

TRAFFIC MONITORING FOR EMERGENCY VEHICLE USING RFID

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Electronics and Communication Engineering

By

Shaik Mahammad Adil (38130202)

Toram Naga Durga Sai Srinivas (38130226)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

**Sathyabama Institute of science And Technology
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI- 600119

MARCH-2022



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119
www.sathyabama.ac.in



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of

SHAIK MAHAMMAD ADIL (38130202),

TORAM NAGA DURGA SAI SRINIVAS (38130226)

who carried out the project entitled **Traffic Monitoring for Emergency Vehicle**

Using RFID under our supervision from December 2021 to March 2022.

Ms.T Gomathi, M.Tech., Ph.D.

Internal guide

Dr.T.Ravi, M.E, Ph.D.

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We **Mr. SHAIK MAHAMMAD ADIL (38130202)** and **Mr. TORAM NAGA DURGA SAI SRINIVAS (38130226)** hereby declare that the Project Report entitled **TRAFFIC MONITORING FOR EMERGENCY VEHICLE USING RFID** done by me under the guidance **Ms. T GOMATHI M.Tech., (Ph.D)** is submitted in partial fulfilment of the requirements for the award of Bachelors of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING.**

DATE:11-03-2022

PLACE: Chennai

S MAHAMMAD ADIL (38130202)
T.N.D.S SRINIVAS (38130226)

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr .N.M.NANDHITHA, M.E., Ph.D.**, Prof & Dean, School of Electrical and Electronics Engineering and **Dr .T.RAVI, M.E., Ph.D.**, Head of the Department, Dept. of Electrical and Electronics Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would also like to express our sincere and deep sense of gratitude to our internal guide **Ms. T GOMATHI M.Tech., Ph.D.**, for their valuable guidance and suggestions during the progressive reviews for our project.

We wish to express our thanks to all Teaching and Non-teaching staff and also panel members of the Department of Electronics and communication engineering who were helpful in many ways for the completion of the project

ABSTRACT:

In developing countries like India population is significantly growing. As the population grows, the numbers of vehicles on the roads are also exponentially increasing, which results in increase in road accidents and traffic jam. Specifically, when an emergency vehicle such as Ambulance or Fire engine gets stuck in traffic jam, saving the human life becomes difficult. Under such circumstances, a promising system which can clear the traffic congestions especially in peak hours and thereby providing a safe path for emergency vehicles is very much essential. In the existing literature, less focus is given towards the problem of providing a clear path for emergency vehicles during traffic congestions. To solve these issues, a RFID-based system is proposed, which manages and regulates the traffic signals at junctions when the emergency vehicle approaches, by allowing the easy passage out of the traffic congestions. The proposed framework is modelled by means of an experimental setup using ARDUINO and LED displays which simulates a real time traffic scenario. The simulation results illustrate the better performance of the proposed framework in terms of detection as well as management of emergency vehicle by providing passage out of traffic congestions during peak hours.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1.	CHAPTER 1 : INTRODUCTION 1.1 GENERAL 1.2 EXISTING SYSTEM 1.2.1 EXISTING SYSTEM DISADVANTAGES 1.3 PROPOSED SYSTEM 1.3.1 PROPOSED SYSTEM ADVANTAGES	1 1 1 2 2
2.	CHAPTER 2: LITERATURE SURVEY	3-7
3.	CHAPTER 3: SCOPE OF THE PROJECT	8
4.	CHAPTER 4 : PROJECT DESCRIPTION 4.1 GENERAL 4.2 BLOCK DIAGRAM 4.3 MODULES NAME 4.4 MODULE DESCRIPTION 4.4.1 MEASURE THE TRAFFIC DENSITY 4.4.2 OPEN THE SIGNAL FOR EMERGENCY VEHICLE 4.4.3 UPDATE THE TRAFFIC INFORMATION 4.5 HARDWARE DESCRIPTION 4.5.1 ARDUINO 4.5.1.1 ARDUINO MEGA	9 9 9 10 10 10 11 11-14 14

	4.5.2 POWER SUPPLY 4.5.3 ULTRASONIC SENSOR 4.5.4 RFID READER AND TAG 4.5.5 LCD 4.5.6 IOT MODULE 4.6 SOFTWARE DESCRIPTION 4.6.1 EMBEDDED C 4.6.2 ARDUINO SOFTWARE IDE	33 34 43 46 52 68 80
5.	CHAPTER 5 : OUTPUT 5.1 OUTPUT	 97
6.	CHAPTER 6 : CONCLUSION 6.1 APPLICATIONS 6.2 FUTURE ENHANCEMENT 6.3 ADVANTAGE 6.4 CONCLUSION 6.5 REFERENCES	98 98 98 99-102

CHAPTER: 1 - INTRODUCTION

1.1 GENERAL

The Internet of Things (IOT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. RFID stands for Radio Frequency Identification. This rapidly-growing technology transmits information wirelessly, through the use of radio waves. RFID requires using a device known as a reader. The reader is needed to retrieve any data stored on a RFID tag. The reader device contains an antenna and a small chip to transmit information via the radio-frequency electromagnetic field. The reader is responsible for picking up identifying information like the unique serial number from package tags. The information is picked up via the antenna which emits radio signals and receives signals back from package tags. These signals are then transmitted in digital format to the appropriate computer system. An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves.

1.2 EXISTING SYSTEM

In existing technique the emergency vehicles cannot move fast in traffic congestions. Then the Traffic density on the road cannot be monitored and cleared.

1.2.1 EXISTING SYSTEM DISADVANTAGES

- Requires lot of man power to clear the traffic congestions.

- Lives in emergency need immediate support which gets delayed due to traffic.

1.3 PROPOSED SYSTEM

In this proposed technique the detection as well as management of emergency vehicle by providing passage out of traffic congestions during peak hours be implemented. The traffic on the road be monitored by use of Internet of things (IOT) and then also use of RFID tag on emergency vehicle also proposed in this technique.

1.3.1 PROPOSED SYSTEM ADVANTAGES

- Emergency vehicles can alter the signal during it approach to save time and lives.
- Traffic density can be monitored and altered accordingly

CHAPTER 2 - LITERATURE SURVEY

Title 1: A framework for traffic management in IOT networks

Authors: Mukesh Taneja

Year: 2016

Description:

Wireless networks for IOT applications support different types or classes of end devices. Each such class results in different uplink and downlink traffic behavior. It is important to identify suitable class for each end device. We first propose a generic framework for this purpose. We propose an element, called Software Controller, which learns device profile using variety of means such as information provided by the device itself, information provided by the associated IoT operator and contextual information using other sources. It can also use machine learning techniques to learn how a device might behave during certain period. Suitable resource management methods are to be associated with such classification schemes. We propose one such resource management method for 802.11ah type of networks. Next, we look at some traffic scenarios that may not be supported well by the existing device classes in some of these networks. Some IoT devices may always communicate low amount of data sporadically but some may need to communicate large amount of uplink or downlink (or bi-directional) data during certain time intervals. For example, an IoT device may need to measure (and report) certain parameters more frequently on detection of certain events, or a network server may want to set certain parameters or upgrade software at an IoT device during some time interval. It becomes important to control uplink / downlink communication opportunities and sleep interval at IoT devices in the

network effectively. We propose a new device class and dynamic switching mechanism to handle such traffic scenarios effectively. We also include a software defined controller that provides for dynamic management of these communication opportunities at IoT devices and Access Points in the network.

Title 2: A Novel Assistive On-ramp Merging Control System for
Dense Traffic Management

Authors: Weihai Chen; Zheng Zhao; Zhong Liu; Peter C. Y. Chen

Year: 2017

Description:

On-ramps are area of frequent traffic congestion. Proper traffic guidance around on-ramp merging areas exerts a positive effect on the relief of traffic congestion. The objective of this paper is to design an assistive ramp-merging control (ARMCON) system. It utilizes knowledge about professional driver behavior and the dynamical relationship among the on-ramp vehicles, to produce timely information so as to guide the on-ramp drivers when merging with the main traffic flow. Under the guidance of ARMCON, disruption of the main traffic on the express way is minimized while a certain merging rate is maintained.

Title 3: Improved Intelligent Transport System for Reliable Traffic Control Management by Adapting Internet of Things

Authors: Ramkumar Eswaraprasad; Linesh Raja

Year: 2017

Description:

As the population increases, vehicle usage has been increased considerably. Traffic becomes the most crucial factor at the time of traffic management which needs to be controlled for the improved traffic management. However traffic management would be more difficult task in case of increased vehicles usage by different number of peoples. Internet of Thing makes things easier by interconnecting vehicles with the server through the internet. IoT can monitor the vehicle periodically and track their location by sending periodic information to the server. This is focused in the proposed research framework by collecting and analyzing the traffic information so that traffic can be controlled very effectively. The significant target of this proposed framework is to carry out a novel IoT based Traffic Management (IoT-TM) that can make short term decision about the traffic management, thus the accurate and efficient traffic clearance can be achieved. In this research method, data set is gathered from the multiple traffic profiles which includes attributes such as time consumption, traffic rate, number of vehicles and so on. These data's would be learned in the training phase by using the Hybrid Artificial Neural Network with Hidden Markov Model (HANN-HMM) which can accurately learn the traffic profile information with reduced time. To perform accurate recognition of the traffic optimized feature selection is done before learning by using Hybrid Ant colony Glow worm swarm optimization approach. The complete interpretation of the aniticipated

investigational framework has been conducted on MATLAB environment from which it is proved that the proposed research method namely IoT-TM can make better decision about the traffic management than the existing research systems.

Title 4: An Automated Game Theoretic Approach for Cooperative Road Traffic Management in Disaster

Authors: Samya Muhuri; Debasree Das; Susanta Chakraborty

Year: 2017

Description:

Transportation system gets paralyzed when a place affected by any catastrophic natural disaster. Some roads get blocked and vehicle density increases in all the other remaining routes. Each vehicle stuck at different position of the road network and tries to reach its destination in minimum time. A cooperative game theory based approach is proposed to regulate road traffic and minimize the waiting time of individual vehicles in any disaster situation. Each vehicle acts as a player and tries to increase its payoff value. Payoff value is based on the different parameters of the vehicle like its arrival time, velocity, priority and traffic density. Payoff value is adjusted according to the existing vehicle density in the road segment. Traffic density of each road is minimized simultaneously which increases the flow of the vehicles in the network cooperatively. Waiting time of a vehicle is directly proportional to the number of adjacent edges of the corresponding node and its density. The method also shows that any vehicle with higher priority will cross the road in minimum time. New alternative road creation process is also proposed based on graph theory for the worst situation. The alternative path will join the affected road with a minimum utilized node in the

network to evacuate the affected traffic. Experimental results suggest that our proposed method achieve minimum average waiting time compare to the existing method. The shortest time travel path is found for each vehicle from its source to the desired destination in post disaster situation.

Title 5: A Review of IoT devices for Traffic Management System

Authors: N. B. Soni; Jaideep Saraswat.

Year: 2017

Description:

In recent times, traffic congestion has increased at a breakneck speed especially in metropolitan cities. Traffic congestion leads to increase in noise pollution, traveling time, pollution, and fuel wastage etc. There are short term and long term causes of traffic congestion. Short term causes include traffic signal failures, inefficient law enforcement, inadequate road infrastructure, accidents etc. Long term causes are attributed to economic growth of the society, changes in lifestyle of people etc. As a result, traffic management has become one of the vital areas to be looked into. It includes monitoring of traffic density, communication, rerouting of traffic to avoid further delay. Internet of things can help in smooth implementation of traffic management system. There are different methods of traffic management - video analysis, wireless sensor network, adaptive traffic control system that are closely knit around IoT devices.

CHAPTER 3 - SCOPE OF THE PROJECT

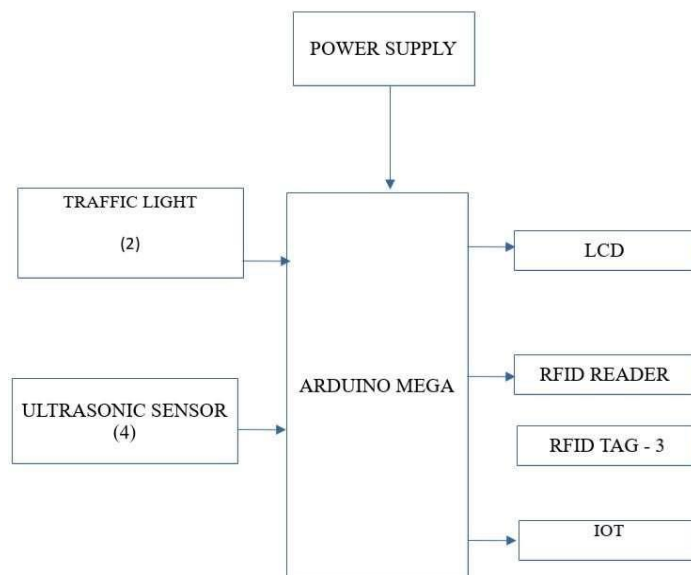
The main aim of the project is to clear the traffic jam for emergency vehicle and providing clear path for the emergency vehicles on the road, by finding density of the traffic on the road. The problem of traffic light control can be solved by RFID based system. With this system, we can consider the priority of different type of vehicles and also consider the density of traffic on the roads by installing RF reader on the road intersections. Radio frequency identification is a technique that uses the radio waves to identify the object uniquely. If at all the Ambulance encounters the traffic jam in the route, the RFID is used as a remote to control the traffic signals. The particular signal is made Green for a predetermined amount of time and after the ambulance passes by, it regains its original flow of sequence of signaling. Each individual vehicle is equipped with special Radio Frequency Identification tags, RFID reader is used to read the tags attached to the vehicle. Here we use passive tags as the life of the passive tag is very long. After the acknowledgment for the user through the mobile app, the particular signal is made Green for some time and after the ambulance passes by, it regains its original flow of sequence of signaling. If, this scheme is fully automated, it finds the ambulance spot, controls the traffic lights. This system controls the traffic lights and save the time in emergency periods. Thus, it acts as a life saver project.

CHAPTER 4 - PROJECT DESCRIPTION

4.1 GENERAL

The security in the existing voting system is not to the expected level. So in the proposed model security is enhanced by adding an IOT based authentication and followed by RFID identification and the Biometric authentication of the voting system which increases security of the model. This can be implemented to avoid lags in security.

4.2 BLOCK DIAGRAM



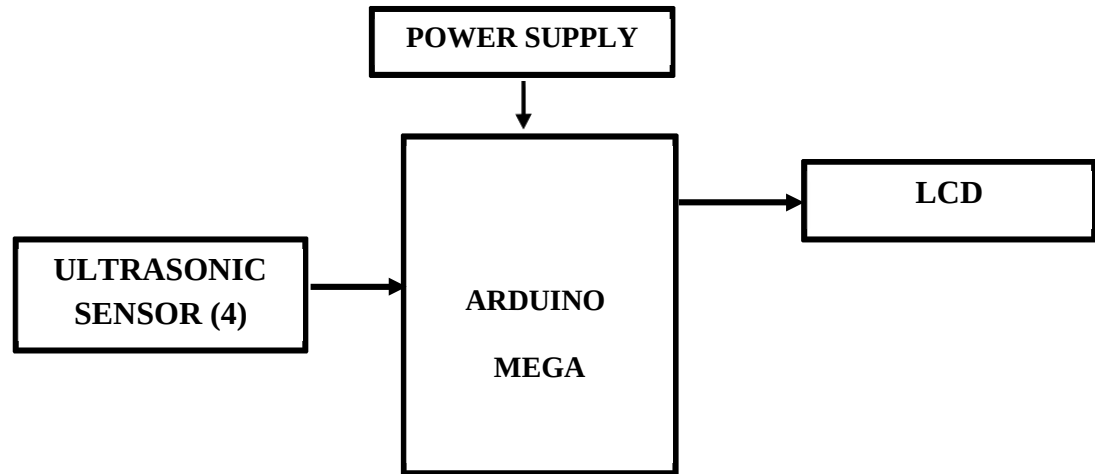
4.3 MODULES NAME:

- Measure the traffic density
- Open the signal for emergency vehicle

- Update the traffic information

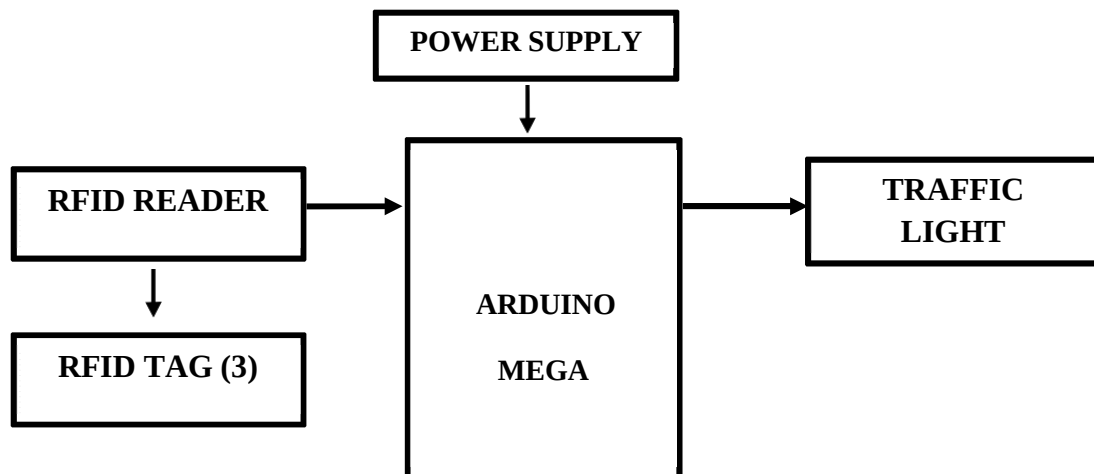
4.4 MODULE DESCRIPTION:

4.4.1 MEASURE THE TRAFFIC DENSITY



The ultrasonic sensor fixed on the road sideways. The use the four ultrasonic sensor we can measure the traffic density. Based on the four ultrasonic sensor the traffic density split into three level like low traffic, medium and high. The LCD is used to show the traffic density.

4.4.2 OPEN THE SIGNAL FOR EMERGENCY VEHICLE

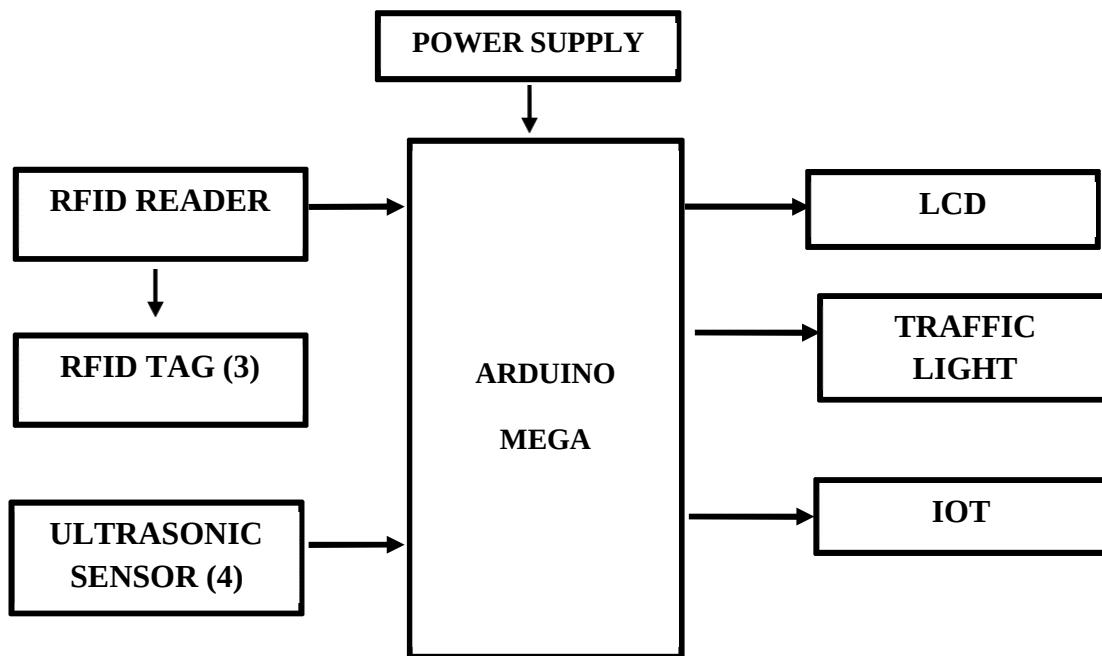


The emergency vehicle has the RFID tag and the RFID reader fixed on the road

side ways before the signal entrance. So whenever the emergency vehicle cross the reader it's automatically detect and change the traffic signal to green. These automatic signal change make an easy way for emergency vehicle to reach the destination point.

4.4.3 UPDATE THE TRAFFIC INFORMATION

If the traffic density is so high the alert will be send through the IOT. And we can monitor the signal present condition is red or green. If the road is good for the emergency vehicle or not like all the information we can get through the IOT.



4.5 HARDWARE DESCRIPTION:

4.5.1 ARDUINO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor,

a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

WHY ARDUINO?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific

instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50.
- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently

based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.

➤ **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

➤ **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

4.5.1.1 ARDUINO MEGA

The MEGA 2560 is designed for more complex projects. With 54 digital I/O pins, 16 analog inputs and a larger space for your sketch it is the recommended board for 3D printers and robotics projects. This gives your projects plenty of room and opportunities.

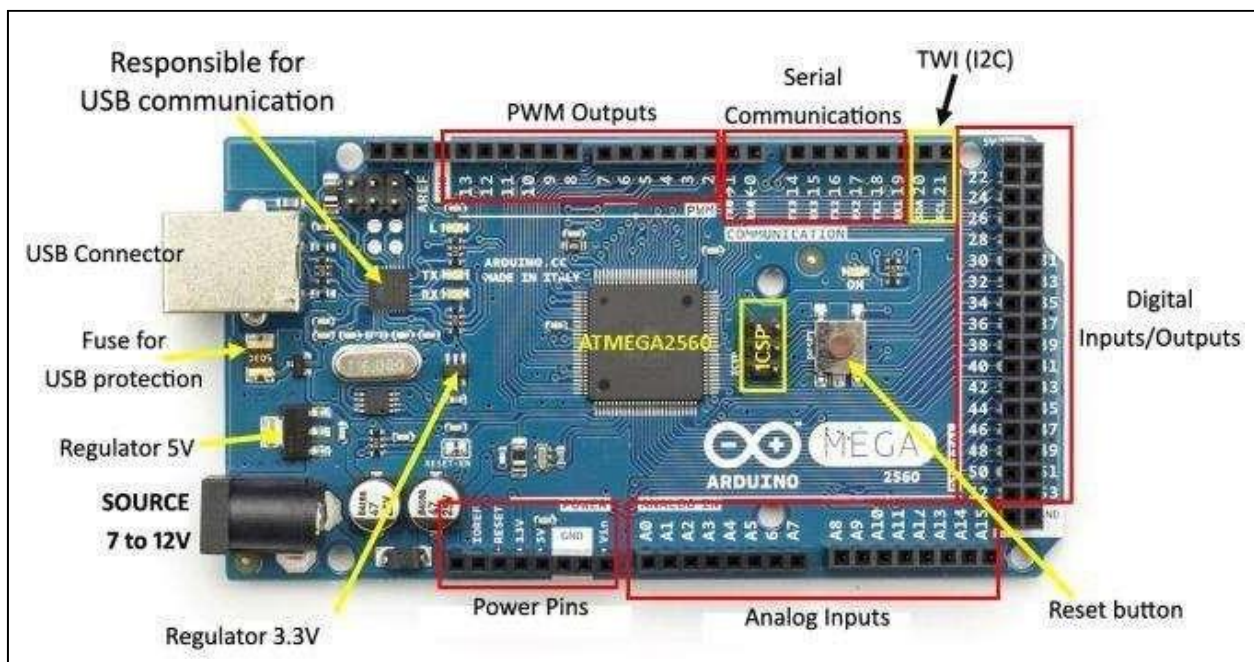


The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. 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. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

TECHNICAL SPECIFICATIONS:

Microcontroller	<u>ATmega2560</u>
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16

DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by boot loader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 m
Width	53.3 mm
Weight	37 g



HARDWARE:

Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available.

Although the hardware and software designs are freely available under copyleft licenses, the developers have requested the name Arduino to be exclusive to the official product and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product. Several Arduino-compatible products commercially released have avoided the project name by using various names ending in -duino. An early Arduino board with an RS-232 serial interface (upper left) and an Atmel ATmega8 microcontroller chip (black, lower right); the 14 digital I/O pins are at the top, the 6 analog input pins at the lower right, and the power connector at the lower left.

Most Arduino boards consist of an Atmel 8-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560) with varying amounts of flash memory, pins, and features. The 32-bit Arduino Due, based on the Atmel SAM3X8E was introduced in 2012. The boards use single or double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed shields. Multiple and possibly stacked shields may be individually addressable via an I²C serial bus. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator. Some designs, such as the LilyPad, run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions.

Arduino microcontrollers are pre-programmed with a boot loader that

simplifies uploading of programs to the on-chip flash memory. The default bootloader of the Arduino UNO is the optiboot bootloader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor-transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used. An official Arduino Uno R2 with descriptions of the I/O locations

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

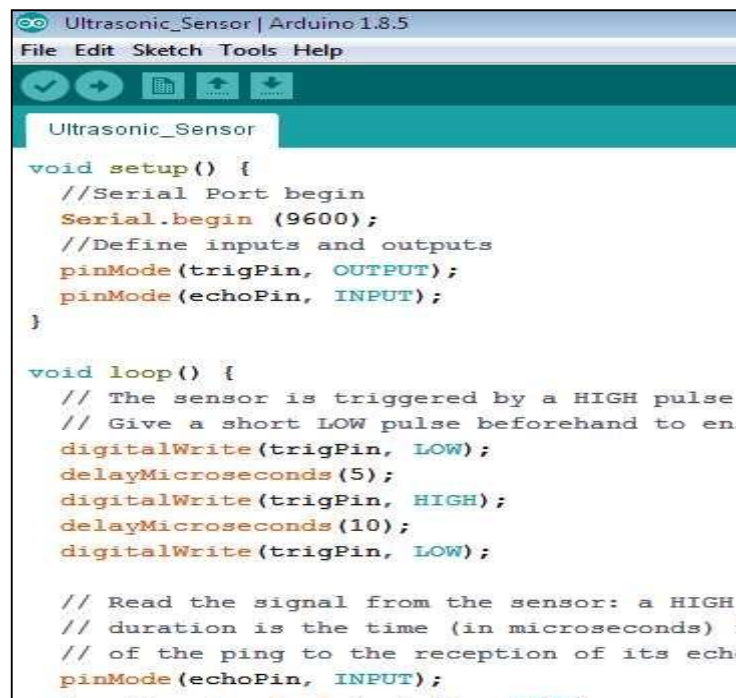
Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically

equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility. PROGRAMMING:

The Mega 2560 board can be programmed with the [Arduino Software \(IDE\)](#). For details, see the [reference](#) and [tutorials](#).

The ATmega2560 on the Mega 2560 comes preprogrammed with a [boot loader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the boot loader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using [Arduino ISP](#) or similar; see these [instructions](#) for details.

A screenshot of the Arduino IDE interface. The title bar reads "Ultrasonic_Sensor | Arduino 1.8.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening files, saving, uploading, and downloading. The main text area shows the code for a sketch named "Ultrasonic_Sensor". The code defines two functions: "void setup()" and "void loop()". The setup function initializes the serial port at 9600 baud, defines the trigPin as an output and the echoPin as an input, and sets their modes. The loop function triggers the sensor by sending a short LOW pulse followed by a HIGH pulse, then reads the echo signal from the echoPin. Comments explain the timing and purpose of each step.

```
Ultrasonic_Sensor | Arduino 1.8.5
File Edit Sketch Tools Help

Ultrasonic_Sensor

void setup() {
  //Serial Port begin
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // The sensor is triggered by a HIGH pulse
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the signal from the sensor: a HIGH pulse
  // duration is the time (in microseconds) from the sending of the pulse
  // of the ping to the reception of its echo
  pinMode(echoPin, INPUT);
  // ... (rest of the code is partially obscured)
}
```

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available in the [Arduino repository](#). The ATmega16U2/8U2 is loaded with a DFU boot loader, which can be activated by: **On Rev1 boards:** connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.

➤ **On Rev2 or later boards:** there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode. You can then use [Atmel's FLIP software](#) (Windows) or the [DFU programmer](#) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU boot loader). See [this user-contributed tutorial](#) for more information.

WARNINGS:

The Mega 2560 has a resettable poly fuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

POWER:

The Mega 2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **Vin.** The input voltage to the board when it's using an external powersource (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.** 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. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.
- **IOREF.** This pin on the 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.

MEMORY:

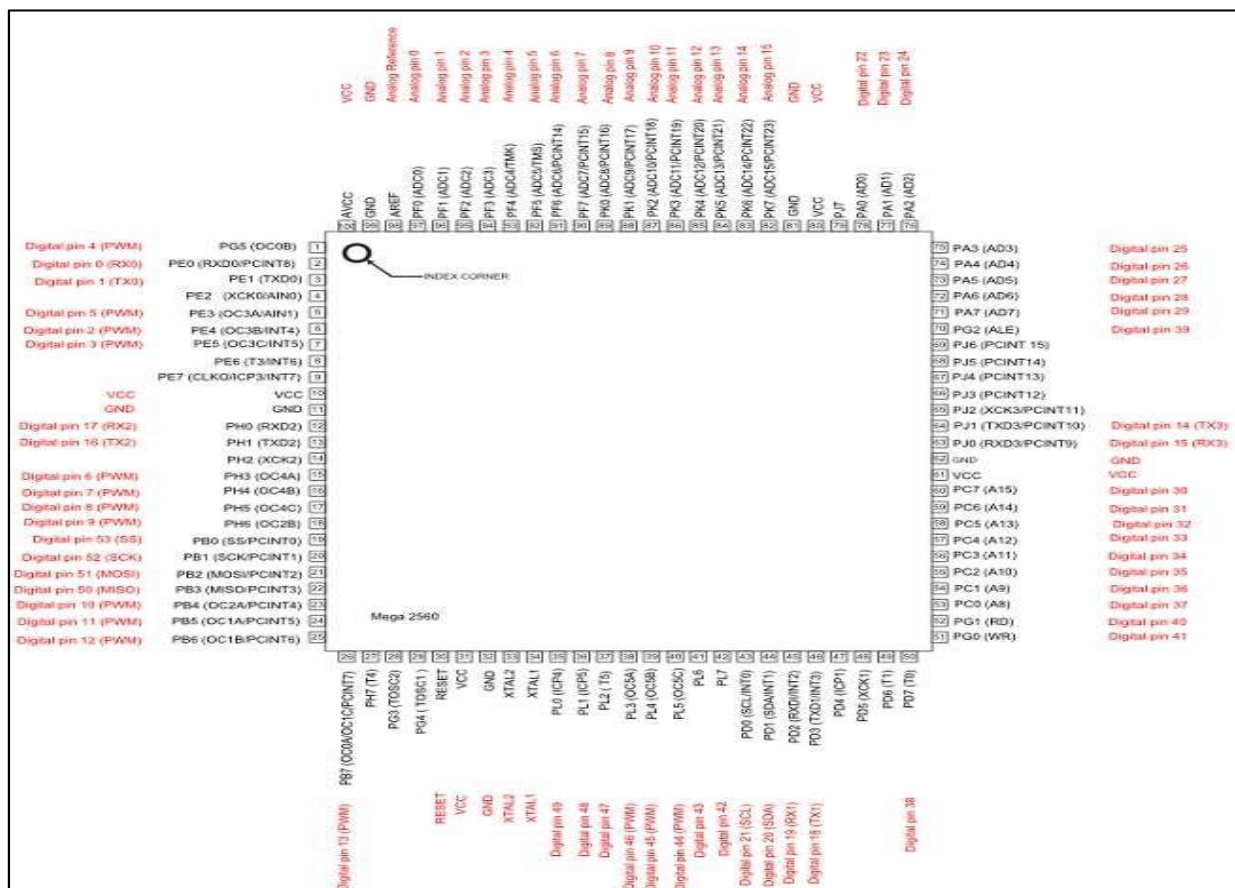
The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

INPUT AND OUTPUT:

ATmega2560-Arduino Pin Mapping.

Below is the pin mapping for the ATmega2560. The chip used in Arduino 2560. There are pin mappings to [Atmega8](#) and [Atmega 168/328](#) as well.

Arduino Mega 2560 PIN diagram



Arduino Mega 2560 PIN mapping table

Pin Number	Pin Name	Mapped Pin Name
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)

16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)
23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET
31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2

34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXDI/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41

52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
65	PJ2 (XCK3/PCINT11)	
66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	

70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15
83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10

88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TMK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog Reference
99	GND	GND
100	AVCC	VCC

Each of the 54 digital pins on the Mega can be used as an input or output, using **pinMode()**, **digitalWrite()**, and **digitalRead()** functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 k ohm. A maximum of 40mA is the value that must not be exceeded to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

➤ **Serial:** 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

➤ **External Interrupts:** 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the [attachInterrupt\(\)](#) function for details.

➤ **PWM:** 2 to 13 and 44 to 46. Provide 8-bit PWM output with the [analogWrite\(\)](#) function.

➤ **SPI:** 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the [SPI library](#). The SPI pins are also broken out on the ICSP header, which is physically compatible with the Arduino /Genuino Uno and the old Duemilanove and Diecimila Arduino boards.

➤ **LED:** 13. 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.

➤ **TWI:** 20 (SDA) and 21 (SCL). Support TWI communication using the [Wire library](#). Note that these pins are not in the same location as the TWI pins on the old Duemilanove or Diecimila Arduino boards.

See also the mapping Arduino Mega 2560 PIN diagram.

The Mega 2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). 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 and analogReference() function.

There are a couple of other pins on the board:

➤ **AREF.** Reference voltage for the analog inputs. Used with analogReference().

➤ **Reset.** Bring this line LOW to reset the microcontroller.

Typically used to add a reset button to shields which block the one on the board.

COMMUNICATION:

The Mega 2560 board has a number of facilities for communicating with a computer, another board, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2 (ATmega 8U2 on the revision 1 and revision 2 boards) on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2/ATmega16U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1.

A SoftwareSerial library allows for serial communication on any of the Mega 2560's digital pins.

The Mega 2560 also supports TWI and SPI communication. The Arduino Software (IDE) includes a Wire library to simplify use of the TWI bus; see the [documentation](#) for details. For SPI communication, use the [SPI library](#).

PHYSICAL CHARACTERISTICS AND SHIELD COMPATIBILITY:

The maximum length and width of the Mega 2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega 2560 is designed to be compatible with most shields designed for the Uno and the older Diecimila or Duemilanove Arduino boards. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Furthermore, the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega 2560 and Duemilanove / Diecimila boards. Please note that I2C is not located on the same pins on the Mega 2560 board (20 and 21) as the Duemilanove / Diecimila boards (analog inputs 4 and 5).

AUTOMATIC (SOFTWARE) RESET:

Rather than requiring a physical press of the reset button before an upload, the Mega 2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough

to reset the chip. The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega 2560 board is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the ATmega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega 2560 board contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

REVISIONS:

The Mega 2560 does not use the FTDI USB-to-serial driver chip used in past designs. Instead, it features the ATmega16U2 (ATmega8U2 in the revision 1 and revision 2 Arduino boards) programmed as a USB-to-serial converter. Revision 2 of the Mega 2560 board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the Arduino board and the current Genuino Mega 2560 have the following improved features:

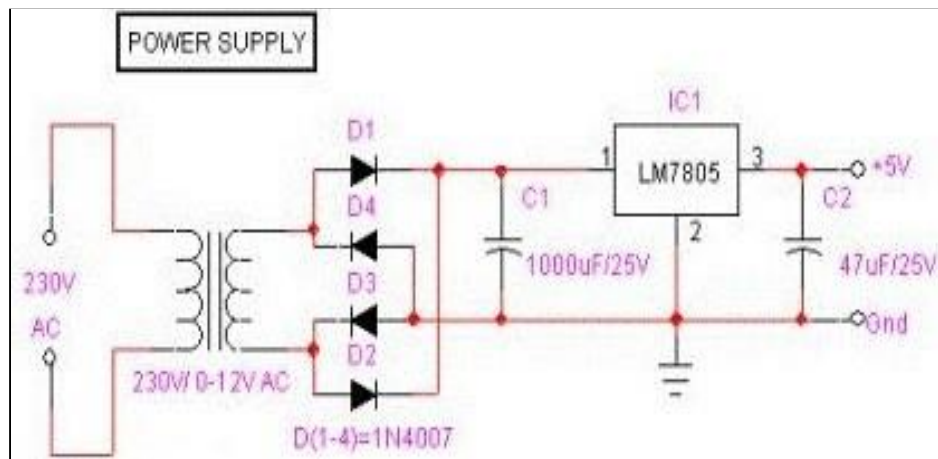
- **1.0 pinout:** SDA and SCL pins - near to the AREF pin - and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the board that uses ATSAM3X8E, that operate with 3.3V. The second one is a not connected pin that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

APPLICATIONS:

- Arduboy, a handheld game console based on Arduino.
- Arduinome, a MIDI controller device that mimics the Monome.
- Ardupilot, drone software and hardware.
- ArduSat, a cubesat based on Arduino.
- C-STEM Studio, a platform for hands-on integrated learning of computing, science, technology, engineering, and mathematics (C-STEM) with robotics.
- Data loggers for scientific research.
- OBduino, a trip computer that uses the on-board diagnostics interface found in most modern cars.

4.5.2 POWER SUPPLY

This section describes how to generate +5V DC power supply

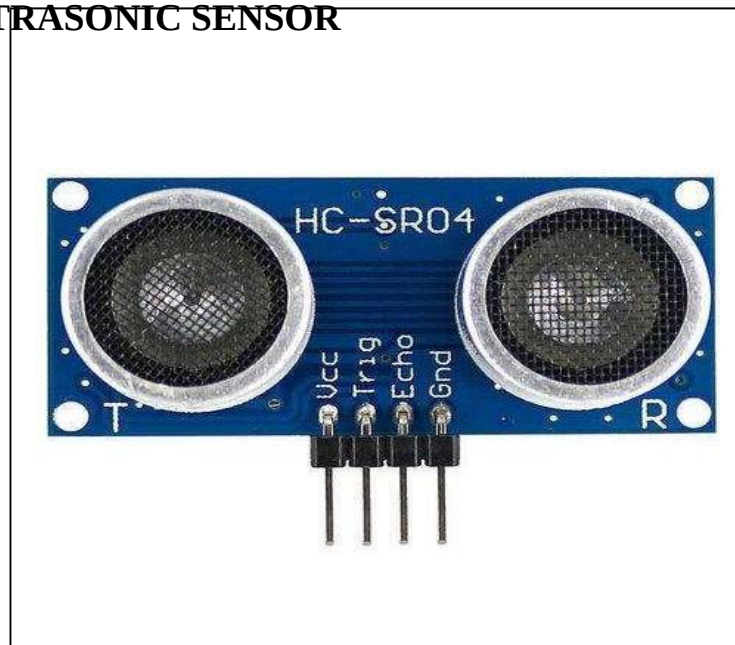


The power supply section is the important one. It should deliver constant output regulated power supply for successful working of the project. A 0-12V/1 mA transformer is used for this purpose. The primary of this transformer is connected in to main supply through on/off switch& fuse for protecting from overload and short circuit protection. The secondary is connected to the diodes to convert 12V AC to 12V DC voltage. And filtered by the capacitors, which is further regulated to +5v, by using IC 7805.

4.5.3 ULTRASONIC SENSOR

Description

The HC-SR04 sensor uses determine an object like offers excellent range detection



ultrasonic sonar to distance to bats do. It non-contact with high

accuracy and stable readings in an easy-to-use package. It comes complete with

ultrasonic transmitter and receiver modules.as the name indicates measure distance by using ultrasonic waves .the sensor head emits ultrasonic wave and receives the wave reflected back from the target .Ultrasonic measures the distance through the target by measuring time between emission and reception

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal.

(2)The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back. IF the signal back, through high level , time of high output IO duration is thetime from sending ultrasonic to returning. Test distance = (high leveltime×velocity of sound (340M/S) / 2,

HC-SR04 Sensor Features

HC-SR04 Specifications

The sensor chosen for the Firefighting Drone Project was the HCSR04. This section contains the specifications and why they are important to the sensor module. The sensor modules requirements are as follows.

- Cost .
- Weight.
- Community of hobbyists and support.

- Accuracy of object detection.
- Probability of working in a smoky environment.
- Ease of use .

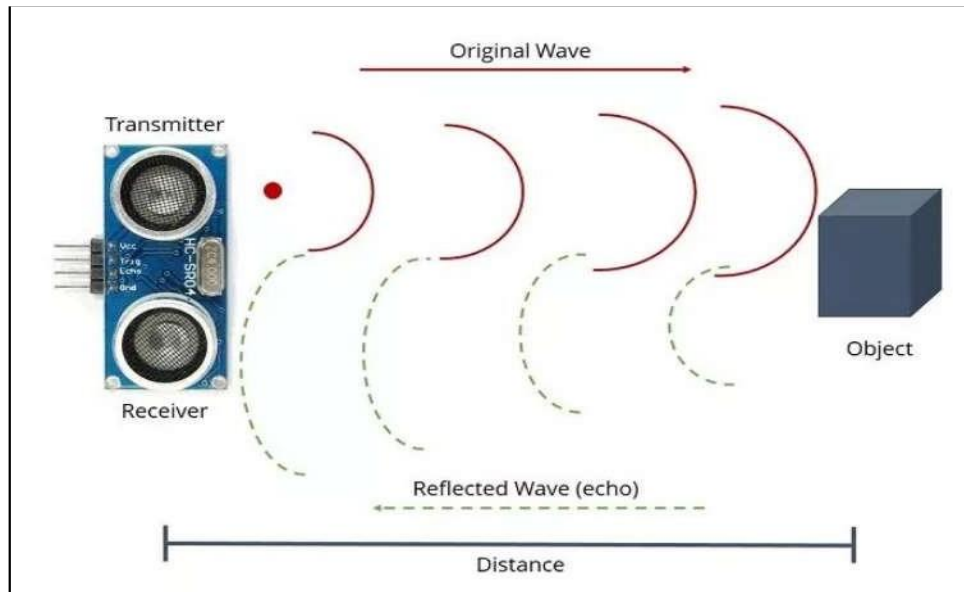
Electric Parameter

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2 cm

MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm
Main parts	Transmitter &receiver
Technology used	Non-contact technology
Resolution	3mm

Working principle of ultrasonic sensor

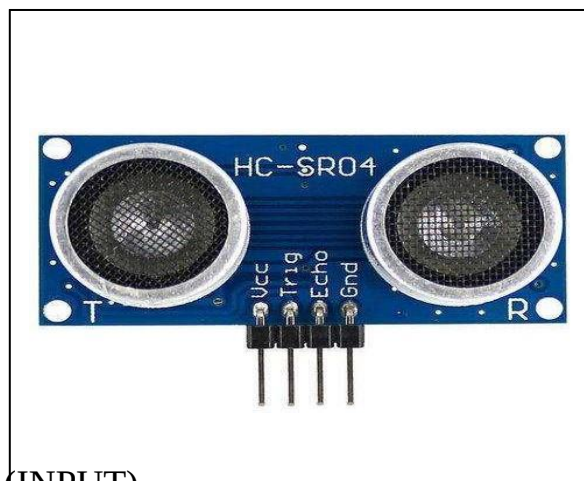
- Ultrasonic sensor transmits sound waves.
- These waves are reflected back from the surface of an object.
- An ultrasonic sensor receives the reflected waves.
- Then it measures the time elapsed during the entire process, from transmission to receiving, it is known as round trip time.
- This time is equal to the distance between an object and the sensor itself.
- I have also provided some visual, so that you can easily understand its working principle.
- Ultrasonic sensor principle is shown in the figure given below.



The time between the transmission and reception of the signal allows us to calculate the distance to an object. This is possible because we know the sound's velocity in the air.

HC-SR04 Ultrasonic Sensor Pinout

Pins

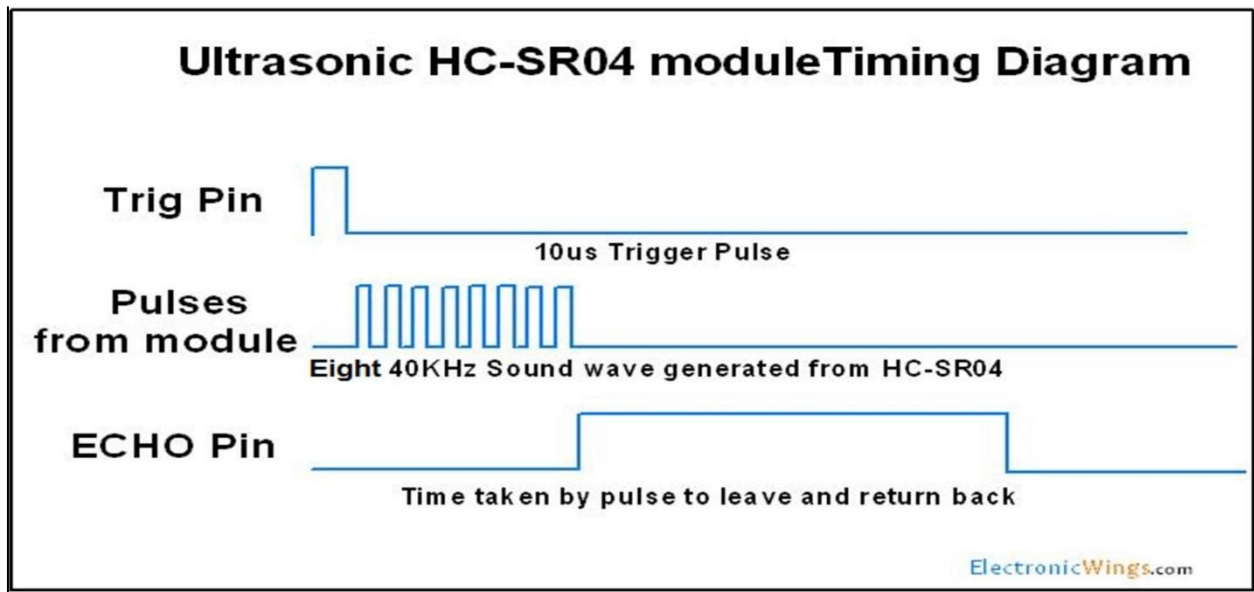


- VCC: +5VDC
- Trig : Trigger (INPUT)
- Echo: Echo (OUTPUT)
- GND: GND

Ultrasonic sensor pin configuration

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

Timing diagram The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: $\mu S / 58 =$ centimeters or $\mu S / 148 =$ inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



How to use the HC-SR04 Ultrasonic Sensor

HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used. Power the Sensor using a regulated +5V through the Vcc and Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.

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

Wiring the HCSR04 to a Microcontroller

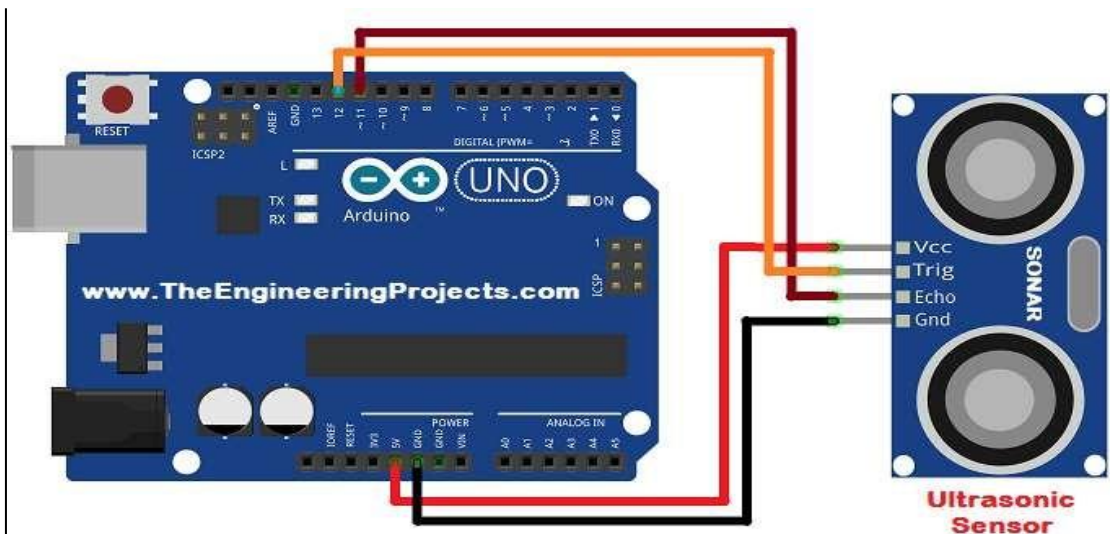
This section only covers the hardware side. For information on how to integrate the software side, look at one of the links below or look into the specific microcontroller you are using.

The HCSR04 has 4 pins: VCC, GND, TRIG and ECHO.

1. VCC is a 5v power supply. This should come from the microcontroller
2. GND is a ground pin. Attach to ground on the microcontroller.
3. TRIG should be attached to a GPIO pin that can be set to HIGH
4. ECHO is a little more difficult. The HCSR04 outputs 5v, which could destroy many microcontroller GPIO pins (the maximum allowed voltage varies). In order to step down the voltage use a single resistor or a voltage divider circuit. Once again this depends on the specific microcontroller you are using, you will need to find out its GPIO maximum voltage and make sure you are below that



Interfacing ultrasonic sensor with arduino uno



In order to get the precise distance measurement, HC-SR04 is mostly used in combination with different Arduino Modules like Arduino Uno and Arduino Mega. You can connect Arduino with this sensor in the following way. First, you need to power up the sensor using 5V DC regulated input to the sensor. Connect the ground pin with the ground of the voltage source. You can also power the sensor module using the Arduino 5V pins as the current drawn by the sensor is less than 15mA, won't be affecting the current ratings of the Arduino Module.

After setting up the initial arrangement, connect both Trig and Echo pins to the I/O pins of the Arduino Board. As mentioned earlier, in order to initialize the measurement process, the Trig pin must be kept high for 10us in the start. The sensor module will start generating sound waves with the frequency around 40,000 Hz per second from the transmitter.

- As the waves bounce back, consequently, the Echo pin will turn on until the sounds waves are received by the receiver. This time will be calculated using Arduino Module.

Applications

- Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
- Can be used to map the objects surrounding the sensor by rotating it.
- Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water
- Speed and direction measurement
- Wireless charging
- Humidifiers
- Medical ultrasonography
- Burglar alarms
- Embedded system
- Depth measurement
- Non-destructive testing

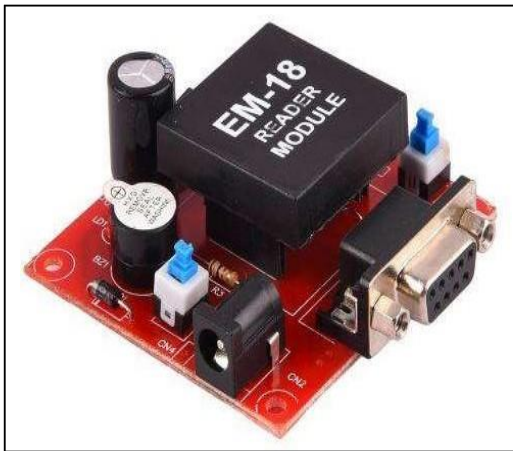
4.5.4 RFID READER AND TAG

An RFID reader is a device that is used to interrogate an RFID tag. The reader has an antenna that emits radio waves; the tag responds by sending back its data.

An RFID tag is a microchip combined with an antenna in a compact package; the packaging is structured to allow the RFID tag to be attached to an object to be tracked. "RFID" stands for Radio Frequency Identification. The tag's antenna picks up signals from an RFID reader or scanner and then returns the signal, usually with some additional data (like a unique serial number or other customized information).

A passive tag is an RFID tag that does not contain a battery; the power is supplied by the reader. When radio waves from the reader are encountered by a passive rfid tag, the coiled antenna within the tag forms a magnetic field. The tag draws power from it, energizing the circuits in the tag. The tag then sends the information encoded in the tag's memory.

The RX and TX pins of RFID reader connected to Tx and Rx pins of 8051 Microcontroller respectively. Then the reader senses the data from the Tag and transmits the sensed data to microcontroller via serial port.



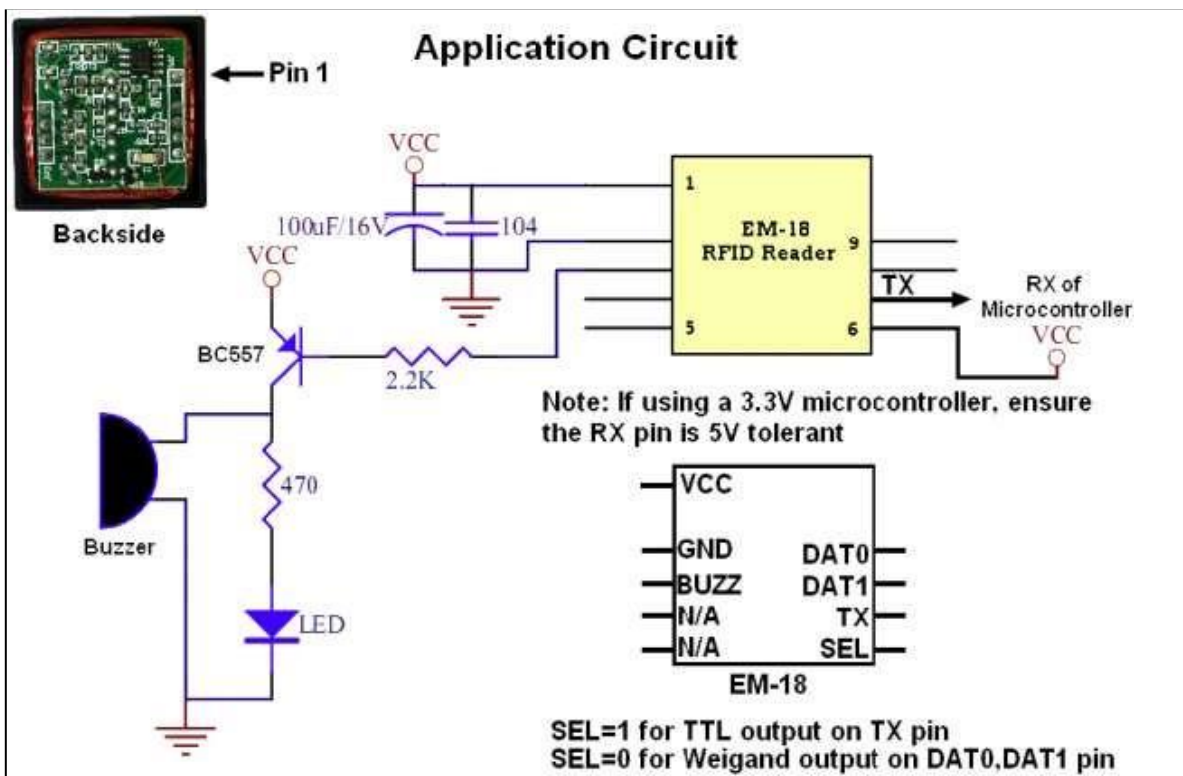
The EM-18 RFID Reader module operating at 125kHz is an inexpensive solution for your RFID based application. The Reader module comes with an on-chip antenna and can be powered up with a 5V power supply. Power-up the module and connect the transmit pin of the module to receive pin of your microcontroller. Show your card within the reading distance and the card number is thrown at the output. Optionally the module can be configured for also a weigand output.

Typical Applications

- Υ E-Payment
- Υ E-Toll Road Pricing
- Υ E-Ticketing for Events
- Υ E-Ticketing for Public Transport
- Υ Access Control
- Υ PC Access
- Υ Authentication
- Υ Printer / Production Equipment

Features

RF Transmit Frequency	125kHz
Supported Standards	EM4001 64-bit RFID tag compatible
Communications Interface	TTL Serial Interface, Wiegand output
Communications Protocol	Specific ASCII
Communications Parameter	9600 bps, 8, N, 1
Power Supply	4.6V - 5.5VDC $\pm$ 10% regulated
Current Consumption	50 mA < 10mA at power down mode.
Reading distance	Up to 100mm, depending on tag
Antenna	Integrated
Size (LxWxH)	32 x 32 x 8mm



4.5.5 LCD

4.5.6 LCD screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

We come across LCD displays everywhere around us. Computers, calculators, television sets, mobile phones, digital watches use some kind of display to display the time. An LCD is an electronic display module which uses liquid crystal to produce a visible image. The 16x2 LCD display is a very basic module commonly used in projects. The 16x2 translates to a display 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5x7 pixel

matrix.

16X2 LCD Pin Functions:

PIN NO.	FUNCTION	NAME
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	VCC
3	Contrast adjustment; the best way is to use a variable resistor such as a potentiometer. The output of the potentiometer is connected to this pin. Rotate the potentiometer knob forward and backwards to adjust the LCD contrast.	Vo / VEE
4	Selects command register when low, and data register when high	RS (Register Select)
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given; Extra voltage push is required to execute the instruction and EN(enable) signal is used for this purpose. Usually, we make it en=0 and when we want to execute the instruction we make it high en=1 for some milliseconds. After this we again make it ground that is, en=0.	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7

15	Backlight VCC (5V)	Led+
----	--------------------	------

16	Backlight Ground (0V)	Led-
----	-----------------------	------

RS (REGISTER SELECT):

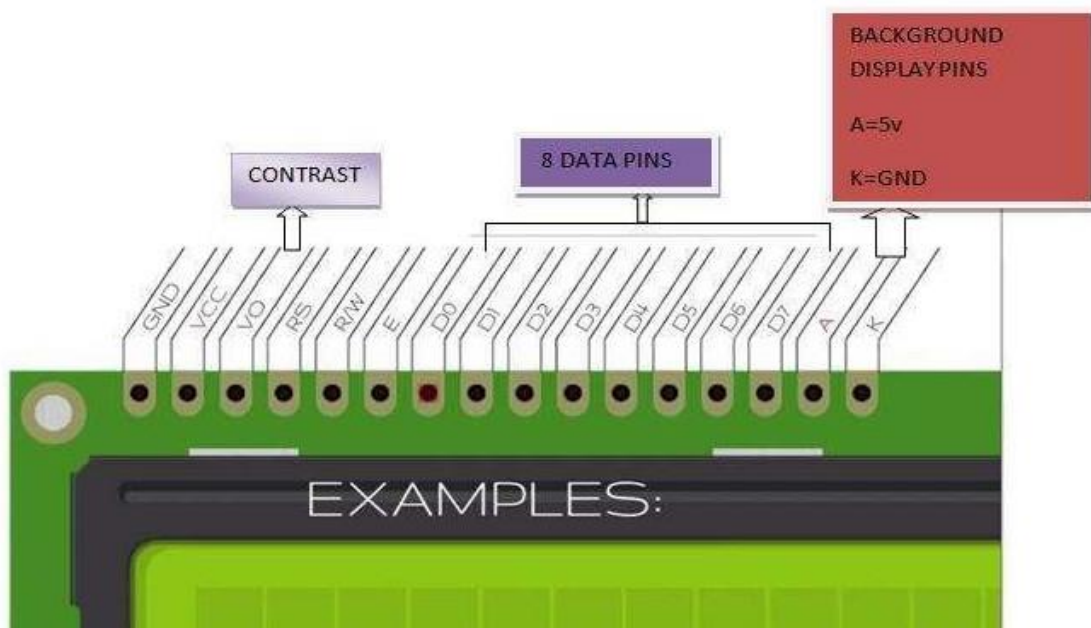
A 16X2 LCD has two registers, namely, command and data. The register select is used to switch from one register to other. RS=0 for command register, whereas RS=1 for data register.

COMMAND REGISTER:

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. Processing for commands happens in the command register.

DATA REGISTER:

The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. When we send data to LCD it goes to the data register and is processed there. When RS=1, data register is selected.



IMPORTANT COMMAND CODES FOR LCD:

SR.NO.	HEX CODE	COMMAND TO LCD INSTRUCTION REGISTER
1	01	Clear display screen
2	02	Return home
3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left

13	14	Shift cursor position to right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)
18	38	2 lines and 5×7 matrix

DISPLAYING CUSTOM CHARACTERS ON 16X2 LCD:

Generating custom characters on LCD is not very hard. It requires the knowledge about custom generated random access memory (CG-RAM) of LCD and the LCD chip controller. Most LCDs contain Hitachi HD4478 controller. CG-RAM is the main component in making custom characters. It stores the custom characters once declared in the code. CG-RAM size is 64 byte providing the option of creating eight characters at a time. Each character is eight byte in size.

CG-RAM address starts from 0x40 (Hexadecimal) or 64 in decimal. We can generate custom characters at these addresses. Once we generate our characters at these addresses, now we can print them on the LCD at any time by just sending simple commands to the LCD. Character addresses and printing commands are below.

In the table above you can see starting addresses for each character with their printing commands. The first character is generated at address 0x40 to 0x47 and is printed on LCD by just sending simple command 0 to the LCD. The second character is generated at address 0x48 to 0x55 and is printed by sending 1 to LCD.

CG-RAM Characters	CG-RAM Address (Hexadecimal)	Commands to display Generated Characters
1 st Character	0x40	0
2 nd Character	0x48	1
3 rd Character	0x56	2
4 th Character	0x64	3
5 th Character	0x72	4
6 th Character	0x80	5
7 th Character	0x88	6
8 th Character	0x96	7

4.5.7 INTERNET OF THINGS (IOT)

The **Internet Of Things (IoT)** is the network of physical devices, vehicles, buildings and other items embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. In 2013 the Global Standards Initiative on Internet of Things (IoT-GSI) defined the IoT as "the infrastructure of the information society. The IoT allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit. When IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids, smart homes, intelligent transportation and smart cities. Each thing is uniquely identifiable through its

embedded computing system but is able to interoperate within the existing Internet infrastructure. Experts estimate that the IoT will consist of almost 50 billion objects by 2020.

INFRASTRUCTURE:

The Internet of Things will become part of the fabric of everyday life. It will become part of our overall infrastructure just like water, electricity, telephone, TV and most recently the Internet. Whereas the current Internet typically connects full-scale computers, the Internet of Things (as part of the Future Internet) will connect everyday objects with a strong integration into the physical world.

1. Plug and Play Integration

If we look at IoT-related technology available today, there is a huge heterogeneity. It is typically deployed for very specific purposes and the configuration requires significant technical knowledge and may be cumbersome. To achieve a true Internet of Things we need to move away from such small-scale, vertical application silos, towards a horizontal infrastructure on which a variety of applications can run simultaneously.

2. Infrastructure Functionality

The infrastructure needs to support applications in finding the things required. An application may run anywhere, including on the things themselves. Finding things is not limited to the start-up time of an application. Automatic adaptation is needed whenever relevant new things become available, things become unavailable or the status of things changes. The infrastructure has to support the monitoring of such changes and the adaptation that is required as a result of the changes.

3. Physical Location and Position

As the Internet of Things is strongly rooted in the physical world, the notion of physical location and position are very important, especially for finding things, but also for deriving knowledge. Therefore, the infrastructure has to support finding things according to location (e.g. geo-location based discovery). Taking mobility into account, localization technologies will play an important role for the Internet of Things and may become embedded into the infrastructure of the Internet of Things.

4. Security and Privacy

In addition, an infrastructure needs to provide support for security and privacy functions including identification, confidentiality, integrity, non-repudiation authentication and authorization. Here the heterogeneity and the need for interoperability among different ICT systems deployed in the infrastructure and the resource limitations of IoT devices (e.g., Nano sensors) have to be taken into account.

Data Management

Data management is a crucial aspect in the Internet of Things. When considering a world of objects interconnected and constantly exchanging all types of information, the volume of the generated data and the processes involved in the handling of those data become critical. A long-term opportunity for wireless communications chip makers is the rise of Machine-to-Machine (M2M) computing, which one of the enabling technologies for Internet of Things. This technology spans a broad range of applications. While there is consensus that M2M is a promising pocket of growth, analyst estimates on the size of the opportunity diverge by a factor of four [16]. Conservative estimates assume roughly 80 million

to 90 million M2M units will be sold in 2014, whereas more optimistic projections forecast sales of 300 million units. Based on historical analyses of adoption curves for similar disruptive technologies, such as portable MP3 players and antilock braking systems for cars, it is believed that unit sales in M2M could rise by as much as a factor of ten over the next five years, see Figure 2.29 [16]. There are many technologies and factors involved in the “data management” within the IoT context. Some of the most relevant concepts which enable us to understand the challenges and opportunities of data management are:

- Data Collection and Analysis
- Big Data
- Semantic Sensor Networking
- Virtual Sensors
- Complex Event Processing.

APPLICATION AREAS:

In the last few years the evolution of markets and applications, and therefore their economic potential and their impact in addressing societal trends and challenges for the next decades has changed dramatically. Societal trends are grouped as: health and wellness, transport and mobility, security and safety, energy and environment, communication and e-society, as presented in Figure 2.15. These trends create significant opportunities in the markets of consumer electronics, automotive electronics, medical applications, communication, etc. The applications in these areas benefit directly by the More-Moore and More-than-Moore semiconductor technologies, communications, networks, and software developments.

a) Cities

Smart Parking: Monitoring of parking spaces availability in the city.

Structural health: Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.

Noise Urban Maps: Sound monitoring in bar areas and centric zones in real time.

Traffic Congestion: Monitoring of vehicles and pedestrian levels to optimize driving and walking routes.

Smart Lightning: Intelligent and weather adaptive lighting in street lights.

Waste Management: Detection of rubbish levels in containers to optimize the trash collection routes.

Intelligent Transportation Systems: Smart Roads and Intelligent Highways with warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.

b) Environment

Forest Fire Detection: Monitoring of combustion gases and preemptive fire conditions to define alert zones.

Air Pollution: Control of CO₂ emissions of factories, pollution emitted by cars and toxic gases generated in farms.

Landslide and Avalanche Prevention: Monitoring of soil moisture, vibrations and earth density to detect dangerous patterns in land conditions.

Earthquake Early Detection: Distributed control in specific places of tremors.

c) Water

Water Quality: Study of water suitability in rivers and the sea for fauna and Eligibility for drinkable use.

Water Leakages: Detection of liquid presence outside tanks and pressure variations along pipes.

River Floods: Monitoring of water level variations in rivers, dams and reservoirs.

d) Energy Smart Grid, Smart Metering

Smart Grid: Energy consumption monitoring and management.

Tank level: Monitoring of water, oil and gas levels in storage tanks and cisterns.

Photovoltaic Installations: Monitoring and optimization of performance in solar energy plants.

Water Flow: Measurement of water pressure in water transportation systems.

Silos Stock Calculation: Measurement of emptiness level and weight of the goods.

e) Security & Emergencies

Perimeter Access Control: Access control to restricted areas and detection of people in non-authorized areas.

Liquid Presence: Liquid detection in data centres, warehouses and sensitive building grounds to prevent break downs and corrosion.

Radiation Levels: Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.

Explosive and Hazardous Gases: Detection of gas levels and leakages in industrial environments, surroundings of chemical factories and inside mines.

f) Industrial Control

M2M Applications: Machine auto-diagnosis and assets control.

Indoor Air Quality: Monitoring of toxic gas and oxygen levels inside chemical plants to ensure workers and goods safety.

Temperature Monitoring: Control of temperature inside industrial and medical fridges with sensitive merchandise.

Ozone Presence: Monitoring of ozone levels during the drying meat process in food factories.

Indoor Location: Asset indoor location by using active (ZigBee, UWB) and passive tags (RFID/NFC).

Vehicle Auto-diagnosis: Information collection from CAN Bus to send real time alarms to emergencies or provide advice to drivers.

g) Agriculture

Wine Quality Enhancing: Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.

Green Houses: Control micro-climate conditions to maximize the production of fruits and vegetables and its quality.

Golf Courses: Selective irrigation in dry zones to reduce the water resources required in the green.

Meteorological Station Network: Study of weather conditions in fields to forecast ice formation, rain, drought, snow or wind changes.

Compost: Control of humidity and temperature levels in alfalfa, hay, straw, etc. to prevent fungus and other microbial contaminants.

h) Domestic & Home Automation

Energy and Water Use: Energy and water supply consumption monitoring to obtain advice on how to save cost and resources.

Remote Control Appliances: Switching on and off remotely appliances to avoid accidents and save energy.

Intrusion Detection Systems: Detection of window and door openings and violations to prevent intruders.

Art and Goods Preservation: Monitoring of conditions inside museums and art warehouses.

i) E-Health

Fall Detection: Assistance for elderly or disabled people living independent.

Medical Fridges: Control of conditions inside freezers storing vaccines, medicines and organic elements.

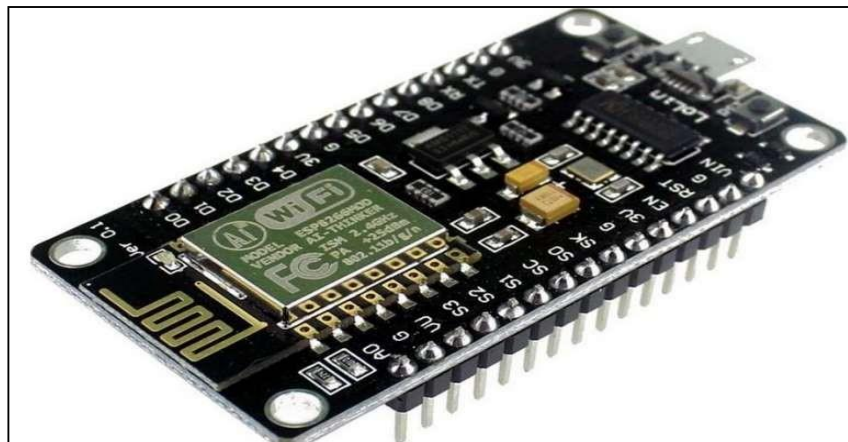
Sportsmen Care: Vital signs monitoring in high performance centres and fields.

Patients Surveillance: Monitoring of conditions of patients inside hospitals and in old people's home.

Ultraviolet Radiation: Measurement of UV sun rays to warn people not to be exposed in certain hours.

ESP-12E BASED NODEMCU

The ESP8266 is the name of a micro controller designed by Espressif Systems. The ESP8266 itself is a self-contained Wi-Fi networking solution offering as a bridge from existing micro controller to Wi-Fi and is also capable of running self-contained applications. This module comes with a built in USB connector and a rich assortment of pin-outs. With a micro USB cable, you can connect NodeMCU devkit to your laptop and flash it without any trouble, just like Arduino. It is also immediately breadboard friendly.

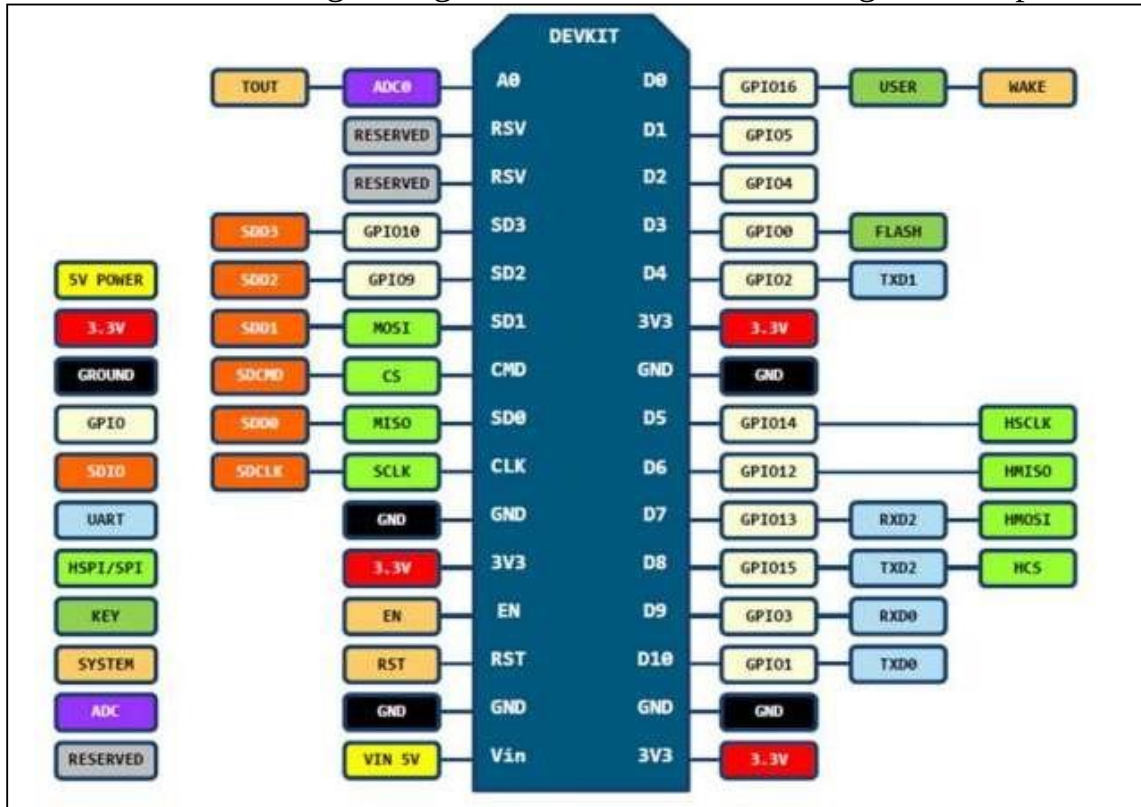


ESP-12E BASED NODEMCU

ESP-12E Wi-Fi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Tensilica L106 integrates industry-leading ultra-low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna. The module supports standard

IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the

add modules to an existing device networking, or building a separate network controller. ESP8266 is high integration wireless SOCs, designed for space and

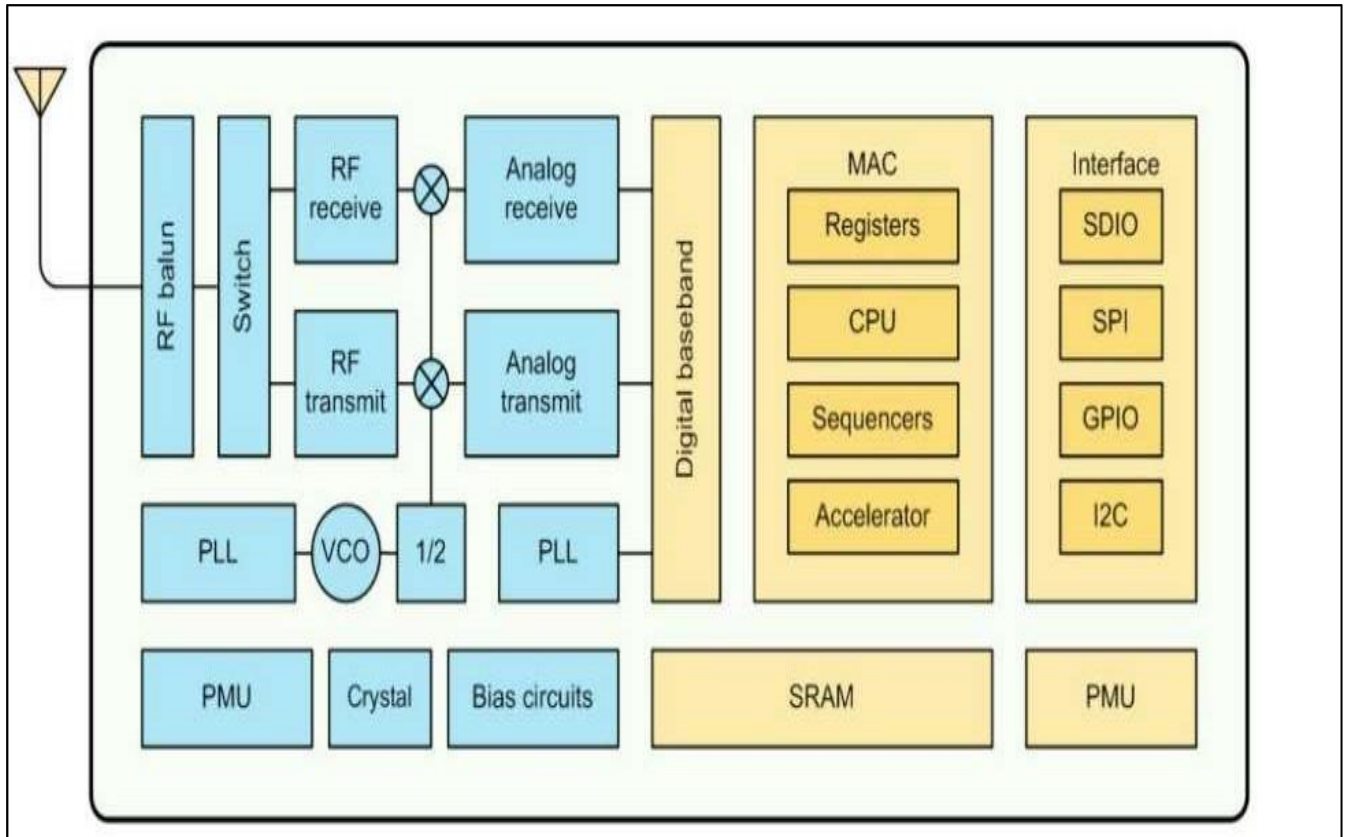


power constrained mobile platform designers. It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

NODEMCU PIN CONFIGURATION

ESP8266EX offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor. When ESP8266EX hosts the application, it boots up directly from an external flash. It has integrated cache to improve the performance of the system in such applications. Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any micro controller based design with simple connectivity (SPI/SDIO or I2C/UART interface).

ESP8266EX is among the most integrated Wi-Fi chip in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.



ESP-12E ARCHITECTURE

ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the Wi-Fi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs; codes for such applications are provided in examples in the SDK. Espressif Systems' Smart Connectivity Platform (ESCP) demonstrates sophisticated system-level features include fast sleep/wake context

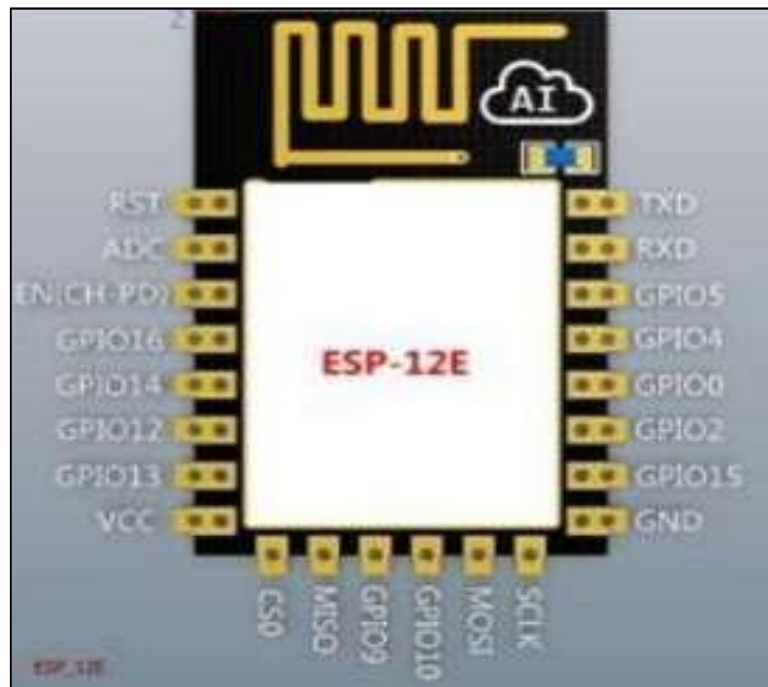
switching for energy-efficient VoIP, adaptive radio biasing. For low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

Features

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation and 0.4s guard interval
- Deep sleep power < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

- +20dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C

ESP-12E



PIN CONFIGURATION & PIN DESCRIPTION

NO.	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result. Input voltage range 0-1v, scope: 0-1024
3	EN	Chip enable pin. Active high
4	IO16	GPIO16; can be used to wake up the chipset from deep sleep mode.
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input
11	IO9	GPIO9
12	IO10	GPIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	IO2	GPIO2; UART1_TXD
18	IO0	GPIO0
19	IO4	GPIO4
20	IO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

PIN MODE

Mode	GPIO15	GPIO0	GPIO2
UART	Low	Low	High
Flash Boot	Low	High	High

Receiver Sensitivity

Parameters	Min	Typical	Max	Unit
Input frequency	2412		2484	MHz
Input impedance		50		Ω
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
Adjacent Channel Rejection				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

Functional Descriptions

MCU

ESP8266EX is embedded with Tensilica L106 32-bit micro controller (MCU), which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz. ESP8266EX is often integrated with external sensors and other specific devices through its GPIOs; codes for such applications are provided in examples in the SDK.

External SPI Flash

This module is mounted with an 4 MB external SPI flash to store user programs. If larger definable storage space is required, a SPI flash with larger memory size is preferred. Theoretically speaking, up to 16 MB memory capacity can be supported. Suggested SPI Flash memory capacity:

- OTA is disabled: the minimum flash memory that can be supported is 512 kB;
 - OTA is enabled: the minimum flash memory that can be supported is 1 MB.
- Several SPI modes can be supported, including Standard SPI, Dual SPI, and Quad SPI.

Crystal

Currently, the frequency of crystal oscillators supported includes 40MHz, 26MHz and 24MHz. The accuracy of crystal oscillators applied should be $\pm 10\text{PPM}$, and the operating temperature range should be between -20°C and 85°C . When using the downloading tools, please remember to select the right crystal

oscillator type. In circuit design, capacitors C1 and C2, which are connected to the earth, are added to the input and output terminals of the crystal oscillator respectively. The values of the two capacitors can be flexible, ranging from 6pF to 22pF, however, the specific capacitive values of C1 and C2 depend on further testing and adjustment on the overall performance of the whole circuit. Normally, the capacitive values of C1 and C2 are within 10pF if the crystal oscillator frequency is 26MHz, while the values of C1 and C2 are 10pF.

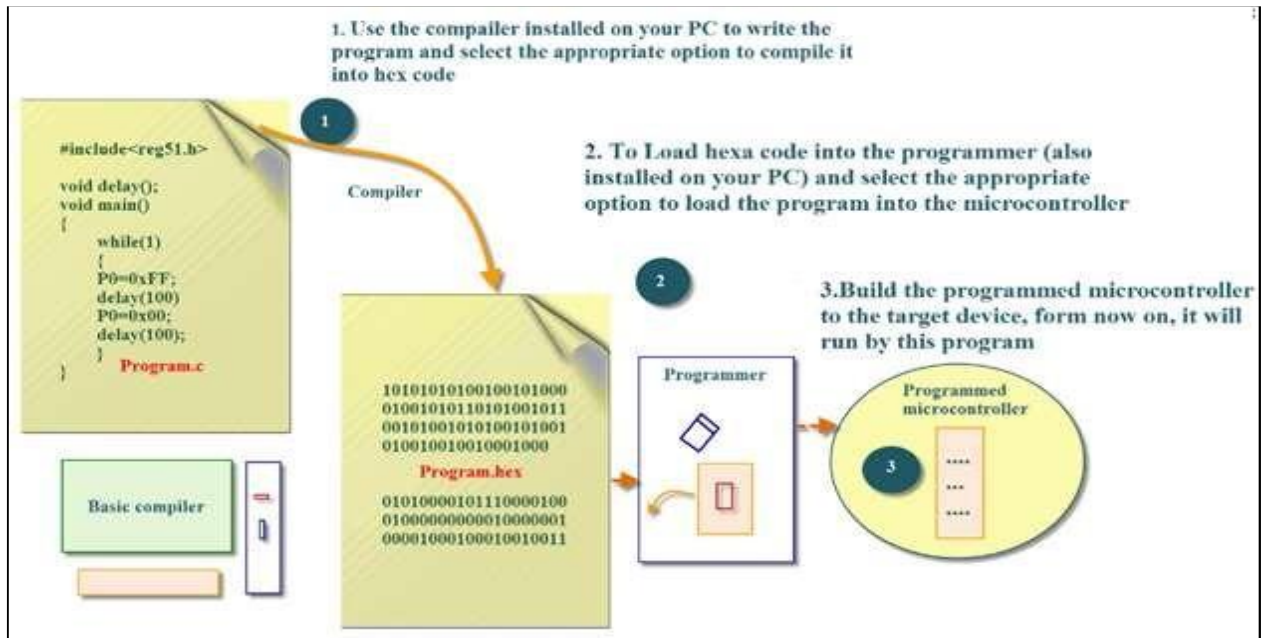
4.6 SOFTWARE REQUIREMENTS:

4.6.1 EMBEDDED C

Embedded C is most popular programming language in software field for developing electronic gadgets. Each processor used in electronic system is associated with embedded software.

Embedded C programming plays a key role in performing specific function by the processor. In day-to-day life we used many electronic devices such as mobile phone, washing machine, digital camera, etc. These all device working is based on microcontroller that are programmed by embedded C.

Let's see the block diagram representation of embedded system programming:



The Embedded C code written in above block diagram is used for blinking the LED connected with Port0 of microcontroller.

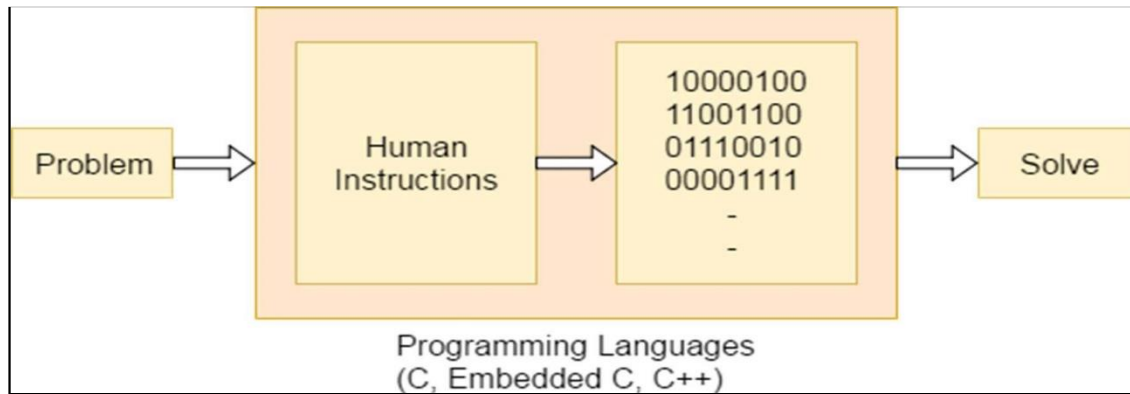
In embedded system programming C code is preferred over other language. Due to the following reasons:

- Easy to understand
- High Reliability
- Portability
- Scalability

EMBEDDED SYSTEM PROGRAMMING:

Basic Declaration

Let's see the block diagram of Embedded C Programming development:

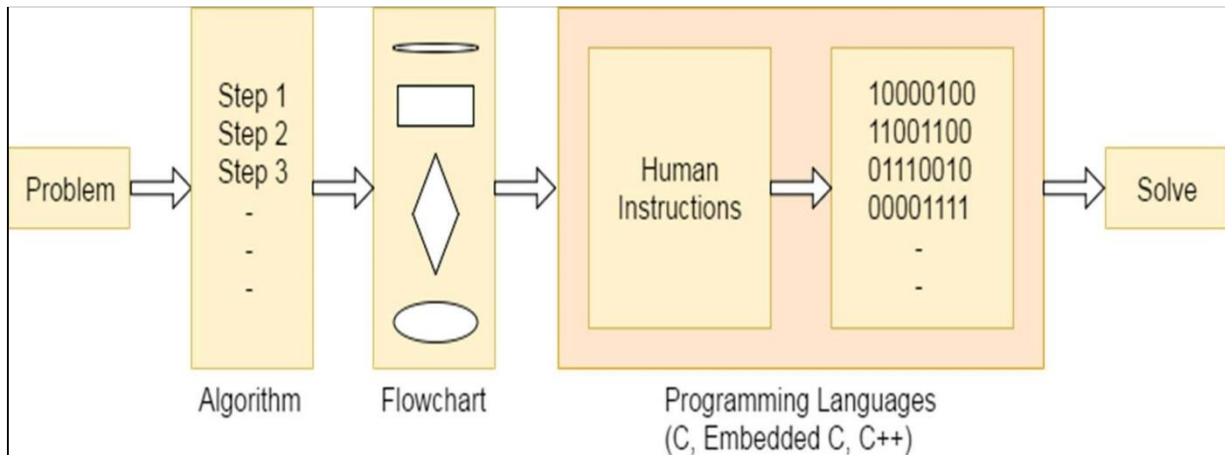


Function is a collection of statements that is used for performing a specific task and a collection of one or more functions is called a programming language. Every language is consisting of basic elements and grammatical rules. The C language programming is designed for function with variables, character set, data types, keywords, expression and so on are used for writing a C program.

The extension in C language is known as embedded C programming language. As compared to above the embedded programming in C is also have some additional features like data types, keywords and header file etc is represented by `#include<microcontroller name.h>`.

BASIC EMBEDDED C PROGRAMMING STEPS:

Let's see the block diagram representation of Embedded C Programming Steps:



The microcontroller programming is different for each type of operating system. Even though there are many operating system are exist such as Windows, Linux, RTOS, etc but RTOS has several advantage for embedded system development.

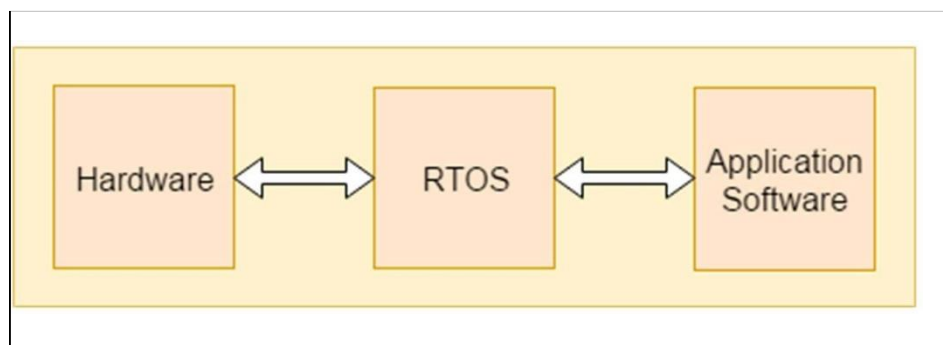
EMBEDDED SYSTEMS:

Embedded System is a system composed of hardware, application software and real time operating system. It can be small independent system or large combinational system.

Our Embedded System tutorial includes all topics of Embedded System such as characteristics, designing, processors, microcontrollers, tools, addressing modes, assembly language, interrupts, embedded c programming, led blinking, serial communication, lcd programming, keyboard programming, project implementation etc.

An embedded system is a combination of three major components:

- **Hardware:** Hardware is physically used component that is physically connected with an embedded system. It comprises of microcontroller based integrated circuit, power supply, LCD display etc.
- **Application software:** Application software allows the user to perform varieties of application to be run on an embedded system by changing the code installed in an embedded system.
- **Real Time Operating system (RTOS):** RTOS supervises the way an embedded system work. It act as an interface between hardware and application software which supervises the application software and provide mechanism to let the processor run on the basis of scheduling for controlling the effect of latencies.



CHARACTERISTICS OF EMBEDDED SYSTEM:

- An embedded system is software embedded into computer hardware that makes a system dedicated to be used for variety of application.
- Embedded system generally used for do specific task that provide real-time output on the basis of various characteristics of an embedded system.

- Embedded system may contain a smaller part within a larger device that used for serving the more specific application to perform variety of task using hardware-software intermixing configuration.
- It provides high reliability and real-time computation ability.

Advantages:

- Same hardware can be used in variety of application.
- Lesser power requirement
- Lower operational cost of system
- Provide high performance and efficiency

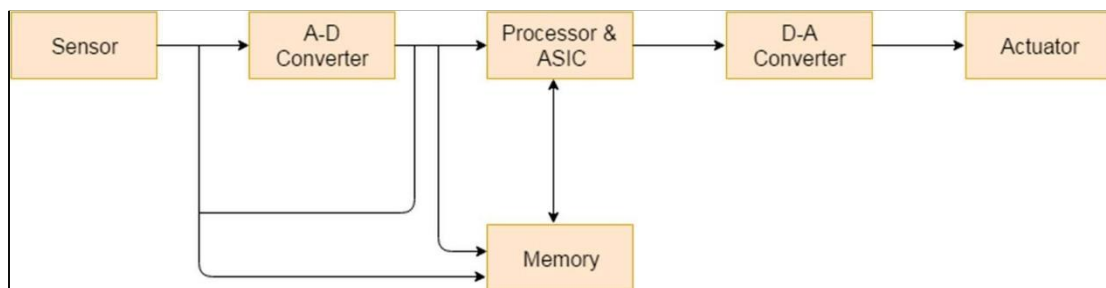
Disadvantages:

- Developing a system required more time. Due to functional complexity.
- Skilled engineers required because one mistake may result in destroying of complete project.

Designing of an embedded system

BASIC STRUCTURE OF AN EMBEDDED SYSTEM:

Let's see the block diagram shows the basic structure of an embedded system.

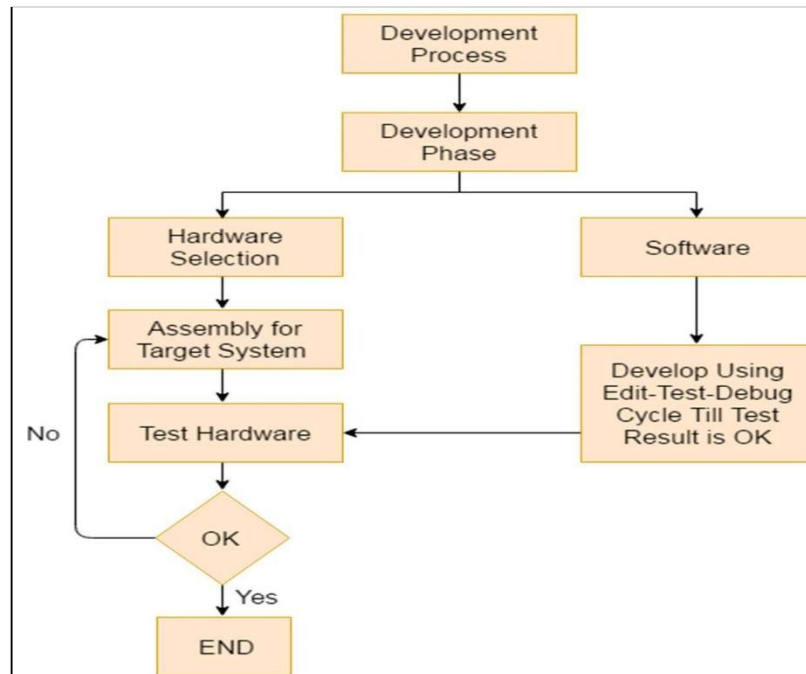


- **Sensor:** Sensor used for sensing the change in environment condition and it generate the electric signal on the basis of change in environment condition. Therefore it is also called as transducers for providing electric input signal on the basis of change in environment condition.
- **A-D Converter:** An analog-to-digital converter is a device that converts analog electric input signal into its equivalent digital signal for further processing in an embedded system.
- **Processor & ASICs:** Processor used for processing the signal and data to execute desired set of instructions with high-speed of operation. Application specific integrated circuit (ASIC) is an integrated circuit designed to perform task specific operation inside an embedded system.
- **D-A Converter:** A digital-to-analog converter is a device that converts digital electric input signal into its equivalent analog signal for further processing in an embedded system.
- **Actuators:** Actuators is a comparator used for comparing the analog input signal level to desired output signal level for providing the error free output from the system.

DESIGN STEPS REQUIRED FOR THE DEVELOPMENT OF EMBEDDED SYSTEM:

Designing steps required for embedded system are different from the design process of another electronic system.

Let's see a flow chart represent the design steps required in the development of an embedded system:



EMBEDDED SYSTEM TOOLS AND PERIPHERALS:

Compiler:

Compiler is used for converting the source code from a high-level programming language to a low-level programming language. It converts the code written in high level programming language into assembly or machine code. The main reason for conversion is to develop an executable program.

Let's see the operations performed by compiler are:

- Code generation
- Code optimization
- Parsing
- Syntax direct translation
- Preprocessing

Cross-Compiler:

If a program compiled is run on a computer having different operating system and hardware configuration than the computer system on which a compiler compiled the program, that compiler is known as cross-compiler.

Decompiler:

A tool used for translating a program from a low-level language to high-level language is called a decompiler. It is used for conversion of assembly or machine code to high-level programming language.

Assembler:

Assembler is embedded system tool used for translating a computer instruction written in assembly language into a pattern of bits which is used by the computer processor for performing its basic operations. Assembler creates an object code by translating assembly language instruction into set of mnemonics for representing each low-level machine operation.

DEBUGGING TOOLS IN AN EMBEDDED SYSTEM:

Debugging is a tool used for reducing the number of error or bugs inside a computer program or an assembled electronic hardware.

Debugging of a compact subsystem is difficult because a small change in one subsystem can create bugs in another system. The debugging used inside embedded system differs in terms of their development time and debugging features.

Let's see the different debugging tools used in embedded system are:

Simulators:

Simulator is a tool used for simulation of an embedded system. Code tested for microcontroller unit by simulating code on the host computer. Simulator is used for model the behavior of the complete microcontroller in software.

Functions of simulators:

Let's see the functions performed by simulator are:

- It defines the processing or processor device family with various version of target system.
- It monitors the detailed information of a source code and symbolic arguments as the execution goes for each single step of operation.
- It simulates the ports of target system for each single step of execution.
- It provides the working status of RAM.
- It monitors the response of system and determines the throughput.
- It provides the complete meaning of the present command.
- It monitors the detailed information of the simulator commands entered from the keyboard or selected from the menu.
- It facilitates synchronization of internal peripherals and delays.

MICROCONTROLLER STARTER KIT:

For developing an embedded system based project a complete microcontroller starter kit is required. The major advantage of this kit over simulator is that they work in real-time operating condition. Therefore it allows the easy input/output functional verification. Consider a microcontroller starter kit consists of:-

- Hardware Printed Circuit Board (PCB)
- In-System Programmer (ISP)

- Some embedded system tools like compiler, assembler, linker, etc
- Sometimes, there is a requirement of an Integrated Development Environment (IDE)

The above component available in microcontroller starter kit is completely enough and the cheapest option available for developing simple microcontroller projects.

Emulators:

An emulator is a software program or a hardware kit which emulates the functions of one computer system into another computer system. Emulators have an ability to support closer connection to an authenticity of the digital object.

It can also be defined as the ability of a computer program in electronic device to emulate another program or device. It focusing on recreating the original computer environment and helps a user to work on any type of application or operating system.

PERIPHERAL DEVICES IN EMBEDDED SYSTEMS:

Communication of an embedded system with an outside environment is done by using different peripheral devices as a combination with microcontroller.

Let's see the different peripheral devices in embedded system are:-

- Universal Serial Bus (USB)
- Networks like Ethernet, Local Area Network(LAN) etc
- Multi Media Cards (SD Cards, Flash memory, etc)
- Serial Communication Interface (SCI) like RS-232, RS-485, RS-422,etc

- Synchronous Serial Communication Interface like SPI, SSC and ESSI
- Digital to Analog/ Analog to Digital (DAC/ADC)
- General Purpose Input/Output (GPIO)
- Debugging like In System Programming (ISP), In Circuit Serial Programming (ICSP), BDM Port, etc

CRITERIA FOR CHOOSING MICROCONTROLLER:

Choosing a microcontroller is essential process in designing of embedded system. While selecting a microcontroller, make sure that it meets the system need and it must be cost effective. We need to decide whether an 8-bit, 16-bit or 32-bit microcontroller is best suitable for the computing needs of a task.

In addition to above, the following points need to be kept in mind while selecting a microcontroller:-

- **speed:** The operational speed of the microcontroller or the highest speed microcontroller can support.
- **Packaging:** Packaging is important for improving the assembling, space and prototyping of an end-product.
- **RAM and ROM:** On the basis of operation of embedded system and memory need for storage data and programs the type of microcontroller required for designing system is decided.
- **Count of I/O pins:** The number of input and output devices connected with the system plays an essential role in choosing the type of microcontroller.
- **Cost per unit:** It is important in terms of final cost of the product in which the microcontroller is to be used.

- **Power consumption:** Power consumption plays an important role formaintaining the efficiency of an embedded system

4.6.2 ARDUINO SOFTWARE IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.



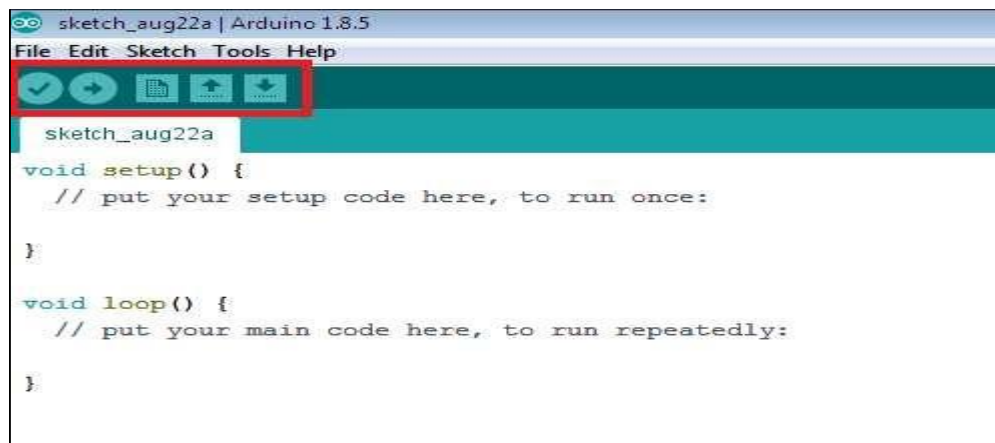
WRITING SKETCHES:

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension **.ino**.

The editor has features for cutting/pasting and for searching/replacing text. The

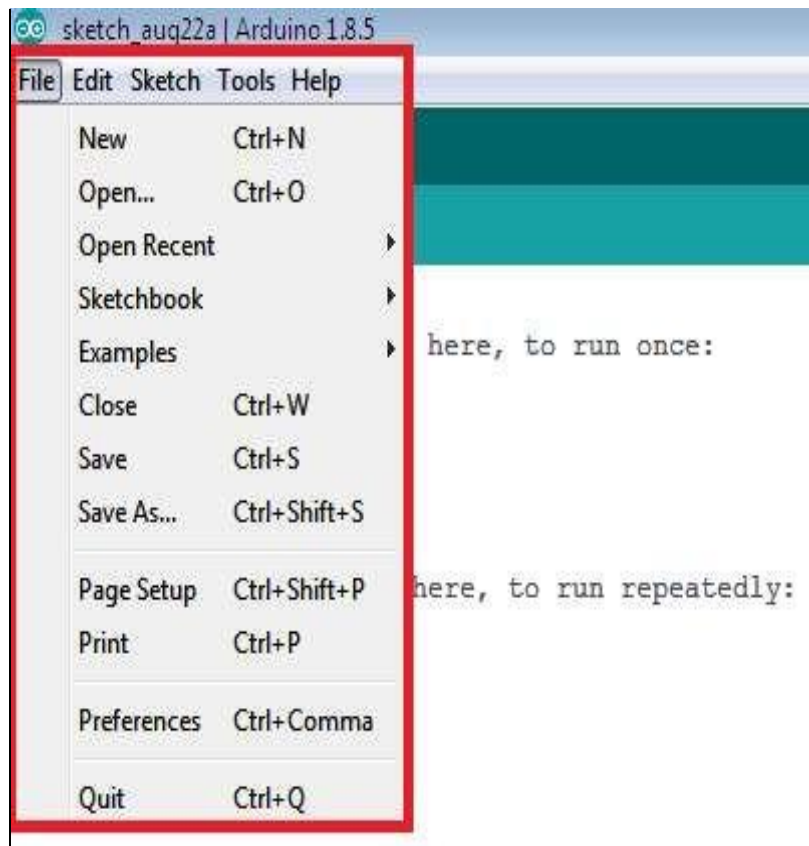
message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

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



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

FILE:



- *New*

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

- *Open*

Allows loading a sketch file browsing through the computer drives and folders.

- *OpenRecent*

Provides a short list of the most recent sketches, ready to be opened.

➤ *Sketchbook*

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

➤ *Examples*

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

➤ *Close*

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

➤ *Save*

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

➤ *Saveas...*

Allows saving the current sketch with a different name.

➤ *PageSetup*

It shows the Page Setup window for printing.

➤ *Print*

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

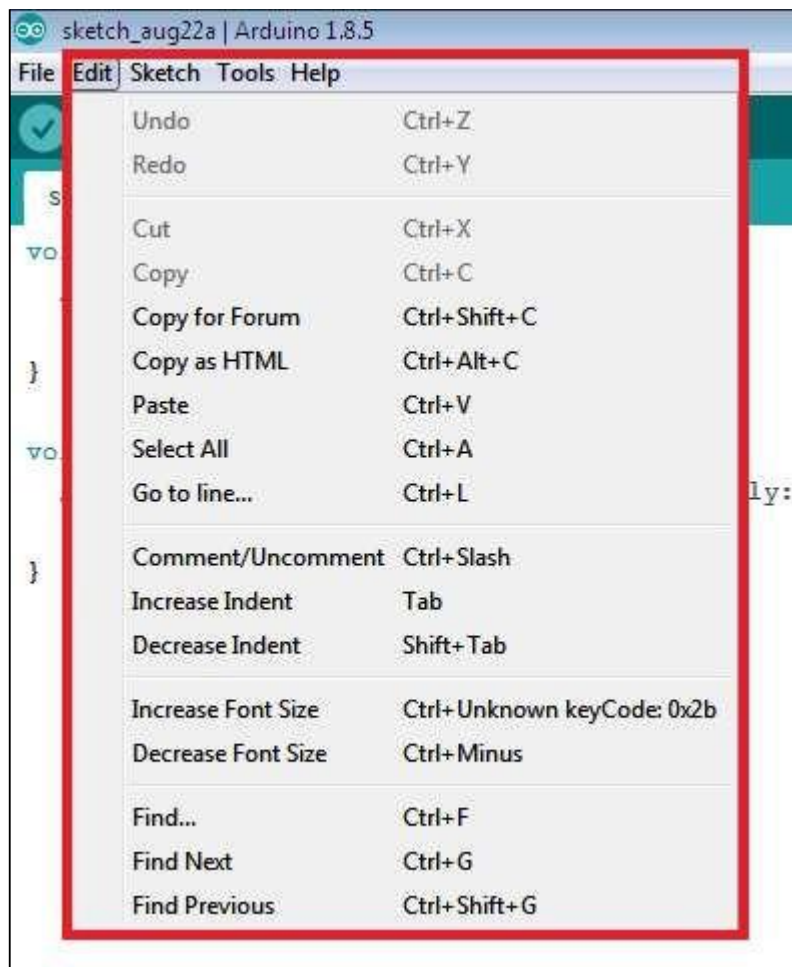
➤ *Preferences*

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

➤ *Quit*

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

EDIT:



➤ *Undo/Redo*

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

➤ *Cut*

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

➤ *Copy*

Duplicates the selected text in the editor and places it into the clipboard.

➤ *CopyforForum*

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

➤ *CopyasHTML*

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

➤ *Paste*

Puts the contents of the clipboard at the cursor position, in the editor.

➤ *SelectAll*

Selects and highlights the whole content of the editor.

➤ *Comment/Uncomment*

Puts or removes the // comment marker at the beginning of each selected line.

➤ *Increase/DecreaseIndent*

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

➤ *Find*

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

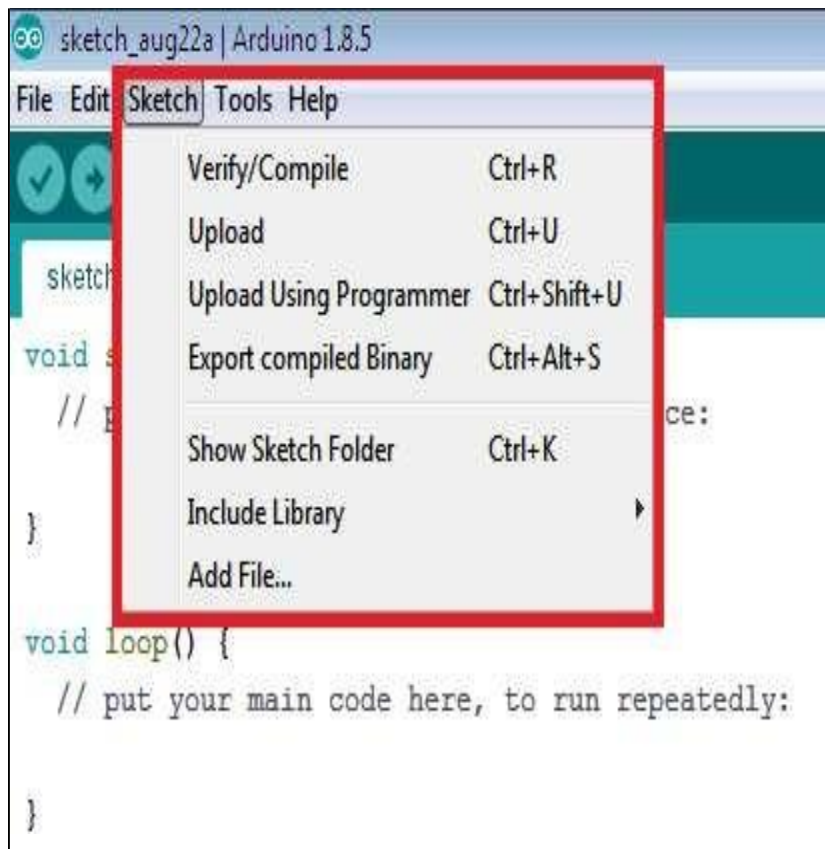
➤ *FindNext*

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

➤ *FindPrevious*

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

SKETCH:



➤ *Verify/Compile*

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

➤ *Upload*

Compiles and loads the binary file onto the configured board through the configured Port.

➤ *Upload Using Programmer*

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

➤ *Export Compiled Binary*

Saves a .hex file that may be kept as archive or sent to the board using other tools.

➤ *Show Sketch Folder*

Opens the current sketch folder.

➤ *Include Library*

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

➤ *Add File...*

Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on

the small triangle icon below the serial monitor one on the right side of the toolbar.

TOOLS:

- *Auto Format*

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

- *Archive Sketch*

Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

- *Fix Encoding & Reload*

Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

- *Serial Monitor*

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

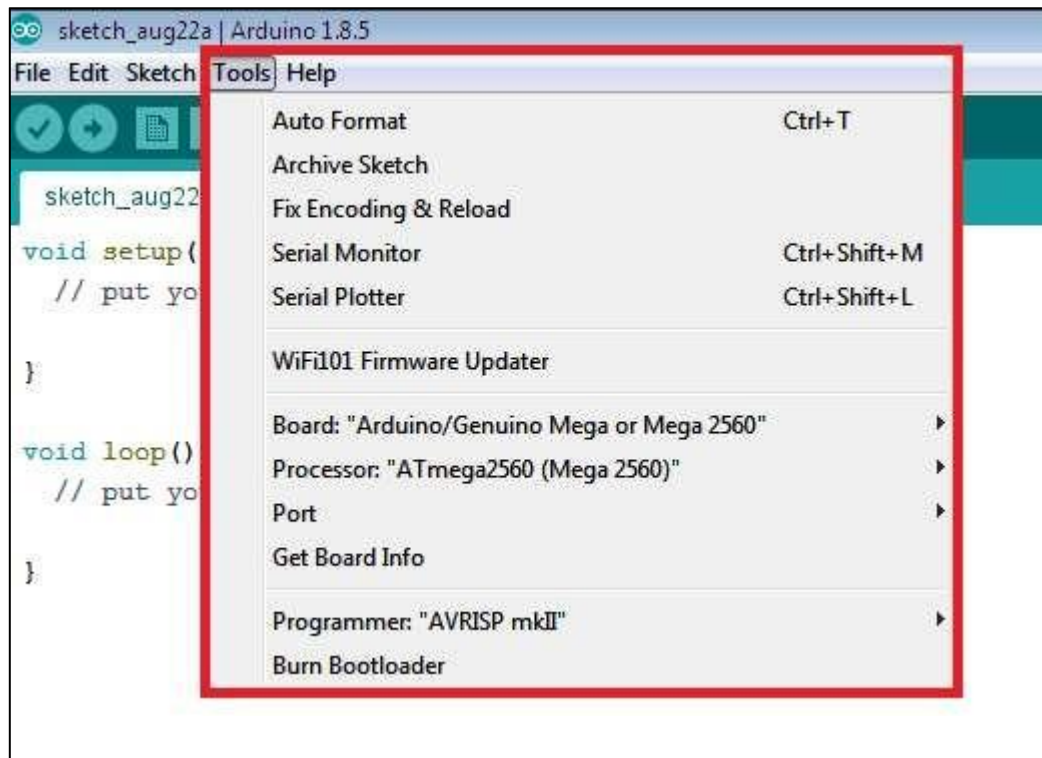
- *Board*

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

- *Port*

This menu contains all the serial devices (real or virtual) on your machine.

It should automatically refresh every time you open the top-level tools menu.



➤ *Programmer*

For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a boot loader to a new microcontroller, you will use this.

➤ *Burn Boot loader*

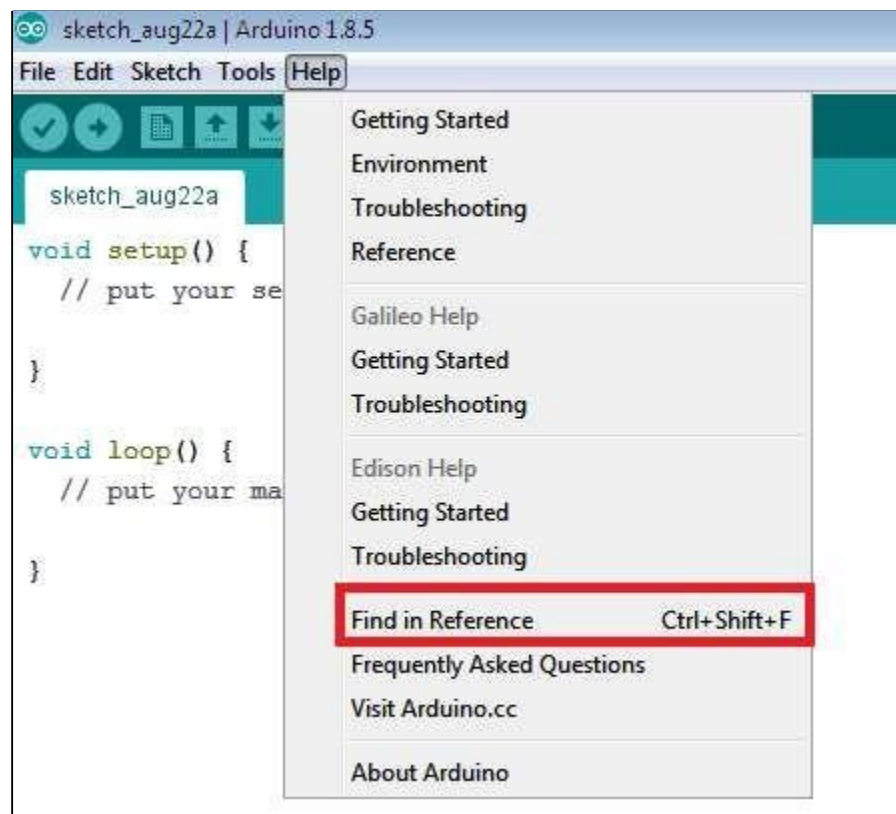
The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally comes without a boot

loader). Ensure that you've selected the correct board from

the **Boards** menu before burning the boot loader on the target board. This command also set the right fuses.

Help:

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



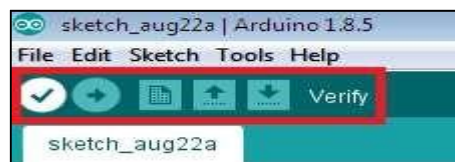
➤ *FindinReference*

This is the only interactive function of the Help menu: it directly selects the

relevant page in the local copy of the Reference for the function or command under the cursor.

SKETCHBOOK:

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



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

TABS, MULTIPLE FILES, AND COMPILATION:

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

UPLOADING:



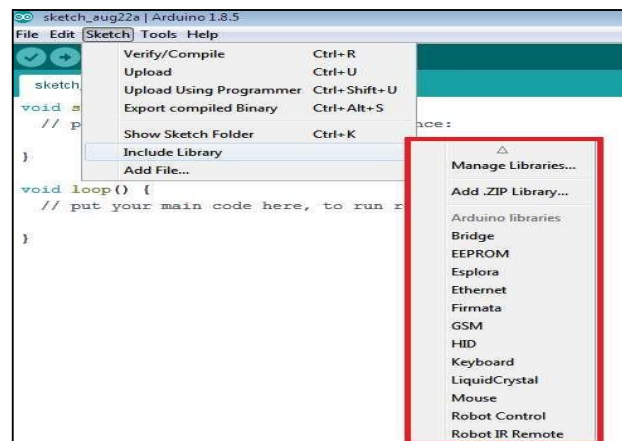
Before uploading your sketch, you need to select the correct items from the **Tools > Board** and **Tools > Port** menus. The boards are described below. On the Mac, the serial port is probably something like **/dev/tty.usbmodem241** (for an Uno or Mega2560 or Leonardo) or **/dev/tty.usbserial-1B1**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

When you upload a sketch, you're using the Arduino **bootloader**, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is

active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The boot loader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

LIBRARIES:

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

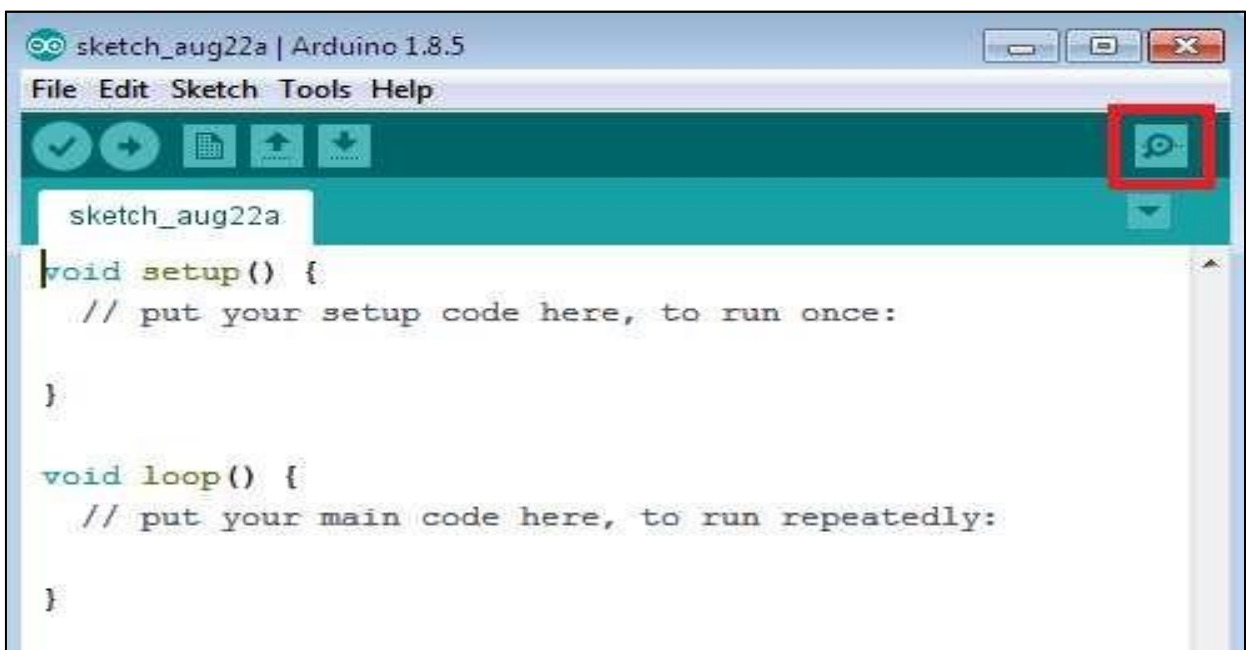


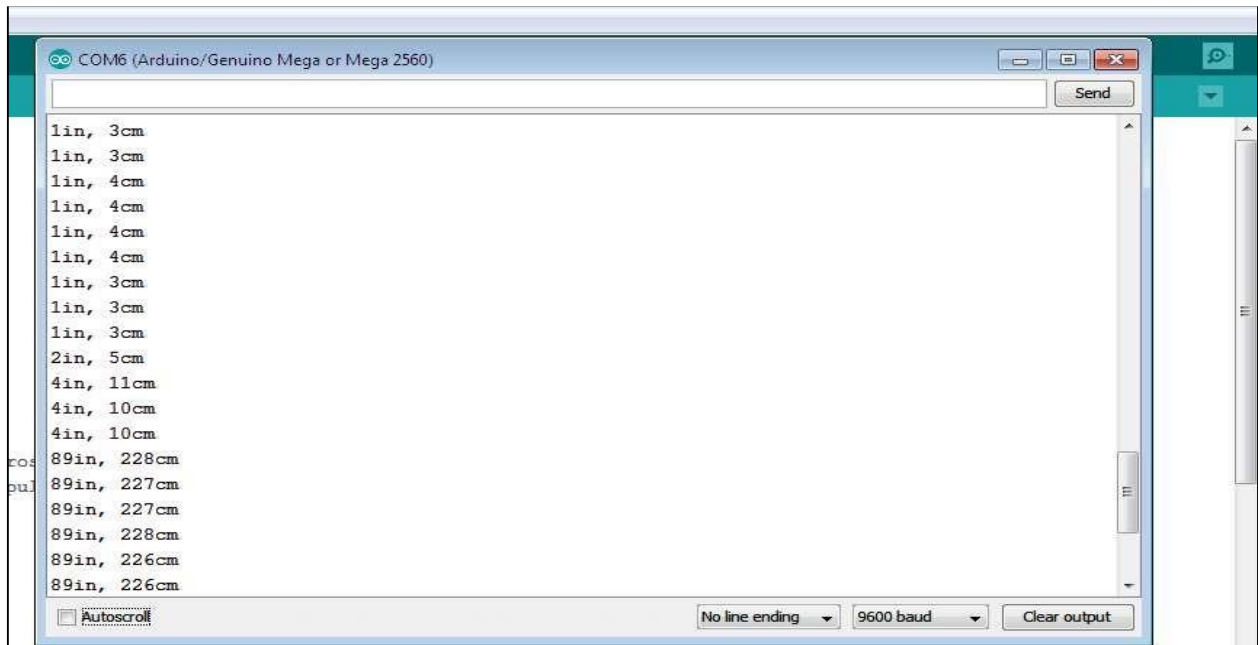
There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See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

SERIAL MONITOR:

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to **Serial.begin** in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

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





PREFERENCES:

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

LANGUAGE SUPPORT:



Since version 1.0.1, the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.)

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

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

CHAPTER: 5 - OUTPUT

5.1 OUTPUT:

The screenshot displays the Adafruit IO web interface. At the top, there is a navigation bar with links for Shop, Learn, Blog, Forums, LIVE!, AdaBox, and IO. On the right side of the navigation bar, there are links for Account and a shopping cart icon showing 0 items. Below the navigation bar, the user's profile 'saisrinivas' is shown, along with links to Dashboards and traffic. The main content area displays a list of traffic events in a table format. The table has columns for Date, Time, Location, and Event. The events are listed as follows:

Date	Time	Location	Event
2022/03/02	8:02PM	Default	traffic EMERGENCY
2022/03/03	8:52AM	Default	traffic EMERGENCY
2022/03/03	8:52AM	Default	traffic TH
2022/03/03	8:52AM	Default	traffic THEFT
2022/03/03	8:52AM	Default	traffic THEFT
2022/03/03	8:52AM	Default	traffic EMERGENCY
2022/03/03	8:52AM	Default	traffic THEFT
2022/03/03	8:52AM	Default	traffic TRAFFIC
2022/03/03	8:52AM	Default	traffic EMERGENCY
2022/03/03	8:52AM	Default	traffic TRAFFIC
2022/03/03	8:52AM	Default	traffic TRAFFIC
2022/03/03	8:52AM	Default	traffic TRAFFIC
2022/03/03	8:52AM	Default	traffic EMERGENCY

At the bottom of the page, there is a footer with links for Get Help, Quick Guides, API Documentation, Learn, IO Plus, and News. On the right side of the footer, there are social media icons for Instagram, Twitter, Facebook, YouTube, and TikTok.

CHAPTER 6 -CONCLUSION

6.1 APPLICATIONS

This method is used in monitoring traffic and clear the traffic for emergency vehicle.

6.2 FUTURE ENHANCEMENT:

- In future we add sound waves and image processing techniques made the traffic clearance efficiently.

6.3 ADVANTAGES:

- Emergency vehicles can alter the signal during it approach to save time and lives.
- Traffic density can be monitored and altered accordingly.

6.4 CONCLUSION:

Traffic management is the most concerned research issue in the real world environment which requires to be focused well for the better traffic management. The foremost objective of this investigational work is to implement a novel framework namely traffic monitoring for emergency vehicle using RFID that can make short term decision about the traffic management, thus the accurate and efficient traffic clearance can be achieved .By clearing the traffic on the road, emergency vehicle can easily pass through and save lives .The traffic also monitored on a webpage.

6.5 REFERENCES :

- [1] N. Cheng *et al.*, “Big data driven vehicular networks,” *IEEE Netw.*, vol. 32, no. 6, pp. 160–167, Nov. 2018.
- [2] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura, and A. A. F. Loureiro, “Vehicular ad hoc networks: A new challenge for localization-based systems,” *Comput. Commun.*, vol. 31, no. 12, pp. 2838–2849, Jul. 2008.
- [3] E. I. Laftchiev, C. M. Lagoa, and S. N. Brennan, “Vehicle localization using in-vehicle pitch data and dynamical models,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 206–220, Feb. 2015.
- [4] B.-H. Lee, J.-H. Song, J.-H. Im, S.-H. Im, M.-B. Heo, and G.-I. Jee, “GPS/DR error estimation for autonomous vehicle localization,” *Sensors*, vol. 15, no. 8, pp. 20779–20798, Aug. 2015.
- [5] K. Jo, K. Chu, K. Lee, and M. Sunwoo, “Integration of multiple vehicle models with an IMM filter for vehicle localization,” in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 746–751.
- [6] R. Parker and S. Valaee, “Vehicular node localization using received signal-strength indicator,” *IEEE Trans. Veh. Technol.*, vol. 56, no. 6, pp. 3371–3380, Nov. 2007.
- [7] M. Rohani, D. Gingras, V. Vigneron, and D. Gruyer, “A new decentralized Bayesian approach for cooperative vehicle localization based on fusion of GPS and VANET based inter-vehicle distance measurement,” *IEEE Intell. Transp. Syst. Mag.*, vol. 7, no. 2, pp. 85–95, Summer 2015.
- [8] N. Drawil and O. Basir, “Vehicular collaborative technique for location estimate correction,” in *Proc. IEEE 68th Veh. Technol. Conf.*, Sep. 2008, pp. 1–5.

- [9] S.-J. Han and J. Choi, "Real-time precision vehicle localization using numerical maps," *ETRI J.*, vol. 36, no. 6, pp. 968–978, Dec. 2014.
- [10] S. Hoermann and P. V. K. Borges, "Vehicle localization and classification using off-board vision and 3-D models," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 432–447, Apr. 2014.