DESIGN AND IMPLEMENTATION OF GPS CONTROLLED ENVIRONMENT MONITORING ROBOT SYSTEM BASED ON IOT

A Project Report

submitted

in partial fulfillment of the requirements for

the award of the degree of

Bachelor of Technology

in

Electronics & Communication Engineering

(JNTUA, Anantapuramu)

By

GUVVALA RAGHAVENDRA	**192N1A0408**
POREDDY BHARATH SIMHA REDDY	**192N1A0415**
BOYA JAYA KUMAR	**192N1A0401**
GOLLAPALLI SAI BHARATHWAJ	**192N1A0405**
N SHAIK RESHMA SIDDIKHA	**202N5A0401**

Under the guidance of

Mr M. RANGA SWAMY M .Tech, (Ph. D)

Assoc. Professor, Dept of ECE

BITS, KURNOOL

Department of Electronics & Communication Engineering

BRINDAVAN

INSTITUTE OF TECHNOLOGY & SCIENCE (BITS-KNL)

(Approved by AICTE & Affiliated to JNTUA)

NH-7, PEDDATEKUR, KURNOOL-518218

May 2023

BRINDAVAN

INSTITUTE OF TECHNOLOGY & SCIENCE (BITS-KNL)

NH-7, PEDDATEKUR, KURNOOL - 518218

CERTIFICATE

This is to certify that the Project Report **DESIGN AND IMPLEMENTATION OF GPS CONTROLLED ENVIRONMENT MONITORING ROBOT SYSTEM BASED ON IOT** is the bonafide record of the Project Work carried out under my Guidance and Supervision by

1.	**GUVVALA RAGHAVENDRA**	**192N1A0408**
2.	**POREDDY BHARATH SIMHA REDDY**	**192N1A0415**
3.	**BOYA JAYA KUMAR**	**192N1A0401**
4.	**GOLLAPALLI SAI BHARATHWAJ**	**192N1A0405**
5.	**N SHAIK RESHMA SIDDIKHA**	**202N5A0401**

in partial fulfillment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering by JNTUA, Anantapuramu and, is submitted in the Department of Electronics and Communication Engineering, Brindavan Institute of Technology & Science, NH-7, Peddatekur, Kurnool.

Head of the Department	**Guide**
Mr. M.RANGASWAMY M. Tech (Ph.D)	**Mr. M.RANGASWAMY M. Tech (Ph.D)**
Assoc. Professor, Dept of ECE	Assoc. Professor, Dept of ECE
Brindavan	Brindavan
Institute of Technology & Science	Institute of Technology & Science
NH-7, Peddatekur, Kurnool.	NH-7, Peddatekur, Kurnool.

Signature of External Examiner:

ACKNOWLEDGMENTS

Our hard work never shines if we do not convey my heartfelt gratitude to those people from whom we got considerable support and encourage during this Project report.

We would like to express our gratitude to our Guide **M. Ranga Swamy** **M.Tech (Ph.D)** **Associate Professor** of Electronics and Communication Engineering, **Brindavan Institute of Technology& Science**, for his constant support and guidance throughout the project work.

We would like to thank **Associate Prof. M. Ranga Swamy**, Head of the Department, Electronics and Communication Engineering, **Brindavan Institute of Technology & Science**, Kurnool for his valuable suggestions from time to time during this Project Work.

We also express our special thanks to **Prof. N. Siva Prasad Reddy (Academic Director) and Dr. A.V. Prathap Kumar Principal**, **Brindavan Institute of Technology & Science**, for their support and encouragement to complete our Project Work.

We would like to thank **Mr. R. David Pranay** is the Project Coordinator, for the help throughout the Project.

We are happy to express our sincere thanks to all teaching and non-teaching staff, Department of Electronics and Communication Engineering for their help. Finally, we thank to all those who helped directly or indirectly in making endeavor a success

.

G. Raghavendra **P. Bharath Simha Reddy**

B. Jaya Kumar **G. Sai Bharathwaj**

N. Shaik Reshma Siddikha

INDEX

TABLE OF CONTENTS

NAME OF CONTENTS	PAGE NO
ABSTRACT	**I**
LIST OF FIGURES	**II**
LIST OF TABLES	**III**
CHAPTER-1	
1. INTRODUCTION	**1**
1.1 Introduction to the Project	1
1.2 Aim of the Proposed System	2
CHAPTER-2	
2. LITERATURE REVIEW	**3**
CHAPTER-3	
3. INTRODUCTION ABOUT EMBEDDED SYSTEM	**5**
3.1 Introduction to Embedded System	5
3.2 Overview of Embedded System	6
CHAPTER-4	
4. DESIGN OF HARDWARE	**9**
4.1 Arduino	9
4.2 Power Supply	13
4.2.1 Transformer	14
4.2.2 Rectifier	15
4.2.3 Filter	15
4.2.4 Voltage Regulator	16
4.3 LCD	16
4.3.1 PIN Functions	16
4.3.2 LCD Screen	18

4.4 Temperature Sensor 19

4.5 ESP8266 Wi-fi 21

4.6 GPS 27

4.7 MQ2-Sensor 35

4.8 Ultrasonic Sensor 45

4.9 Buzzer 47

CHAPTER-5

5. DESIGN OF SOFTWARE 53

5.1 Introduction to Arduino IDE Software 53

5.2 Software Steps 53

CHAPTER-6

6. PROJECT DESCRIPTION 58

6.1 Block Diagram 58

6.2 Software Requirements 58

6.3 Hardware Requirements 59

6.4 Working 59

CHAPTER-7

7. RESULT 61

CHAPTER-8

8. CONCLUSION 63

CHAPTER-9

9. FUTURE SCOPE 64

CHAPTER-10

10. REFERENCES 65

SOURCE CODE

ABSTRACT

Environmental monitoring systems are often designed to measure and log the current status of an environment or to establish trends in environmental parameters. In this project, we proposed an autonomous robotic system that is designed and implemented to monitor environmental parameters such as temperature, humidity, air quality, and harmful gas concentration. The robot has GPS coordinates, and it can store data on the IOT platform. The mobile robot is controlled by a smartphone which runs an app built on the Android platform. The whole system is realized using a cost-effective-based embedded system called Arduino which communicates through a wireless network to the IOT platform, where data are stored, processed and can be accessed using a computer or any smart device from anywhere. The system can update sensor data to IOT server every high of each sensor and gives the GPS location. The stored data can be used for further analysis of the reduction of pollution, save energy and provide an overall living environment enhancement. The robotic system has designed for cost effective remote monitoring environmental parameters without any human intervention to avoid health risk efficiently. A proof-of-concept prototype has been developed to illustrate the effectiveness of the proposed system.

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
3.2.1	Building blocks of the hardware Of an Embedded systems	7
4.1.1	Arduino Uno R3 Board	10
4.2(a)	Block Diagram of Power Supply	14
4.2(b)	Schematic Diagram of power Supply	14
4.2.2	Bridge Rectifier	15
4.3	LCD	16
4.3.1	LCD Screen Circuit Diagram	18
4.4	Temperature Sensor	21
4.5	ESP8266 Wi-Fi	22
4.5.1	Espressif Wi-Fi Modules	26
4.6.1	Space Segment	29
4.6.2	Control Segment	30
4.6.3	Evolution of Mobile Phone	33
4.6.4	GPS RS-J32	33
4.6.5	GPS Modem	35
4.7.1	MQ2-Sensor	35
4.7.2	MQ2-Sensor Module	36
4.7.3	555 Timer Circuit	38
4.7.4	555 Functional Schematic Diagram	39
4.7.5	H-Bridge	41
4.8	Basic Sensor Operation Principle	46
4.9	Buzzer Pin Configuration	47
6.1	Block Diagram	58
7.1	Hardware Kit	61
7.2	Temperature & Humidity	62
7.3	Output	62

LIST OF TABLES

TABLE NO	NAME OF THE TABLE	PAGE NO
4.3.1	LCD PIN Functions	17
4.3.2	Basic Commands of LCD	19
4.5.1	ESPRESSIF Modules	25
4.7.1	Operation of the motor	42

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1. INTRODUCTION TO THE PROJECT

Observing of environment is information assortment and environment parameters data gathering. Observing and surveying the maintainability of our regular health is additionally significant for proficient environmental arranging, policymaking and natural contamination goal. This conveys the wellbeing danger of manual observing for an incredibly polluted region. Building up a gadget would be a successful alternative for distant observing to such an extent that the checking can be performed with no human obstruction. As of late, researchers have been utilizing frameworks as information gathering instruments to all the more likely comprehend natural cycles. Track climatic conditions and, at the front line, the most recent moving remote sensor request to send an adaptable and distant observing framework, an effective stage that empowers clients to control their regular introduction to air contaminations by giving data on air quality created by different detecting foundations. The sensors control the air quality on an intermittent premise. Information can be followed and gotten to utilizing mobile phones or an Internet-empowered PC from anyplace. The execution incorporates air quality, CO, CO2, and temperature and dampness sensors to screen the surrounding condition. The capacity to gather explicit Environmental discharges has gotten widespread in metropolitan urban communities because of advances in innovation and quickened financial turn of events. The objective of this investigation is accordingly to plan an ease, cloud-based keen framework called Cloud-based Smart Device Environment Monitoring (CEMSD) that tracks different natural boundaries, for example, air quality, clamor, temperature and stickiness. The Raspberry Pi 3 (RPI 3) Model B and thusly a microchip with DHT11 temperature mugginess sensor, Grove-Loudness sensor, Shinyei PPD42NS particulate issue (PM) woods dust sensor, COZIR wide range 100%carbondioxide (CO2) sensor and MQ131 Ozone (O3) gas sensor are utilized to make the CEMSD. The CEMSD gathers and sends information from focused estimating areas to a cloud worker by means of remote organization or cell organization, where information is gathered, and available through a PC or any savvy gadget is in boom today.

This innovation is applied on the Internet of Things is the organization of physical items containing embedded technology that assists with building individuals to machines or machines communication. This venture essentially bolsters an independent system that offers a dynamic datasheet on the city condition's boundaries. The machine utilizes Arduino, a minimal effort low-power ARM based minicomputer. It can convey through Local Area Network (LAN) or outer module Wi-Fi. Client orders are handled utilizing the language of Python on the Raspberry Pi. Other terminal gadgets, for example, Laptop, Smart Phone and Tablet blessed with the web office can follow the information. This framework offers admittance to constant data about a metropolitan domain that incorporates boundaries, for example, temperature, humidity, pressure, CO and harmful outflows from air.

Observing and monitoring the environmental parameters have become essential to know the condition of environment. Also, measuring the sustainability of our health has become important for the purpose of capable ecological positioning, for strategy creation and pollution-control. This shows the well-being hazard of physical observation in the case of extremely polluted area. In this case, designing device is useful for remote monitoring so that scrutinizing the polluted area can be done with no human intervention. Many researchers utilized different frameworks to collect information using instruments that are more likely to realize natural cycles. The tracking of all kind of climatic circumstances have become an efficient method and remote monitoring framework have been performed using remote sensor. These sensors could control and monitor quality of air.

1.2. AIM OF THE PROPOSED SYSTEM

With the help of our project we aim to provide a self-working autonomous robotic system that records the data and sends it to the user without any human intervention. The robot is controlled and handled through GPS and it records the data of that environment continuously. So when the user wants to know the data he can access the data in his mobile or through any other device through an application.

CHAPTER-2

LITERATURE REVIEW

2. LITERATURE REVIEW

Existing environmental monitoring systems discussed in this section are 1. Cyber-physical system for environmental monitoring with recent advances in wireless sensor technology, single-board computers and short-range communication technologies, remote sensing applications have improved towards solutions that encompass ubiquitous computing. Cyber-Physical device was once proposed for environmental monitoring of ambient stipulations in indoor spaces. 2. Climate monitoring using Raspberry Pi Shete R. and Agrawal S. present the framework for monitoring the metropolis environment. Raspberry Pi used for implanting the system. However, no emphasis has given on particulate matter which left the environment monitoring system in complete. 3. Cloud-based smart device Cloud-based Environment Monitoring Smart Device that monitors different environmental parameters such as air quality, noise, temperature, and humidity. The device collects and sends data from targeted measurement locations through a wireless network or cellular network to a cloud server. 4. Air quality monitoring system based on IOT using raspberry pi the proposed method of this paper presents a real time air quality monitoring system which includes various parameters: carbon monoxide, carbon dioxide, temperature, humidity and air pressure. Internet of Things converging with cloud computing offers a novel technique for better management of data coming from different sensors, collected and transmitted by low power, low cost ARM based microcontroller Raspberry pi .

Furthermost, a detailed study on wireless type of technologies prove to create a framework of remote sensors. This study led to advancements in the field of wireless technology both in the aspects of technological as well as economical. Here, the main issue is the choice of the communication. The gathered information could be transferred using IP address to any place i.e., using web platform. Next is choosing a right micro-controller. In a Cloud-based Smart System for Environment Control" proposed by Biao Jiang and Christian F.Huacon illustrates that the natural pollution has become uncontrollable in all the metropolitan-urban zones due to advancements in modernization as well as faster economic progress.

This has led to development of cloud-based smart device known as CEMSD i.e., Cloud-based Environment Monitoring Smart Device. This is used to detect all-natural margins such as noise level, humidity, level of temperature, and the quality of air. In the paper titled "Mobile Robots Navigation in Indoor Environments Using Kinect Sensor" by Diogo Santos Ortiz Correa, Diego Fernando Sciotti, Marcos Gomes Prado, Daniel Oliva Sales, DenisFernando Wolf, Fernando Santos Osório- a new kind of surveilling system is demonstrated for monitoring indoor environments. The system is divided into two parts. In the first part, it consists of navigation system involved with Kinect sensor to detect obstacles. The second part involves ANN i.e., artificial neural network in order to identify various configurations. The main limitation of this paper is it is limited to indoor applications. In the paper titled "Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision" by Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju And Tomi Westerlund, it represents multi-robot systems which support SAR (Search and Rescue) type of operations.

Techniques such as active perception are applied to SAR which helps to understand and analyse the SAR personnel actions. In the paper titled "Design of Wireless Mobile Environment Monitoring System Based on Spherical Amphibious Robots" by Shuxiang Guo, Xujie Yang, Jian Guo, Chunying Li, it represents an environment monitoring system which monitors data of aquaculture. To display the obtained data, controlling interface is designed using LabVIEW. The communication between robot and the PC is established using XBee module. The robot is embedded with pressure sensors, temperature and humidity sensors and HD cameras to capture the data. In this, a PID controller is used to detect and monitor the fish in various depths. In the paper titled "Autonomous Environmental Monitoring by Self-powered Biohybrid Robot" by Kan Shoji, Keisuke Morishima, Yoshitake Akiyama, Nobuhumi Nakamura and Hiroyuki Ohno, a self-powered monitoring system for tracking the environmental conditions is introduced. These are the bio-hybrid robots used for generating electric power from their body fluid.

CHAPTER-3

INTRODUCTION ABOUT EMBEDDED SYSTEMS

3. INTRODUCTION ABOUT EMBEDDED SYSTEMS

3.1 INTRODUCTION OF EMBEDDED SYSTEM:

An embedded system is a combination of software and hardware to perform a dedicated task. Some of the main devices used in embedded products are Microprocessors and Microcontrollers.

Microprocessors are commonly referred to as general purpose processors as they simply accept the inputs, process it and give the output. In contrast, a microcontroller not only accepts the data as inputs but also manipulates it, interfaces the data with various devices, controls the data and thus finally gives the result.

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. A good example is the microwave oven. Almost every household has one, and tens of millions of them are used every day, but very few people realize that a processor and software are involved in the preparation of their lunch or dinner.

This is in direct contrast to the personal computer in the family room. It too is comprised of computer hardware and software and mechanical components (disk drives, for example). However, a personal computer is not designed to perform a specific function rather; it is able to do many different things. Many people use the term general-purpose computer to make this distinction clear. As shipped, a general-purpose computer is a blank slate; the manufacturer does not know what the customer will do wish it. One customer may use it for a network file server another may use it exclusively for playing games, and a third may use it to write the next great American novel.

Frequently, an embedded system is a component within some larger system. For example, modern cars and trucks contain many embedded systems. One embedded system controls the anti-lock brakes, other monitors and controls the vehicle's emissions, and a third displays information on the dashboard. In some cases, these embedded systems are connected by some sort of a communication network, but that is certainly not a requirement.

At the possible risk of confusing you, it is important to point out that a general-purpose computer is itself made up of numerous embedded systems. For example, my computer consists of a keyboard, mouse, video card, modem, hard drive, floppy drive, and sound card-each of which is an embedded system. Each of these devices contains a processor and software and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over analog telephone line. That's it and all of the other devices can be summarized in a single sentence as well.

3.2. OVERVIEW OF EMBEDDED SYSTEM:

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is also called the 'firmware'.

The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application.

For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run *for* a long time you don't need to reload new software. Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in Fig. the building blocks are:

a. Central Processing Unit (CPU)
b. Memory (Read-only Memory and Random Access Memory)
c. Input Devices
d. Output devices
e. Communication interfaces
f. Application-specific circuitry

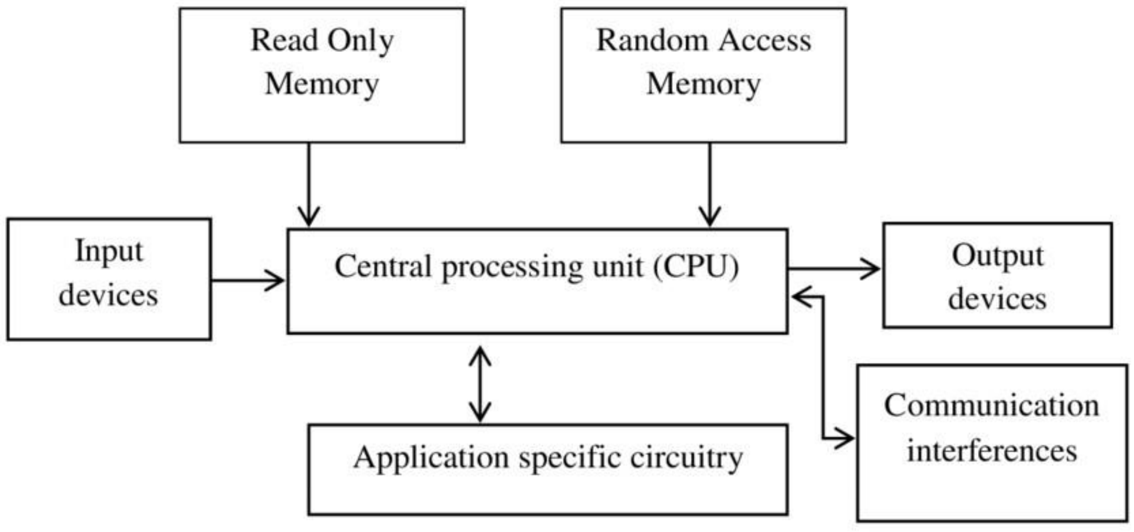

Read Only Memory
Random Access Memory
Input devices
Central processing unit (CPU)
Output devices
Application specific circuitry
Communication interferences

Fig: 3.2.1 Building blocks of the hardware of an embedded system

➢ CENTRAL PROCESSING UNIT (CPU):

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog to digital converter etc.

So, for small applications, a micro-controller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. D5P is used mainly for applications in which signal processing is involved such as audio and video processing.

➢ MEMORY:

The memory is categorized as Random Access Memory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is program is executed

➢ INPUT DEVICES:

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device for user interaction; they take inputs from sensors or transducers 1’fnd produce electrical signals that are in turn fed to other systems.

➢ OUTPUT DEVICES:

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

➢ COMMUNICATION INTERFACES:

The embedded systems may need to, interact with other embedded systems at they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), IEEE 1394, Ethernet etc.

➢ APPLICATION-SPECIFIC CIRCUITRY:

Sensors, transducers, special processing and control circuitry may be required fat an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware has to design in such a way that the power consumption is minimized.

CHAPTER-4

DESIGN OF HARDWARE

4. DESIGN OF HARDWARE

This chapter briefly explains about the Hardware implementation of IOT BASED SMART HELMENT SYSTEM FOR COAL MINE INDUSTRY. It discusses the circuit diagram of each module in detail.

4.1 ARDUINO:

The most common version of Arduino is the Arduino Uno. This board is what most people are talking about when they refer to an Arduino. The Uno is one of the more popular boards in the Arduino family and a great choice for beginners. There are different revisions of Arduino Uno, below detail is the most recent revision (Rev3 or R3).

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

Microcontroller	:	ATmega328
Operating Voltage	:	5V
Input Voltage (recommended)	:	7-12V
Input Voltage (limits)	:	6-20V
Digital I/O Pins	:	14 (of which 6 provide PWM output)
Analog Input Pins	:	6
DC Current per I/O Pin	:	40 mA
DC Current for 3.3V Pin	:	50 mA
Flash Memory	:	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	:	2 KB (ATmega328)
EEPROM	:	1 KB (ATmega328)
Clock Speed	:	16 MHz
Length	:	68.6 mm
Width	:	53.4 mm

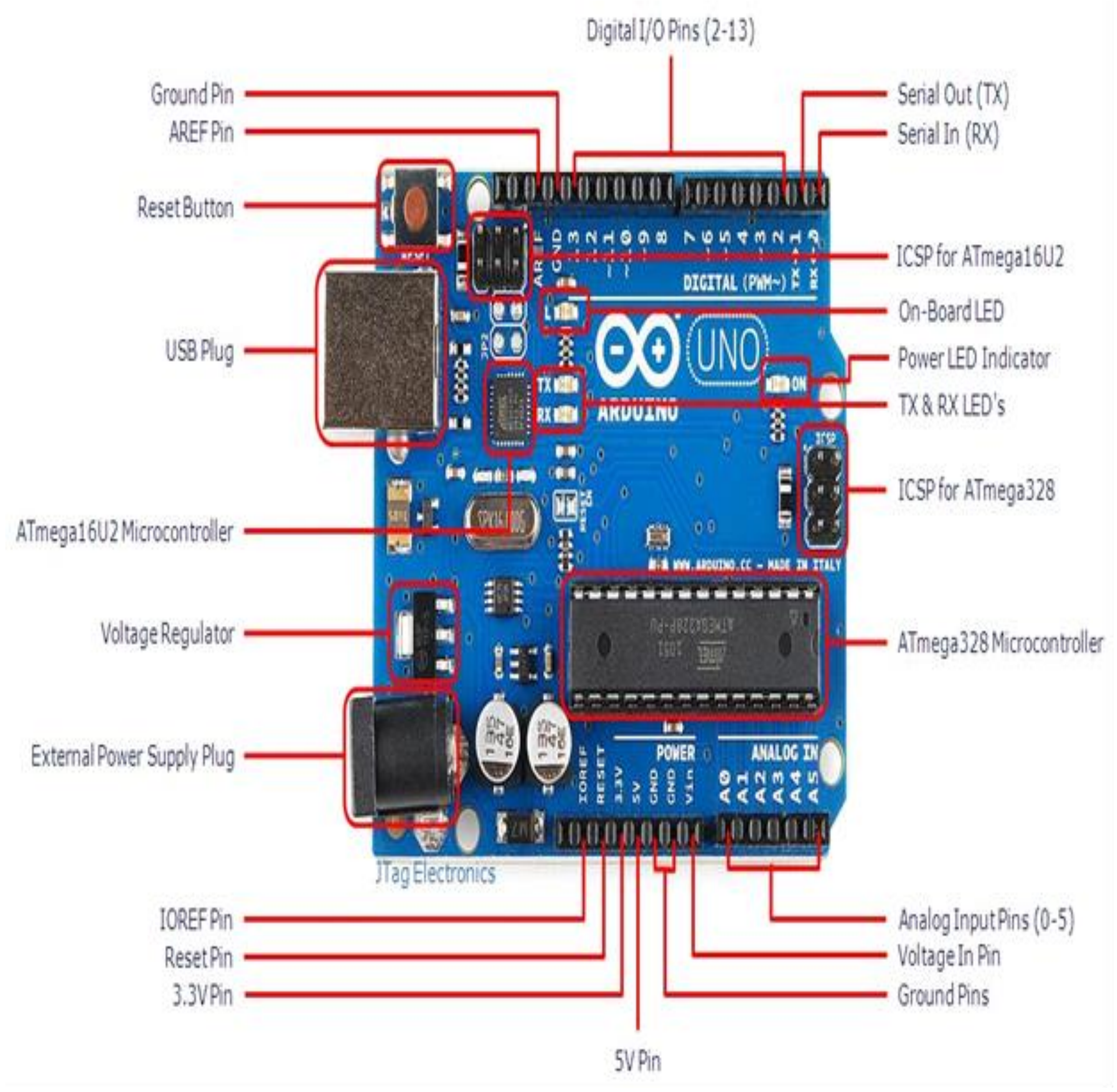


Fig: 4.1.1 Arduino Uno R3 Board

- **USB Plug & External Power Supply Plug**

Every Arduino board needs a way to be connected to a power source. The Arduino Uno can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. The power source is selected automatically. The USB connection is also how you will load code onto your Arduino board.

.

- **Voltage Regulator**

The voltage regulator is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

- **Power Pins**

Voltage In Pin – The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
5V Pin – This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 – 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. It's not recommended.3.3V Pin – A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

- **Ground Pins**

There are several GND pins on the Arduino, any of which can be used to ground your circuit.

- **IOREF Pin**

This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

- **Input and Output Pins**

Each of the 14 digital pins on the Uno can be used as an input or output. They operate at 5 volts. These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED). Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-5k Ohms. In addition, some pins have specialized functions.

- **Serial Out (TX) & Serial In (RX)**

Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- **External Interrupts**

Pins 2 and 3 can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

PWM – You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). Think of these pins as being able to simulate analog output (like fading an LED in and out).

SPI – Pins 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). SPI stands for Serial Peripheral Interface. These pins support SPI communication using the SPI library.

Analog Input Pins – Labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read. By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF Pin (Stands for Analog Reference). Most of the time you can leave this pin alone). Additionally, some pins have specialized functionality:

TWI – Pins A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

- **Reset Pin**

  Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

- **LED Indicators**

  Power LED Indicator – Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON'. This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

On-Board LED – There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off. This useful to quickly check if the board has no problem as some boards has a pre-loaded simple blinking LED program in it.

TX & RX LEDs – These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program on to the board).

Reset Button: Pushing the reset button temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times.

4.2. POWER SUPPLY:

The power supplies are designed to convert high voltage AC mains electricity to a suitable low voltage supply for electronic circuits and other devices. A power supply can by broken down into a series of blocks, each of which performs a particular function. A d.c power supply which maintains the output voltage constant irrespective of a.c mains fluctuations or load variations is known as "Regulated D.C Power Supply".

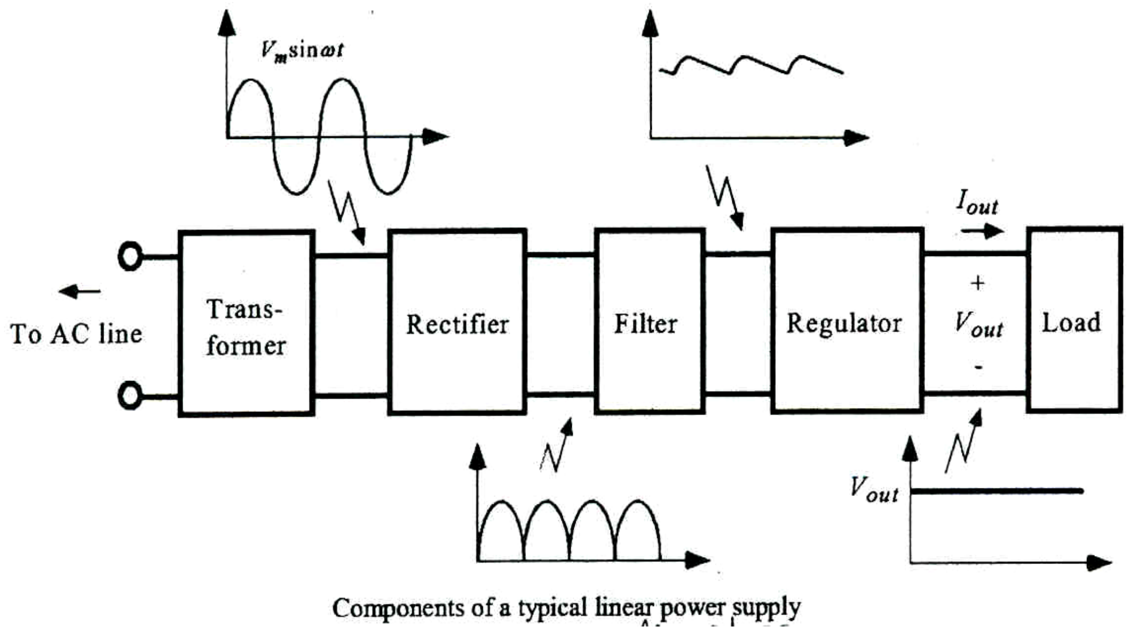


Fig:4.2(a) Block Diagram of Power Supply

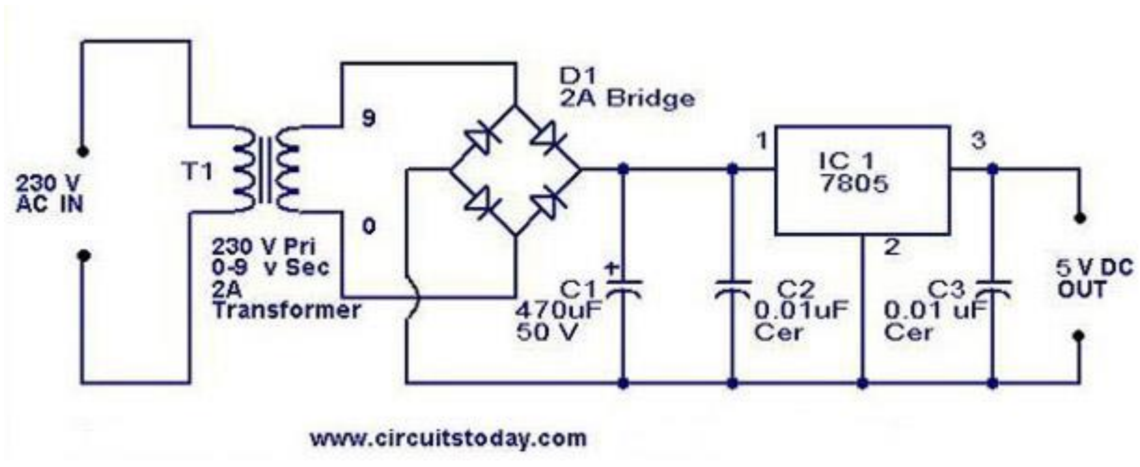


Fig:4.2(b) Schematic Diagram of Power Supply

4.2.1. TRANSFORMER:

A transformer is an electrical device which is used to convert electrical power from one electrical circuit to another without change in frequency.

When AC is applied to the primary winding of the power transformer it can either be stepped down or up depending on the value of DC needed. In our circuit the transformer of 230v/12-0-12v is used to perform the step down operation where a 230V AC appears as 12V AC across the secondary winding.

4.2.2. RECTIFIER:

A circuit which is used to convert a.c to dc is known as RECTIFIER. The process of conversion a.c to d.c is called “rectification.

Bridge Rectifier:

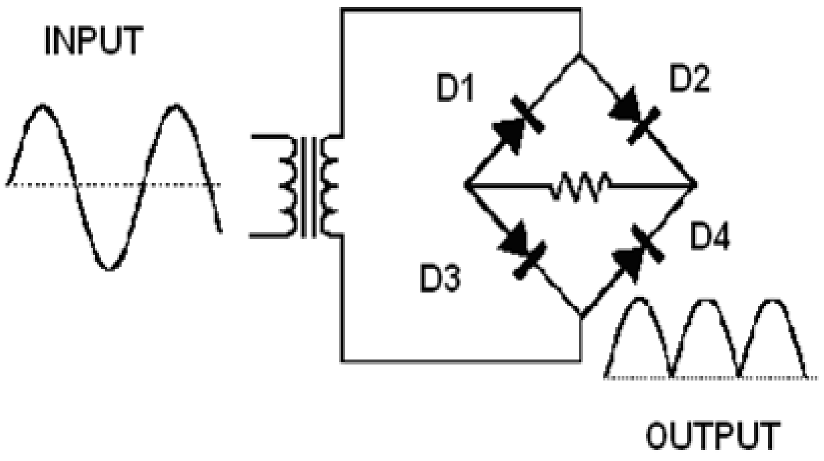


Fig: 4.2.2 Bridge Rectifier

OPERATION:

During positive half cycle of secondary, the diodes D2 and D3 are in forward biased while D1 and D4 are in reverse biased. During negative half cycle of secondary voltage, the diodes D1 and D4 are in forward biased while D2 and D3 are in reverse biased.

4.2.3. FILTER:

A Filter is a device which removes the a.c component of rectifier output but allows the d.c component to reach the load. We have seen that the ripple content in the rectified output of half wave rectifier is **121%** or that of full-wave or bridge rectifier or bridge rectifier is **48%** such high percentages of ripples is not acceptable for most of the applications. Ripples can be removed by one of the following methods of filtering. A capacitor, in parallel to the load, provides an easier by –pass for the ripples voltage though it due to low impedance. At ripple frequency and leave the d.c.to appears the load.

4.2.4. VOLTAGE REGULATOR:

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power supply of 5V and 12V are required. In order to obtain these voltage levels, 7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers 05, 12 represent the required output voltage.

4.3. LCD:

A model described here is for its low price and great possibilities most frequently used in practice. It is based on the HD44780 microcontroller (Hitachi) and can display messages in two lines with 16 characters each. It displays all the alphabets, Greek letters, punctuation marks, mathematical symbols etc. In addition, it is possible to display symbols that user makes up on its own. Automatic shifting message on display (shift left and right), appearance of the pointer, backlight etc. are considered as useful characteristics.

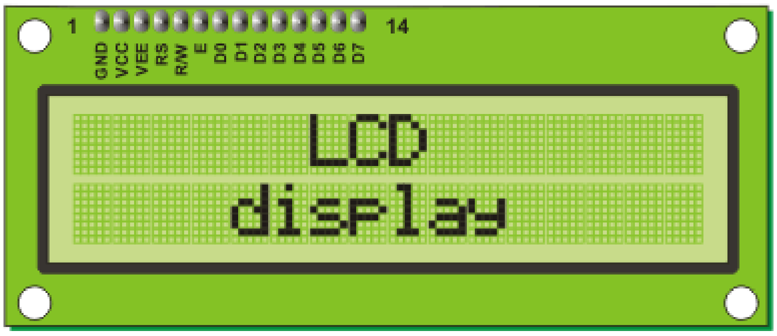


Fig: 4.3. LCD

4.3.1. PINS FUNCTIONS:

There are pins along one side of the small printed board used for connection to the microcontroller. There are total of 14 pins marked with numbers (16 in case the background light is built in). Their function is described in the table below:

Table 4.3.1. LCD Pin Functions

Function	Pin Number	Name	Logic State	Description
Ground	1	Vss	-	0v
Power supply	2		-	+5v
Contrast	3		-	0-VDD
Control of operating	4	RS	0	D0 – D7 are interpreted as commands
			1	D0 – D7 are interpreted as data
	5	RW	0	Write data (from controller to LCD)
			1	Read data (from LCD to controller)
	6	E	0	Access to LCD disabled
			1	Normal operating
				Data/commands are transferred to LCD
Data/Commands	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1
	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	13	D6	0/1	Bit 6
	14	D7	0/1	Bit 7 MSB

4.3.2. LCD SCREEN:

LCD screen consists of two lines with 16 characters each. Each character consists of 5x7 dot matrix. Contrast on display depends on the power supply voltage and whether messages are displayed in one or two lines. For that reason, variable voltage 0-Vdd is applied on pin marked as Vee. Trimmer potentiometer is usually used for that purpose. Some versions of displays have built in backlight (blue or green diodes). When used during operating, a resistor for current limitation should be used (like with any LE diode).

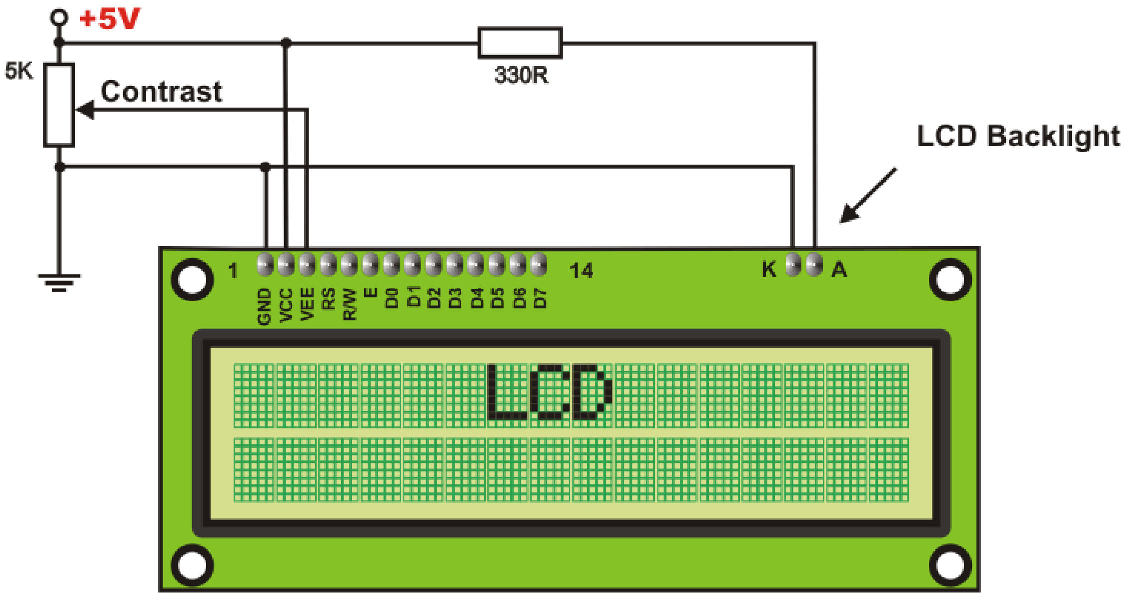


Fig: 4.3.1 LCD Screen Circuit Diagram

➢ **LCD Basic Commands:**

All data transferred to LCD through outputs D0-D7 will be interpreted as commands or as data, which depends on logic state on pin RS: RS = 1 - Bits D0 - D7 are addresses of characters that should be displayed. Built in processor addresses built in “map of characters” and displays corresponding symbols. Displaying position is determined by DDRAM address. This address is either previously defined or the address of previously transferred character is automatically incremented. RS = 0 - Bits D0 - D7 are commands which determine display mode. List of commands which LCD recognizes are given in the table below:

Table 4.3.2 Basic Commands Of LCD

Command	**RS**	**RW**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**Execution Time**
Clear display	0	0	0	0	0	0	0	0	0	1	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	x	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	40uS
Display on/off control	0	0	0	0	0	0	1	D	U	B	40uS
Cursor/Display Shift	0	0	0	0	0	1	D/C	R/L	x	x	40uS
Function set	0	0	0	0	1	DL	N	F	x	x	40uS
Set CGRAM address	0	0	0	1	CGRAM address						40uS
Set DDRAM address	0	0	1	DDRAM address							40uS
Read “BUSY” flag (BF)	0	1	BF	DDRAM address							-

4.4. TEMPERATURE SENSOR (LM35):

In order to monitor the temperature continuously and compare this with the set temperature preprogrammed in the microcontroller, initially this temperature value has to be read and fed to the microcontroller. This temperature value has to be sensed. Thus a sensor has to be used and the sensor used in this project is LM35. It converts temperature value into electrical signals.

LM35 series sensors are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Celsius temperature. The LM35 requires no external calibration since it is internally calibrated. . The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4°C at room temperature and ±3⁄4°C over a full −55 to +150°C temperature range.

The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air.

Features

1. Calibrated directly in ° Celsius (Centigrade)
2. Linear + 10.0 mV/°C scale factor
3. 0.5°C accuracy guaranteed (at +25°C)
4. Rated for full −55° to +150°C range
5. Suitable for remote applications
6. Low cost due to wafer-level trimming
7. Operates from 4 to 30 volts
8. Less than 60 µA current drain
9. Low self-heating, 0.08°C in still air
10. Low impedance output, 0.1 W for 1 mA load

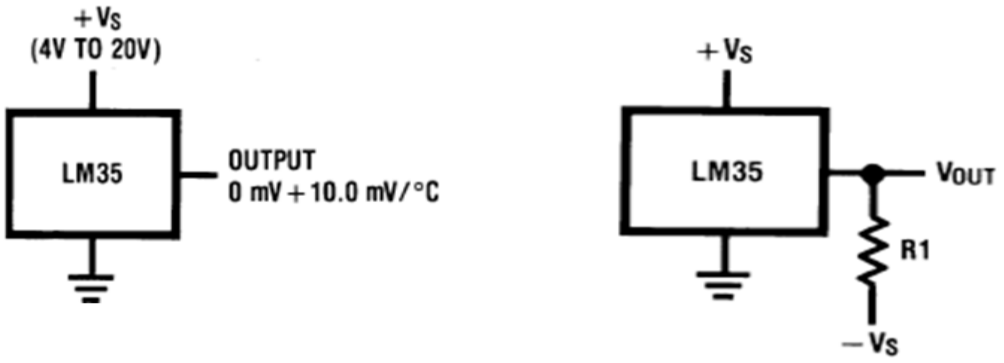


Fig: 4.4. Temperature Sensor

ROLE OF LM35:

In this project, the temperature is to be monitored continuously and if the temperature exceeds the set value preprogrammed in the microcontroller, a buzzer indication is provided in the circuit to alert the people in the industry to stop the process immediately. Thus the temperature sensor LM35 has to read the temperature continuously and the microcontroller has to compare this temperature value with the set temperature preprogrammed in it. When this temperature exceeds the set value, the microcontroller sends an indication to the buzzer which gives a loud noise.

4.5. ESP8266 WI-FI

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer, Espressif Systems. The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer, Ai-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at the time there was almost no English-language documentation on the chip and the commands it accepted. The very low price and the fact that there were very few external components on the module which suggested that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.The **ESP8285** is an ESP8266 with 1 MB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi.

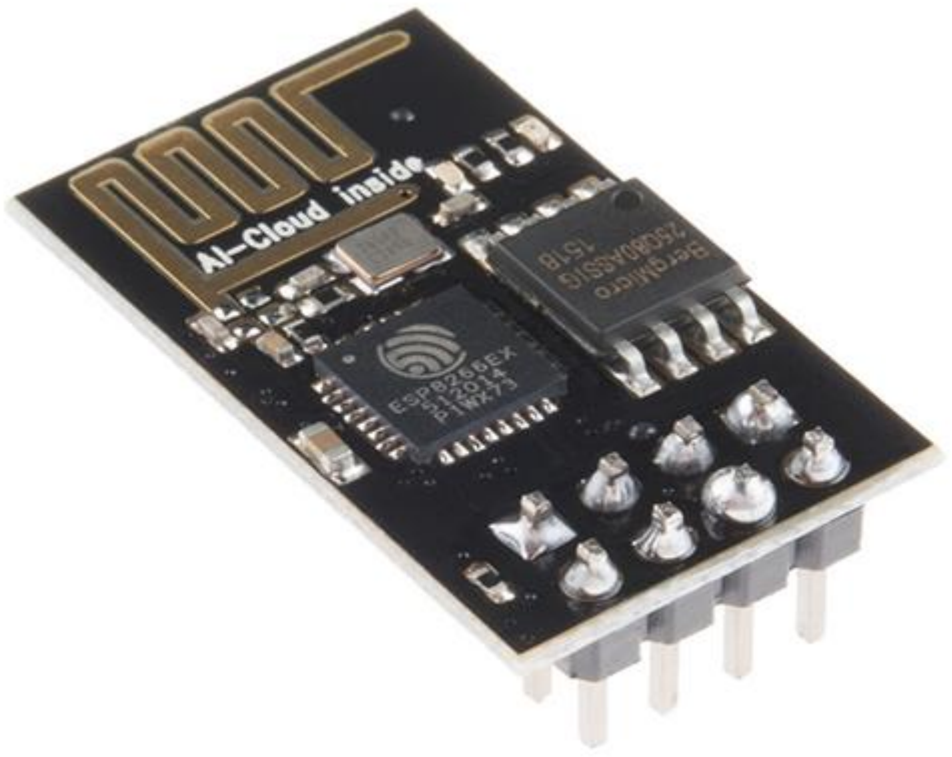


Fig: 4.5 ESP8266 Wi-Fi Module

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz
- Memory:
- 32 KB instruction RAM
- 32 KB instruction cache RAM
- 80 KB user data RAM
- 16 KB ETS system data RAM
- External QSPI flash: up to 16 MB is supported (512 KB to 4 MB typically included)

- IEEE 802.11 b/g/n Wi-Fi
- Integrated TR switch, balun, LNA, power amplifier and matching network
- WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI
- I²C (software implementation)
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 10-bit ADC (successive approximation ADC)

Both the CPU and flash clock speeds can be doubled by overclocking on some devices. CPU can be run at 160 MHz and flash can be sped up from 40 MHz to 80 MHz.[citation needed] Success varies chip to chip.

In late October 2014, Espressif Systems released a software development kit (SDK) that allowed the chip to be programmed, removing the need for a separate microcontroller. Since then, there have been many official SDK releases from Espressif; Espressif maintains two versions of the SDK – one that is based on FreeRTOS and the other based on callbacks.

An alternative to Espressif's official SDK is the open source ESP-Open-SDK[8] that is based on the GCC tool chain. ESP8266 uses the Cadence Tensilica L106 microcontroller and the GCC tool chain is open-sourced and maintained by Max Filippov. Another alternative is the "Unofficial Development Kit" by Mikhail Grigorev.

Other SDKs (mostly open source) include:

- NodeMCU – A Lua-based firmware.

- Arduino – A C++ based firmware. This core enables the ESP8266 CPU and its Wi-Fi components to be programmed like any other Arduino device. The ESP8266 Arduino Core is available through GitHub.
- PlatformIO (https://platformio.org/platforms/espressif8266) – A cross-platform IDE and unified debugger which sits on top of Arduino code and libraries.
- MicroPython – A port of Micro Python (an implementation of Python for embedded devices) to the ESP8266 platform.
- ESP8266 BASIC – An open source basic interpreter specifically tailored for the internet of things. Self-hosting browser based development environment.
- Zbasic for ESP8266 – A subset of Microsoft's widely used Visual Basic 6 which has been adapted as a control language for the ZX microcontroller family and the ESP8266.
- Espruino – An actively maintained JavaScript SDK and firmware, closely emulating Node.js. Supports a few MCUs, including the ESP8266.

- Mongoose OS – An open source Operating System for connected products. Supports ESP82666 and ESP32. Develop in C or JavaScript
- ESP-Open-SDK – Free and open (as much as possible) integrated SDK for ESP8266/ESP8285 chips.
- ESP-Open-RTOS – Open source FreeRTOS-based ESP8266 software framework.
- Zerynth – IOT framework that allows programming ESP8266 and other microcontrollers using Python.

Table 4.5.1. ESPRESSIF Modules

Name	**Active pins**	**Pitch**	**Form factor**	**LEDs**	**Antenna**	**Shielded**	**Dimensions (mm)**	**Notes**
ESP-WROOM-02[14]	18	1.5 mm	2×9 castellated	No	PCB trace	Yes	18 × 20	FCC ID 2AC7Z-ESPWROOM 02.
ESP-WROOM-02D[15]	18	1.5 mm	2×9 castellated	No	PCB trace	Yes	18 × 20	FCC ID 2AC7Z-ESPWROOM 02D. Revision of ESP-WROOM-02 compatible with both 150-mil and 208-mil flash memory chips.
ESP-WROOM-02U[15]	18	1.5 mm	2×9 castellated	No	U.FL socket	Yes	18 × 20	Differs from ESP-WROOM-02D in that includes an U.FL compatible antenna socket connector.
ESP-WROOM-S2[16]	20	1.5 mm	2×9 castellated	No	PCB trace	Yes	16 × 23	FCC ID 2AC7Z-ESPWROOM S2

In the table above (and the two tables which follow), "Active pins" include the GPIO and ADC pins with which you can attach external devices to the ESP8266 MCU. The "Pitch" is the space between pins on the ESP8266 module, which is important to know if you are going to breadboard the device. There are several antenna options for ESP-xx boards including a trace antenna, an on-board ceramic antenna, and an external connector which allows you to attach an external Wi-Fi antenna. Since Wi-Fi communications generates a lot of RFI (Radio Frequency Interference), governmental bodies like the FCC like shielded electronics to minimize interference with other devices. Some of the ESP-xx modules come housed within a metal box with an FCC seal of approval stamped on it. First and second world markets will likely demand FCC approval and shielded Wi-Fi devices

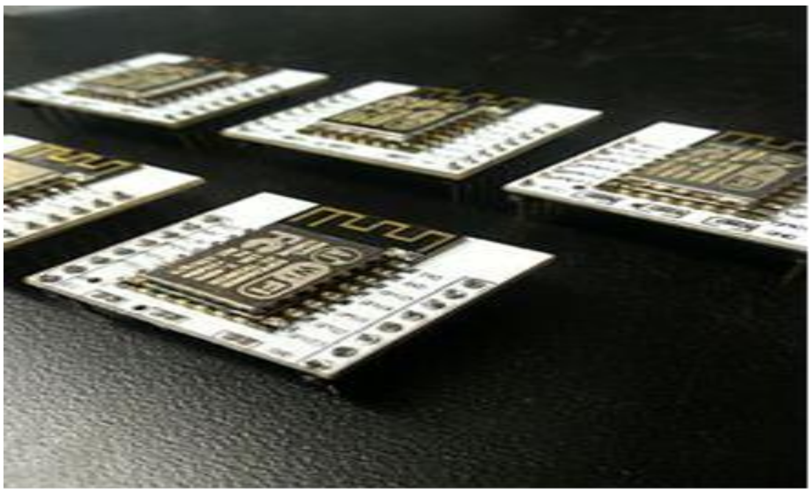


Fig: 4.5.1 Espressif Wi-fi Modules

These are the first series of modules made with the ESP8266 by the third-party manufacturer *Ai-Thinker* and remain the most widely available. To form a workable development system they require additional components, especially a serial TTL-to-USB adapter and an external 3.3 volt power supply. Novice ESP8266 developers are encouraged to consider larger ESP8266 Wi-Fi development boards like the Node MCU which includes the USB-to-UART bridge and a Micro-USB connector coupled with a 3.3 volt power regulator already built into the board. When project development is complete, these components are not needed anymore and it can be considered using these cheaper ESP-xx modules as a lower power, smaller footprint option for production runs.

4.6. GPS:

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather, anywhere on or near the Earth, where there is an unobstructed line of sight to four or more GPS satellites. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver.

The GPS program provides critical capabilities to military, civil and commercial users around the world. In addition, GPS is the backbone for modernizing the global air traffic system.

The GPS project was developed in 1973 to overcome the limitations of previous navigation systems, integrating ideas from several predecessors, including a number of classified engineering design studies from the 1960s. GPS was created and realized by the U.S. Department of Defense (DoD) and was originally run with 24 satellites. It became fully operational in 1994.

Advances in technology and new demands on the existing system have now led to efforts to modernize the GPS system and implement the next generation of GPS III satellites and Next Generation Operational Control System (OCX). Announcements from the Vice President and the White House in 1998 initiated these changes. In 2000, U.S. Congress authorized the modernization effort, referred to as GPS III.

In addition to GPS, other systems are in use or under development. The Russian Global Navigation Satellite System (GLONASS) was in use by only the Russian military, until it was made fully available to civilians in 2007. There are also the planned European Union Galileo positioning system, Chinese Compass navigation system, and Indian Regional Navigational Satellite System.

Basic Components of GPS

A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites high above the Earth. Each satellite continually transmits messages that include

- the time the message was transmitted
- satellite position at time of message transmission

The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the speed of light. These distances along with the satellites' locations are used with the possible aid of trilateration, depending on which algorithm is used, to compute the position of the receiver. This position is then displayed, perhaps with a moving map display or latitude and longitude; elevation information may be included.

Many GPS units show derived information such as direction and speed, calculated from position changes.

In typical GPS operation, four or more satellites must be visible to obtain an accurate result. Three satellites might seem enough to solve for position since in typical GPS operation the surfaces of three spheres intersect in only two points and only one of these two points is near earth. However, a three satellite solution requires an extremely precise clock. Clocks accurate enough (typically atomic clocks) are very expensive, large, power hungry. To avoid the need to use these very expensive clocks, receivers use much less expensive clocks and use the Bancroft or other algorithm discussed in the section, "Navigation equations", to compute the three components of receiver position and the time correction required from the data provided by four satellites. The very accurately computed time is sometimes hidden by GPS receivers, which use only the location. A few specialized GPS applications do however use the time; these include time transfer, traffic signal timing, and synchronization of cell phone base stations.

Although four satellites are required for normal operation, fewer apply in special cases. If one variable is already known, a receiver can determine its position using only three satellites. For example, a ship or aircraft may have known elevation. Some GPS receivers may use additional clues or assumptions such as reusing the last known altitude, dead reckoning, inertial navigation, or including information from the vehicle computer, to give a (possibly degraded) position when fewer than four satellites are visible.

Structure

The current GPS consists of three major segments. These are the space segment (SS), a control segment (CS), and a user segment (US). The U.S. Air Force develops, maintains, and operates the space and control segments. GPS satellites broadcast signals from space, and each GPS receiver uses these signals to calculate its three-dimensional location (latitude, longitude, and altitude) and the current time.

The space segment is composed of 24 to 32 satellites in medium Earth orbit and also includes the payload adapters to the boosters required to launch them into orbit. The control segment is composed of a master control station, an alternate master control station, and a host of dedicated and shared ground antennas and monitor stations. The user segment is composed of hundreds of thousands of U.S. and allied military users of the secure GPS Precise Positioning Service, and tens of millions of civil, commercial, and scientific users of the Standard Positioning Service (GPS navigation devices).

Space segment:

Fig.4.6.1. Space Segment

The space segment (SS) is composed of the orbiting GPS satellites or Space Vehicles (SV) in GPS parlance. The GPS design originally called for 24 SVs, eight each in three approximately circular orbits, but this was modified to six orbital planes with four satellites each. The orbits are centered on the Earth, not rotating with the Earth, but instead fixed with respect to the distant stars. The six orbit planes have approximately 55° inclination (tilt relative to Earth's equator) and are separated by 60° right ascension of the ascending node (angle along the equator from a reference point to the orbit's intersection). The orbital period is one-half a sidereal day, i.e. 11 hours and 58 minutes.

The orbits are arranged so that at least six satellites are always within line of sight from almost everywhere on Earth's surface. The result of this objective is that the four satellites are not evenly spaced (90 degrees) apart within each orbit. In general terms, the angular difference between satellites in each orbit is 30, 105, 120, and 105 degrees apart which, of course, sum to 360 degrees.

Orbiting at an altitude of approximately 20,200 km (12,600 mi); orbital radius of approximately 26,600 km (16,500 mi), each SV makes two complete orbits each sidereal day, repeating the same ground track each day. This was very helpful during development because even with only four satellites, correct alignment means all four are visible from one spot for a few hours each day. For military operations, the ground track repeat can be used to ensure good coverage in combat zones.

As of March 2008, there are 31 actively broadcasting satellites in the GPS constellation, and two older, retired from active service satellites kept in the constellation as orbital spares. The additional satellites improve the precision of GPS receiver calculations by providing redundant measurements. With the increased number of satellites, the constellation was changed to a non-uniform arrangement. Such an arrangement was shown to improve reliability and availability of the system, relative to a uniform system, when multiple satellites fail. About nine satellites are visible from any point on the ground at any one time (see animation at right), ensuring considerable redundancy over the minimum four satellites needed for a position.

Control segment:

Fig.4.6.2. Control Segment

The control segment is composed of

1. a master control station (MCS),
2. an alternate master control station,
3. four dedicated ground antennas and
4. six dedicated monitor stations

The MCS can also access U.S. Air Force Satellite Control Network (AFSCN) ground antennas (for additional command and control capability) and NGA (National Geospatial-Intelligence Agency) monitor stations. The flight paths of the satellites are tracked by dedicated U.S. Air Force monitoring stations in Hawaii, Kwajalein, Ascension Island, Diego Garcia, Colorado Springs, Colorado and Cape Canaveral, along with shared NGA monitor stations operated in England, Argentina, Ecuador, Bahrain, Australia and Washington DC. The tracking information is sent to the Air Force Space Command MCS at Schriever Air Force Base 25 km (16 mi) ESE of Colorado Springs, which is operated by the 2nd Space Operations Squadron (2 SOPS) of the U.S. Air Force. Then 2 SOPS contacts each GPS satellite regularly with a navigational update using dedicated or shared (AFSCN) ground antennas (GPS dedicated ground antennas are located at Kwajalein, Ascension Island, Diego Garcia, and Cape Canaveral). These updates synchronize the atomic clocks on board the satellites to within a few nanoseconds of each other, and adjust the ephemeris of each satellite's internal orbital model. The updates are created by a Kalman filter that uses inputs from the ground monitoring stations, space weather information, and various other inputs.

Satellite maneuvers are not precise by GPS standards. So to change the orbit of a satellite, the satellite must be marked *unhealthy*, so receivers will not use it in their calculation. Then the maneuver can be carried out, and the resulting orbit tracked from the ground. Then the new ephemeris is uploaded and the satellite marked healthy again.

The Operation Control Segment (OCS) currently serves as the control segment of record. It provides the operational capability that supports global GPS users and keeps the GPS system operational and performing within specification. OCS successfully replaced the legacy 1970's-era mainframe computer at Schriever Air Force Base in September 2007.

After installation, the system helped enable upgrades and provide a foundation for a new security architecture that supported the U.S. armed forces. OCS will continue to be the ground control system of record until the new segment, Next Generation GPS Operation Control System (OCX), is fully developed and functional.

The new capabilities provided by OCX will be the cornerstone for revolutionizing GPS's mission capabilities, and enabling Air Force Space Command to greatly enhance GPS operational services to U.S. combat forces, civil partners and myriad of domestic and international users.

The GPS OCX program also will reduce cost, schedule and technical risk. It is designed to provide 50% sustainment cost savings through efficient software architecture and Performance-Based Logistics. In addition, GPS OCX expected to cost millions less than the cost to upgrade OCS while providing four times the capability.

The GPS OCX program represents a critical part of GPS modernization and provides significant information assurance improvements over the current GPS OCS program.

- OCX will have the ability to control and manage GPS legacy satellites as well as the next generation of GPS III satellites, while enabling the full array of military signals.
- Built on a flexible architecture that can rapidly adapt to the changing needs of today's and future GPS users allowing immediate access to GPS data and constellations status through secure, accurate and reliable information.
- Empowers the warfighter with more secure, actionable and predictive information to enhance situational awareness.
- Enables new modernized signals (L1C, L2C, and L5) and has M-code capability, which the legacy system is unable to do.
- Provides significant information assurance improvements over the current program including detecting and preventing cyberattacks, while isolating, containing and operating during such attacks.
- Supports higher volume near real-time command and control capability.

User segment

Fig: 4.6.3 Evolution of Mobile Phones

The user segment is composed of hundreds of thousands of U.S. and allied military users of the secure GPS Precise Positioning Service, and tens of millions of civil, commercial and scientific users of the Standard Positioning Service. They may also include a display for providing location and speed information to the user. A receiver is often described by its number of channels: this signifies how many satellites it can monitor simultaneously. Originally limited to four or five, this has progressively increased over the years so that, as of 2007, receivers typically have between 12 and 20 channels.

Fig: 4.6.4 GPS RS-J32

GPS receivers may include an input for differential corrections, using the RTCM SC-104 format. This is typically in the form of an RS-232 port at 4,800 bit/s speed. Data is actually sent at a much lower rate, which limits the accuracy of the signal sent using RTCM. Receivers with internal DGPS receivers can outperform those using external RTCM data. As of 2006, even low-cost units commonly include Wide Area Augmentation System (WAAS) receivers.

Many GPS receivers can relay position data to a PC or other device using the NMEA 0183 protocol. Although this protocol is officially defined by the National Marine Electronics Association (NMEA references to this protocol have been compiled from public records, allowing open source tools like GPS to read the protocol without violating intellectual property laws. Other proprietary protocols exist as well, such as the SiRF and MTK protocols. Receivers can interface with other devices using methods including a serial connection, USB, or Bluetooth.

A GPS tracking unit is a device, normally carried by a moving vehicle or person, that uses the Global Positioning System to determine and track its precise location, and hence that of its carrier, at intervals. The recorded location data can be stored within the tracking unit, or it may be transmitted to a central location database, or Internet-connected computer, using a cellular(GPRS or SMS), radio, or satellite modem embedded in the unit. This allows the asset's location to be displayed against a map backdrop either in real time or when analysing the track later, using GPS tracking software. Data tracking software is available for smart phones with GPS capability. The GPS was originally developed for use by the United States military, but in the 1980s, the United States government allowed the system to be used for civilian purposes. Though the GPS satellite data is free and works anywhere in the world, the GPS device and the associated software must be bought or rented.

A GPS device can retrieve from the GPS system location and time information in all weather conditions, anywhere on or near the Earth. A GPS reception requires an unobstructed line of sight to four or more GPS satellites, and is subject to poor satellite signal conditions. In exceptionally poor signal conditions, for example in urban areas, satellite signals may exhibit multipath propagation where signals bounce off structures, or are weakened by meteorological conditions. Obstructed lines of sight may arise from a tree canopy or inside a structure, such as in a building, garage or tunnel. Today, most standalone GPS receivers are used in automobiles.

The GPS capability of smart phones may use assisted GPS (A-GPS) technology, which can use the base station or cell towers to provide the device location tracking capability, especially when GPS signals are poor or unavailable. However, the mobile network part of the A-GPS technology would not be available when the smartphone is outside the range of the mobile reception network, while the GPS aspect would otherwise continue to be available.

The Russian Global Navigation Satellite System (GLONASS) was developed contemporaneously with GPS, but suffered from incomplete coverage of the globe until the mid-2000s. GLONASS can be added to GPS devices to make more satellites available and enabling positions to be fixed more quickly and accurately, to within 2 meters.

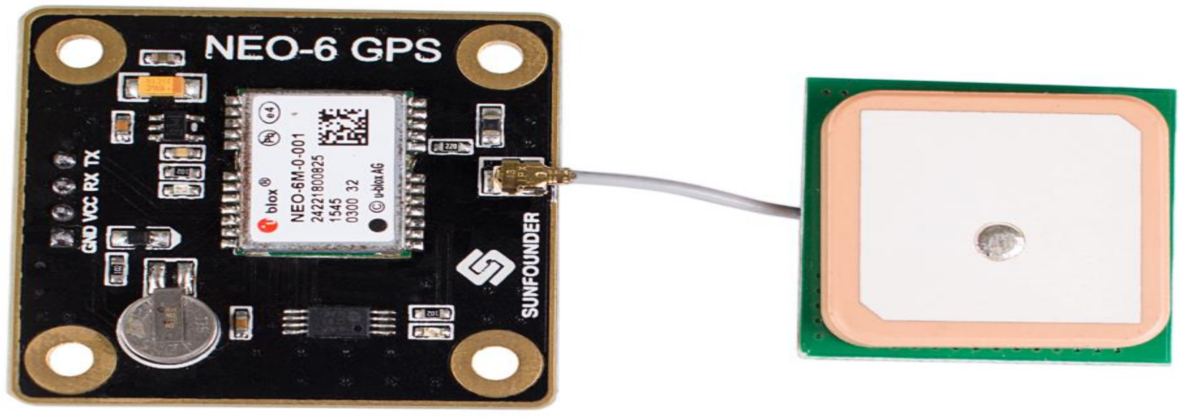


Fig: 4.6.5 GPS MODEM

4.7. MQ2- SENSOR

Fig: 4.7.1 MQ2-Sensor

DESCRIPTION

MQ2 flammable gas and smoke sensor detects the concentrations of combustible gas in the air and outputs its reading as an analog voltage. The sensor can measure concentrations of flammable gas of 300 to 10,000 ppm. The sensor can operate at temperatures from -20 to 50°C and consumes less than 150 mA at 5 V.

Connecting five volts across the heating (H) pins keeps the sensor hot enough to function correctly. Connecting five volts at either the A or B pins causes the sensor to emit an analog voltage on the other pins. A resistive load between the output pins and ground sets the sensitivity of the detector. Please note that the picture in the datasheet for the top configuration is wrong. Both configurations have the same pin out consistent with the bottom configuration. The resistive load should be calibrated for your particular application using the equations in the datasheet, but a good starting value for the resistor is 20 kΩ.

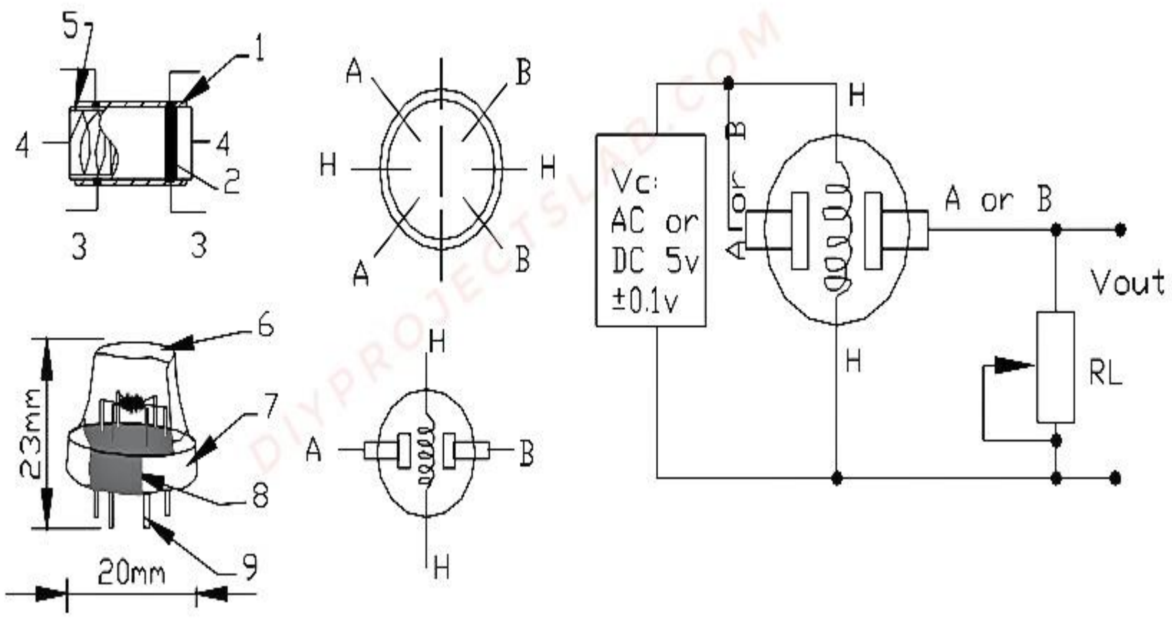


Fig: 4.7.2 MQ2-Sensor Module

FEATURES

- Wide detecting scope
- Fast response and High sensitivity
- Stable and long life
- Simple drive circuit

They are used in gas leakage detecting equipment in family and industry, are suitable for detecting of LPG, i-butane, propane, methane, alcohol, Hydrogen, smoke.

SENSITVITY ADJUSTMENT

Resistance value of MQ-2 is difference to various kinds and various concentration gases. So, when using these components, sensitivity adjustment is very necessary. We recommend that you calibrate the detector for 1000ppm liquefied petroleum gas<LPG>, or 1000ppm iso-butane<i-C_4H_{10}>concentration in air and use value of Load resistance that (RL) about 20 KΩ (5KΩ to 47 KΩ).

When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence.

Driver circuit:

This circuit diagram shows how a 555 timer IC is configured to function as a basic monostable multivibrator. A monostable multivibrator is a timing circuit that changes state once triggered, but returns to its original state after a certain time delay. It got its name from the fact that only one of its output states is stable. It is also known as a 'one-shot'.

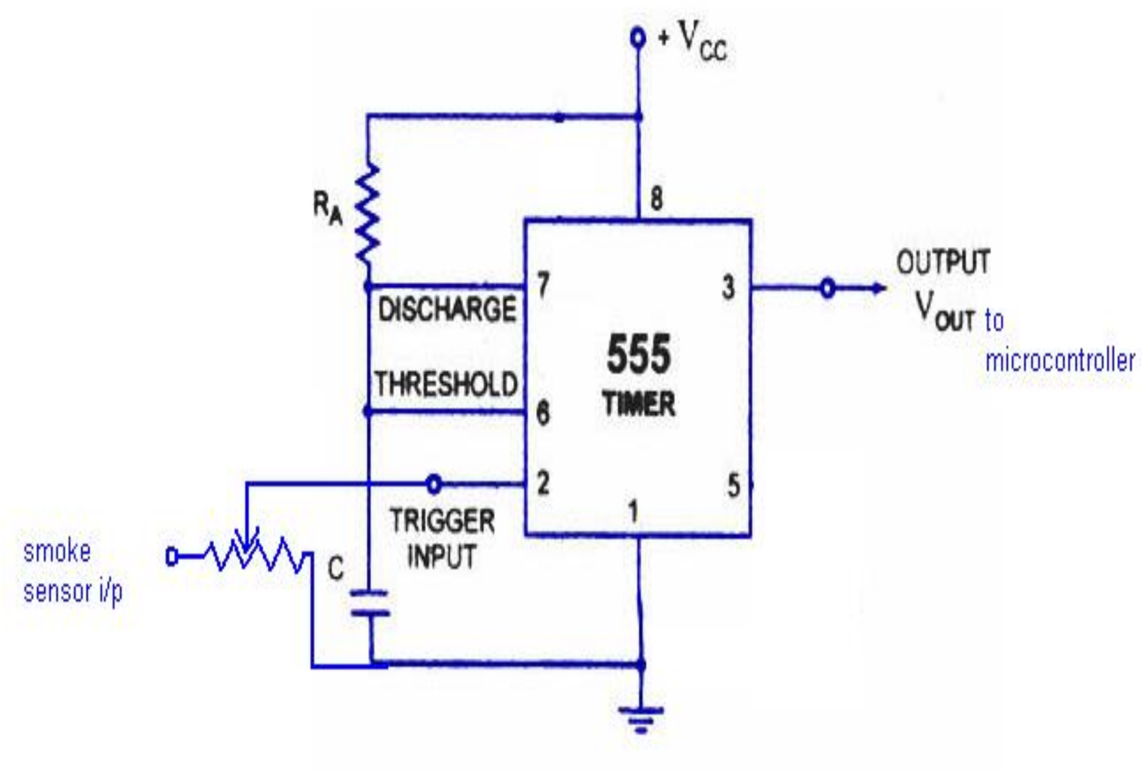


Fig: 4.7.3 555 Timer Circuit

In this circuit, a negative pulse applied at pin 2 triggers an internal flip-flop that turns off pin 7's discharge transistor, allowing C1 to charge up through R1. At the same time, the flip-flop brings the output (pin 3) level to 'high'. When capacitor C1 as charged up to about 2/3 Vcc, the flip-flop is triggered once again, this time making the pin 3 output 'low' and turning on pin 7's discharge transistor, which discharges C1 to ground. This circuit, in effect, produces a pulse at pin 3 whose width t is just the product of R1 and C1, i.e., t=R1C1.

THEORY OF OPERATION

Looking at the functional schematic shown you can see that pin 7 is a transistor going to ground.

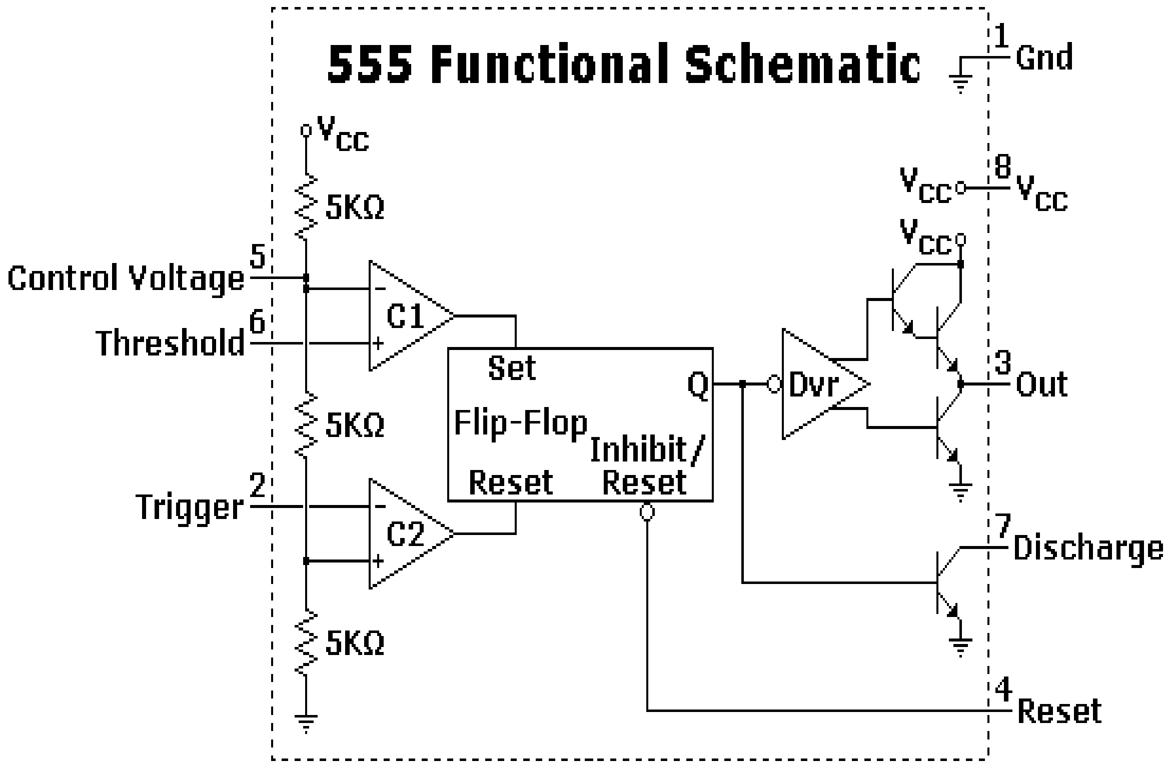
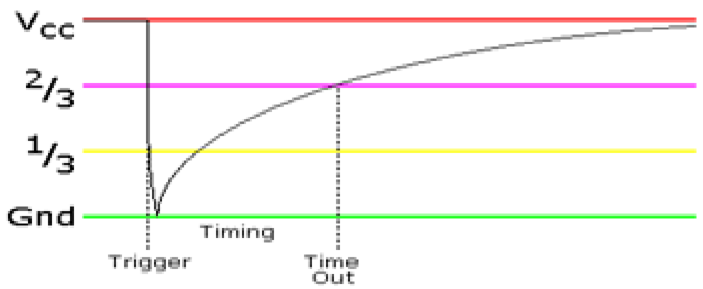


Fig. 4.7.4 555 Functional Schematic Diagram

This transistor is simply a switch that normally conducts until pin 2 (which is connected through the comparator C1, which feeds the internal flip flop) is brought low, allowing the capacitor Ct to start charging. Pin 7 stays off until the voltage on Ct charges to 2/3 of the power supply voltage, where the timer times out and pin 7 transistor turns on again, its normal state in this circuit. The following figure shows the sequence of switching, with red being the higher voltages and green being ground (0 volts), with the spectrum in between since this is fundamentally an analog circuit.



555 Timing Cycle

THEORY OF DC MOTOR

The speed of a DC motor is directly proportional to the supply voltage, so if we reduce the supply voltage from 12 Volts to 6 Volts, the motor will run at half the speed. How can this be achieved when the battery is fixed at 12 Volts? The speed controller works by varying the average voltage sent to the motor. It could do this by simply adjusting the voltage sent to the motor, but this is quite inefficient to do. A better way is to switch the motor's supply on and off very quickly. If the switching is fast enough, the motor doesn't notice it, it only notices the average effect.

When you watch a film in the cinema, or the television, what you are actually seeing is a series of fixed pictures, which change rapidly enough that your eyes just see the average effect - movement. Your brain fills in the gaps to give an average effect.

Now imagine a light bulb with a switch. When you close the switch, the bulb goes on and is at full brightness, say 100 Watts. When you open the switch it goes off (0 Watts). Now if you close the switch for a fraction of a second, then open it for the same amount of time, the filament won't have time to cool down and heat up, and you will just get an average glow of 50 Watts. This is how lamp dimmers work, and the same principle is used by speed controllers to drive a motor. When the switch is closed, the motor sees 12 Volts, and when it is open it sees 0 Volts. If the switch is open for the same amount of time as it is closed, the motor will see an average of 6 Volts, and will run more slowly accordingly. The graph below shows the speed of a motor that is being turned on and off

H-BRIDGE:

An H-bridge is an electronic circuit which enables DC electric motors to be run forwards or backwards. These circuits are often used in robotics. H-bridges are available as integrated circuits, or can be built from discrete components.

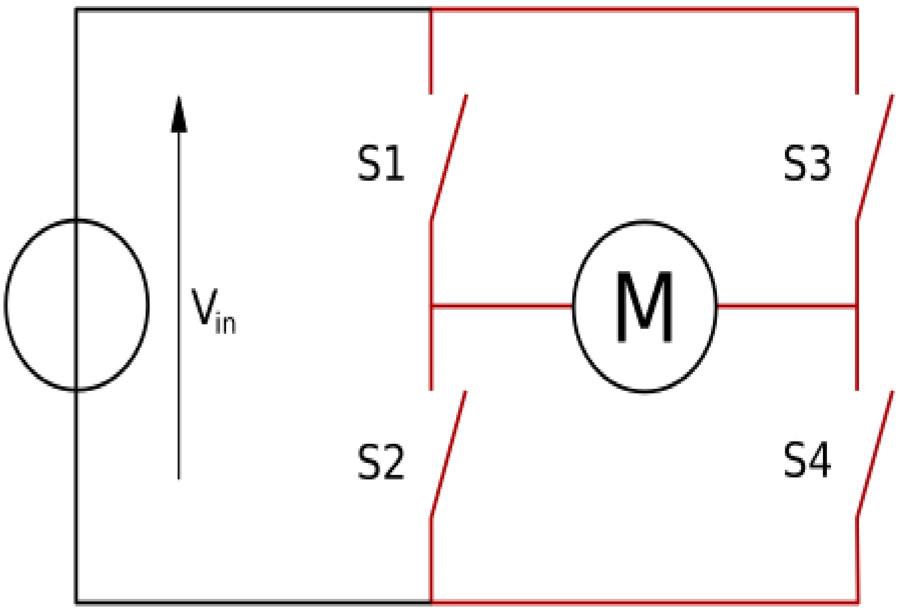


Fig.4.7.5. H-Bridge

The two basic states of a H-bridge. The term "H-bridge" is derived from the typical graphical representation of such a circuit. An H-bridge is built with four switches (solid-state or mechanical). When the switches S1 and S4 (according to the first figure) are closed (and S2 and S3 are open) a positive voltage will be applied across the motor. By opening S1 and S4 switches and closing S2 and S3 switches, this voltage is reversed, allowing reverse operation of the motor.

Using the nomenclature above, the switches S1 and S2 should never be closed at the same time, as this would cause a short circuit on the input voltage source. The same applies to the switches S3 and S4. This condition is known as shoot-through.

Operation

The H-Bridge arrangement is generally used to reverse the polarity of the motor, but can also be used to 'brake' the motor, where the motor comes to a sudden stop, as the motors terminals are shorted, or to let the motor 'free run' to a stop, as the motor is effectively disconnected from the circuit. The following table summarizes operation.

Table: 4.7.1. Operation of the motor

S1	**S2**	**S3**	**S4**	**Result**
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
0	0	0	0	Motor free runs
0	1	0	1	Motor brakes

H-Bridge Driver:

The switching property of this H-Bridge can be replace by a Transistor or a Relay or a MOSFET or even by an IC. Here we are replacing this with an IC named L293D as the driver whose description is as given below.

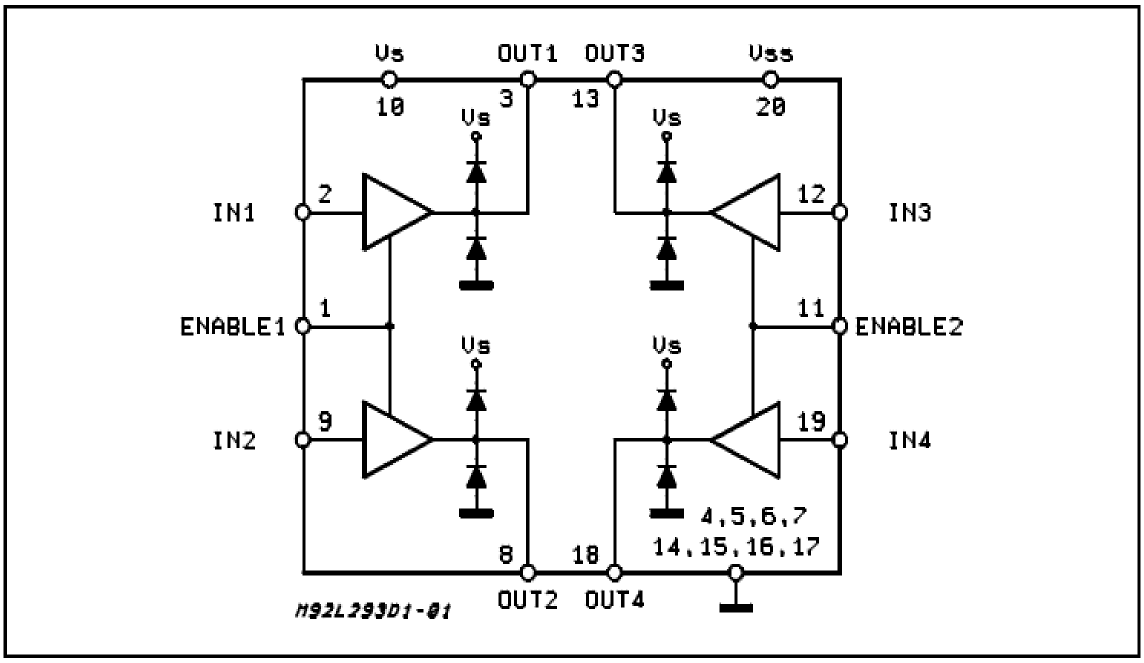
Features:

- 600mA OUTPUT CURRENT CAPABILITY
- PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non-repetitive)
- PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V
- (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoids, DC and stepping motors) and switching power transistors. To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included. This device is suitable for use in switching applications at frequencies up to 5 kHz. The L293D is assembled in a 16 lead plastic package which has 4 centre pins connected together and used for heatsinking The L293DD is assembled in a 20 lead surface mount which has 8 centre pins connected together and used for heatsinking.

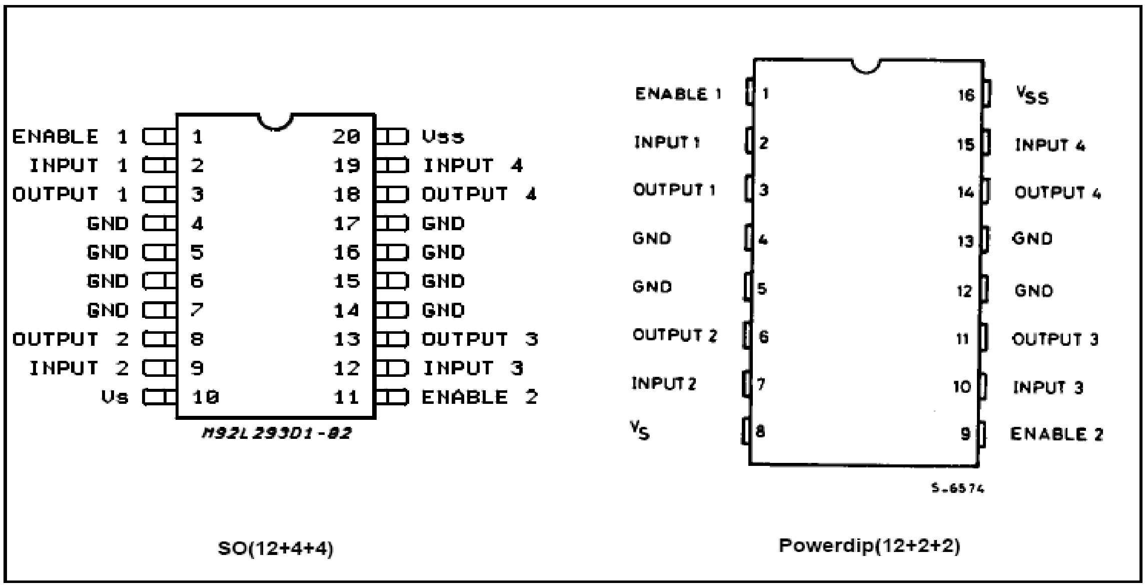
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_i	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 μs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS



4.8. Ultrasonic sensor:

The sensor is primarily intended to be used in security systems for detection of moving objects, but can be effectively involved in intelligent children's toys, automatic door opening devices, and sports training and contact-less-speed measurement equipment.

Introduction

Modern security systems utilize various types of sensors to detect unauthorized object access attempts. The sensor collection includes infrared, microwave and ultrasound devices, which are intended to detect moving objects. Each type of sensor is characterized by its own advantages and drawbacks. Microwave sensors are effective in large apartments because microwaves pass through dielectric materials. But these sensors consist of expensive super-high frequency components and their radiation is unhealthy for living organisms.

Infrared sensors are characterized by high sensitivity, low cost and are widely used. But, these sensors can generate false alarm signals if heating systems are active or temperature change speed exceeds some threshold level.

Ultrasound motion detection sensors are characterized by small power consumption, suitable cost and high sensitivity. That is why this kind of sensor is commonly used in home, office and car security systems. Existing ultrasound sensors consist of multiple passive and active components and are relatively complicated for production and testing. Sensors often times require a laborious tuning process.

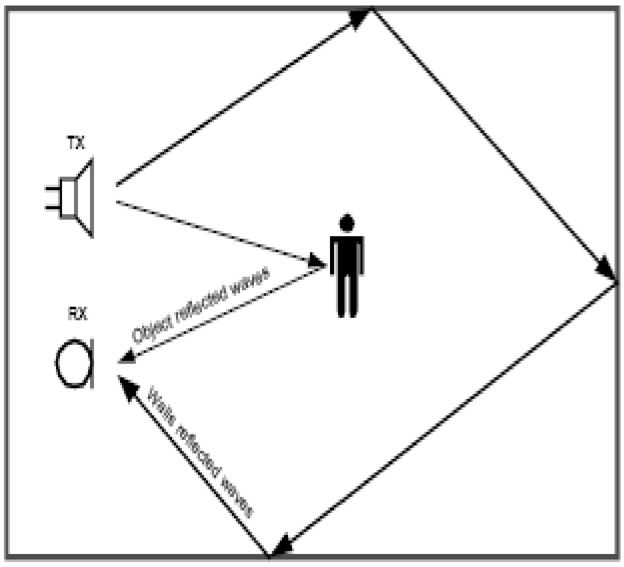


Fig: 4.8. Basic Sensor Operation Principle

The ultrasound transmitter **TX** is emitting ultrasound waves into sensor ambient space continuously. These waves are reflecting from various objects and are reaching ultrasound receiver **RX**. There is a constant interference figure if no moving objects are in the placement.

Any moving object changes the level and phase of the reflected signal, which modifies the summed received signal level. Most low cost sensors (car security systems, for instance) perform reflected signal amplitude analysis to detect moving objects. In spite of implementation simplicity, this detection method is characterized by a high sensitivity to noise signals. For example, heterogeneous airflows, sensor vibrations, room window and door deformations, and gusts can change the interference figure and generate false alarm signals.

4.9. BUZZER:

What is a Buzzer: Working & Its Applications

There are many ways to communicate between the user and a product. One of the best ways is audio communication using a buzzer IC. So during the design process, understanding some technologies with configurations is very helpful. So, this article discusses an overview of an audio signaling device like a beeper or a buzzer and its working with applications.

What is a Buzzer?

An audio signaling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

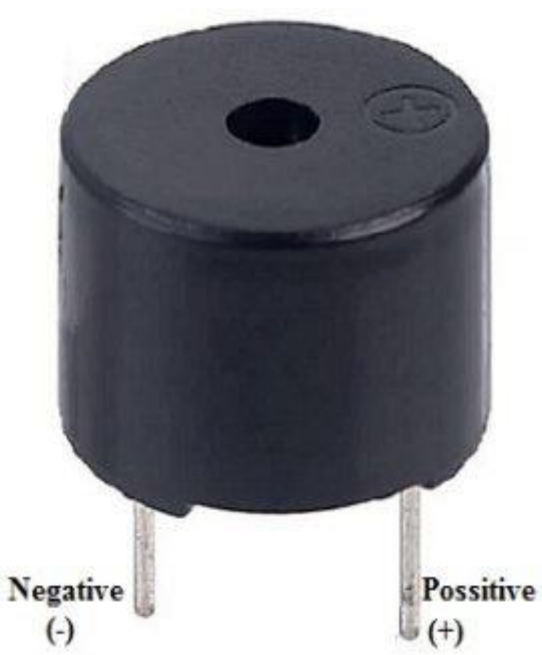


Fig: 4.9. Buzzer Pin Configuration

The pin configuration of the buzzer is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the '+' symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the '-'symbol or short terminal and it is connected to the GND terminal.

History

The history of an electromechanical buzzer and piezoelectric is discussed below.

Electromechanical

This buzzer was launched in the year 1831 by an American Scientist namely Joseph Henry but, this was used in doorbells until they were eliminated in 1930 in support of musical bells, which had a smooth tone.

Piezoelectric

These buzzers were invented by manufacturers of Japanese & fixed into a broad range of devices during the period of 1970s – 1980s. So, this development primarily came due to cooperative efforts through the manufacturing companies of Japanese. In the year 1951, they recognized the Application Research Committee of Barium Titanate that allows the corporations to be cooperative competitively & bring about numerous piezoelectric creations. The main function of this is to convert the signal from audio to sound.

Specifications

The specifications of the buzzer include the following.

- Colour is black
- The frequency range is 3,300Hz
- Operating Temperature ranges from – 20° C to +60°C
- Operating voltage ranges from 3V to 24V DC
- The sound pressure level is 85dBA or 10cm
- The supply current is below 15Ma

Types of Buzzer:

A buzzer is available in different types which include the following.

- Piezoelectric
- Electromagnetic
- Mechanical
- Electromechanical
- Magnetic

Piezoelectric

As the name suggests, the piezoelectric type uses the piezoelectric ceramic’s piezoelectric effect & pulse current to make the metal plate vibrate & generate sound. This kind of buzzer is made with a resonance box, multi resonator, piezoelectric plate, housing, impedance matcher, etc. Some of the buzzers are also designed with LEDs.

The multi resonator of this mainly includes ICs and transistors. Once the supply is given to this resonator, it will oscillate and generates an audio signal with 1.5 to 2.kHz. The impedance matcher will force the piezoelectric plate to produce sound.

Electromagnetic

This type of buzzer is made with a magnet, solenoid coil, oscillator, housing, vibration diaphragm, and magnet. Once the power supply is given, the oscillator which produces the audio signal current will supply throughout the solenoid coil to generate a magnetic field.

Sometimes, the vibration diaphragm will vibrate & generates sound under the magnet & solenoid coil interaction. The frequency range of this ranges from 2 kHz to 4 kHz.

Mechanical

These types of buzzers are subtypes of electromagnetic, so the components used in this type are also similar. But the main difference is that the vibrating buzzer is placed on the outside instead of the inside.

Electromechanical

The designing of these types of buzzers can be done with a bare metal disc & an electromagnet. The working principle of this is similar to magnetic and electromagnetic. It generates sound throughout the disc movement & magnetism.

Magnetic

Like a piezo type, magnetic is also used to generate a sound but they are different due to core functionality. The magnetic type is more fixed as compared to the piezo type because they work through a magnetic field.

Magnetic buzzers utilize an electric charge instead of depending on piezo materials to generate a magnetic field, after that it permits another element of the buzzer to vibrate & sound.The applications of magnetic buzzers are similar to the piezo type in household devices, alarms such as watches, clocks & keyboards.

Working Principle

The working principle of a buzzer depends on the theory that, once the voltage is given across a piezoelectric material, then a pressure difference is produced. A piezo type includes piezo crystals among two conductors. Once a potential disparity is given across these crystals, then they thrust one conductor & drag the additional conductor through their internal property. So this continuous action will produce a sharp sound signal.

Mounting Configurations

The mounting configurations of buzzers include the following.

- Panel Mount
- Wire Leads
- Screw Terminals

- Through Hole
- Spring Contact
- Surface Mount

How to use a Buzzer?

A buzzer is an efficient component to include the features of sound in our system or project. It is an extremely small & solid two-pin device thus it can be simply utilized on breadboard or PCB. So in most applications, this component is widely used.There are two kinds of buzzers commonly available like simple and readymade. Once a simple type is power-driven then it will generate a beep sound continuously.

This buzzer uses a DC power supply that ranges from 4V – 9V. To operate this, a 9V battery is used but it is suggested to utilize a regulated +5V/+6V DC supply. Generally, it is connected through a switching circuit to switch ON/OFF the buzzer at the necessary time interval.

How to Choose a Buzzer?

While choosing a buzzer or speaker, many principles need to consider like the following.

- Size of the product
- Consumption of Current
- Type of terminal
- Frequency Voltage
- Volume
- Type
- AC/DC Voltage
- The tone is Continuous/Pulsed
- Fixing – Pins, Leads/Surface Mount
- Output of Sound

- Feedback Option
- Piezo Elements

Advantages: The advantages of the buzzer include the following

- Simply Compatible
- Frequency Response is Good
- Size is small
- Energy Consumption is less
- The Range of Voltage usage is Large
- Sound Pressure is high

Disadvantages: The disadvantages of the buzzer include the following

- Controlling is a little hard
- Generates Annoying Sound
- Training is necessary to know how to repair the condition without just turning off.

Applications: The applications of the buzzer include the following.

- Communication Devices
- Electronics used in Automobiles
- Alarm Circuits
- Portable Devices
- Security Systems
- Timers
- Household Appliances
- Electronic Metronomes
- Sporting Events
- Annunciator Panels
- Game Show

CHAPTER-5

DESIGN OF SOFTWARE

5. DESIGN OF SOFTWARE

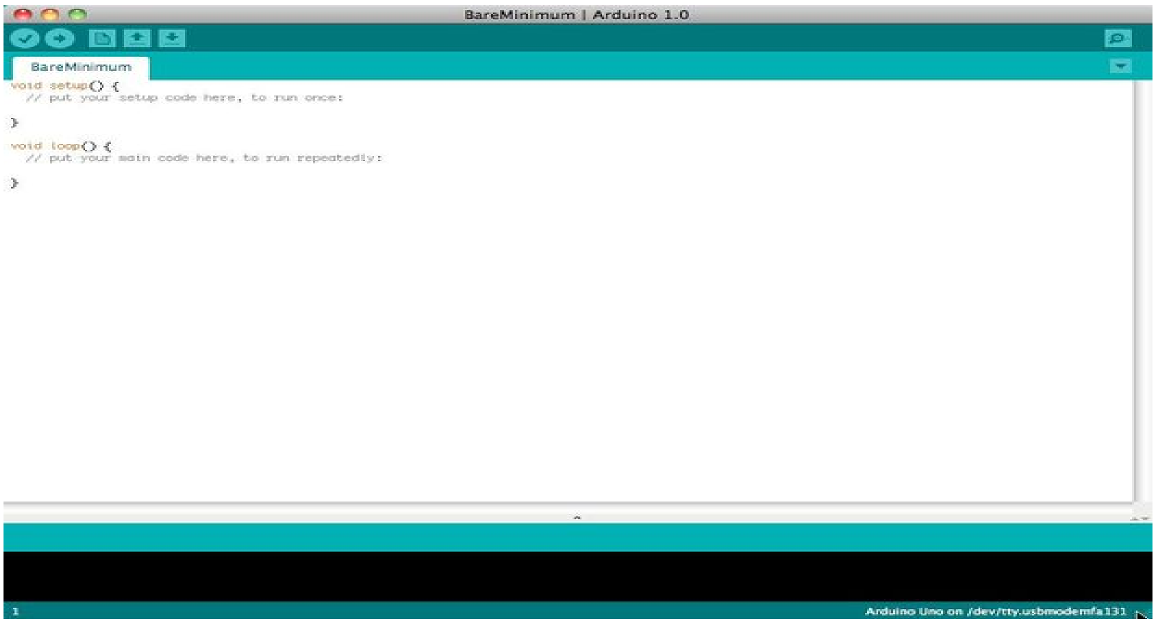
5.1. INTRODUCTION TO ARDUINO IDE SOFTWARE:

This is free software (evaluation version) which solves many of the pain points for an embedded system developer. This software is an Integrated Development Environment (IDE), which integrated text editor to write program, a compiler and it will convert your source code into HEX file. Here is simple guide to start working with Arduino IDE Vision which can be used for:

- Writing programs in Arduino IDE
- Compiling and assembling programs
- Debugging programs

5.2. SOFTWARE STEPS:

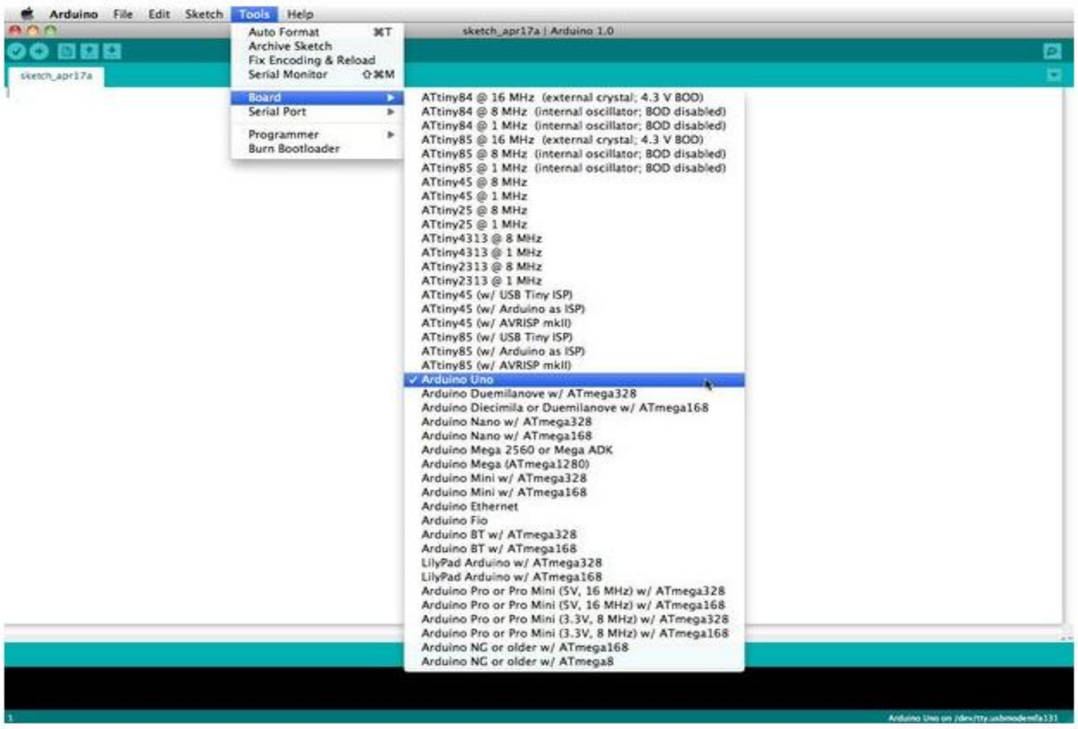
Before you can start doing anything with the Arduino, you need to download and install the Arduino IDE (integrated development environment).


BareMinimum | Arduino 1.0
BareMinimum

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

1
Arduino Uno on /dev/tty.usbmodemfa131

After the opening IDE the settings are changed in order to connect to the Arduino.


Arduino File Edit Sketch Tools Help
sketch_apr17a | Arduino 1.0
sketch_apr17a
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
ATtiny84 @ 16 MHz (external crystal; 4.3 V BOD)
ATtiny84 @ 8 MHz (internal oscillator; BOD disabled)
ATtiny84 @ 1 MHz (internal oscillator; BOD disabled)
ATtiny85 @ 16 MHz (external crystal; 4.3 V BOD)
ATtiny85 @ 8 MHz (internal oscillator; BOD disabled)
ATtiny85 @ 1 MHz (internal oscillator; BOD disabled)
ATtiny45 @ 8 MHz
ATtiny45 @ 1 MHz
ATtiny25 @ 8 MHz
ATtiny25 @ 1 MHz
ATtiny4313 @ 8 MHz
ATtiny4313 @ 1 MHz
ATtiny2313 @ 8 MHz
ATtiny2313 @ 1 MHz
ATtiny45 (w/ USB Tiny ISP)
ATtiny45 (w/ Arduino as ISP)
ATtiny45 (w/ AVRISP mkII)
ATtiny85 (w/ USB Tiny ISP)
ATtiny85 (w/ Arduino as ISP)
ATtiny85 (w/ AVRISP mkII)
✓ Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Uno on /dev/tty.usbmodemfa131

Before you can start doing anything in the Arduino programmer, you must set the board-type and serial port.

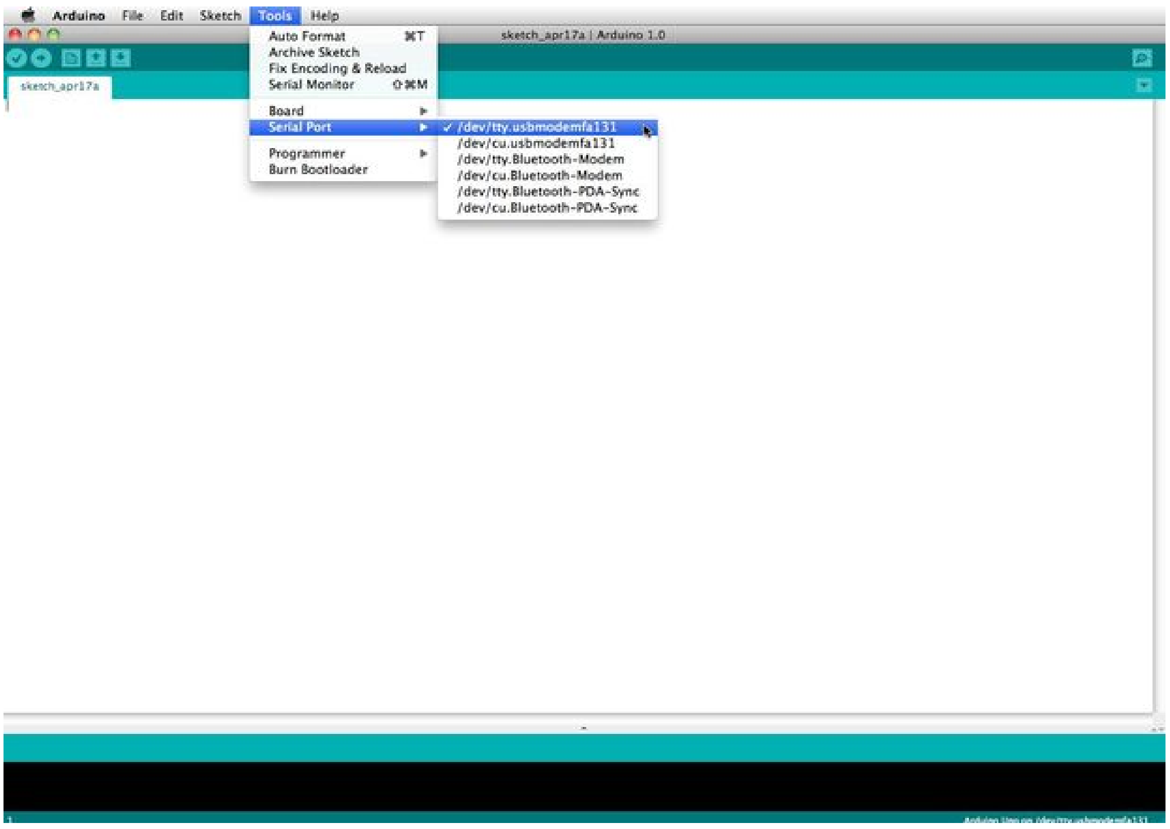
To set the board, go to the following:

Tools --> Boards

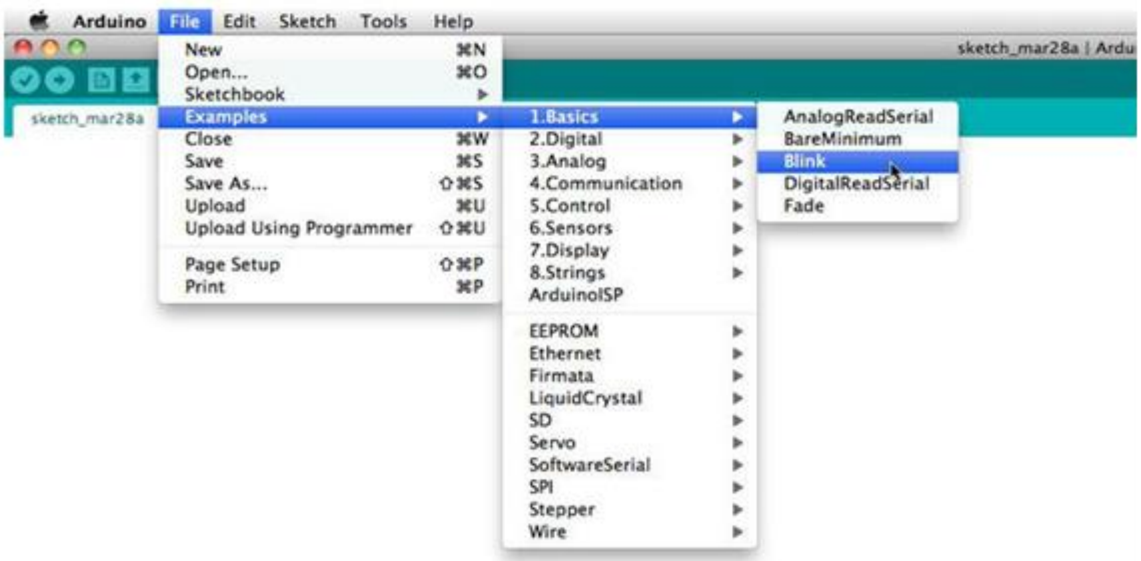
Select the version of board that you are using. Since I have an Arduino Uno plugged in, I obviously selected "Arduino Uno."

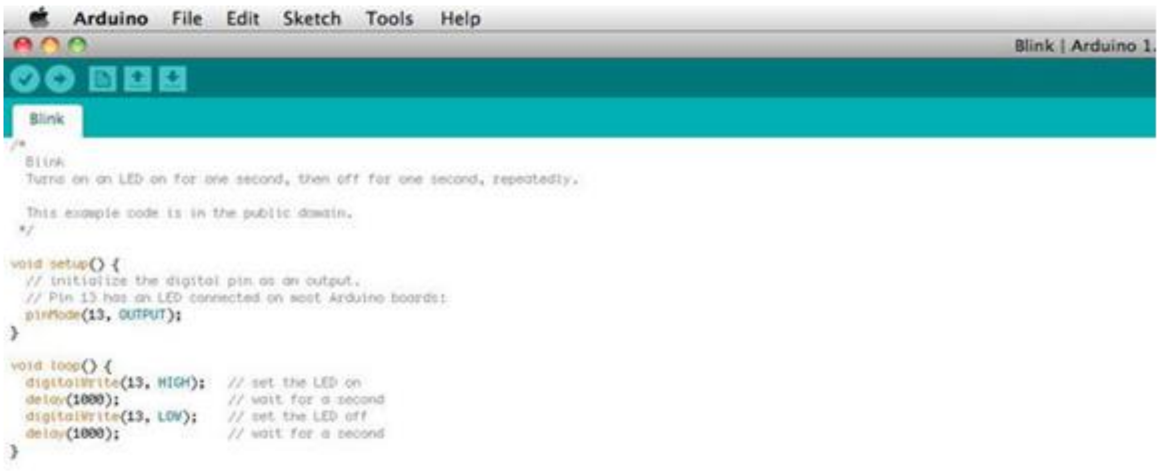
To set the serial port, go to the following:

Tools --> Serial Port



Arduino programs are called sketches. The Arduino programmer comes with a ton of example sketches preloaded. This is great because even if you have never programmed anything in your life, you can load one of these sketches and get the Arduino to do something.

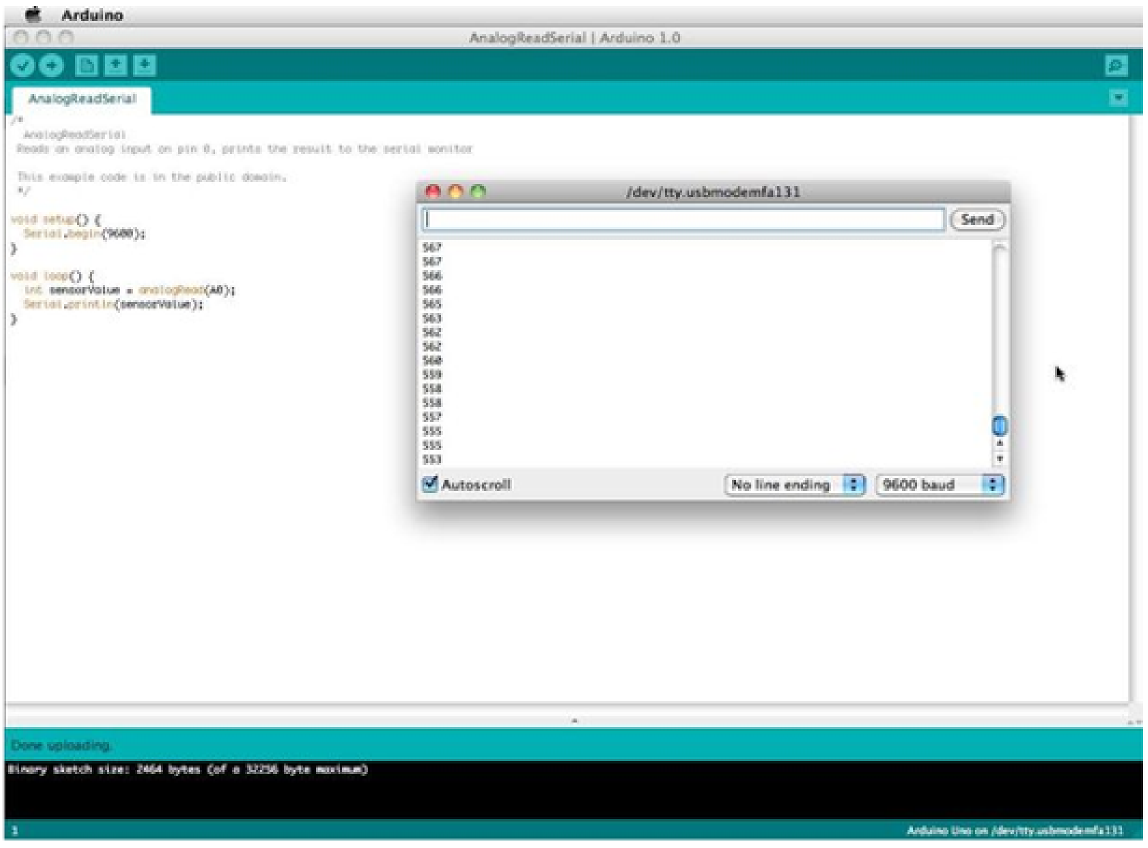




```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

The serial monitor allows your computer to connect serially with the Arduino. This is important because it takes data that your Arduino is receiving from sensors and other devices and displays it in real-time on your computer. Having this ability is invaluable to debug your code and understand what number values the chip is actually receiving.

For instance, connect center sweep (middle pin) of a potentiometer to A0, and the outer pins, respectively, to 5v and ground. Next upload the sketch shown below:
File --> Examples --> 1.Basics --> Analog Read Serial
Click the button to engage the serial monitor which looks like a magnifying glass. You can now see the numbers being read by the analog pin in the serial monitor. When you turn the knob the numbers will increase and decrease.
The numbers will be between the range of 0 and 1023. The reason for this is that the analog pin is converting a voltage between 0 and 5V to a discreet number.



```
/*
  AnalogReadSerial
 Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

```
567
567
566
566
565
563
562
562
560
559
558
558
557
555
555
553
```

```
Binary sketch size: 2464 bytes (of a 32256 byte maximum)
```

CHAPTER-6

PROJECT DESCRIPTION

6. PROJECT DESCRIPTION

This chapter deals with working and circuits of “Design and implementation of GPS controlled environment monitoring robot system based on IOT”. It can be simply understood by its block diagram &circuit diagram.

6.1. BLOCK DIAGRAM:

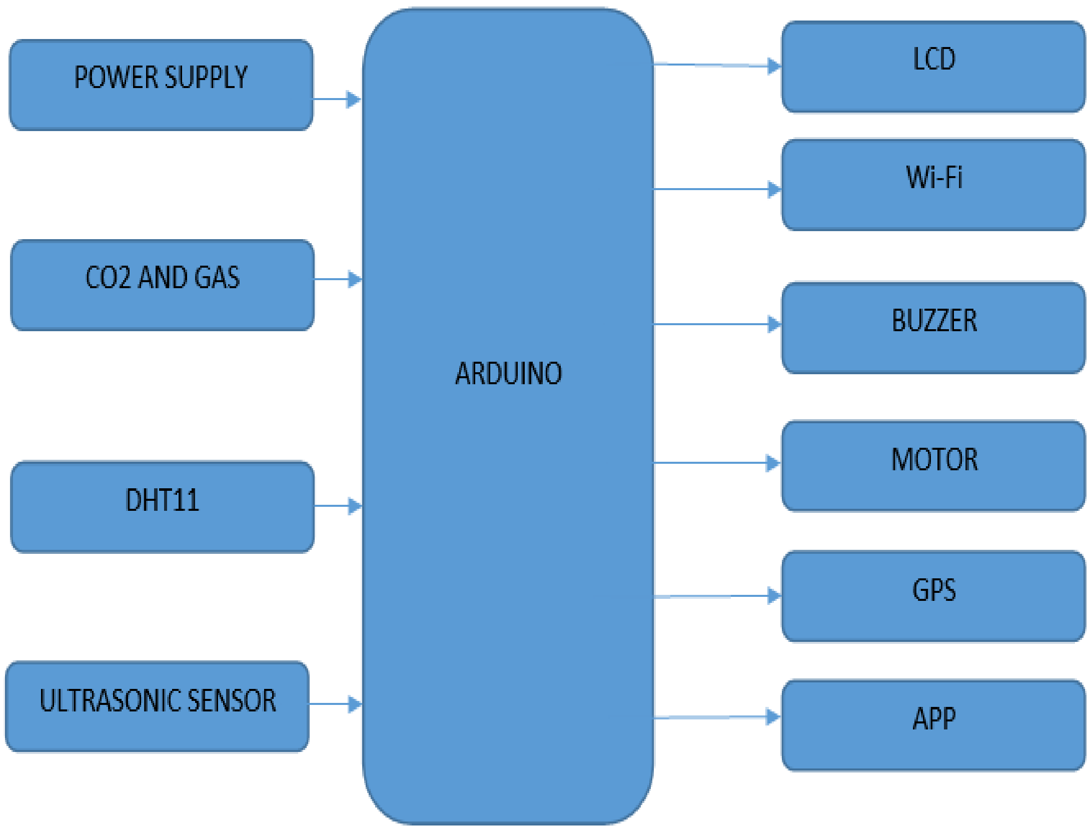


Fig 6.1 Block Diagram

6.2. SOFTWARE REQUIREMENTS:

- Arduino
- Embedded C language

6.3. HARDWARE REQUIREMENTS:

LM 35, ARDUINO, DC MOTOR

6.4. WORKING:

The figure shows the block diagram of our proposed system wherein all the devices are connected to the microcontroller i.e., Arduino .With the help of motor driver, robot moves to the location, Sensors takes the input from the environment and sends it to the arduino which than sends it to the android app, when asked

- The working of our project can be divided into two aspects Environment monitoring system, Navigation control system and a Core which handles the operation
- Environment monitoring system consists of the sensors namely gas sensors, air quality sensors, and temperature and humidity sensors.
- These sensors start to collect the data from the environment after reaching the specified location and give it to the core for further processing.
- Navigation system involves the GPS, Motor driver circuit and a Bluetooth module.
- This unit helps in movement of robot to the desired location with the help of GPS coordinates and movement is controlled by Bluetooth module /Internet through the user.
- The core used here is Raspberry pi 3 which governs the other two Systems. The data obtained from the sensing unit is processed and is given to the android application build for this project when the user wants the data and can access it from any part of the world.
- As the data is continuously updated, the supervisor or the user can check for any abnormality and can act accordingly.
- Thus the data is collected without human intervention which was the main objective of our project.

The robotic gadget proposed was intended to coordinate the inserted equipment, programming, and IOT modules. The design of the gadget is appeared which shows the IOT and ARM implanted automated gadget block graphs. There are two parts of the whole mechanical framework:

1. Environment Monitoring System: This framework is answerable for social event sensor information and transferring the information accumulated to the IOT stage.

2. Navigation and Control System: This framework's basic role is to explore and control the development of the automated framework as coordinated by the application.

CHAPTER-7

RESULT

7. RESULT

The Arduino-UNO microcontroller is attached to three detectors and a WiFi module in our suggested framework. The 3 sensors gather readings, analyze pollutant concentrations in the environment, and display the results on the consecutive screen of the Arduino IDE.The detector in more advantageous locations acquires pollution details around the device. These sensor measurements are subsequently transferred to the server with the assistance of the WiFi module.

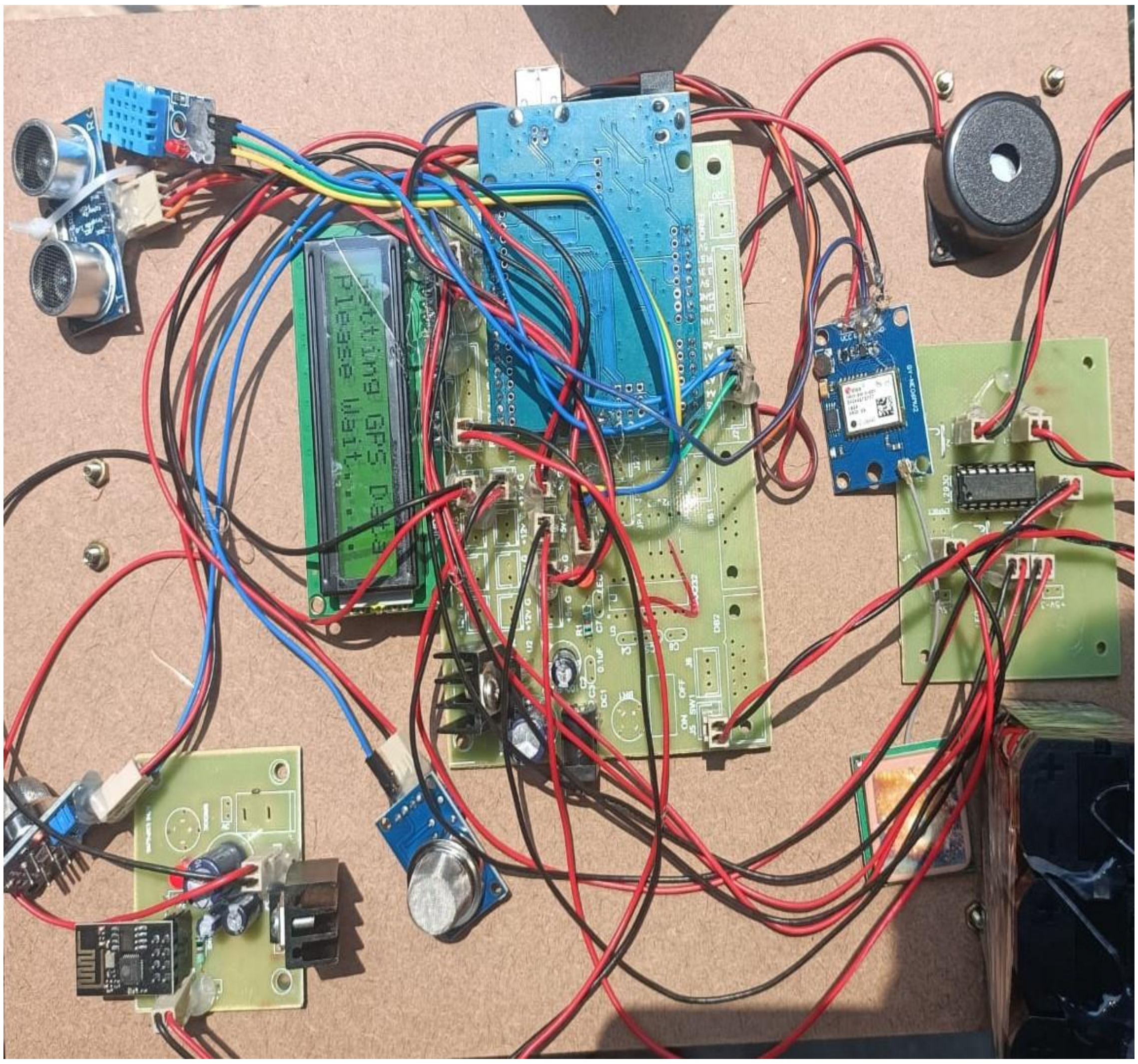
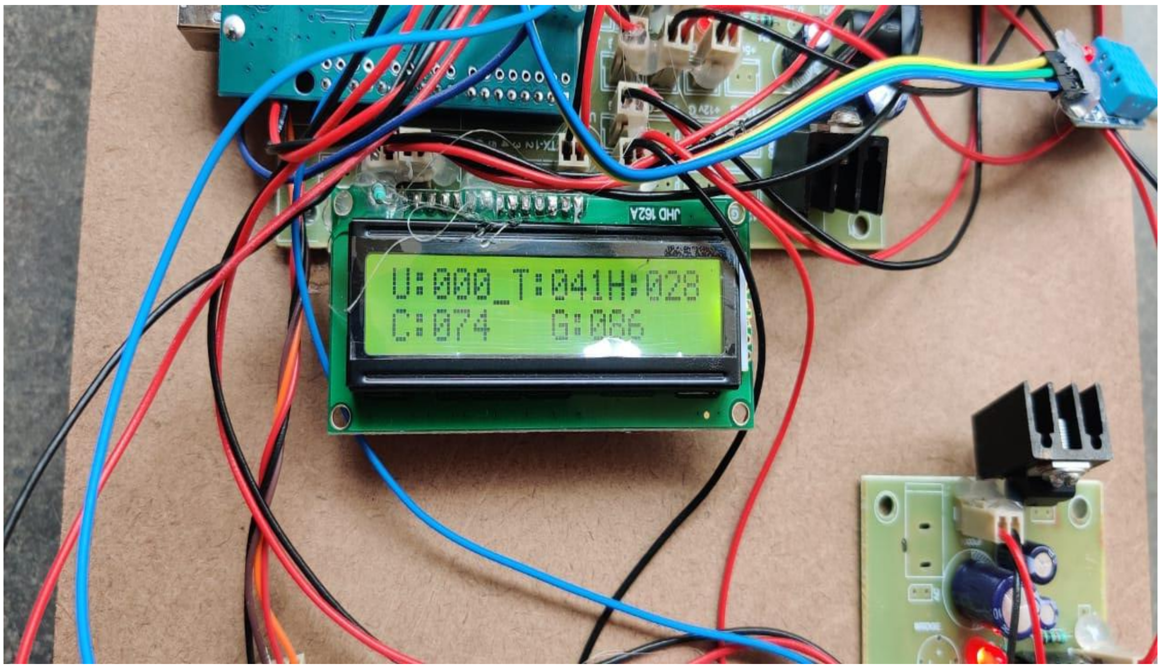


Fig.7.1. Hardware Kit



```
U:000_T:041H:028
C:074    G:086
```

Fig.7.2. Temperature & Humidity

Fig.7.3. Output

CHAPTER-8

CONCLUSION

8. CONCLUSION

Study and execution of a structure is done to follow environmental boundaries utilizing a situation. A low-power answer for setting up an environmental framework is given by the framework. The gadget is tried in both indoor and outside settings, and the ecological conditions from sensor information are effectively adjusted. Plan and usage of a GPS controlled robot for IOT based observing of natural boundaries have been accomplished. The manufactured robot-based installed gadget with the IOT stage can follow the ecological boundaries, and conservative and practical air quality estimation. The outcomes got are viewed as helpful in checking the natural conditions progressively. The assembled App empowers clients to handily control and explore the robot.

The system to watch the air of environment using Arduino microcontroller, IOT Technology is proposed to enhance quality of air. With the utilization of IOT technology enhances the method of monitoring various aspects of environment like air quality monitoring issue proposed. Here the using of MQ2 gas sensor gives the sense of various sort of dangerous gas and Arduino is that the heart of this project. We implemented our system on the Arduino system monitor and the 16x2 LCD display by using the MQ2 sensor and the system displays the values that it received from the MQ2 sensor and will show the corresponding air quality rating as per the PPM value.

CHAPTER-9

FUTURE SCOPE

9. FUTURE SCOPE

This project can also be implemented using a Wi-Fi module that displays the AQI (Air Quality Index) on a web server continuously. The server can be accessed by everyone, anytime and it can keep them informed about the air quality of a particular location. We can also connect multiple Arduino's and display the AQI of multiple areas and compare them to find which area is more polluted and which one is cleaner. We can also use the GSM module to send messages to users when the air quality goes above a particular limit i.e. the citizens can be alerted if the AQI goes above 300 and the air quality turns bad. This can keep them well prepared. Additional sensors to measure temperature, pressure, humidity can be added. Since many air quality monitoring systems in Mumbai are not in operational condition, this project can be implemented for the benefit of the city. The project can also be implemented on a large scale by installing large LED screens across the city that displays the AQI and its corresponding category. I can also show warnings so that the citizens can take precautions when required.

CHAPTER-10

REFERENCES

10. REFERENCES

1. J. J. Acevedo, J. Messias, J. Capitan, R. Ventura, L. Merino, andP. U. Lima, "A dynamic weighted area assignment based on a particle filter for active cooperative perception", IEEE Robot. Autom. Lett., vol. 5, no. 2, pp. 736–743, Apr. 2020.
2. Y. Wei and R. Zheng, „„Informative path planning for mobile sensing with reinforcement learning,"" 2020, arXiv:2002.07890. [Online]. Available: http://arxiv.org/abs/2002.07890
3. Shuxiang Guo, Xujie Yang, Jian Guo, Chunying Li, "Design of Wireless Mobile Environment Monitoring System Based on Spherical Amphibious Robots"- Proceedings of 2018 IEEE International Conference on Mechatronics and Automation August 5 - 8, Changchun, China.
4. Biao Jiang and Christian Huacón, "Cloud-based smart device for environment monitoring," 2017 IEEE Conference on Technologies for Sustainability (SusTech), 12-14 Nov. 2017, DOI: 10.1109/SusTech.2017.8333472.
5. Sanjib KumarDeb; JahedHossen Rokky; Tuton Chandra Mallick; Juliana Shetara, "2017 4th International Conference on Advances in Electrical Engineering (ICAEE)," 28-30 Sept. 2017, DOI: 10.1109/ICAEE.2017.8255367.
6. Rohini Shete and Sushma Agrawal, "IoT based urban climate monitoring using Raspberry Pi", 2016 International Conference on Communication and Signal Processing (ICCSP), 6-8 April 2016, DOI: 10.1109/ICCSP.2016.7754526.
7. Keyur K Patel and Sunil M Patel,"Internet ofThings-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges," International Journal of Engineering Science and Computing, Volume 6, Issue No. 5, May 2016.
8. Kan Shoji, Keisuke Morishima, Yoshitake Akiyama, Nobuhumi Nakamura and Hiroyuki Ohno, "Autonomous Environmental Monitoring by Self-powered Biohybrid Robot"- Proceedings of2016 IEEE International Conference on Mechatronics and Automation August 7 - 10, Harbin, China.

9. Dr. Ovidiu Vermesan SINTEF, Norway, Dr. Peter FriessEU, Belgium, “Internet of Things–From Research and Innovation to MarketDeployment”, river publishers‟ series in communications, 2014.

10. Dr. Ovidiu Vermesan SINTEF, Norway, Dr. Peter FriessEU, Belgium, “Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems”, river publishers‟ series in communications, 2013.

11. Diogo Santos Ortiz Correa, Diego Fernando Sciotti, Marcos Gomes Prado, Daniel Oliva Sales, Denis Fernando Wolf, Fernando Santos Osório, “Mobile Robots Navigation in Indoor Environments Using Kinect Sensor”-2012 Second Brazilian Conference on Critical Embedded Systems.

12. O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, et al., “Internet of Things Strategic Research Agenda”, Chapter 2 in Internet of Things -Global Technological and Societal Trends, River Publishers, 2011.

13. Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju And Tomi Westerlund, “Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision”-Digital Object Identifier 10.1109/ACCESS.2020.3030190

SOURCE CODE:

```
#include <LiquidCrystal.h>
#include <stdio.h>

#include <SoftwareSerial.h>
SoftwareSerial mySerial(8, 9);

LiquidCrystal lcd(6, 7, 5, 4, 3, 2);

#include <Wire.h>
#include "dht.h"

const int trigPin = A4;
const int echoPin = A3;

int m1a = 10;
int m1b = 11;
int m2a = 12;
int m2b = 13;

#define dht_apin A2
dht DHT;

int buzzer = A5;



int tempc=0,humc=0;

int rtr1=0;
int dist1=0,dist2=0,dist3,sts1=0;

long duration;
int distanceCm, distanceInch;

 unsigned char rcv,count,gchr='x',gchr1='x',robos='s';

 char pastnumber[10];
 char gpsval[50];
// char dataread[100] = "";
// char lt[15],ln[15];


int i=0,k=0;
int  gps_status=0;
float latitude=0;
float logitude=0;
String Speed="";
String gpsString="";
char *test="$GPRMC";


int hbtc=0,hbtc1=0,rtrl=0;

unsigned char gv=0,msg1[10],msg2[11];
 float lati=0,longi=0;
unsigned int lati1=0,longi1=0;
unsigned char flat[5],flong[5];
unsigned char finallat[8],finallong[9];


 int ii=0;

float vout=0;

int sti=0;
String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete


void okcheck()
{
  unsigned char rcr;
  do{
      rcr = Serial.read();
    }while(rcr != 'K');
}

unsigned int ultra_dist()
{
   digitalWrite(trigPin, LOW);
   delayMicroseconds(5);
   digitalWrite(trigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigPin, LOW);
   duration = pulseIn(echoPin, HIGH);
   distanceCm= duration*0.034/2;
   //distanceInch = duration*0.0133/2;
   //lcd.setCursor(2,0);
   //convertl(distanceCm);
   dist1 = distanceCm;
   return dist1;
}


void send_link()
 {
    Serial.write("AT+CMGS=\"");
    Serial.write(pastnumber);
    Serial.write("\"\r\n");  delay(2500);
    Serial.write("https://www.google.co.in/search?client=opera&q=");
    for(ii=0;ii<=6;ii++){Serial.write(finallat[ii]);}
    Serial.write("%2C");
    for(ii=0;ii<=7;ii++){Serial.write(finallong[ii]);}
    Serial.write(0x1A);delay(4000);delay(4000);
 }

void beep()
{
  digitalWrite(buzzer, LOW);delay(2500);  digitalWrite(buzzer, HIGH);
}
void serialFlush()
{
  while(Serial.available() > 0)
  {
    char t = Serial.read();
  }
}

void setup()
{
 Serial.begin(9600);//serialEvent();
 mySerial.begin(9600);

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

  pinMode(m1a, OUTPUT);pinMode(m1b, OUTPUT);
  pinMode(m2a, OUTPUT);pinMode(m2b, OUTPUT);
  pinMode(buzzer, OUTPUT);

  digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
  digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
  digitalWrite(buzzer, HIGH);

  lcd.begin(16, 2);lcd.cursor();
  lcd.print("GPS Controlled Env");
  lcd.setCursor(0,1);
  lcd.print(" Monitoring Robot ");
  delay(1000);

get_gps();
gps_convert();

    lcd.clear();
    lcd.setCursor(0,0);
     for(ii=0;ii<=6;ii++) lcd.write(finallat[ii]);

     lcd.setCursor(0,1);
     for(ii=0;ii<=7;ii++) lcd.write(finallong[ii]);

delay(1000);
//gsminit();

  Serial.write("AT\r\n");  delay(2500);
  Serial.write("ATE0\r\n");                 okcheck();
  Serial.write("AT+CWMODE=3\r\n"); delay(2500);//          okcheck();
  Serial.write("AT+CIPMUX=1\r\n"); delay(2500);//     okcheck();
  Serial.write("AT+CIPSERVER=1,23\r\n"); delay(2500);//  okcheck();
  do{
     rcv = Serial.read();
    }while(rcv != 'C');
  lcd.clear();lcd.print(" Connected ");

  delay(1500);

   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("U:");//2-3-4,0
   lcd.setCursor(6,0);
   lcd.print("T:");//8-9-10,0
   lcd.setCursor(11,0);
   lcd.print("H:");//13-14-15,0
   lcd.setCursor(0,1);
   lcd.print("C:");//2-3-4,1
   lcd.setCursor(8,1);
   lcd.print("G:");//10,1
   //serialEvent();

   serialFlush();
}


char wifi_data[20];
int co2v=0;
int gasv=0;

int cntlmk=0;
void loop()
{

for(rtr1=0;rtr1<5;rtr1++)
   {
    dist2 = ultra_dist();
    dist1 = (dist1 + dist2);
      delay(10);
   }

 dist1 = (dist1/5);


lcd.setCursor(2,0);convertl(dist1);delay(10);

if(dist1 < 10)
  {
   sts1++;
   if(sts1 >= 4)
     { sts1=0;
      digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
      digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
     beep();

     Serial.write("AT+CIPSEND=0,18\r\n");delay(2000);
     Serial.write("U:");converts(dist1);Serial.write("-
Obstacle\r\n");delay(3000);

     Serial.write("AT+CIPSEND=0,21\r\n"); delay(2000);
     Serial.write("GPS:");
     for(ii=0;ii<=6;ii++) Serial.write(finallat[ii]);
     for(ii=0;ii<=7;ii++) Serial.write(finallong[ii]);
     Serial.write("\r\n");                delay(3000);
     }
  }
else
  {
    sts1=0;
  }
```

```
    DHT.read11(dht_apin);

      tempc = DHT.temperature;
      humc  = DHT.humidity;

    lcd.setCursor(8,0);convertl(tempc);
    lcd.setCursor(13,0);convertl(humc);

  if(tempc > 40)
    {
     digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
     digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
     beep();

    Serial.write("AT+CIPSEND=0,17\r\n");delay(2000);
    Serial.write("High
Temp:");converts(tempc);Serial.write("\r\n");delay(3000);

    Serial.write("AT+CIPSEND=0,21\r\n"); delay(2000);
    Serial.write("GPS:");
    for(ii=0;ii<=6;ii++) Serial.write(finallat[ii]);
    for(ii=0;ii<=7;ii++) Serial.write(finallong[ii]);
    Serial.write("\r\n");                delay(3000);
    }
  if(humc > 75)
    {
     digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
     digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
     beep();

    Serial.write("AT+CIPSEND=0,16\r\n");delay(2000);
    Serial.write("High
Hum:");converts(tempc);Serial.write("\r\n");delay(3000);

    Serial.write("AT+CIPSEND=0,21\r\n"); delay(2000);
    Serial.write("GPS:");
    for(ii=0;ii<=6;ii++) Serial.write(finallat[ii]);
    for(ii=0;ii<=7;ii++) Serial.write(finallong[ii]);
    Serial.write("\r\n");                delay(3000);
    }

  co2v = analogRead(A0);
  lcd.setCursor(2,1);convertl(co2v);

  if(co2v > 150)
    {
       digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
     digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
     beep();

    Serial.write("AT+CIPSEND=0,16\r\n");delay(2000);
    Serial.write("High
Co2:");converts(co2v);Serial.write("\r\n");delay(3000);

    Serial.write("AT+CIPSEND=0,21\r\n"); delay(2000);
    Serial.write("GPS:");
    for(ii=0;ii<=6;ii++) Serial.write(finallat[ii]);
    for(ii=0;ii<=7;ii++) Serial.write(finallong[ii]);
```

```
     Serial.write("\r\n");                delay(3000);
     }

   gasv = analogRead(A1);
   lcd.setCursor(10,1);convertl(gasv);
   if(gasv > 150)
     {
        digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
      digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
      beep();

     Serial.write("AT+CIPSEND=0,16\r\n");delay(2000);
     Serial.write("High
Gas:");converts(gasv);Serial.write("\r\n");delay(3000);

     Serial.write("AT+CIPSEND=0,21\r\n"); delay(2000);
     Serial.write("GPS:");
     for(ii=0;ii<=6;ii++) Serial.write(finallat[ii]);
     for(ii=0;ii<=7;ii++) Serial.write(finallong[ii]);
     Serial.write("\r\n");                delay(3000);
     }

   /*
    if(stringComplete)
      {
       if(inputString[1] == 'f')
         {
          digitalWrite(m1a, HIGH);digitalWrite(m1b, LOW);
          digitalWrite(m2a, HIGH);digitalWrite(m2b, LOW);
          lcd.setCursor(15,1);lcd.print("F");
         }
       if(inputString[1] == 'b')
         {
          digitalWrite(m1a, LOW);digitalWrite(m1b, HIGH);
          digitalWrite(m2a, LOW);digitalWrite(m2b, HIGH);
          lcd.setCursor(15,1);lcd.print("B");
         }
       if(inputString[1] == 'l')
         {
          digitalWrite(m1a, HIGH);digitalWrite(m1b, LOW);
          digitalWrite(m2a, LOW);digitalWrite(m2b, HIGH);
          lcd.setCursor(15,1);lcd.print("L");
         }
       if(inputString[1] == 'r')
         {
          digitalWrite(m1a, LOW);digitalWrite(m1b, HIGH);
          digitalWrite(m2a, HIGH);digitalWrite(m2b, LOW);
          lcd.setCursor(15,1);lcd.print("R");
         }
       if(inputString[1] == 's')
         {
          digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
          digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
          lcd.setCursor(15,1);lcd.print("S");
         }

       inputString = "";
       stringComplete = false;
      }
    */


    while(Serial.available())
         {
          char inChar = (char)Serial.read();
           if(inChar == '*')
             {sti=1;
             }
           if(sti == 1)
             {
              wifi_data[cntlmk] = inChar;
              cntlmk++;
             }
           if(inChar == '#')
             {sti=0;
              wifi_data[cntlmk-1] = '\0';
              cntlmk=0;
              gchr = wifi_data[1];

                      lcd.setCursor(15,1);lcd.write(gchr);
       if(gchr == 'f')
         {
          digitalWrite(m1a, HIGH);digitalWrite(m1b, LOW);
          digitalWrite(m2a, HIGH);digitalWrite(m2b, LOW);
          delay(1500);
         }
       if(gchr == 'b')
         {
          digitalWrite(m1a, LOW);digitalWrite(m1b, HIGH);
          digitalWrite(m2a, LOW);digitalWrite(m2b, HIGH);
          delay(1500);
         }
       if(gchr == 'l')
         {
          digitalWrite(m1a, HIGH);digitalWrite(m1b, LOW);
          digitalWrite(m2a, LOW);digitalWrite(m2b, HIGH);
          delay(1500);
         }
       if(gchr == 'r')
         {
          digitalWrite(m1a, LOW);digitalWrite(m1b, HIGH);
          digitalWrite(m2a, HIGH);digitalWrite(m2b, LOW);
          delay(1500);
         }
       if(gchr == 's')
         {
          digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
          digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
         }
               stringComplete = true;
             inputString="";
             }
       }
 }

 /*
```

```
void serialEvent()
{
   while(Serial.available())
        {
         char inChar = (char)Serial.read();
          if(inChar == '*')
            {sti=1;
            }
          if(sti == 1)
            {
                inputString += inChar;
            }
          if(inChar == '#')
            {sti=0;

      if(inputString[1] == 'f')
        {inputString="";
         digitalWrite(m1a, HIGH);digitalWrite(m1b, LOW);
         digitalWrite(m2a, HIGH);digitalWrite(m2b, LOW);
         lcd.setCursor(15,1);lcd.print("F");
        }
      if(inputString[1] == 'b')
        {inputString="";
         digitalWrite(m1a, LOW);digitalWrite(m1b, HIGH);
         digitalWrite(m2a, LOW);digitalWrite(m2b, HIGH);
         lcd.setCursor(15,1);lcd.print("B");
        }
      if(inputString[1] == 'l')
        {inputString="";
         digitalWrite(m1a, HIGH);digitalWrite(m1b, LOW);
         digitalWrite(m2a, LOW);digitalWrite(m2b, HIGH);
         lcd.setCursor(15,1);lcd.print("L");
        }
      if(inputString[1] == 'r')
        {inputString="";
         digitalWrite(m1a, LOW);digitalWrite(m1b, HIGH);
         digitalWrite(m2a, HIGH);digitalWrite(m2b, LOW);
         lcd.setCursor(15,1);lcd.print("R");
        }
      if(inputString[1] == 's')
        {inputString="";
         digitalWrite(m1a, LOW);digitalWrite(m1b, LOW);
         digitalWrite(m2a, LOW);digitalWrite(m2b, LOW);
         lcd.setCursor(15,1);lcd.print("S");
        }
              stringComplete = true;
              inputString="";
            }
        }
}
*/




/*
void serialEvent()
{
  while (Serial.available())
        {

         char inChar = (char)Serial.read();
           if(inChar == '*')
             {
              gchr = Serial.read();
             }
           if(inChar == '#')
             {
              gchr1 = Serial.read();
             }
        }
}*/


int readSerial(char result[])
{
  int i = 0;
  while (1)
  {
    while (Serial.available() > 0)
    {
      char inChar = Serial.read();
      if (inChar == '\n')
         {
          result[i] = '\0';
          Serial.flush();
          return 0;
         }
      if (inChar != '\r')
         {
          result[i] = inChar;
          i++;
         }
    }
  }
}


int readSerial1(char result[])
{
  int i = 0;
  while (1)
  {
    while (Serial.available() > 0)
    {
      char inChar = Serial.read();
      if (inChar == '*')
         {
          result[i] = '\0';
          Serial.flush();
          return 0;
         }
      if (inChar != '*')
         {
          result[i] = inChar;
          i++;
         }
     }
   }
 }




void gpsEvent()
{
  gpsString="";
  while(1)
  {
   //while (gps.available()>0)            //Serial incoming data from
GPS

    while (mySerial.available() > 0)
   {
    //char inChar = (char)gps.read();
    char inChar = (char)mySerial.read();
     gpsString+= inChar;                    //store incoming data from
GPS to temparary string str[]
     i++;
    // Serial.print(inChar);
     if (i < 7)
     {
      if(gpsString[i-1] != test[i-1])         //check for right string
      {
        i=0;
        gpsString="";
      }
     }
    if(inChar=='\r')
    {
     if(i>60)
     {
       gps_status=1;
       break;
     }
     else
     {
       i=0;
     }
    }
  }
   if(gps_status)
    break;
  }
}

void get_gps()
{

  lcd.clear();
  lcd.print("Getting GPS Data");
  lcd.setCursor(0,1);
  lcd.print("Please Wait.....");
```

```
   gps_status=0;
   int x=0;
   while(gps_status==0)
   {
    gpsEvent();
    int str_lenth=i;
    coordinate2dec();
    i=0;x=0;
    str_lenth=0;
   }
}

void coordinate2dec()
{
  String lat_degree="";
    for(i=19;i<=20;i++)
      lat_degree+=gpsString[i];

  String lat_minut="";
     for(i=21;i<=22;i++)
      lat_minut+=gpsString[i];
     for(i=24;i<=25;i++)
      lat_minut+=gpsString[i];



  String log_degree="";
    for(i=32;i<=34;i++)
      log_degree+=gpsString[i];
  String log_minut="";
    for(i=35;i<=36;i++)
      log_minut+=gpsString[i];
    for(i=38;i<=39;i++)
      log_minut+=gpsString[i];


    Speed="";
    for(i=45;i<48;i++)          //extract longitude from string
      Speed+=gpsString[i];

     float minut= lat_minut.toFloat();
     minut=minut/60;
     float degree=lat_degree.toFloat();
     latitude=degree+minut;

     minut= log_minut.toFloat();
     minut=minut/60;
     degree=log_degree.toFloat();
     logitude=degree+minut;
}

void gps_convert()
{
  if(gps_status)
  {
 // Serial.println(gpsString);
```

```
  if(gpsString[0] == '$' && gpsString[1] == 'G' && gpsString[2] == 'P'
&& gpsString[3] == 'R' && gpsString[4] == 'M' && gpsString[5] == 'C')
    {
     //
Serial.println("Don1111111111111111111111111111111111111111111111
11\r\n");
    //
Serial.write(gpsString[18]);Serial.write(gpsString[19]);Serial.write(g
psString[20]);Serial.write(gpsString[21]);Serial.write(gpsString[22]);
     //lcd.setCursor(0,0);
     for(ii=0;ii<9;ii++)
       {
        //lcd.write(gpsString[19+ii]);
        msg1[ii] = gpsString[19+ii];
        //Serial.write(msg1[ii]);
       }
       //Serial.println("\r\n");
     //lcd.setCursor(0,1);
      for(ii=0;ii<10;ii++)
       {
        //lcd.write(gpsString[32+ii]);
        msg2[ii] = gpsString[32+ii];
       // Serial.write(msg2[ii]);
       }

// Serial.println(msg1);
// Serial.println(msg2);

       //lati = (((msg1[2]-48)*100000) +((msg1[3]-48)*10000) +
((msg1[5]-48)*1000) + ((msg1[6]-48)*100) + ((msg1[7]-48)*10) +
(msg1[8]-48));
       //longi = (((msg2[3]-48)*100000) + ((msg2[4]-48)*10000) +
((msg2[6]-48)*1000) + ((msg2[7]-48)*100) + ((msg2[8]-48)*10) +
(msg2[9]-48));

       lati = (((msg1[2]-48)*1000) + ((msg1[3]-48)*100) + ((msg1[5]-
48)*10) + (msg1[6]-48));
       longi = (((msg2[3]-48)*1000) + ((msg2[4]-48)*100) + ((msg2[6]-
48)*10) + (msg2[7]-48));

      // converts(lati);Serial.write("-");
      // converts(longi);Serial.write("\r\n");

       lati = (lati/60);  longi = (longi/60);

       lati = (lati*100); longi = (longi*100);
       lati1 = lati;      longi1 = longi;

// Serial.write("After ");
  //      converts(lati1);Serial.write("-");
    //   converts(longi1);Serial.write("\r\n");


              convlat(lati); convlong(longi);
        finallat[0] = msg1[0];
        finallat[1] = msg1[1];
        finallat[2] = '.';
        finallat[3] = flat[0]; finallat[4] = flat[1];finallat[5] =
flat[2];finallat[6] = flat[3];finallat[7] = '\0';


        finallong[0] = msg2[0];
        finallong[1] = msg2[1];
        finallong[2] = msg2[2];
        finallong[3] = '.';
        finallong[4] = flong[0];finallong[5] = flong[1];finallong[6] =
flong[2];finallong[7] = flong[3];finallong[8] = '\0';


    }
  }
}

 void convlat(unsigned int value)
      {
            unsigned int a,b,c,d,e,f,g,h;

      a=value/10000;
      b=value%10000;
      c=b/1000;
      d=b%1000;
      e=d/100;
      f=d%100;
      g=f/10;
      h=f%10;


      a=a|0x30;
      c=c|0x30;
      e=e|0x30;
    g=g|0x30;
      h=h|0x30;

  // dlcd(a);
//   dlcd(c);dlcd(e); dlcd(g);dlcd(h);//lcddata('A');//lcddata('
');lcddata(' ');


              flat[0] = c;
              flat[1] = e;
              flat[2] = g;
            flat[3] = h;


          }

 void convlong(unsigned int value)
      {
           unsigned int a,b,c,d,e,f,g,h;

      a=value/10000;
      b=value%10000;
      c=b/1000;
      d=b%1000;
      e=d/100;
      f=d%100;
      g=f/10;
      h=f%10;


      a=a|0x30;
      c=c|0x30;
      e=e|0x30;
    g=g|0x30;
      h=h|0x30;

  // dlcd(a);
//   dlcd(c);dlcd(e); dlcd(g);dlcd(h);//lcddata('A');//lcddata('
');lcddata(' ');


               flong[0] = c;
               flong[1] = e;
               flong[2] = g;
             flong[3] = h;


         }


/*
void coordinate2dec()
{
  String lat_degree="";
    for(i=20;i<=21;i++)
      lat_degree+=gpsString[i];

  String lat_minut="";
     for(i=22;i<=28;i++)
      lat_minut+=gpsString[i];
  String log_degree="";
    for(i=32;i<=34;i++)
      log_degree+=gpsString[i];
  String log_minut="";
    for(i=35;i<=41;i++)
      log_minut+=gpsString[i];

    Speed="";
    for(i=45;i<48;i++)          //extract longitude from string
      Speed+=gpsString[i];

     float minut= lat_minut.toFloat();
     minut=minut/60;
     float degree=lat_degree.toFloat();
     latitude=degree+minut;

     minut= log_minut.toFloat();
     minut=minut/60;
     degree=log_degree.toFloat();
     logitude=degree+minut;
}*/

void gsminit()
```

```
{
  Serial.write("AT\r\n");                    okcheck();
  Serial.write("ATE0\r\n");                  okcheck();
  Serial.write("AT+CMGF=1\r\n");             okcheck();
  Serial.write("AT+CNMI=1,2,0,0\r\n");       okcheck();
  Serial.write("AT+CSMP=17,167,0,0\r\n");    okcheck();

  lcd.clear();
  lcd.print("SEND MSG STORE");
  lcd.setCursor(0,1);
  lcd.print("MOBILE NUMBER");
  do{
     rcv = Serial.read();
    }while(rcv != '*');
     readSerial(pastnumber);

  lcd.clear();
  lcd.print(pastnumber);

    Serial.write("AT+CMGS=\"");
    Serial.write(pastnumber);
    Serial.write("\"\r\n"); delay(3000);
    Serial.write("Mobile no. registered\r\n");
    Serial.write(0x1A);
    //pastnumber[10]='\0';
    delay(4000);

  //delay(1000);
}


/*
int gpsgain(char result[])
{
  int i = 0;
  char rcvv;

  while (1)
  {
    while (Serial.available() > 0)
    {
      lp:
      char inChar = Serial.read();
      result[i] = inChar;
      if(result[0] == '$')
        {
          i++;
       //   result[i] = inChar;
        }
      if(result[0] != '$')
        {
          i=0;
        }
      if(i == 5)
        {
         if(result[0] == '$' && result[1] == 'G' && result[2] == 'P'
&& result[3] == 'R' && result[4] == 'M' && result[5] == 'C')
           {
             goto lp;
           }
         else
           {
            i=0;
           }
        }
      if(i == 46)
        {
            result[47] = '\0';
            Serial.flush();


lt[0]=result[21];lt[1]=result[22];lt[2]=result[23];lt[3]=result[24];lt
[4]=result[25];lt[5]=result[26];

lt[6]=result[27];lt[7]=result[28];lt[8]=result[29];lt[9]=result[30];lt
[10]=result[31];lt[11]='\0';


ln[0]=result[33];ln[1]=result[34];ln[2]=result[35];ln[3]=result[36];ln
[4]=result[37];ln[5]=result[38];

ln[6]=result[39];ln[7]=result[40];ln[8]=result[41];ln[9]=result[42];ln
[10]=result[43];ln[11]=result[44];ln[12]='\0';

            return 0;
        }
    }
  }
}
*/

/*
void keypad()
{
   char kn=0,valk=0;

   lcd.setCursor(0,1);

  while(1)
  {
    if(digitalRead(swi) == LOW)
      {delay(1000);
       while(digitalRead(swi) == LOW);
         valk++;
         if(valk >= 9)
           {
            valk=9;
           }
           lcd.setCursor(kn,1);  convertk(valk);
      }
    if(digitalRead(swd) == LOW)
      {delay(1000);
       while(digitalRead(swd) == LOW);
         valk--;
         if(valk <= 0)
           {
            valk=0;
           }
           lcd.setCursor(kn,1);  convertk(valk);
      }
    if(digitalRead(swe) == LOW)
      {delay(1000);
       while(digitalRead(swe) == LOW);

        password[kn] = (valk+48);
         kn++;

         lcd.setCursor(kn,1);
            valk=0;
         if(kn == 4)
           {kn=0;
            break;
           }

      }
  }
}
*/
void converts(unsigned int value)
{
  unsigned int a,b,c,d,e,f,g,h;

      a=value/10000;
      b=value%10000;
      c=b/1000;
      d=b%1000;
      e=d/100;
      f=d%100;
      g=f/10;
      h=f%10;


      a=a|0x30;
      c=c|0x30;
      e=e|0x30;
      g=g|0x30;
      h=h|0x30;


   Serial.write(a);
   Serial.write(c);
   Serial.write(e);
   Serial.write(g);
   Serial.write(h);
}

void convertl(unsigned int value)
{
  unsigned int a,b,c,d,e,f,g,h;

      a=value/10000;
      b=value%10000;
      c=b/1000;
      d=b%1000;
      e=d/100;
      f=d%100;
      g=f/10;
      h=f%10;


      a=a|0x30;
      c=c|0x30;
      e=e|0x30;
      g=g|0x30;
      h=h|0x30;


   //lcd.write(a);
   //lcd.write(c);
   lcd.write(e);
   lcd.write(g);
   lcd.write(h);
}

void convertk(unsigned int value)
{
  unsigned int a,b,c,d,e,f,g,h;

      a=value/10000;
      b=value%10000;
      c=b/1000;
      d=b%1000;
      e=d/100;
      f=d%100;
      g=f/10;
      h=f%10;


      a=a|0x30;
      c=c|0x30;
      e=e|0x30;
      g=g|0x30;
      h=h|0x30;


  // lcd.write(a);
  // lcd.write(c);
  // lcd.write(e);
  // lcd.write(g);
   lcd.write(h);
}
```

THANK YOU