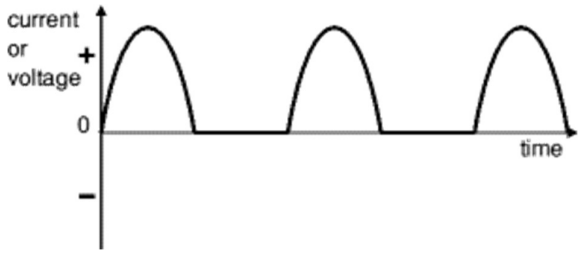


Fig 5.1.6.2 Output

5.1.7 REGULATOR

Voltage regulator ICs are available with fixed (typically 5, 12 and 15V) or variable output voltages. They are also rated by the maximum current they can pass. Negative voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current ('overload protection') and overheating ('thermal protection').

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although

designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and current.

Many of the fixed voltage regulator ICs has 3 leads and look like power transistors, such as the 7805 +5V 1A regulator shown on the right. They include a hole for attaching a heatsink if necessary.

Positive regulator

1. input pin
2. ground pin
3.output pin

It regulates the positive voltage

Negative regulator

1.ground pin
2.input pin
3.output pin

It regulates the negative voltage

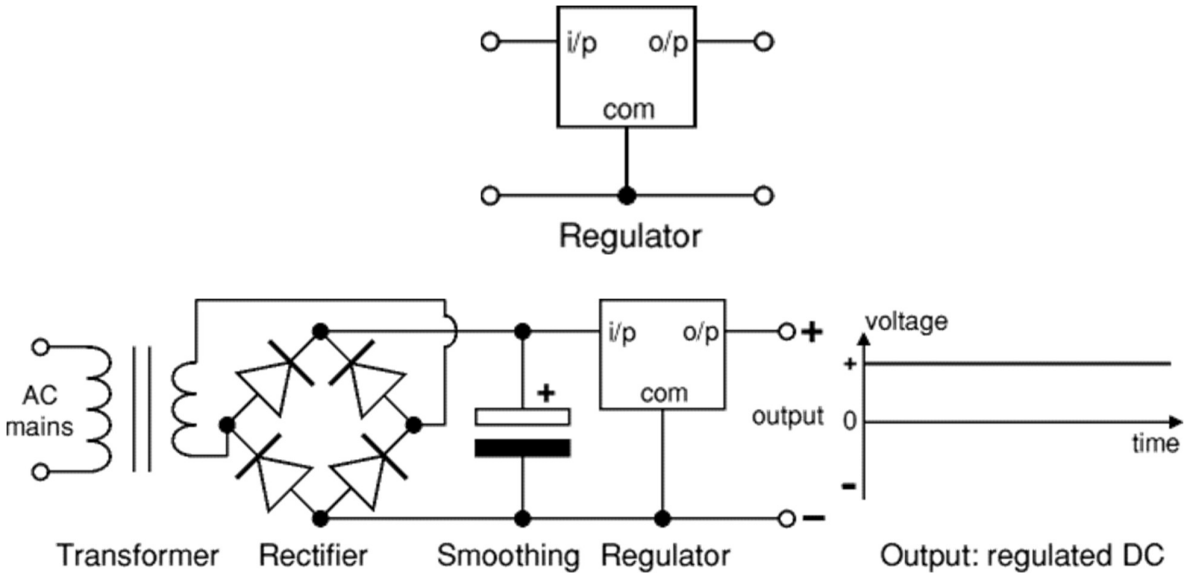


Fig 5.1.7.1 Regulator

The regulated DC output is very smooth with no ripple. It is suitable for all electronic circuits.

5.1.8 NodeMCU

Fig 5.1.2.1 NodeMCU

NodeMCU V3 is an open-source firmware and microcontroller that plays a vital role in designing an IoT product using a few script lines.

Multiple GPIO pins on the board allow us to connect the board with other peripherals and are capable of generating PWM, I2C, SPI, and UART serial communications.

- The interface of the module is mainly divided into two parts including both Firmware and Hardware where former runs on the ESP8266 Wi-Fi SoC and later is based on the ESP-12 module.

The firmware is based on Lua – A scripting language that is easy to learn, giving a simple programming environment layered with a fast-scripting language that connects you with a well-known developer community.

And open-source firmware gives you the flexibility to edit, modify and rebuilt the existing module and keep changing the entire interface until you succeed in optimizing the module as per your requirements.

- USB to UART converter is added on the module that helps in converting USB data to UART data which mainly understands the language of serial communication. Instead of the regular USB port, Micro-USB port is included in the module that connects it with the computer for dual purposes: programming and powering up the board.
- The board incorporates status LED that blinks and turns off immediately, giving you the current status of the module if it is running properly when

connected with the computer. The ability of module to establish a flawless Wi-Fi connection between two channels make it an ideal choice for incorporating it with other embedded devices like Raspberry Pi.

5.1.9 NodeMCU V3 Pinout

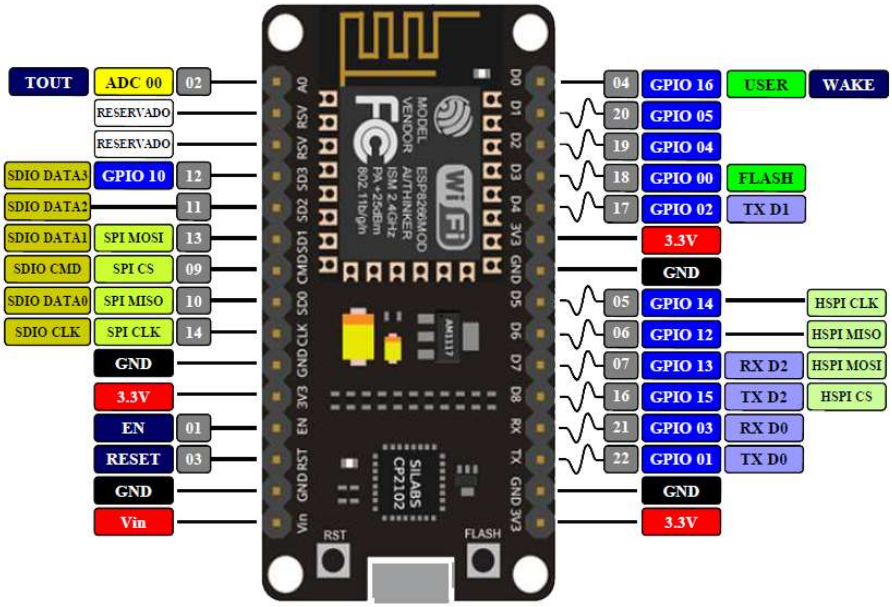


fig 5.1.9.1 Nodemcu pinout

•There is a candid difference between Vin and VU where former is the

•There is a candid difference between Vin and VU where former is the

NodeMCU V3 comes with a number of GPIO Pins.

•There is a candid difference between Vin and VU where former is the

There is a candid difference between Vin and VU where former is the regulated voltage that may stand somewhere between 7 to 12 V while later is the power voltage for USB that must be kept around 5 V.

Features

1.Open-source

2.Arduino-like hardware

3.Status LED

4.Micro-USB port

5.Reset/Flash buttons

6.Interactive and Programmable

7.Low cost

8.ESP8266 with inbuilt Wi-Fi

9.USB to UART converter

10.GPIO pins

11. Arduino-like hardware IO

12. Advanced API for hardware IO, which can dramatically reduce the redundant work for configuring and manipulating hardware.

13.Code like Arduino, but interactively in Lua script.

14.Nodejs style network API

15.Event-driven API for network applications, which facilitates developers writing code running on a 5mm*5mm sized MCU in Nodejs style.

16.Greatly speed up your IOT application developing process.

17.Lowest cost WI-F

5.1.10 ULTRA SONIC SENSOR: GENRAL DESCRPTION

Ultrasonic sensors work on a principle similar to sonar which evaluates distance of a target by interpreting the echoes from ultrasonic sound waves. This ultrasonic module measures the distance accurately which provides 0cm - 400cm with a gross error of 3cm. Its compact size, higher range and easy usability make it a handy sensor for distance measurement and mapping. The module can easily be interfaced to micro controllers where the triggering and measurement can be done using two pins. The sensor transmits an ultrasonic wave and produces an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo

pulse width, the distance to target can easily be calculated. Features non-contact measurement with blinding from 0-1cm*.

FEATURE:

- Easy To Use 4-Pin Breakout.
- Range: 2cm-400cm Non-Contact Measurement Function.
- Ranging Accuracy: ±3cm (Incremental Towards Maximum Range).
- Measure Angle: 15º.
- Operates On 40khz Ultrasonic Frequency Range.
- Operating Voltage Of 4.8 V To 5.5 V (± 0.2v Max).
- Operating Temperature Range: 0c to 60c (± 10%).
- Separate Inputs for Trigger and Received Echo.

PIN DESCRPTION

With the sensor oriented as shown alongside, locate Pin-1 as the 1st pin (refer figure 1) on the left-hand side.

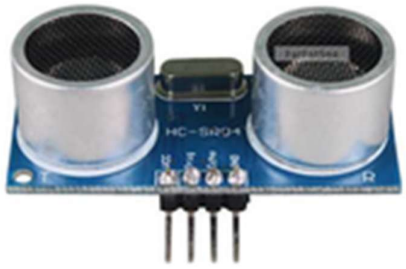


fig 5.1.10.1 Ultrasonic Sensor

5.1.11 MALE CONNECTOR SIDE

parameter	specification
Dimensions	45 x 20 x 15 mm (±
Pin-out Pitch	2.54mm male berg
Interface	VCC, GND, SDA, SCL

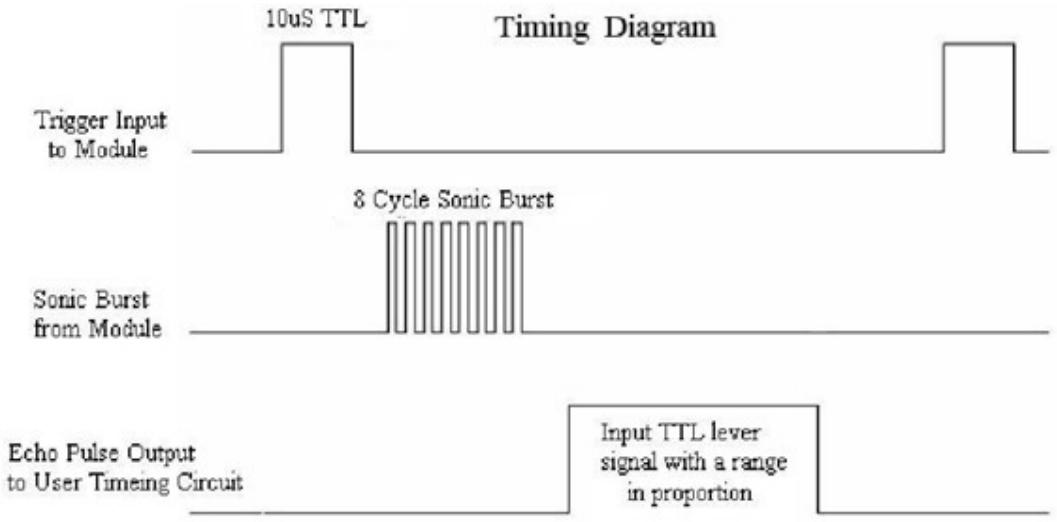
5.1.12 SPECIFICATIONS

Pin No.	Signal
1	VCC (5V supply)
2	Trigger Pulse Input
3	Echo Pulse Output
4	GND (0V)

5.1.13 OPERATION

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8-cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion.You can calculate the range through the time interval between sending trigger signal and receiving echo signal.

we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



5.1.13.1 Timing Diagram

5.2 SERVO MOTOR

5.2.1 SERVO MOTOR THEORY

There are some special types of application of electrical motor where rotation of the motor is required for just a certain angle not continuously for long period of time. For these applications some special types of motor are required with some special arrangement which makes the motor to rotate a certain angle for a given electrical input (signal). For this purpose, servo motor comes into picture. This is normally a simple DC motor which is controlled for specific angular rotation with help of additional servomechanism (a typical closed loop feedback control system). Now day's servo system has huge industrial applications. Servo motor applications are also commonly seen in remote controlled toy cars for controlling direction of motion and it is also very commonly used as the motor which moves the tray of a CD or DVD player. Beside these there are other hundreds of servo motor applications we see in our daily life. The main reason behind using a servo is that it provides angular precision, i.e. it will only rotate as much we want and then stop and wait for next signal to take further action. This is unlike a normal electrical motor which starts rotating as and when power is applied to it and the rotation continues until we switch off the power.

We cannot control the rotational progress of electrical motor; but we can only control the speed of rotation and can turn it ON and OFF.

5.2.2 SERVO MOTOR WORKING PRINCIPLE

Before understanding the working principle of servo motor, we should understand first the basic of servomechanism. A servo system mainly consists of three basic components – a controlled device, a output sensor, a feedback system This is an automatic closed loop control system. Here instead of controlling a device by applying variable input signal, the device is controlled by a feedback signal generated by comparing output signal and reference input signal. When reference input signal or command signal is applied to the system, it is compared with output reference signal of the system produced by output sensor, and a third signal produced by feedback system. This third signal acts as input signal of controlled device. This input signal to the device presents as long as there is a logical difference between reference input signal and output signal of the system. After the device achieves its desired output, there will be no longer logical difference between reference input signal and reference output signal of the system. Then, third signal produced by comparing theses above said signals will not remain enough to operate the device further and to produce further output of the system until the next reference input signal or command signal is applied to the system. Hence the primary task of a servomechanism is to maintain the output of a system at the desired value in the presence of disturbances.

5.2.3 WORKING PRINCIPLE OF SERVO MOTOR

A servo motor is basically a DC motor (in some special cases it is AC motor) along with some other special purpose components that make a DC motor a servo. In a servo unit, you will find a small DC motor, a potentiometer, gear arrangement and an intelligent circuitry. The intelligent circuitry along with the potentiometer makes the servo to rotate according to our wishes.

As we know, a small DC motor will rotate with high speed but the torque generated by its rotation will not be enough to move even a light load. This is where the gear system inside a servomechanism comes into picture. The gear mechanism will take high input speed of the motor (fast) and at the output, we will get a output speed which is slower than original input speed but more practical and widely applicable. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. This output port of the potentiometer is connected with one of the input terminals of the error detector amplifier. Now an electrical signal is given to another input terminal of the error detector amplifier. Now difference between these two signals, one comes from potentiometer and another comes from external source, will be amplified in the error detector amplifier and feeds the DC motor. This amplified error signal acts as the input power of the dc motor and the motor starts rotating in desired direction. As the motor shaft progresses the potentiometer knob also rotates as it is coupled with motor shaft with help of gear arrangement. As the position of the potentiometer knob changes there will be an electrical signal produced at the potentiometer port. As the angular position of the potentiometer knob progresses the output or feedback signal increases. After desired angular position of motor shaft the potentiometer knob is reaches at such position the electrical signal generated in the potentiometer becomes same as of external electrical signal given to amplifier. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer. As the input signal to the motor is nil at that position, the motor stops rotating. This is how a simple conceptual servo motor works.

5.2.4 SERVO MOTOR CONTROL

For understanding servo motor control let us consider an example of servomotor that we have given a signal to rotate by an angle of 45° and then stop and wait for

further instruction. The shaft of the DC motor is coupled with another shaft called output shaft, with help of gear assembly. This gear assembly is used to step down the high rpm of the motor's shaft to low rpm at output shaft of the servo system.

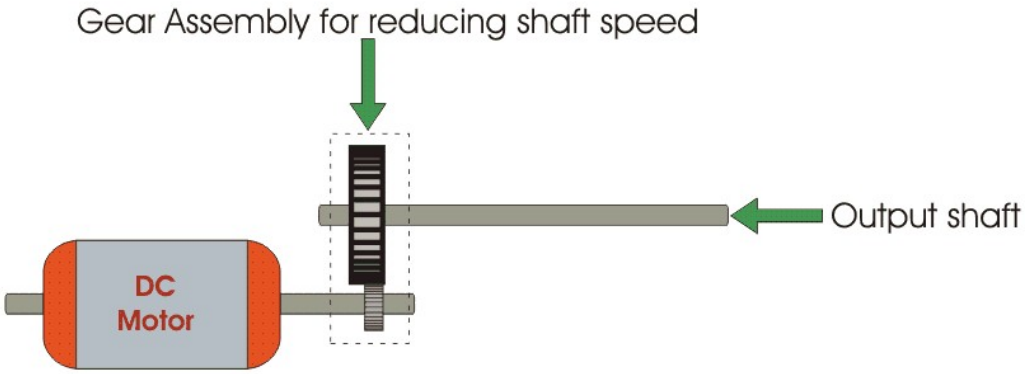


Fig 5.2.4.1 Servo motor control

The voltage adjusting knob of a potentiometer is so arranged with the output shaft by means of another gear assembly, that during rotation of the shaft, the knob also rotates and creates a varying electrical potential according to the principle of potentiometer. This signal i.e. electrical potential is increased with angular movement of potentiometer knob along with the system shaft from 0° to 45°. This electrical potential or voltage is taken to the error detector feedback amplifier along with the input reference commends i.e. input signal voltage.

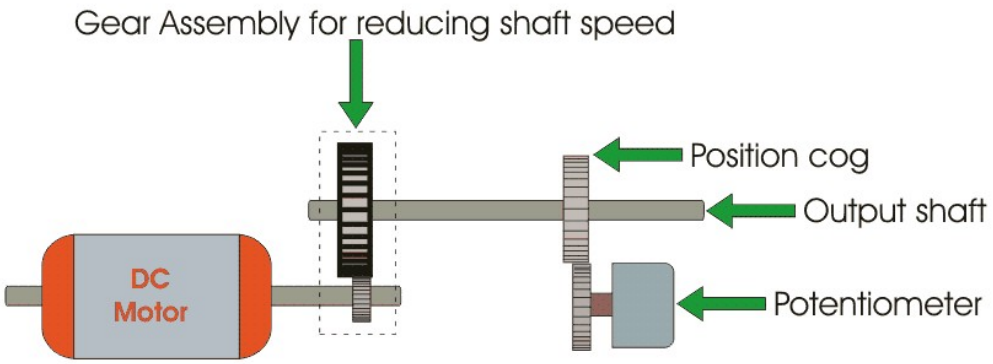


Fig 5.2.4.2 Servo Motor Control

As the angle of rotation of the shaft increases from 0° to 45° the voltage from potentiometer increases. At 45° this voltage reaches to a value which is equal to the given input command voltage to the system. As at this position of the shaft, there is no difference between the signal voltage coming from the potentiometer and reference input voltage (command signal) to the system, the output voltage of the amplifier becomes zero.

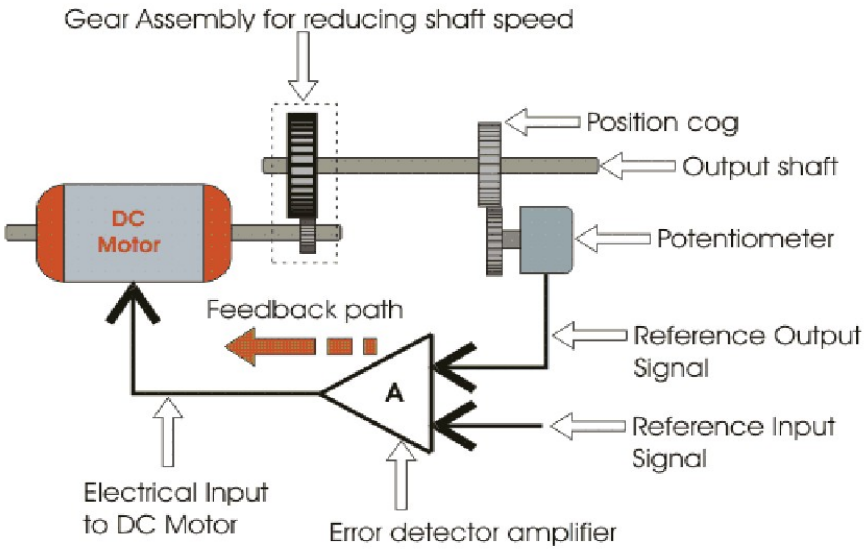


Fig 5.2.4.3 Servo motor control

5.2.5 SERVO MOTOR CIRCUIT

As per the picture given above the output electrical voltage signal of the amplifier, acts as input voltage of the DC motor. Hence the motor will stop rotating after the shaft rotates by 45°. The motor will be at this rest position until another command is given to the system for further movement of the shaft in desired direction. From this example we can understand the most basic servo motor theory and how servo motor control is achieved.

From this basic working principle of servo motor it can be concluded. The shaft of the servo is connected to a potentiometer. The circuitry inside the servo, to which the potentiometer is connected, knows the position of the servo. The current position will be compared with the desired position continuously with the help of an Error Detection Amplifier. If a mismatch is found, then an error signal is provided at the output of the error amplifier and the shaft will rotate to go the exact location required. Once the desired location is reached, it stops and waits.

5.2.6 CONTINUOUS ROTATION SERVO MOTORS

Continuous rotation servo motors are actually a modified version of what the servos are actually meant to do, that is, control the shaft position. The 360° rotation servos are actually made by changing certain mechanical connections inside the servo. However, certain manufacturer like parallax sells these servos as well. With the continuous rotation servo, you can only control the direction and speed of the servo, but not the position. Two of the most popular Servo motor manufacturers are FUTABA and HITEC.

5.2.7 GAS SENSOR

The GAS sensor is a chemical optical sensor utilizing the acidic nature of gas for detection. It consists of a gas-permeable membrane in which a pH-sensitive luminescence dye is immobilized together with a buffer and an inert reference luminescent dye. Gas permeating into the membrane changes the internal pH of the buffer. With this changes the luminescence of the pH- sensitive dye. Together with the inert reference dye internal referencing is made for detection of the luminescence lifetime of the sensor. The measurement signal detected by the pGAS mini correlates to the partial pressure of gas ambient.

- Modified atmospheres
- Indoor air quality
- Stowaway detection
- Cellar and gas stores
- Marine vessels
- Greenhouses
- Land fill gas
- Confined spaces
- Cryogenics
- Ventilation management
- Mining
- Rebreathers (SCUBA)
    - For HVAC applications, Gas sensors can be used to monitor the quality of air and the tailored need for fresh air, respectively.
    - Measuring gas levels indirectly determines how many people are in a room, and ventilation can be adjusted accordingly. See demand-controlled ventilation (DCV).
    - In applications where direct temperature measurement is not applicable, NDIR sensors can be used.
- The sensors absorb ambient infrared radiation (IR) given off by a heated surface.

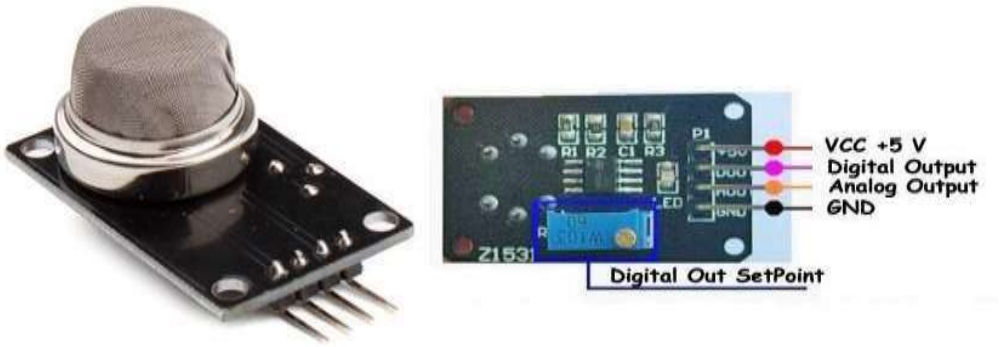


Fig 5.2.7.1 Gas Sensor

MQ2 SPECIFICATION

MQ-2 gas sensor has high sensitivity to LPG, Propane and Hydrogen, also could be used to Methane and other combustible steam, it is with low cost and suitable for different application.

Good sensitivity to Combustible gas in wide range

* High sensitivity to LPG, Propane and Hydrogen
* Long life and low cost
* Simple drive circuit

CHAPTER 6
SYSTEM SPECIFICATIONS

6.1 SOFTWARE REQUIREMENTS

6.1.1 ARDUINO IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - 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 and Genuine hardware to upload programs and communicate with them.

WRITING SKETCHES

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension. ion. The editor 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 Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension. pde. It is possible to open these files with version 1.0; you will be prompted to save the sketch with the. ino extension on save.

- **Verify**
  Checks your code for errors compiling it.
- **Upload**
  Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

- **New**
  Creates a new sketch.
- **Open**
  Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.
  Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketch book menu instead
- **Save**
  Saves your sketch.
- **Serial Monitor**
  Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

FILE

- **New**
  Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- **Open**
  Allows to load a sketch file browsing through the computer drives and folders.
- **Open Recent**
  Provides a short list of the most recent sketches, ready to be opened.

- **Sketchbook**

  Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.
- **Examples**

  Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.
- **Close**

  Closes the instance of the Arduino Software from which it is clicked.
- **Save**

  Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as." window.
- **Save as...**

  Allows saving the current sketch with a different name.
- **Page Setup**

  It shows the Page Setup window for printing.
- **Print**

  Sends the current sketch to the printer according to the settings defined in Page Setup.
- **Preferences**

  Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.
- **Quit**

  Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

EDIT

- **Undo/Redo**

  Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.
- **Cut**

  Removes the selected text from the editor and places it into the clipboard.
- **Copy**

  Duplicates the selected text in the editor and places it into the clipboard.
- **Copy for Forum**

  Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.
- **Copy as HTML**

  Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.
- **Paste**

  Puts the contents of the clipboard at the cursor position, in the editor.
- **Select All**

  Selects and highlights the whole content of the editor.
- **Comment/Uncomment**

  Puts or removes the // comment marker at the beginning of each selected line.
- **Increase/Decrease Indent**

  Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.
- **Find**

  Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

- **Find Next**

  Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.
- **Find Previous**

  Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

SKETCH

- **Verify/Compile**

  Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.
- **Upload**

  Compiles and loads the binary file onto the configured board through the configured Port.
- **Upload Using Programmer**

  This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Bootloader command must be executed.
- **Export Compiled Binary**

  Saves a .hex file that may be kept as archive or sent to the board using other tools.
- **Show Sketch Folder**

  Opens the current sketch folder.
- **Include Library**

  Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

- **Add File...**

  Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side on the toolbar.

TOOLS

- **Auto Format**

  This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.
- **Archive Sketch**

  Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.
- **Fix Encoding & Reload**

  Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.
- **Serial Monitor**

  Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.
- **Board**

  Select the board that you're using. See below for descriptions of the various boards.
- **Port**

  This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.
- **Programmer**

  For selecting a hardware programmer when programming a board or chip and

not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

- **Burn Bootloader**
  The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuine board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

HELP

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

- Find in Reference
  This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

SKETCHBOOK

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ion file extension. Previous versions use the. pde extension. You may still open. pde named files in version 1.0 and later, the software will automatically rename the extension to. ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

UPLOADING

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/try. usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Emelyanova or earlier USB board), or/dev/tty. USA19QW1b1P1.1 (for a serial board connected with a Key span USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the sports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx,/dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the File menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to

upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

LIBRARIES

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #includestatements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

THIRD-PARTY HARDWARE

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "Arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

SERIAL MONITOR

Displays serial data being sent from the Arduino or Genuine board (USB or serial board). To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down that matches the rate passed to Serial. Begin in your sketch. Note that on Windows, Mac or Linux, the Arduino or Genuine board will reset (rerun your sketch execution to the beginning) when you connect with the serial monitor.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

PREFERENCES

Some preferences can be set in the preferences dialog (found under the Arduino menu on the Mac, or File on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

LANGUAGE SUPPORT

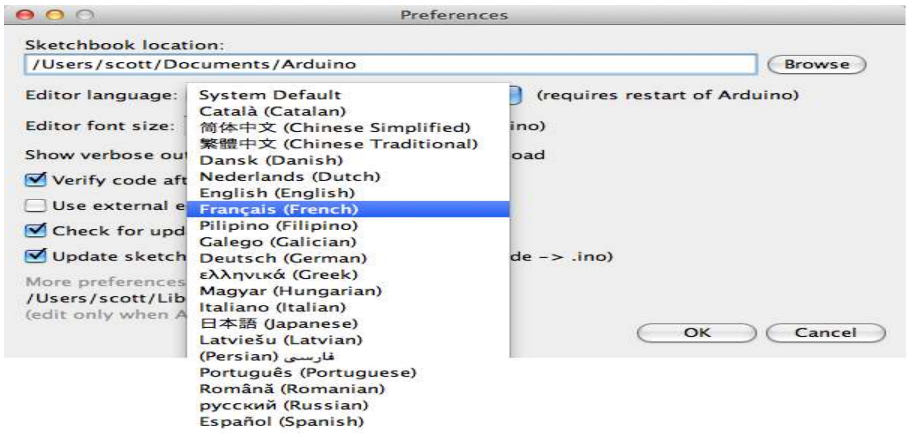


Fig 6.1.1 Language support

Since version 1.0.1, the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your

operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

If you would like to change the language manually, start the Arduino Software (IDE) and open the Preferences window. Next to the Editor Language there is a dropdown menu of currently supported languages. Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English.

You can return the software to its default setting of selecting its language based on your operating system by selecting System Default from the Editor Language drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.

BOARDS

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection, you'll want to check it before burning the bootloader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built-in support for the boards in the following list, all based on the AVR Core. The Boards Manager included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- Arduino Yùn

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino/Genuine Uno

  An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Diecimila or Demilune w/ ATmega168

  An ATmega168 running at 16 MHz with auto-reset.
- Arduino Nano w/ ATmega328

  An ATmega328 running at 16 MHz with auto-reset. Has eight analog inputs.
- Arduino/Genuine Mega 2560

  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- Arduino Mega

  An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- Arduino Mega ADK

  An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- Arduino Leonardo

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino Micro

  An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- Arduino Esplora

  An ATmega32u4 running at 16 MHz with auto-reset.

- Arduino Mini w/ ATmega328

  An ATmega328 running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Ethernet

  Equivalent to Arduino UNO with an Ethernet shield: An ATmega328 running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Fio

  An ATmega328 running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ATmega328, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino BT w/ ATmega328

  ATmega328 running at 16 MHz the bootloader burned (4 KB) includes codes to initialize the on-board Bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM.
- Lilypad Arduino USB

  An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
- Lilypad Arduino

  An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328

  An ATmega328 running at 16 MHz with auto-reset. Equivalent to Arduino Demilune or Nano w/ ATmega328; 6 Analog In, 14 Digital I/O and 6 PWM.
- Arduino NG or older w/ ATmega168

  An ATmega168 running at 16 MHz without auto-reset. Compilation and upload is equivalent to Arduino Decimal or Demilune w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.

- Arduino Robot Control

  An ATmega328 running at 16 MHz with auto-reset.
- Arduino Robot Motor

  An ATmega328 running at 16 MHz with auto-reset.
- Arduino Gemma

  An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

6.1.2 EMBEDDED C

Embedded C is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems. Historically, embedded C programming requires nonstandard extensions to the C language in order to support exotic features such as fixed-point arithmetic, multiple distinct memory banks, and basic I/O operations.

In 2008, the C Standards Committee extended the C language to address these issues by providing a common standard for all implementations to adhere to. It includes a number of features not available in normal C, such as, fixed-point arithmetic, named address spaces, and basic I/O hardware addressing.

Embedded C uses most of the syntax and semantics of standard C, e.g., main () function, variable definition, datatype declaration, conditional statements (if, switch, case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc.

A Technical Report was published in 2004 and a second revision in 2006.

NECESSITY

During infancy years of microprocessor-based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check for correct execution of the program. Some 'very fortunate' developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well. As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the embedded programming language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

As assembly language programs are specific to a processor, assembly language didn't offer portability across systems. To overcome this disadvantage, several high-level languages, including C, came up.

Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn't find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications. Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of C in the embedded systems, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of embedded systems using C. Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly difficult to manage in assembly language. Finding good assembly programmers has also become difficult nowadays. Hence high-level languages are preferred for embedded systems programming.

ADVANTAGES

- It is small and simpler to learn, understand, program and debug.
- Compared to assembly language, C code written is more reliable and scalable, more portable between different platforms.
- C compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
- Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
- As C combines functionality of assembly language and features of high-level languages, C is treated as a 'middle-level computer language' or 'high level assembly language'.
- It is fairly efficient.
- It supports access to I/O and provides ease of management of large embedded projects.
- Java is also used in many embedded systems but Java programs require the Java Virtual Machine (JVM), which consumes a lot of resources. Hence it is not used for smaller embedded devices.

EMBEDDED SYSTEMS PROGRAMMING

Embedded systems programming is different from developing applications on desktop computers. Key characteristics of an embedded system, when compared to PCs, are as follows:

- Embedded devices have resource constraints (limited ROM, limited RAM, limited stack space, less processing power)
- Components used in embedded system and PCs are different; embedded systems typically use smaller, less power consuming components.

- Embedded systems are more tied to the hardware.

Two salient **features of Embedded Programming** are code speed and code size. Code speed is governed by the processing power, timing constraints, whereas code size is governed by available program memory and use of programming language. Goal of embedded system programming is to get maximum features in minimum space and minimum time.

Embedded systems are programmed using different type of languages:

- Machine Code
- Low level language, i.e., assembly
- High level language like C, C++, Java, Ada, etc.
- Application-level language like Visual Basic, scripts, Access, etc.

Assembly language maps mnemonic words with the binary machine codes that the processor uses to code the instructions. Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly difficult to manage in assembly language. Finding good assembly programmers has also become difficult nowadays. Hence high-level languages are preferred for embedded systems programming.

Use of **C in embedded systems** is driven by following advantages

- It is small and reasonably simpler to learn, understand, program and debug.

- C Compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
- Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
- As C combines functionality of assembly language and features of high-level languages, C is treated as a 'middle-level computer language' or 'high level assembly language'
- It is fairly efficient
- It supports access to I/O and provides ease of management of large embedded projects.

Many of these advantages are offered by other languages also, but what sets C apart from others like Pascal, FORTRAN, etc. is the fact that it is a middle level language; it provides direct hardware control without sacrificing benefits of high-level languages.

Compared to other high-level languages, C offers more flexibility because C is relatively small, structured language; it supports low-level bit-wise data manipulation. Compared to assembly language, C Code written is more reliable and scalable, more portable between different platforms (with some changes). Moreover, programs developed in C are much easier to understand, maintain and debug. Also, as they can be developed more quickly, codes written in C offers better productivity. C is based on the philosophy 'programmers know what they are doing'; only the intentions are to be stated explicitly. It is easier to write good code in C & convert it to an efficient assembly code (using high quality compilers) rather than writing an efficient code in assembly itself. Benefits of assembly language programming over C are negligible when we compare the ease with which C programs are developed by programmers.

Objected oriented language, C++ is not apt for developing efficient programs in

resource constrained environments like embedded devices. Virtual functions & exception handling of C++ are some specific features that are not efficient in terms of space and speed in embedded systems. Sometimes C++ is used only with very few features, very much as C.

And, also an object-oriented language, is different than C++. Originally designed by the U.S. DOD, it didn't gain popularity despite being accepted as an international standard twice (Ada83 and Ada95). However, Ada language has many features that would simplify embedded software development.

Java is another language used for embedded systems programming. It primarily finds usage in high-end mobile phones as it offers portability across systems and is also useful for browsing applications. Java programs require Java Virtual Machine (JVM), which consume lot of resources. Hence it is not used for smaller embedded devices.

Dynamic C and B# are some proprietary languages which are also being used in embedded applications.

Efficient embedded C programs must be kept small and efficient; they must be optimized for code speed and code size. Good understanding of processor architecture embedded C programming and debugging tools facilitate this.

DIFFERENCE BETWEEN C AND EMBEDDED C

Though **C and embedded C** appear different and are used in different contexts, they have more similarities than the differences. Most of the constructs are same; the difference lies in their applications.

C is used for desktop computers, while **embedded C** is for microcontroller-based applications. Accordingly, C has the luxury to use resources of a desktop PC like memory, OS, etc. While programming on desktop systems, we need not bother about memory. However, embedded C has to use with the limited resources (RAM, ROM,

I/Os) on an embedded processor. Thus, program code must fit into the available program memory. If code exceeds the limit, the system is likely to crash.

Compilers for C (ANSI C) typically generate OS dependent executables. **Embedded C** requires compilers to create files to be downloaded to the microcontrollers/microprocessors where it needs to run. Embedded compilers give access to all resources which is not provided in compilers for desktop computer applications.

Embedded systems often have the real-time constraints, which is usually not there with desktop computer applications.

Embedded systems often do not have a console, which is available in case of desktop applications.

So, what basically is different while programming with **embedded C** is the mindset; for embedded applications, we need to optimally use the resources, make the program code efficient, and satisfy real time constraints, if any. All this is done using the basic constructs, syntaxes, and function libraries of 'C'.

6.1.3 BLYNK IOT

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. It's really simple to set everything up and you'll start tinkering in less than 5 mins. 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 Wi-Fi, Ethernet or this new ESP8266 chip, Blynk will get you online and ready for the Internet Of Your Things.

Fig 6.1.3.1 Blynk Application

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: – It allows you to create amazing interfaces for your projects using various widgets which are provided.

Blynk Server: – It is responsible for all the communications between the smartphone and hardware. You can use the 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: – It enables communication, for all the popular hardware platforms, with the server and process all the incoming and outcoming commands. Now imagine, every time you press a Button in the Blynk app, the message travels to 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.

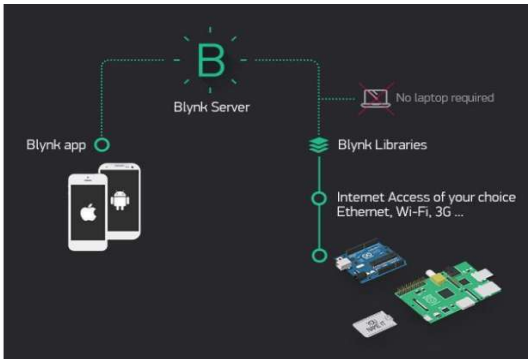


Fig 6.1.3.2 Data Processing

Characteristics of Blynk are:

- Similar API & UI for all supported hardware & devices
- Connection to the cloud can be done using Ethernet, Wi-Fi, Bluetooth, BLE and USB (Serial)
- Set of easy-to-use Widgets
- Direct pin manipulation with no code writing
- Easy to integrate and add new functionality using virtual pins
- History data monitoring via History Graph widget
- Device-to-Device communication using Bridge Widget
- Sending emails, tweets, push notifications, etc.

CHAPTER 7

CONCLUSION

The rapid growth of urban populations has significantly increased the challenges associated with waste management. Traditional systems, based on fixed schedules and manual inspections, are no longer efficient in addressing the growing needs of cleanliness and hygiene in cities. Overflowing garbage bins and undetected harmful gas emissions contribute to environmental pollution and pose serious health risks. Therefore, there is a strong need for smarter, automated solutions that can adapt to real-time waste conditions.

The proposed IoT-enabled garbage management system effectively addresses these challenges by integrating ultrasonic sensors for monitoring garbage levels and gas sensors for detecting harmful emissions. The use of a Nedelcu microcontroller ensures seamless data processing and transmission to a cloud-based platform, enabling timely alerts and proactive responses. Additionally, the automation of the bin lid through a servo motor enhances hygiene by reducing physical contact, making the system more user-friendly and safer.

By leveraging real-time data transmission and mobile/web dashboard integration, the system offers a scalable, efficient, and sustainable solution for modern urban waste management. It not only ensures timely waste collection and reduces the risks associated with waste overflow but also contributes to the development of smart cities. Overall, the project demonstrates how IoT and automation can be harnessed to create a cleaner, safer, and more sustainable living environment.

APPENDIX

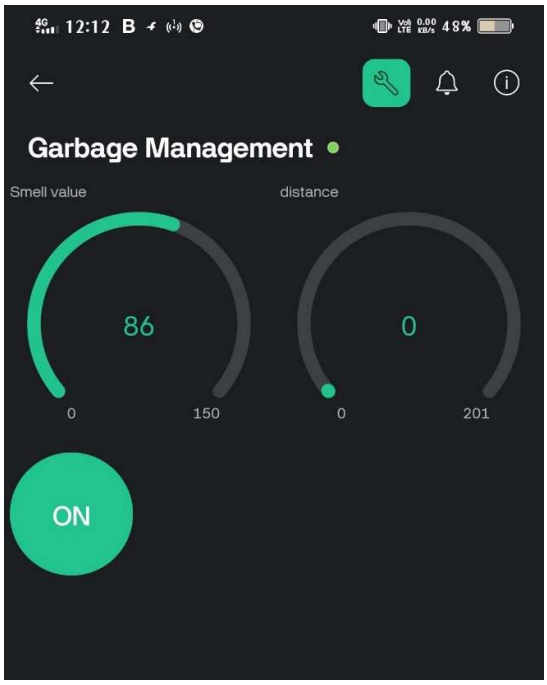
8.1 SOURCE CODE

```
#define BLYNK_TEMPLATE_ID "TMPL3IMXxlbEc"
#define BLYNK_TEMPLATE_NAME "Garbage Management"
#define BLYNK_AUTH_TOKEN "CmaocZV0XpUO5v4v-Q0camlo3hLtV9qB"
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Servo.h>

char auth [] = BLYNK_AUTH_TOKEN;
char ssid [] = "PROJECT";
char pass [] = "12341234";
// Define pins
#define trig D2   // Ultrasonic trigger pin
#define echo D1   // Ultrasonic echo pin
#define gas A0    // Gas sensor input
#define servoPin D3

Servo myServo;
int duration;
int distance;
int state = 0;
void setup () {
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);

  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(gas, INPUT);
```

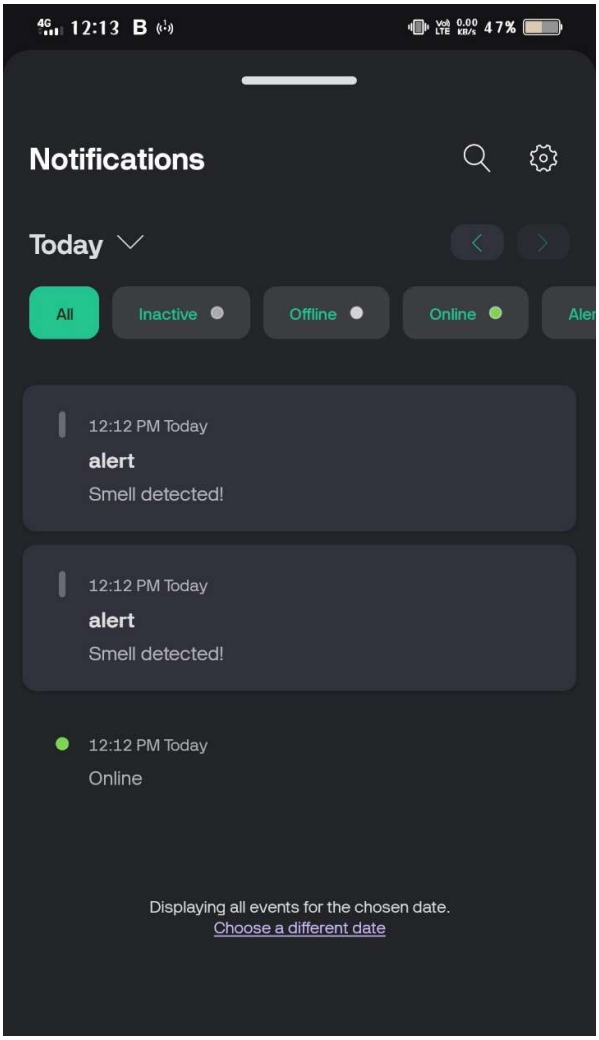
```
  pinMode(servoPin, OUTPUT);

  myServo.attach(servoPin);
  myServo.write(180);
}

void loop () {
  Blynk.run();
  int gasValue = analogRead(gas);

  digitalWrite(trig, LOW);
  delay (10);
  digitalWrite(trig, HIGH);
  delay (10);
  digitalWrite(trig, LOW);

  duration = pulseIn(echo, HIGH);
  distance = duration * 0.030 / 2;

  Serial.print("Gas Value: ");
  Serial.print(gasValue);
  Serial.print(" || ");
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  Blynk.virtualWrite(V0, distance);
  Blynk.virtualWrite(V1, gasValue);
if (state == 10)
{
  if (distance > 0 && distance <= 3) {
    for (int i = 180; i >= 0; i--) {
```

```
    myServo.write(0);
    delay (15);
   }
   delay (1000);
   for (int i = 0; i <= 180; i++) {
    myServo.write(i);
    delay (15);
   }
   Serial.println("Bin is closed");
   Blynk.logEvent("alert", " Bin is closed");
  }
  else if (gasValue > 90) {
   Serial.println("Smell detected!");
   myServo.write(0);
   Blynk.logEvent("alert", "Smell detected!");
  }
  else {
   myServo.write(180);
  }
}
  delay (1000);
}

BLYNK_WRITE(V2) {
  if (param.asInt() == 1) {
   state = 10;
  }
  else {
   state = 0;
  }
}
```

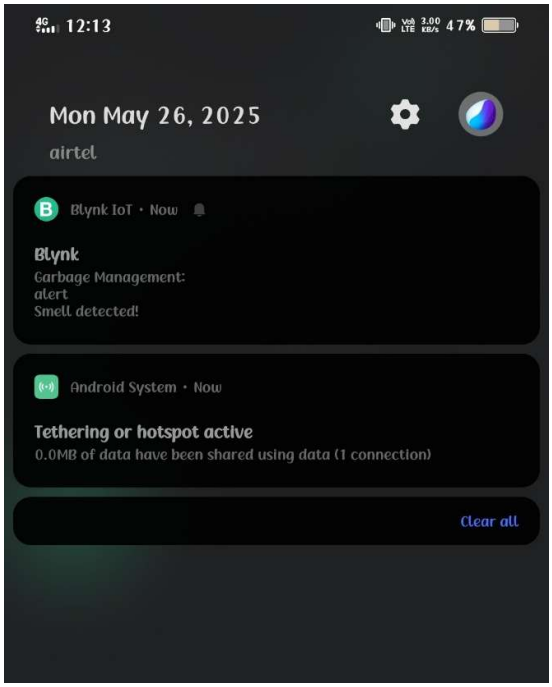
8.2 SCREENSHOTS



Application Interface

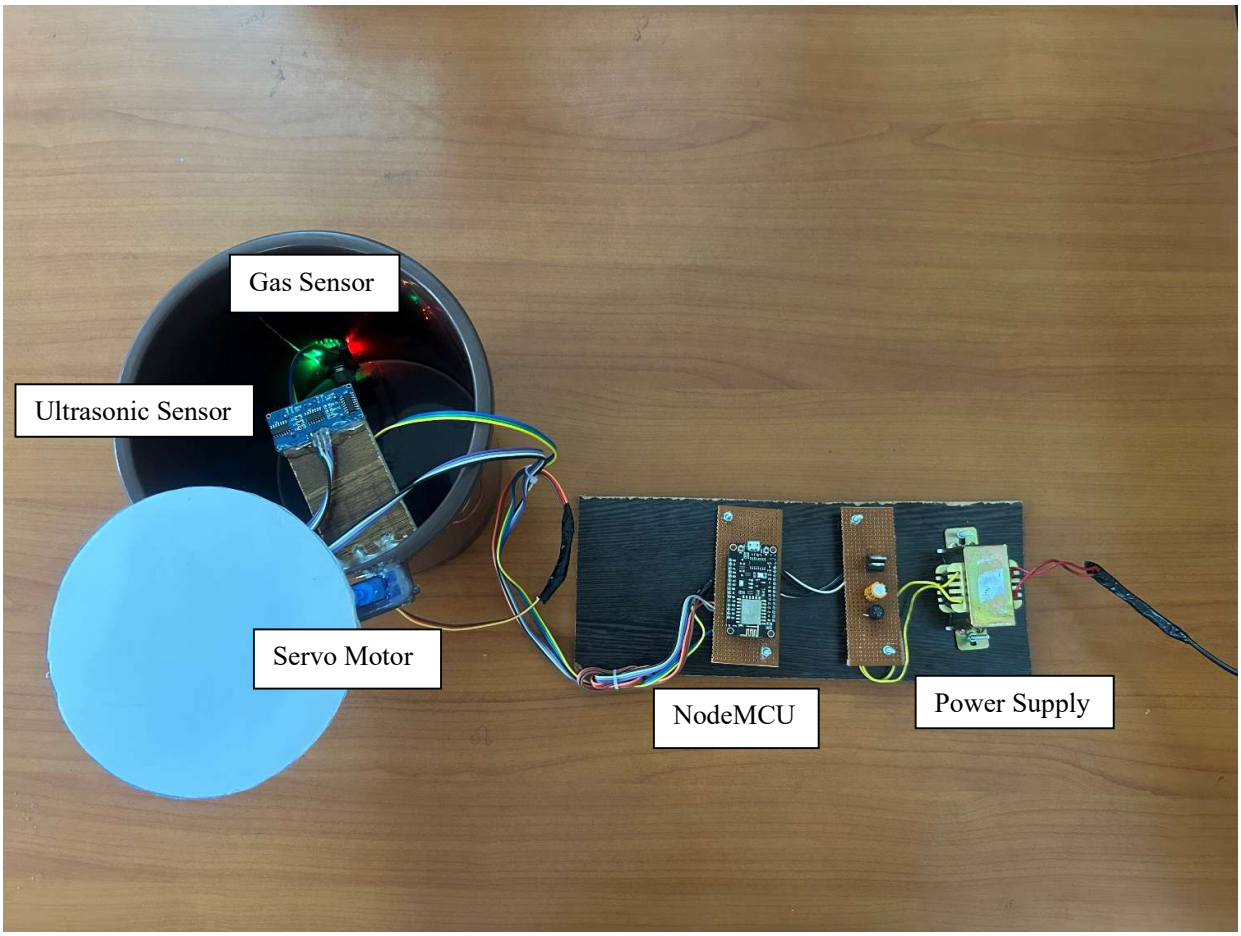


Smell detection notification



Notification alert

8.3 HARDWARE IMPLEMENTATION



REFERENCES

• S. **Agarwal and B. Bhushan (2020)**, "Waste Monitoring with Real-Time Notifications using IoT," *Proc. IEEE Int. Conf. Emerg. Trends Eng. Technol.*, pp. 110–115.

• L. **Banerjee et al. (2020)**, "Waste Segregation and Level Monitoring Using IoT," *Proc. Int. Conf. Innov. Comput. Technol.*, pp. 255–259.

• S. **Bansal et al. (2023)**, "IoT-Driven Smart Bin Management System," *Proc. Med. Image Comput. Comput.-Assist. Interv. (MICCAI)*, pp. 204–215.

• N. **R. Gopal et al. (2019)**, "Automated Smart Bin Using NodeMCU and Ultrasonic Sensor," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 9, no. 2, pp. 400–405.

• M. **R. Gupta et al. (2021)**, "Design and Implementation of Smart Garbage Monitoring System," *Int. J. Comput. Appl.*, vol. 183, no. 23, pp. 5–9.

• S. **S. M. Ilyas et al. (2023)**, "Self-Supervised Learning for Garbage Monitoring," *IEEE J.*

• R. **K. Jha and M. M. Rashid (2019)**, "Low-Cost IoT Framework for Smart Waste Collection," *Proc. Int. Conf. Smart Grid Clean Energy Technol.*, pp. 345–350.

• V. **R. Kumar and P. Thomas (2019)**, "Cloud-Enabled Smart Bin Alert System for Clean Cities," *Proc. IEEE Glob. Humanit. Technol. Conf.*, pp. 183–187.

• T. **Li and Y. Zhang (2021)**, "Gas Sensor Integration for Urban Waste Detection," *IEEE Sens. J.*, vol. 21, no. 15, pp. 16589–16597.

• K. **Maheshwaran et al. (2024)**, "Smart Waste Management Using IoT and Sensor Integration," *Int. J. Eng. Res. Technol.*, vol. 43, no. 2, pp. 1123–1135.

• M. **S. Nafiz et al. (2022)**, "Deep Learning-Based AI Assistants for Waste Level Monitoring," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 8092–8105.

• P. **V. Nair and R. Ramesh (2021)**, "Urban Waste Monitoring and Alert System Using Wi-Fi Modules," *Proc. IEEE Smart Sustain. Cities Conf.*, pp. 221–225.

• H. **Patel and R. Shah (2021)**, "IoT Based Smart Waste Monitoring System," *Proc. Int. Conf. Smart Technol. Sustain. Dev.*, pp. 125–130.

• A. **Sharma and K. Singh (2020)**, "Real-Time Solid Waste Monitoring Using Wireless Sensor Networks," *Int. J. Smart Sens. Intell. Syst.*, vol. 10, no. 3, pp. 567–574.

• G. **White et al. (2022)**, "Waste Classification at the Edge for Smart Bins," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 3121–3130.