# ABSTRACT

Boating is a popular activity, and there are millions of boaters worldwide. Boat accidents are just as likely to occur as car accidents. Every day we hear many stories about boat trips that ended up tragically. Boaters can keep themselves and their passengers safe by learning about responsible boat operation, etiquette, and the rules of the waterways. In this paper, we propose a safety and accident prevention system for boat owners. Just as big automotive companies are developing technologies to help prevent accidents on roads, it would be a good idea to implement these safety measures for boats as well. The proposed system offers several features including a smart radar system, a crash detection system, and an application connected to the Coast Guard. The proposed system would help to reduce the boat accidents and save the humans life to protect the any kind of boat accidentsBoat design and construction has been an integral part of the history of a strong  nation like India. Over the years, there have been numerous changes in the design and development of boats.We intend to assist in the crafting and grassroots support of legislation which will result in more boaters having the education and technology necessary to avoid accidents or survive should and incident occur.The Foundation will work tirelessly to increase the awareness and availability of tools and technology that can help prevent near-shore and offshore boating accidents and improve victims chances for survival and recovery.Together, we can make recreationalboating and water sports safer for all to enjoy.Boating accidents seem pretty self-explanatory: they are accidents that happen on or with a boat. However, it covers a wide variety of different accidents and vessels, from sailboats to speedboats to personal watercraft (jet skis) to even hovercraft. The most common kinds of accidents include either hitting another vessel or hitting an obstacle, like a reef or a dock.

# CHAPTER-1

# INTRODUCTION

## 1.1 INTRODUCTION

The United States Coast Guard (USCG) has the legal responsibility to collect, analyze, and publish recreational boating accident data and statistical information for the fifty states, five U.S. territories, and the District of Columbia. Boating accident statistics are compiled and used for many purposes, such as: identifying trends; characterizing accident causes; assessing the contributions of operator error, mechanical malfunctions, and environmental factors; and evaluating the possible benefits of government initiatives (e.g., boater education, legislation/regulation, and boat construction standards) to reduce the risks associated with recreational boating activity. Complete and accurate accident data are essential for these purposes.  Over the years, USCG and state boating authorities have made many improvements in the quality, accuracy, and relevance of the boating accident and investigation data captured by the Boating Accident Report Database (BARD) System. For example  though many improvements in the collection of boating accident data and reporting of statistics have been made, opportunities for further improvement remain.

## 1.1 METHODOLOGY

The safe navigation of ships is of high societal concern. A promising approach for analyzing waterway risks is using non-accident critical events as surrogate indicators of collision accidents .These are typically detected in data from the Automatic Identification System (AIS). Recognizing the significant interest in this approach, this article provides a review and analysis of methodsbased on the detection of non-accident critical events from AIS data, which aim to provide insight into maritime water way risk  . Considering also recent calls for increased focus on foundational issues in risk research and safety science, each method

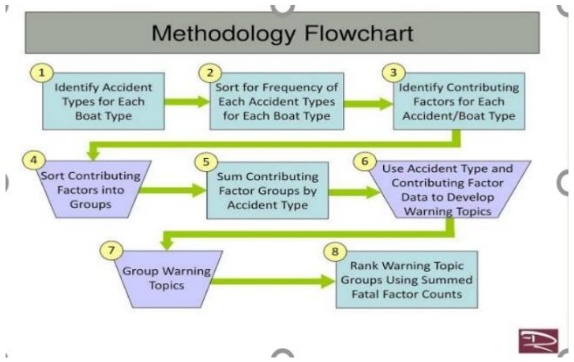

Methodology Flowchart
1
Identify Accident Types for Each Boat Type
2
Sort for Frequency of Each Accident Types for Each Boat Type
3
Identify Contributing Factors for Each Accident/Boat Type
4
Sort Contributing Factors into Groups
5
Sum Contributing Factor Groups by Accident Type
6
Use Accident Type and Contributing Factor Data to Develop Warning Topics
7
Group Warning Topics
8
Rank Warning Topic Groups Using Summed Fatal Factor Counts

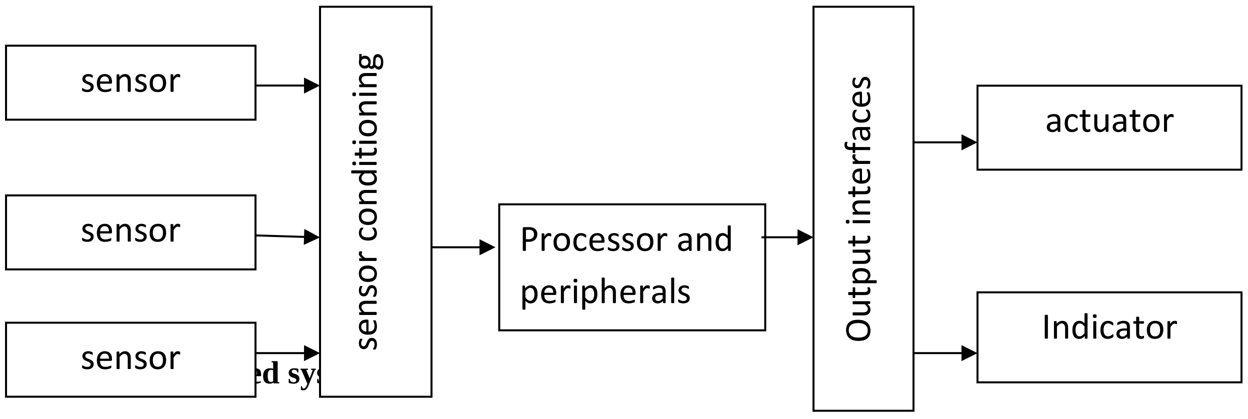
**Fig 1.1 Methodology flow chart**

In the literature is critically reviewed based on five questions: How are non-accident critical events defined? What is the accident-theoretical basis of the method? How are non-accident critical events ranked

## 1.3 SIGNIFICANCE OF THE WORK

### 1.3.1 EMBEDDED SYSTEM

A precise definition of embedded systems is not easy. Simply stated, all computing systems other than general purpose computer (with monitor, keyboard, etc.) are embedded systems. System is a way of working, organizing or performing one or many tasks according to a fixed set of rules, program or plan. In other words, an arrangement in which all units assemble and work together according to a program or plan. An embedded system is a system that has software embedded into hardware, which makes a system dedicated for an application or specific part of an application or product or part of a larger system. It processes a fixed set of pre-programmed instructions to control electromechanical equipment which may be part of an even larger system (not a computer with keyboard, display, etc). A general-purpose definition of embedded systems is that they are devices used to control, monitor or assist the operation of equipment, machinery or plant. “Embedded” reflects the fact that they are an integral part of the system. In many cases, their “ embeddedness ” may be such that their presence is far from obvious to the casual observer. Block diagram of a typical embedded system is shown in fig.



**ed sy**

An embedded system is an engineering artifact involving computation that is subject to physical constraints (reaction constraints and execution constraints) arising through interactions of computational processes with the physical world. Reaction constraints originate from the behavioral requirements & specify deadlines, throughput, and jitter whereas execution constraints originate from the implementation requirements & put bounds on available processor speeds, power, memory and hardware failure rates.  An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is designed for a specific function or for specific function within a larger system. Industrial machines, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys as well as mobile devices are all possible locations for an embedded system.

Embedded systems are computing systems, but can range from having no user interface (UI) for example, on devices in which the embedded system is designed to perform a single task to complex graphical user interfaces (GUI), such as in mobile devices. User interfaces can include buttons, LEDs, touch screen sensing and more. Some systems use remote user interfaces as well.

### 1.3.2 EMBEDDED SYSTEM HARDWARE (MICRO PROCESSER-BASED, MICRO CONTROLLER-BASED)

Embedded systems can be microprocessor or microcontroller based. In either case, there is an integrated circuit (IC) at the heart of the product that is generally designed to carry out computation for real-time operations. Microprocessors are visually indistinguishable from microcontrollers, but whereas the microprocessor only implements a central processing unit (CPU) and thus requires the addition of other components such as memory chips, microcontrollers are designed as self-contained systemsMicrocontrollers include not only a CPU, but also memory and peripherals such as flash memory, RAM or serial communication ports. Because microcontrollers tend to implement full (if relatively low computer power) systems.They are frequently put to use on more complex tasks. Microcontrollers are used, for example, in the operations of vehicles, robots, medical devices and home appliances, among others. At the higher end of microcontroller capability, the term system-on-a-chip (SOC) is often used, though there's no exact delineation in terms of RAM, clock speed and so on.The embedded market was estimated to be in excess of $140 billion in 2013, with many analysts projecting a market larger than $20 billion by 2020. Manufacturers of chips for embedded systems include many main stays of the computer world, such as Apple, IBM, Intel and Texas Instruments, but also numerous other companies that are less familiar to those outside the field.  One highly influential vendor in this space has been ARM, which began as an outgrowth of Acorn, a U.K. maker of early PCs. The RISC-based architecture of the ARM chip, produced under license by other companies, has been widely used in mobile phones and PDAs and remains the most widely deployed SOC in the embedded world.

### 1.3.3 EMBEDDED SYSTEM SOFTWARE

A typical industrial microcontroller is quite unsophisticated compared to a typical enterprise desktop computer and generally depends on a simpler, less-memory-intensive program environment. The simplest devices run on bare metal and are programmed directly using the chip CPU's machine code language. Often, however, embedded systems use operating systems or language platforms tailored to embedded use, particularly where real-time operating environments must be served.

At higher levels of chip capability, such as those found in SOC designers have increasingly decided that the systems are generally fast enough and tasks tolerant of slight variations in reaction time that "near-real-time" approaches are suitable. In these instances, stripped-down versions of the Linux operating system are commonly deployed, though there are also other operating systems that have been pared down to run on embedded systems, including Embedded Java and Windows IOT (formerly Windows Embedded).Generally, storage of programs and operating systems on embedded devices make use either of flash or rewritable flash memory.

### 1.3..4DEBUGGING EMBEDDED SYSTEMS

One area where embedded systems part ways with the operating systems and development environments of other, larger-scale computers is in the area of debugging. Whereas program mers working desktop computer environments have systems that can run both the code being developed and separate debugger applications that monitor the actions of the development codeas it is executed, embedded system programmers generally are afforded no such luxuries. Some programming languages run on microcontrollers with sufficient efficiency that rudimentary interactive debugging is available directly on the chip. Additionally, processors often have CPU debuggers that can be controlled -- and thus control program execution -- via a JTAG or similar debugging port.

In many instances, however, programmers of embedded systems need tools that attach a separate debugging system to the target system via a serial or other port. In this scenario, the programmer can see the source code on the screen of a conventional personal computer just as would be the case in the debugging of software on a desktop computer. A separate, frequently used approach is to run software on a PC that emulates the physical chip in software, thus making it possible to debug the performance of the software as if it were running on an actual, physical chip.

Broadly speaking, embedded systems have received more attention to testing and debugging because a great number of devices using embedded controls are designed for use in situations where safety and reliability are top priorities

### 1.3.5 CHARACTERISTICS

- ➢ Embedded systems are application specific & single functioned; application is known apriority, the programs are executed repeatedly.
- ➢ Efficiency is of paramount importance for embedded systems. They are optimized for energy, code size, execution time, weight & dimensions, and cost.
- ➢ Embedded systems are typically designed to meet real time constraints; a real time system reacts to stimuli from the controlled object/ operator within the time interval dictated by the environment. For real time systems, right answers arriving too late (or even too early) are wrong.
- ➢ Embedded systems often interact (sense, manipulate & communicate) with external world through sensors and actuators and hence are typically reactive systems; a reactive system is in continual interaction with the environment and executes at a pace determined by that environment. They generally have minimal or no user interface.

Hardware. The hardware of **embedded systems** is based around microprocessors and microcontrollers

.

### 1.3.6ADVANTAGES OF EMBEDDED OPERATING SYSTEM

- Small size and faster to load
- More specific to one task
- Easy to manage
- Low cost
- Spend less resources
- These operating system is dedicated to one device so performance is good and use less resources like memory and micro-processors

### 1.3.7 DISADVANTAGES OF EMBEDDED OPERATING SYSTEM

- Difficult to upgrade
- If any problem occurs then you need to reset settings
- Nearly not scalable
- Hardware is limited
- Troubleshooting is difficult
- Difficult to transfer data from one system to other

## 1.4 ORGANIZATION

### 1.4.1 PROJECT OVERVIEW

Missing Boat Identification This project has three buttons, one emergency button and help button and Overload Button has a sensor Wind sensor feature This project is useful for fisher men because it happens whenever Fisher Men goes to Fishing  Purpose.

Case 1 Emergency Mode:-

Emergency means feel whenever that boat is in the middle of the river Otherwise the boat will stop in that time Emergency button will be used click on that button at that time information to station near the emergency town when the boat is sinking. From there tracing boat location then it will be known then that station members they will come and help those on boat

**Case 2 Help Mode:-**Anyone who has ever pressed the button and run out of fuel in that time passes information to the nearby station for help using  GPS and GSM modules they Come help along with the bot engineer and they carrier the some fuel bottle and they will bring what they need

**Case 3 Over Weight Mode:-**When fisher men go to fishing purpose in that time  some body full the boat with fish they filled overload in that overload sensor will be active and shop the boat motor automatic in this case sensor care about boat stability

## 1.1 **CONCLUSION**.

Recognizing the significant interest in this approach, this article provides a review and analysis of methods based on the detection of non-accident critical events from AIS data, which aim to provide insight into maritime waterway risk. Considering also recent calls for increased focus on foundational issues in risk research and safety science, each method in the literature is critically reviewed based on five questions: How are non-accident critical events defined? What is the accident-theoretical basis of the method? How are non-accident critical events ranked? How is the method used? To what extent has the method been validated? Based on the results, it is concluded that focus is needed to build evidence of the validity of the models' results, if these are to be effectively used for waterway risk analysis. As a prerequisite, more focus is needed on how exactly non accident critical events are defined, and what factors are involved in the relation between their occurrence and accident involvement

# CHAPTER-2

# ( LITERATURE REVIEW )

## 2.1 LITERATURE REVIEW

conditions could be natural phenomena such as current, tide and tidal stream, severe wind, reduced visibility (fog, heavy snow and rain), storm seas, darkness etc. affecting the ship or those controlling her.

• Technical failures are shortcomings within the ship, such as corrosion, steering failure, engine failure, or hull failure arising from defective materials or construction, or by the shore-based installations, such as aids to navigation

• Route conditions may include navigational error like over reliance on inaccurate nautical charts, charts of suspect reliability or based upon old surveys, narrow channels with abrupt and angular windings, allowing for very limited maneuverability and exposed to dense marine traffic, such as the Turkish Straits, anchorage contiguous to traffic separation lanes, confined marine areas with insufficient sea-room as well as navigational hazards such as shoals, reefs, wrecks etc.

• Ship-related factors could be the weakness of a ship, associated with her larger size, hence less maneuvering capability and stability or draught constraints.

## 2.2 SHIPPING ACCIDENTS AND MARINE ENVIRONMENT

The Term “marine accident and incident”and “marine casualty” denote undesirablle. Events in connection with ship operations (IMO, 1996).The term “marine incident” is used to denote undesirable marine events, i.e. marine accidents, incidents and near missing events (Weintrit, 2009, p.248).An accident is an undesired event that results in adverse consequences, for example injury, loss of life, economic loss, environmental damage, and damage to or loss of property. Accidents are due to an unexpected combination of conditions 0or events. Despite of the fact that “accident” and“incident” terms states different meanings and consequences, the term “accident” is used in this paper Royal Vasishta’, which capsized in AP’s Godavari with 77 onboard, retrieved after 38 days The ill-fated boat with nearly 77 passengers and crew had capsized in river Godavari as it passed Through whirlpools at Kacaluru on September 15. This type of activities we will overcome the situation and reduce and limit the boat accidents that’s why We are choosing this project

## 2.3 OVERVIEW OF ACCIDENTS

While boating, but there isn't anything we do that doesn't have some degree of risk associated with it. So the question becomes: How risky is boating compared to other activities? Based on an analysis of the BoatUS Marine Insurance claim files between January 1, 2009, and December 31, 2013, an estimated five people died per 100,000 registered boats. According to the Coast Guard 2013 Recreational Boating Statistics, 560 people died in recreational boating accidents in 2013, or 4.7 per 100,000 registered recreational vessels. In 2012, 14 people per 100,000 registered cars and 60 people .

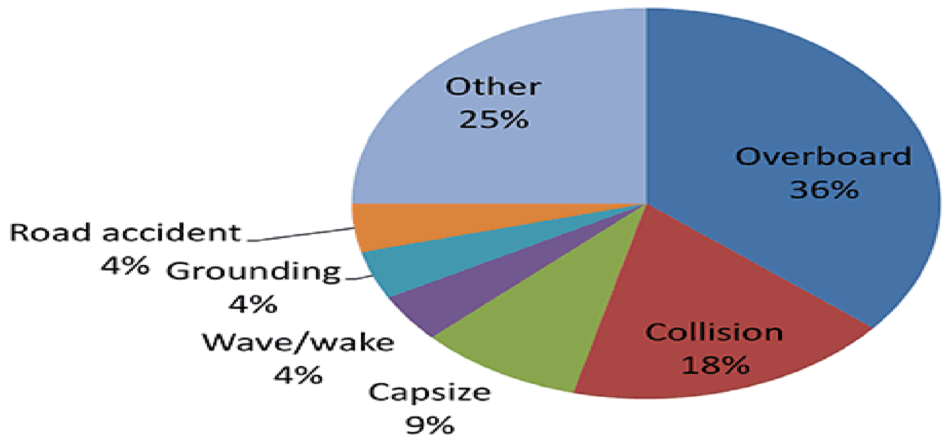


**Fig 2.1 overview of accidents**

The Foundation will work tirelessly to increase the awareness and availability of tools and technology that can help prevent near-shore and offshore boating accidents and improve victims chances for survival and recovery.Together, we can make recreationalboating and water sports safer for all to enjoy.Boating accidents seem pretty self-explanatory: they are accidents that happen on or with a boat. However, it covers a wide variety of different accidents and vessels, from sailboats to speedboats to personal watercraft (jet skis) to even hovercraft. The most common kinds of accidents include either hitting another vessel or hitting an obstacle, like a reef or a dock.

# CHAPTER-3

# ( DESIGN ASPECTS )

## 3.1 UNDERSTANDING ARDUINO NANO

The Arduino board is designed in such a way that it is very easy for beginners to get started with microcontrollers. This board especially is breadboard friendly is very easy to handle the connections. Let's start with powering the Board.

### 3.1.1POWERING YOU ARDUINO NANO

There are totally three ways by which you can power your Nano

#### USB JACK:

Connect the mini USB jack to a phone charger or computer through a cable and it will draw power required for the board to function

#### VIN PIN:

The Vin pin can be supplied with a unregulated 6-12V to power the board. The on-board voltage regulator regulates it to +5V

#### +5V PIN:

If you have a regulated +5V supply then you can directly provide this o the +5V pin of the Arduino

### 3.1.2 INPUT/OUTPUT

There are totally 14 digital Pins and 8 Analog pins on your Nano board. The digital pins can be used to interface sensors by using them as input pins or drive loads by using them as output pins. A simple function like pinMode() and digitalWrite() can be used to control their operation. The operating voltage is 0V and 5V for digital pins. The analog pins can measure analog voltage from 0V to 5V using any of the 8 Analog pins using a simple function liken analogRead()These pins apart from serving their purpose can also be used for special purposes which are discussed below:

- Serial Pins 0 (Rx) and 1 (Tx): Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.

- External Interrupt Pins 2 and 3: These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

- PWM Pins 3, 5, 6, 9 and 11: These pins provide an 8-bit PWM output by using analogWrite() function.
- SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK): These pins are used for SPI communication.
- In-built LED Pin 13: This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.
- I2C A4 (SDA) and A5 (SCA): Used for IIC communication using Wire library.
- AREF: Used to provide reference voltage for analog inputs with analogReference() function.

- Reset Pin: Making this pin LOW, resets the microcontroller.

These special functions and their respective pins are illustrated in the arduino nano pin diagram shown above.

## 2.1MM HOW TO USE ARDUINO NANO

It will hardly take 5-10 minutes to upload you first program to Arduino Nano. All you need the Arduino IDE an USB cable and your Nano board itself.

## 3.1.3 DOWNLOAD AND INSTALL ARDUINO

The first step would be install the Arduino IDE which is available for download for free from the below link. After installing Arduino you might also want to install the drivers (link given below) for you Arduino to communicate with your Computer

Arduino IDE Download

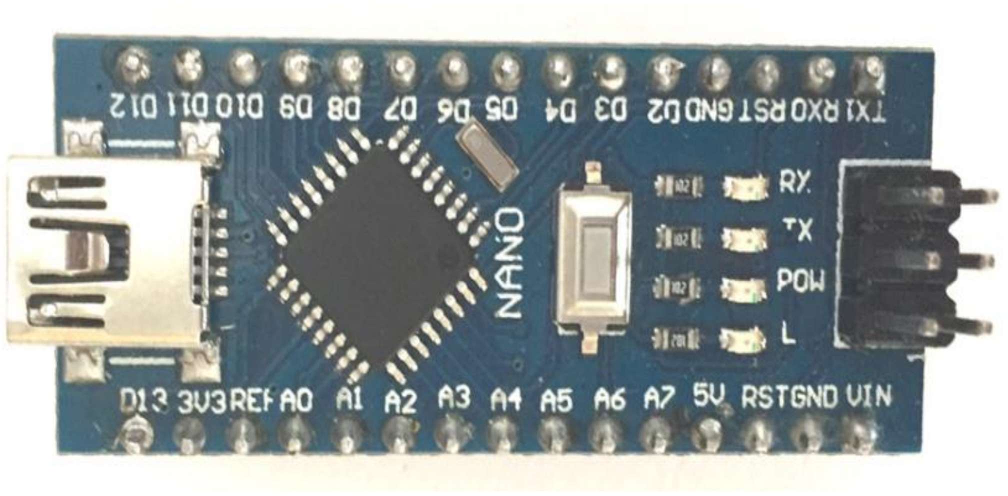
Driver  Download

## 3.1.4 UPLOADING YOUR FIRST PROGRAM

Once arduino IDE is installed on the computer, connect the board with computer using USB cable. Now open the arduino IDE and choose the correct board by selecting Tools>Boards>Arduino/Nano, and choose the correct Port by selecting Tools>Port. Arduino

Uno is programmed using Arduino programming language based on Wiring. To get it started with Arduino Uno board and blink the built-in LED, load the example code by selecting Files>Examples>Basics>Blink. Once the example code (also shown below) is loaded into your IDE, click on the 'upload' button given on the top bar. Once the upload is finished, you should see the Arduino's built-in LED blinking.

## 3.2 NANO ARDUINO PIN DIAGRAM


D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 GND RST RX0 TX1
RX
TX
POW
L
NANO
D13 3V3 REF A0 A1 A2 A3 A4 A5 A6 A7 5V RST GND VIN

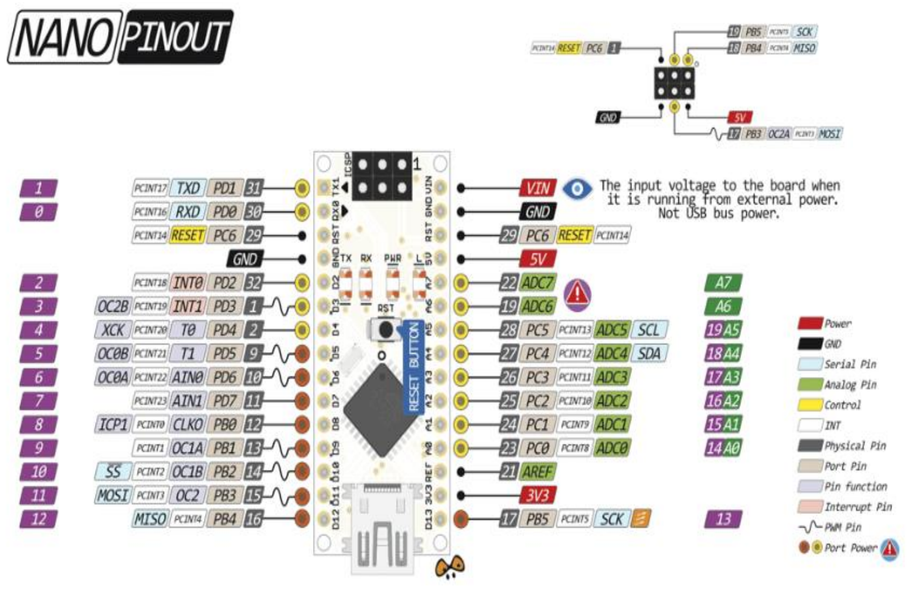


Fig 3.1 N ARDUINO NANO

### 3.2.1Arduino Nano Pin Configuration

| **Pin Category** | **Pin Name** | **Details** |
|---|---|---|
| Power | **Vin, 3.3V, 5V, GND** | **Vin:** Input voltage to Arduino when using an external power source (6-12V)<br>.**5V:** Regulated power supply used to power microcontroller and other components on the board.**3.3V:**<br>3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.<br>**GND:** Ground pins. |
| Reset | **Reset** | Resets the microcontroller. |
| Analog Pins | **A0 – A7** | Used to measure analog voltage in the range of 0-5V |
| Input/Output Pins | **Digital Pins D0 - D13** | Can be used as input or output pins. 0V (low) and 5V (high) |
| Serial | Rx, **Tx** | Used to receive and transmit TTL serial data. |
| External Interrupts | 2, 3 | To trigger an interrupt. |
| PWM | 3, 5, 6, 9, 11 | Provides 8-bit PWM output. |

TABLE 3.1 ARDUINO NANO PIN CONFIGURATION

## 3.2.2 ARDUINO NANO TECHNICAL SPECIFICATIONS

| | |
|---|---|
| Microcontroller | ATmega328P – 8 bit AVR family microcontroller |
| Operating Voltage | 5V |
| Recommended Input Voltage for Vin pin | 7-12V |
| Analog Input Pins | 6 (A0 – A5) |
| Digital I/O Pins | 14 (Out of which 6 provide PWM output) |
| DC Current on I/O Pins | 40 mA |
| DC Current on 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (2 KB is used for Bootloader) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Frequency (Clock Speed) | 16 MHz |
| Communication | IIC, SPI, USART |

**Table 3.2 Arduino nano technical specifications**

## 3.3 WIND SPEED METER (ANEMOMETER)

Anemometer is a device used to measure wind speed and considered an effective weather station instrument. Anemometer is designed to be equipped outdoor in order to efficiently transfer the signal along to the microcontroller [2, 6]. In fact, most of the research papers and the projects, do not implement such a sensitive element as the proposed wind speed sensor. This sensor is an ultimate element that integrates weather station attributes. In this work, wind speed in a local specific location is measured, stored, and demonstrated directly on the proposed LCD display. Moreover, the Arduino code is proposed to show up the maximum wind speed measured lastly such that if the wind blown up hardly in a specific moment, the maximum value of the wind that time is stored and shown in the part assigned to measure the

peak point of the wind. The peak value of the wind differs from time to time depending on the maximum wind strength that is proposed to be stored in the proposed database system, and shown alongside over the display as long as there exists no greater read than the last value. The working principle of the proposed wind speed sensor depends considerably on the construction of this tidy anemometer.

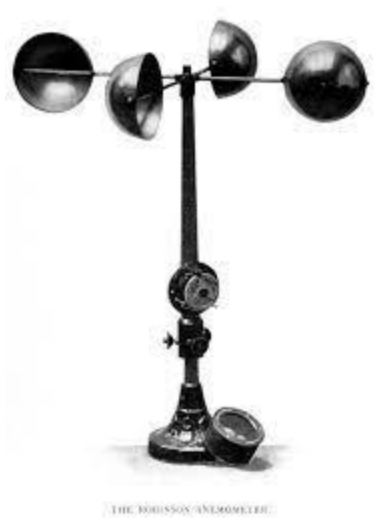


**Fig.3.3. Interior Structure of the Anemometer**

The output terminals of the anemometer consist of three wires identified by Vcc, GND, and Signal wire. As an important point, the proposed anemometer needs external separated power (battery) in a range of (9 – 12) Volt.

## 3.4 SIM808 / GMS/GPS- MODULE

SIM808 GPS Tracker is an IOT (Internet of things) Solution based on the ATmega328 and GPRS/GSM GPS module SIM808. it intergrate a micro Controller Atmega 328, GRRS/GSM module SIM808, which is the upgrade version of SIM900, power management and storage, to make the SIM808 GPS Tracker ready for real project for IOT projects such as smart-home, outdoor monitoring, shared bicycle, etc. The SIM808 GPS Tracker based on the Arduino, users can program it with Arduino IDE, which is very easy especially suit for the none-programmers. There are also guide for users to learn how to create the first IOT project with this board, with which the starters can learn the hardware and programming skill quickly. Features: λ BAT Input Voltage: 3.4-4.2V λ ATmega328: 8MHz, 32KB flash, 2KB SRAM λ Micro SIM connector λ Integrated Power Control System λ AT command interface with "auto baud" detection Quad-band: 850/900/1800/1900Mz λ Send and receive GPRS data (TCP/IP, HTTP, etc.) λ GPS L1 C/A code λ 22 tracking /66 acquisition channels λ Tracking: -165 dBm λ Cold starts: -148 dBm λ Time-To-First-Fix：Cold starts-32s (typ.), Hot starts-1s (typ.)， Warm starts-5s (typ.) λ Accuracy: approx. 2.5 meters channel@makerfabs.com www.makerfabs.com λ Interface: I2C/SPI/UART/18*GPIO λ Arduino compatible λ Working Temperature: -40 – 85℃ λ Default baud rate: 115200 λ Size: 40*55mm

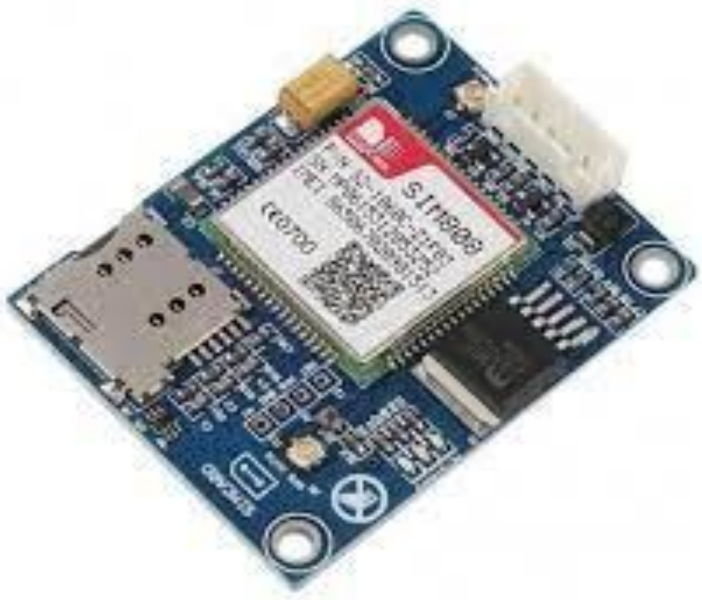


Fig.3.4. SIM80

## 3.5 BRIEF DESCRIPTION ON LCD MODULES

LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability and programmer friendly. Most of us would have come across these displays in our day to day life, either at PCO's or calculators. The appearance and the pinouts have already been visualized above now let us get a bit technical.

**16×2 LCD** is named so because; it has 16 Columns and 2 Rows. There are a lot of combinations available like, 8×1, 8×2, 10×2, 16×1, etc. but the most used one is the 16×2 LCD. So, it will have (16×2=32) 32 characters in total and each character will be made of 5×8 Pixel Dots. A Single character with all its Pixels is shown in the below picture.

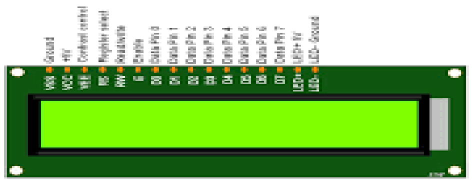


Fig.3.5 16x2 LCD

Now, we know that each character has (5×8=40) 40 Pixels and for 32 Characters we will have (32×40) 1280 Pixels. Further, the LCD should also be instructed about the Position of the Pixels. Hence it will be a hectic task to handle everything with the help of MCU, hence an **Interface IC like HD44780**is used, which is mounted on the backside of the LCD Module itself. The function of this IC is to get the **Commands and Data** from the MCU and process them to display meaningful information onto our LCD Screen. You can learn how to interface an LCD using the above mentioned links. If you are an advanced programmer and would like to create your own library for interfacing your Microcontroller with this LCD module then you have to understand the HD44780 IC is working and commands which can be found its datasheet.

## 3.6 RELAY SWITCH

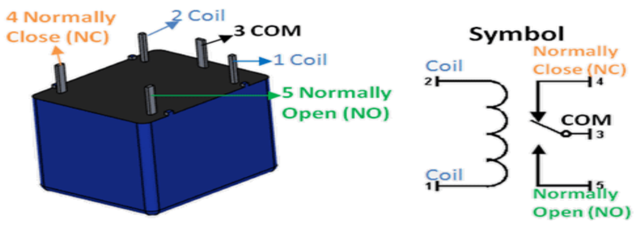


Fig 3.6 **RELAY PIN OUT**

## PIN CONFIGURATION

| Pin Number | Pin Name | Description |
|---|---|---|
| 1 | Coil End 1 | Used to trigger(On/Off) the Relay, Normally one end is connected to 12V and the other end to ground |
| 2 | Coil End 2 | Used to trigger(On/Off) the Relay, Normally one end is connected to 12V and the other end to ground |
| 3 | Common (COM) | Common is connected to one End of the Load that is to be controlled |
| 4 | Normally Close (NC) | The other end of the load is either connected to NO or NC. If connected to NC the load remains connected before trigger |
| 5 | Normally Open (NO) | The other end of the load is either connected to NO or NC. If connected to NO the load remains disconnected before trigger |

## TABLE 3.3 RELAY PIN DESCRIPATION

### 3.6.1 HOW TO USE A RELAY?

Relays are most commonly used switching device in electronics. There are two important parameters of relay, first is the Trigger Voltage, this is the voltage required to turn on the relay that is to change the contact from Common → NC to Common → NO. The other parameter is your Load Voltage & Current, this is the amount of voltage or current that the NC, NO or Common terminal of the relay could withstand, in our case for DC it is maximum of 30V and 10A. Make sure the load you are using falls into this range. The above diagram is for **relay triggering circuit.** Since the relay has 12V trigger voltage we have used a +12V DC supply to one end of the coil and the other end to ground through a switch. For switching we are using a transistor as a switching device. You can also notice a diode connected across the coil of the relay, this diode is called the Fly back Diode. The purpose of the diode is to protect the switch from high voltage spike that can produced by the relay coil. As shown one end of the load can be connected to the Common pin and the other end is either connected to NO or NC. If connected to NO the load remains disconnected before trigger and if connected to NC the load remains connected before trigger

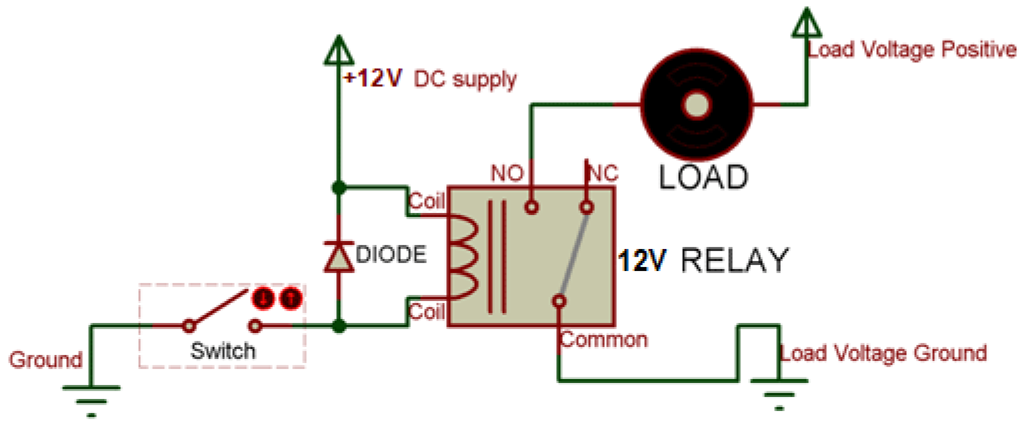


Fig 3.6.1 **RELAY CONNECTION**

## 3.7 PUSH BUTTON

Fig 3.7.1 PUSH BUTTON

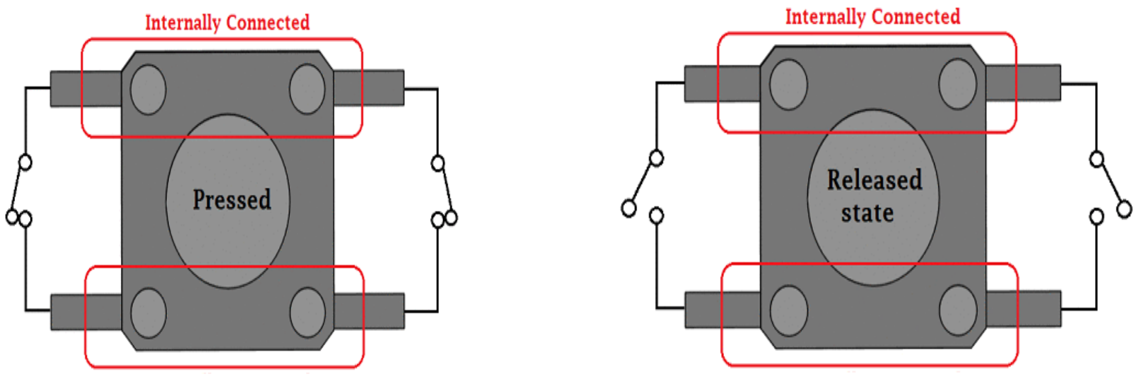


**FIG 3.7.2 PUSH BUTTON LOGIC DIAGRAM**

Push-Buttons are normally-open **tactile switches**. Push buttons allow us to power the circuit or make any particular connection only when we press the button. Simply, it makes the circuit connected when pressed and breaks when released. A push button is also used for triggering of the SCR by gate terminal. These are the most common buttons which we see in our daily life electronic equipment's. Some of the applications of the Push button are mentioned at the end of the article.

## FEATURES

- Prevent flux rise by the insert-molded terminal
- Snap-in mount terminal
- Contact Bounce: MAX 5mS
- Crisp clicking by tactile feedback

- Dielectric Withstanding Voltage 250V AC for 1 minute

## APPLICATIONS

- Calculators
- Push-button telephones
- Kitchen appliances
- Magnetic locks
- Various other mechanical and electronic devices, home and commercials.

## 3.8 POWER MODULE

In today's world most of the electronic appliances needs DC voltages from 3-12 Volts for operation. An adaptor is a device which is used to convert high AC voltages to low DC voltage. Adaptors come in various shapes, sizes & configuration depending on the use.

Current rating: Since you'll probably be connecting other things to the Arduino (LEDs, LCDs, servos) you should get an adapter that can supply at least 500mA, or even 1000 mA (1 amphere). That way you can be sure you have enough juice to make each component of the circuit function reliably

In some cases adaptors are used to provide power to electronic appliances like video games modems etc and in some cases they are used to charge the primary battery of the electronic device and also act as 1alternate power source like mobile chargers, laptop chargers, cell chargers etc. A typical internal diagram of adaptor is shown in the figure below

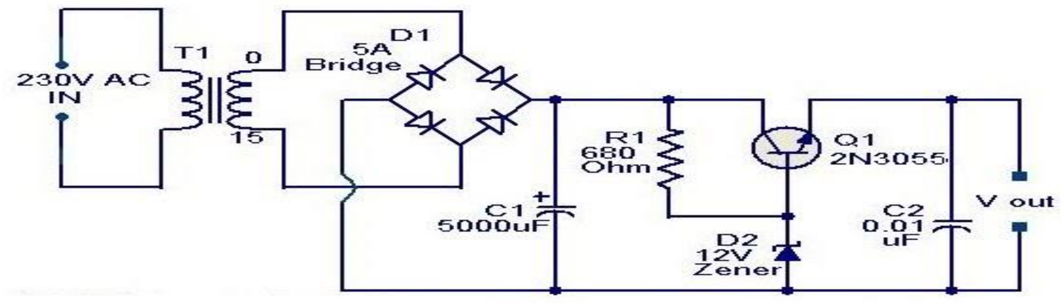


**Fig 3.8 INTERNAL DIAGRAM OF POWER MODULE**

An adaptor works in these 3 simple steps:

1. AC voltage is stepped down using a Transformer

2. A rectifier circuit changes the AC signal to DC signal.

3. A capacitor filters the signal to smooth DC waveform.

## TRANSFORMER

Usually, DC voltages are required to operate various electronic equipment and these voltages are 5V, 9V or 12V. But these voltages cannot be obtained directly. Thus the a.c input available at the mains supply i.e., 230V is to be brought down to the required voltage level. This is done by a transformer. Thus, a step down transformer is employed to decrease the voltage to a required level.

## RECTIFIER

The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification.

## FILTER

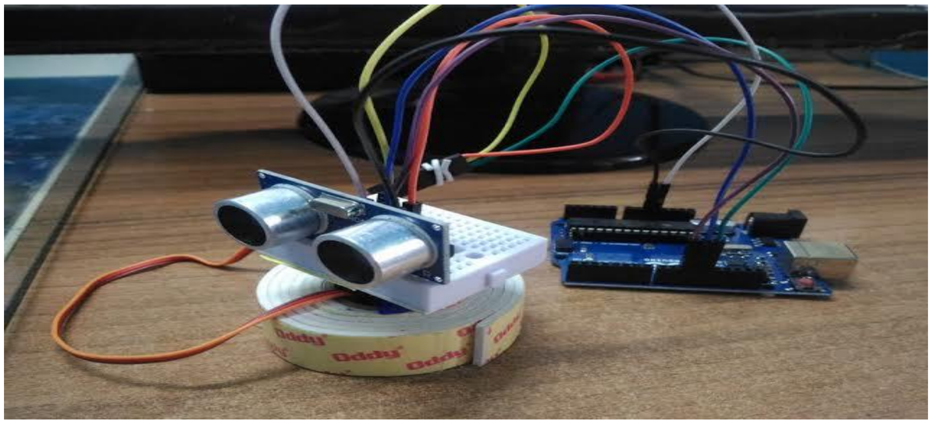
Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied, D.C. voltage received at this point changes. Therefore a regulator is applied at the output stage

## VOLTAGE REGULATOR (7805)

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power supply of 5V and 12V are required. In order to obtain these voltage levels, 7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers05,12 represent the required output voltage level

## 3.9 ULTRASONIC SENSOR RADOR

The authors propose a radar made with the ultrasonic sensor type HC - SR04, developed for Arduino. The novelty added by this paper is the development of specific libraries aiming to interface the sensor produced for Arduino with the microcontroller PIC 18F, manufactured by Microchip. We have taken into account the temperature influence on the ultrasound. The radar can detect objects located at distances between 2 cm and 4 meters **Fig**



**3.9 ultrasonic sensor rador**

The object detection is performed inside an angle of 180 degrees. The motion sensor that covers this angle uses a stepper motor. Data are presented on a LCD display, the motor movement being synchronized with the browsing of the radar screen. Synchronization is made by programming the timers of the microcontroller. These timers generate interruptions at preset moments, correlated with the completion of a specific area of the graphic display

## 3.10 SOFTWARE INSTALLATION

After learning about the main parts of the Arduino UNO board, we are ready to learn how to set up the Arduino IDE. Once we learn this, we will be ready to upload our program on Arduino board.In this section, we will learn in easy steps, how to set up the Arduino IDE on our computer and prepare the board to receive the program via USB cable.

**Step 1** − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image

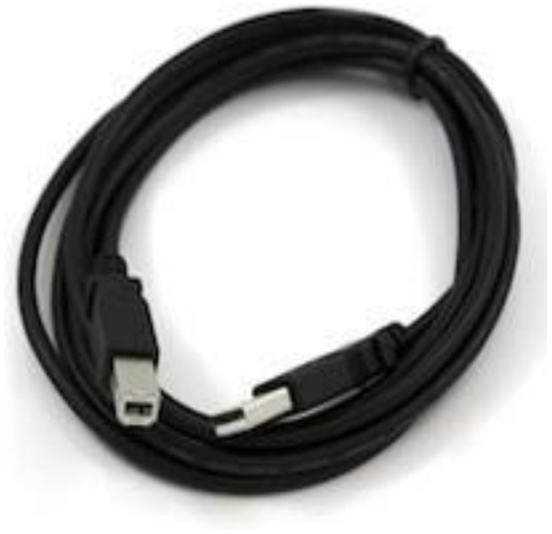


**Fig 3.9USB 2.0 CABLE TYPE A/B**

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

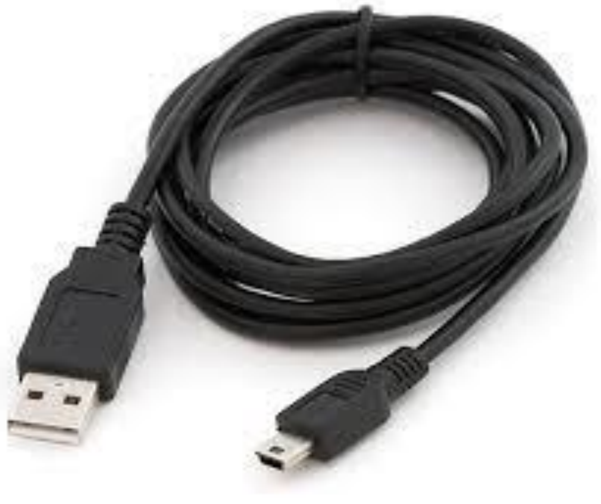
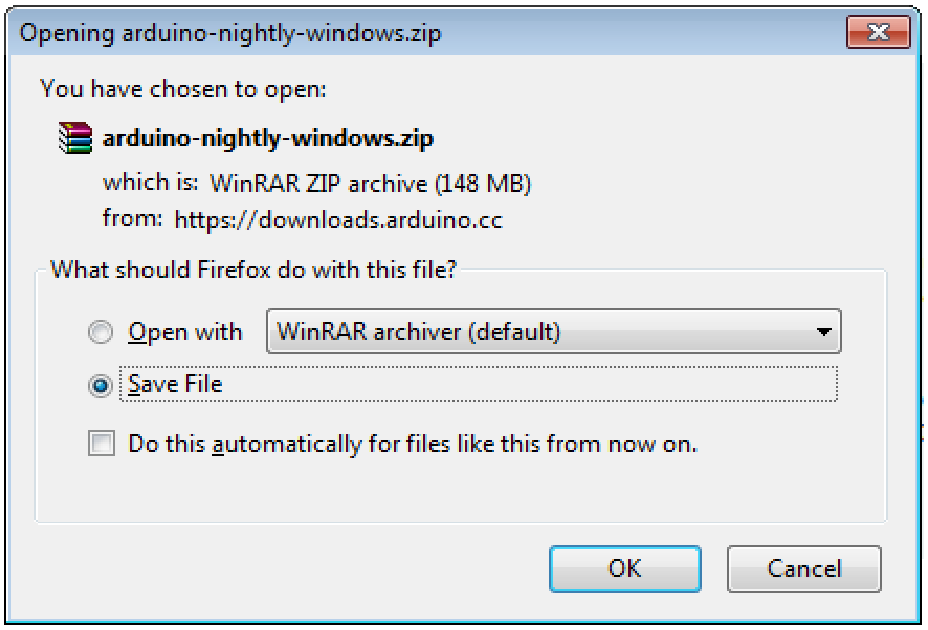


**Fig 3.10 USB 2.0A TO USB 2 MINI B**

## Step 2 − Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.



Opening arduino-nightly-windows.zip

You have chosen to open:

**arduino-nightly-windows.zip**

which is: WinRAR ZIP archive (148 MB)

from: https://downloads.arduino.cc

What should Firefox do with this file?

( ) Open with WinRAR archiver (default)

[✓ Save File]

[ ] Do this automatically for files like this from now on.

**Fig 3.11 DOWNLOADING ARDUINO SOFTWARE**

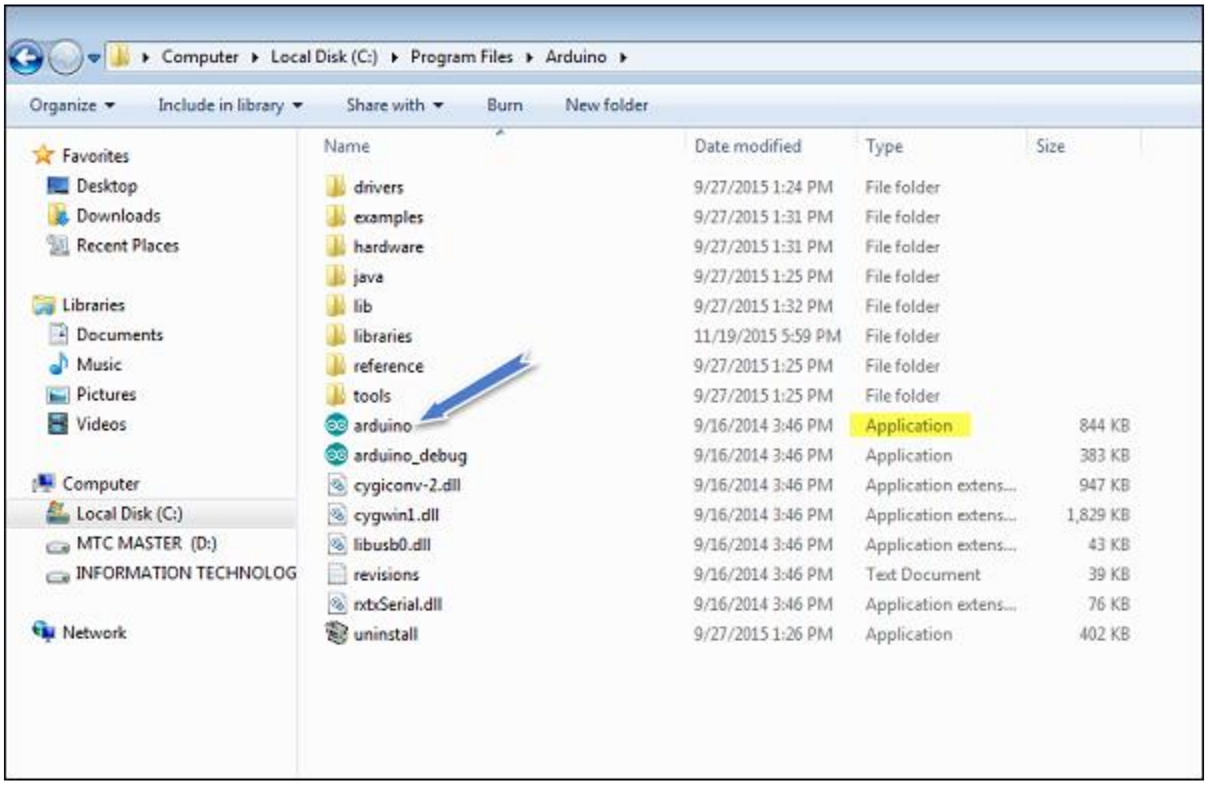
## Step 3 − Power up your board.

The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

## STEP 4 − LAUNCH ARDUINO IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



| Name | Date modified | Type | Size |
|---|---|---|---|
| drivers | 9/27/2015 1:24 PM | File folder | |
| examples | 9/27/2015 1:31 PM | File folder | |
| hardware | 9/27/2015 1:31 PM | File folder | |
| java | 9/27/2015 1:25 PM | File folder | |
| lib | 9/27/2015 1:32 PM | File folder | |
| libraries | 11/19/2015 5:59 PM | File folder | |
| reference | 9/27/2015 1:25 PM | File folder | |
| tools | 9/27/2015 1:25 PM | File folder | |
| arduino | 9/16/2014 3:46 PM | Application | 844 KB |
| arduino_debug | 9/16/2014 3:46 PM | Application | 383 KB |
| cygiconv-2.dll | 9/16/2014 3:46 PM | Application extens... | 947 KB |
| cygwin1.dll | 9/16/2014 3:46 PM | Application extens... | 1,829 KB |
| libusb0.dll | 9/16/2014 3:46 PM | Application extens... | 43 KB |
| revisions | 9/16/2014 3:46 PM | Text Document | 39 KB |
| rxtxSerial.dll | 9/16/2014 3:46 PM | Application extens... | 76 KB |
| uninstall | 9/27/2015 1:26 PM | Application | 402 KB |

**Fig 3.12 INSTALLATION OF ARDUINO IDE**

## Step 5 − Open your first project.

Once the software starts, you have two options −

- Create a new project.
- Open an existing project example.

To create a new project, select File → **New** . To open an existing project example, select File → Example → Basics → Blink .Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list

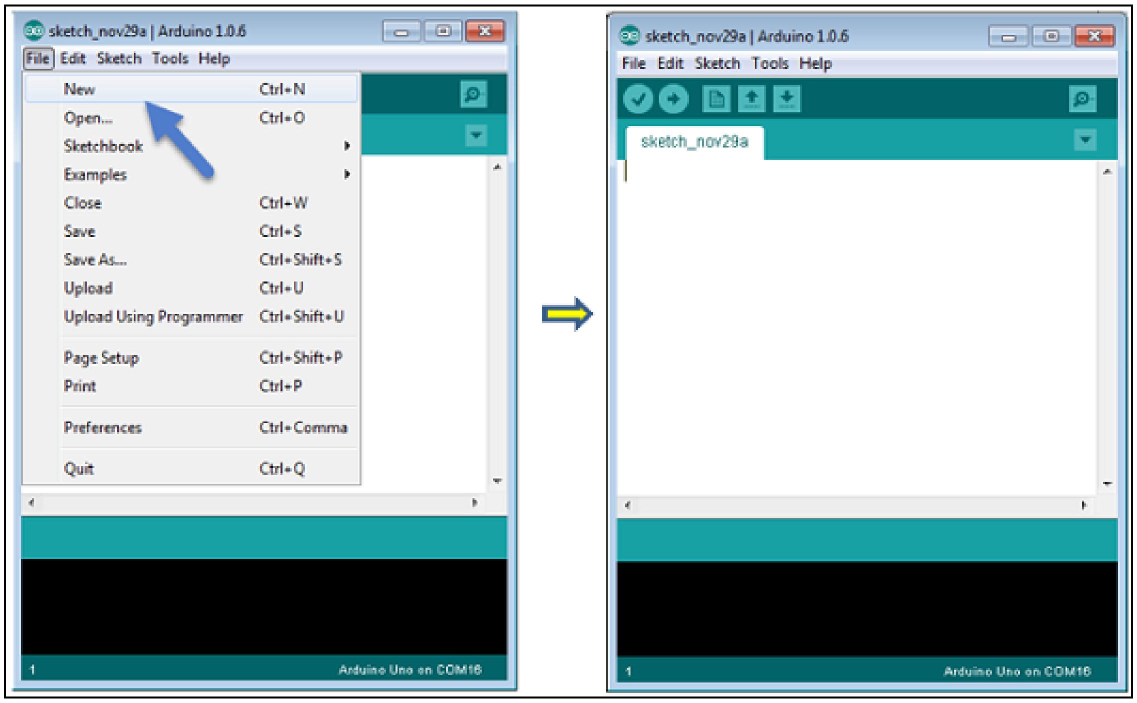
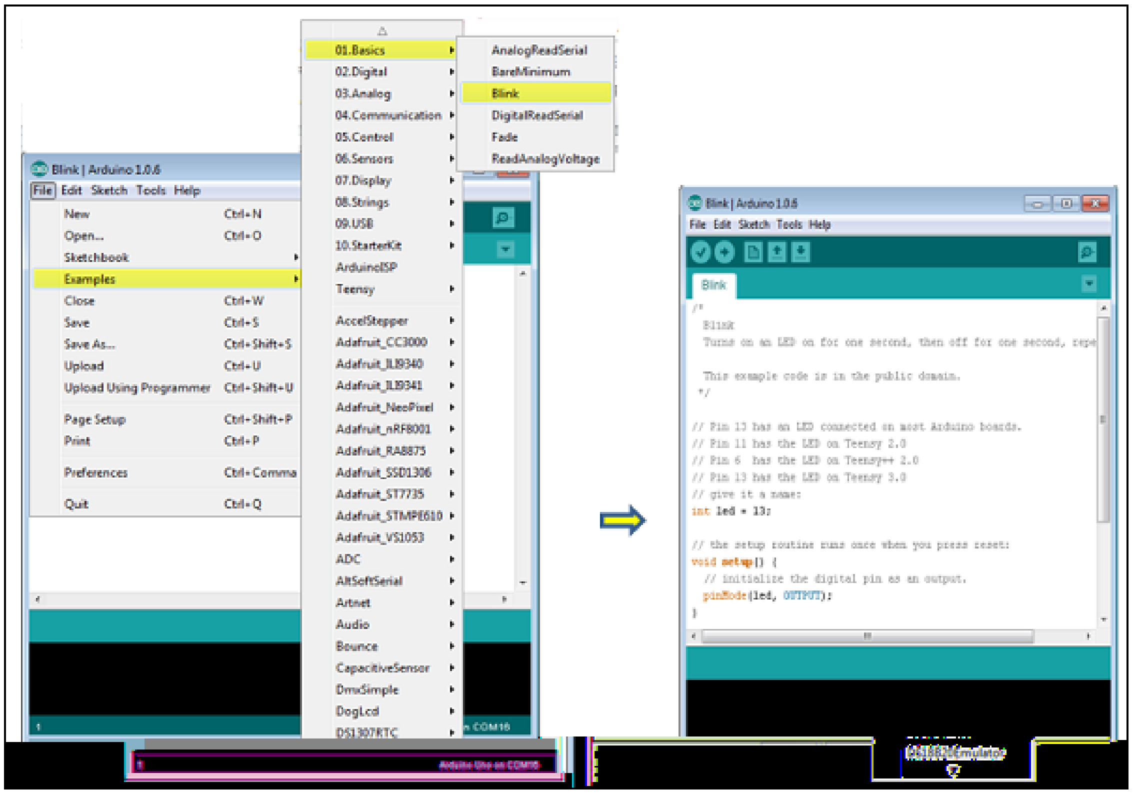


**Fig 3.13 CREATEING NEW PROJECT**



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

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

**Fig 3.14 RUN EXAMPLE PROGRAM**

## STEP 6 – SELECT YOUR ARDUINO BOARD.

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools → Board and select your board.

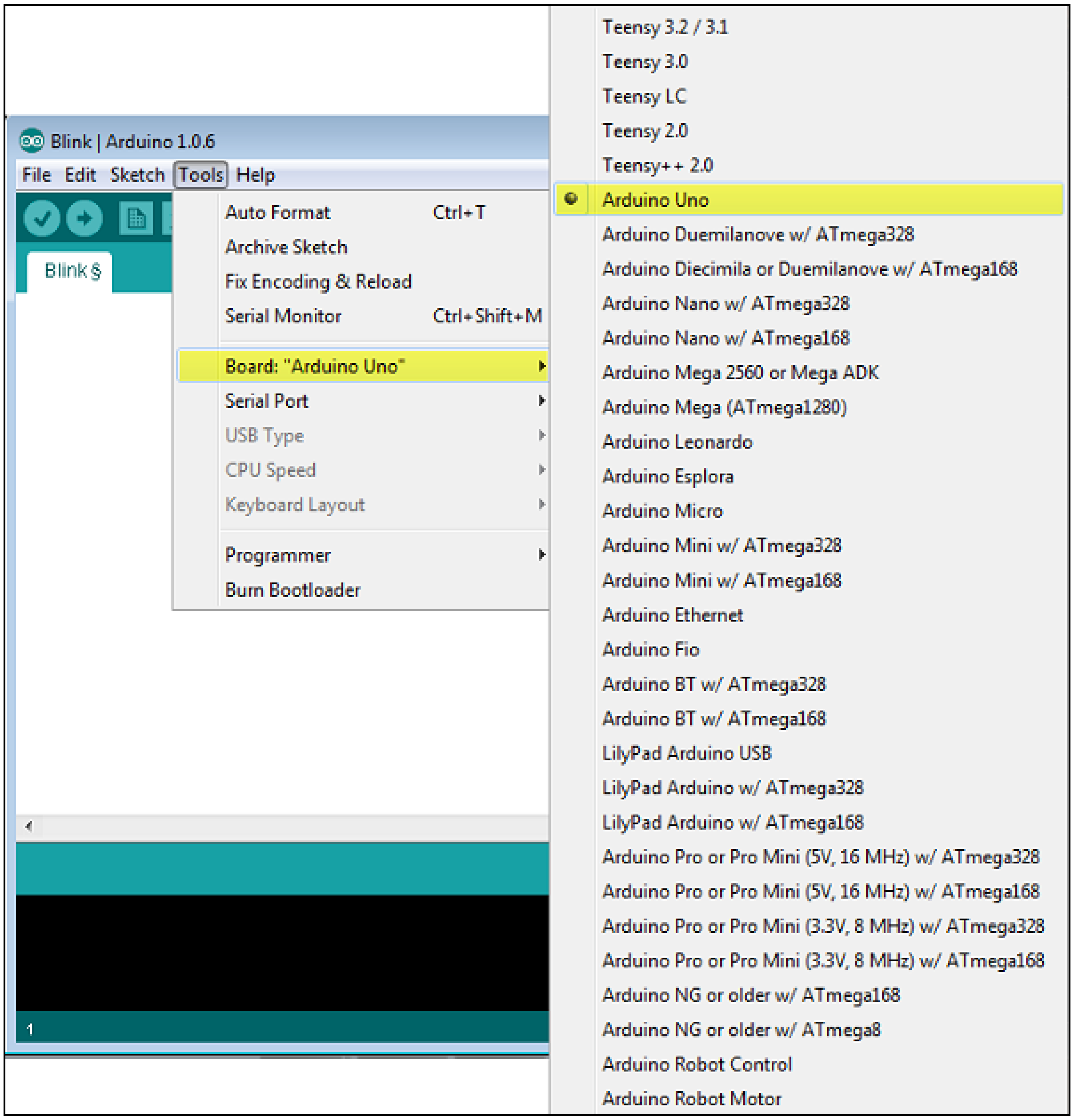

Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
1
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

**Fig 3.15ARDUINO SELECTION IN IDE**

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

## STEP 7 – SELECT YOUR SERIAL PORT.

Select the serial device of the Arduino board. Go to **Tools** → **Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

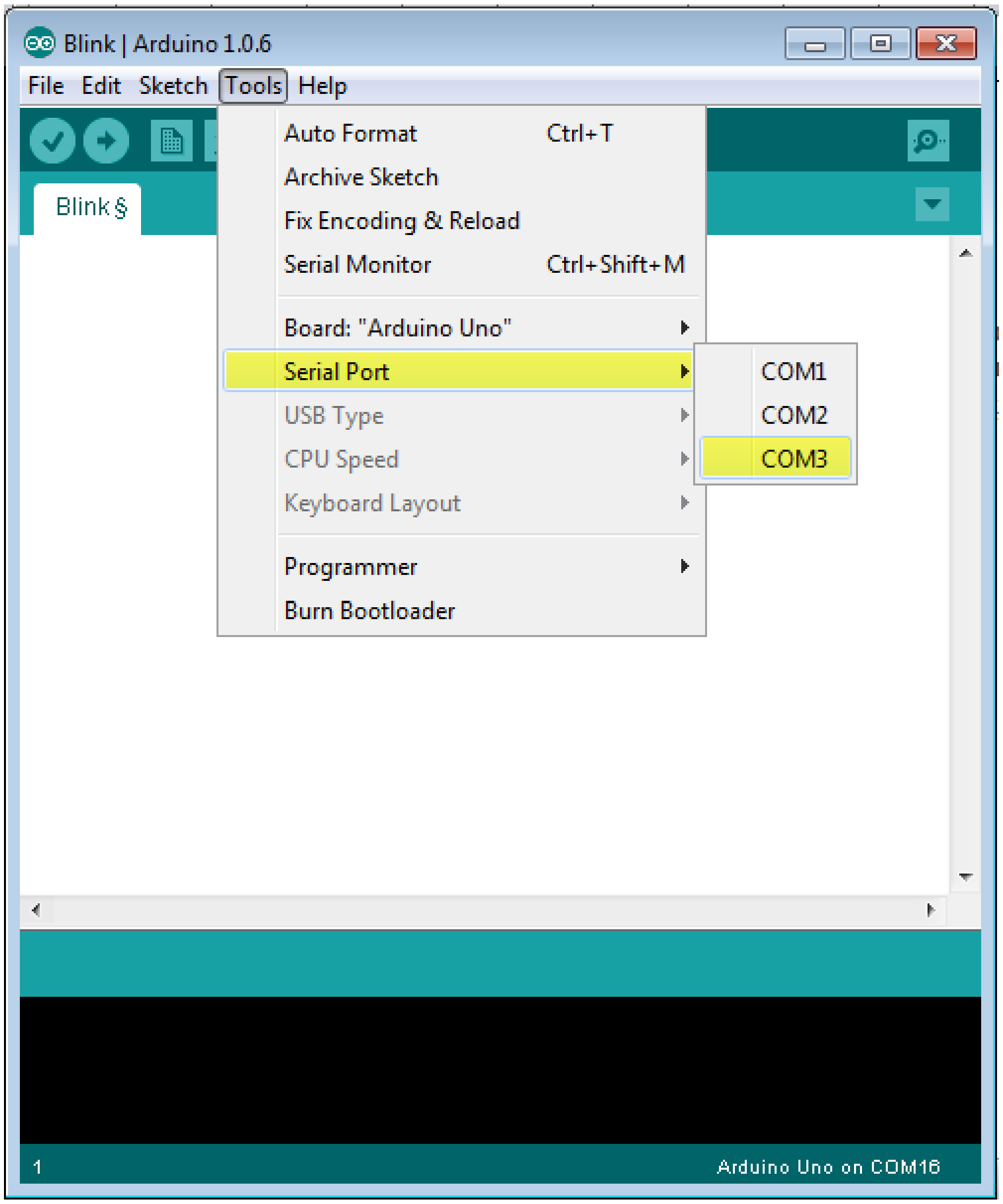


**Fig 3.16 SETTING PORT**

## STEP 8 − UPLOAD THE PROGRAM TO YOUR BOARD.

Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.

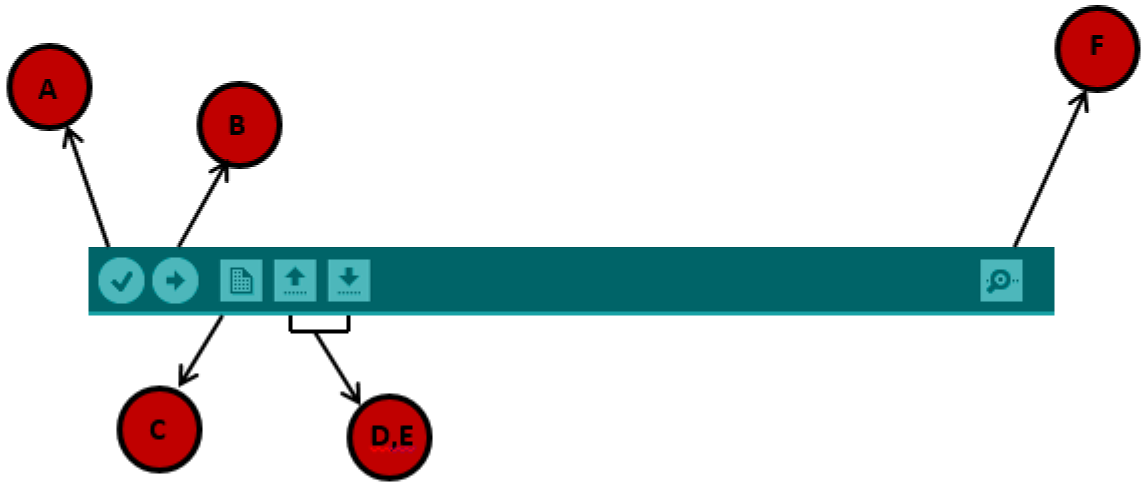


**Fig 3.17 STATUS BAR IN ARDUINO SOFTWARE**

**A** − Used to check if there is any compilation error.

**B** − Used to upload a program to the Arduino board.

**C** − Shortcut used to create a new sketch.

**D** − Used to directly open one of the example sketch.

**E** − Used to save your sketch.

**F** − Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

**Note** − If you have an Arduino Mini, NG, or other board, you need to press the reset button physically on the board, immediately before clicking the upload button on the Arduino Software.

# CHAPTER-4

# (IMPLEMENTATION ASPECTS)

## 4.1 IMPLEMENTATION OF THE BLOCK DIAGRAM

In this project we used the arduino board has heart of the project it is operated at 5V dc voltage so that we used the 5VDC power supply module and we interfaced with the wind sensor and overload sensors and SIM 808 module and 16X2 LCD display this hole circuit is enclosed with the pvc box

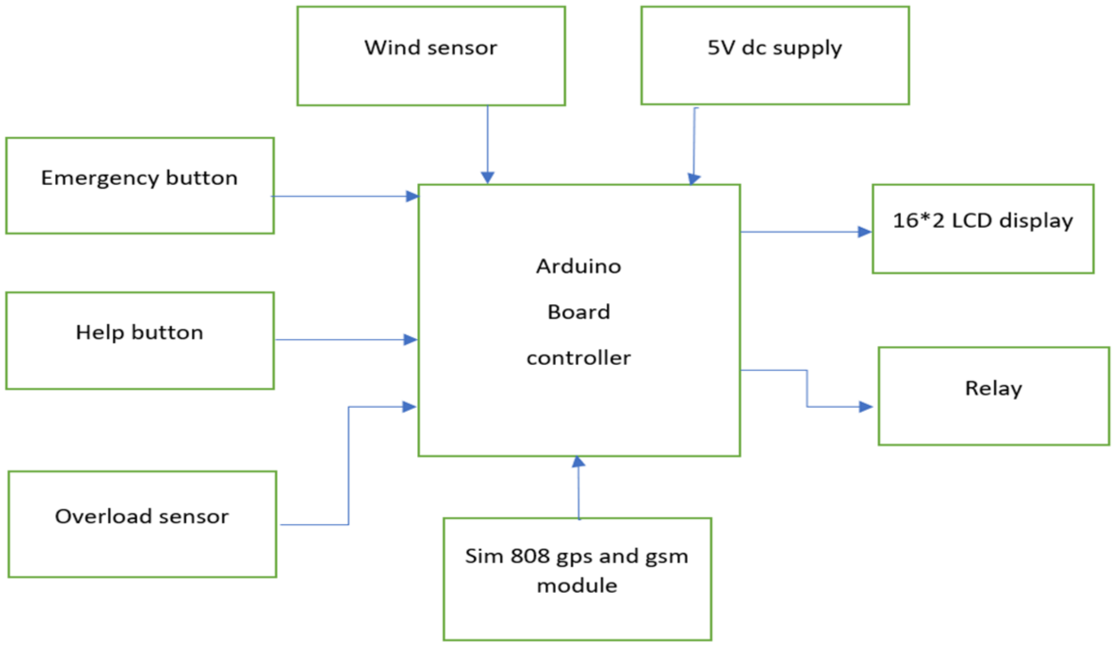


Fig 4.1 Functional Description

In this circuit have 3 button

1.Emergency button

2.Help button

3.Overload sensor

### 4.1. 1 WORKING OF EMERGENCY BUTTON:-

When we pressed the Emergency button then single is passes to arduino as per the code the single is transmitter to satellite by using GSM module after that single is transmitter to near Emergency station at location of boat then parallel display the hole operation in 16X2 LCD display in that time track the location of boat using GPS

## 4.1.2 WORKING OF HELP BUTTON :-

When we pressed the help button then single is passes to arduino as per the code the single is transmitter to satellite by using GSM module after that single is transmitter to near help station at location of boat then parallel display the hole operation in 16X2 LCD display in that time track the location of boat using GPS

## 4.1.3 WORKING OF OVERLOAD BUTTON :-

When the boat is overload based on the fishes weight then the boats is unstable position in that time overload sensor is activated then the single is passed to the arduino then the relay is switched to the normal closed to normal open in that time motor of the boat is off and it displayed the boat is overload and also monitor the whether condition and it is also have the wind sensor and it is displayed in display Identification of missing boat using GPS can be used to trace the exact location of boat, the message will be sent to the nearest load center the rescue team will be alerted and will protect the humans. This can be mainly operated in three ways

- Overload protection
- Over wind protection
- Lack of fuel indication
- Radar system

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.

### 4.1.4 APPLICATIONS

- A boat is designed to float on water and can be used for
- travel,recreation, sports, fishing, transportation, military use and for rescue operations.
- Boats have been used by humans since the earliest civilizations when they were simply made of logs and reeds tied together to make rafts.

### 4.1.5 ADVANTAGES

- Monitored by the exact location of the ships
- Monitored all the sensor immediately with no delay during accidents
- It can be used to being under the below water level by the radar system
- it can be limiting by 75 percentage of boat accidents
- human protection availability is more
- Sensors will be more accurate

## PROJECT PROGRAMME

```
#include <Wire.h>

#include <SoftwareSerial.h>

#include <LiquidCrystal_I2C.h> // Library for LCD

LiquidCrystal_I2C lcd(0x27, 16, 2);

SoftwareSerial sim808(10,11);

char phone_no1[] = "7032325943"; // replace with your phone no.

char phone_no[] = "9848689981"; // replace with your phone no.

String data[5];

#define DEBUG true

String state,timegps,latitude,longitude;

String textMessage;
```

```
const int relay = 16;

String motorState = "HIGH";

const int trigPin = 2;

const int echoPin = 3;

const int ledPin = 13;

long duration;

int distanceCm;

int safetyDistance;

//long duration;

//int distanceCm, distanceInch;

const int buttonPin = 8;

const int buttonPin1 = 9;

const int buttonPin2 = 12;// the number of the pushbutton pin

const int buzzer = 15;

int button1State = 0;     // variable for reading the pushbutton status

int button2State = 0;     // variable for reading the pushbutton status

int button3State = 0;

int serial_in; // variable for reading the pushbutton status

double x = 0;

double y = 0;

double a = 0;

double b = 0;
```

```
const int sensorPin = A0; //Defines the pin that the anemometer output is connected

const int numReadings = 2; //Defines number of reading to calculate average windspeed

int readings[numReadings];  // the readings from the analog input

int readIndex = 0;  // the index of the current reading

int totalWind= 0;   // the running total

int averageWind = 0;  // the average

int inputPin = A0;

int sensorValue = 0; //Variable stores the value direct from the analog pin

float sensorVoltage = 0; //Variable that stores the voltage (in Volts) from the anemometer being
sent to the analog pin

float sensorVoltage2 = 0; //Variable that stores the voltage (in Volts) from the anemometer being
sent to the analog pin

float windSpeed = 0; // Wind speed in meters per second (m/s)

float voltageConversionConstant = .004882814; //This constant map the value provided from the
analog read function, which ranges from 0 to 1023, to actual voltage, which ranges from 0V to
5V

int sensorDelay = 2000; //Delay between sensor readings, measured in milliseconds (ms)

//Anemometer Technical Variables

//The following variables correspond to the anemometer.

float voltageMin = .4; // Mininum output voltage from anemometer in mV.

float windSpeedMin = 0; // Wind speed in meters/sec corresponding to minimum voltage

float voltageMax = 2.0; // Maximum output voltage from anemometer in mV.

float windSpeedMax = 32; // Wind speed in meters/sec corresponding to maximum voltage float
duration, distance;
```

```
//Setup Variables

void setup()

{

  lcd.init();

  lcd.backlight();

sim808.begin(19200);

 Serial.begin(9600);

delay(50);

lcd.begin(16,2);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(ledPin, OUTPUT);

lcd.print(" Weclome To ");

lcd.setCursor(0,1);

lcd.print(" DNR college ");

delay(2000);

lcd.clear();

pinMode(buttonPin, INPUT);

pinMode(buttonPin1, INPUT);

pinMode(buttonPin2, INPUT);

pinMode(relay, OUTPUT);
```

```
Serial.println("SIM808 ready...");

sim808.print("AT+CSMP=17,167,0,0"); // set this parameter if empty SMS received

delay(100);

sim808.print("AT+CMGF=1\r");

delay(400);

sendData("AT+CGNSPWR=1",1000,DEBUG);

delay(50);

sendData("AT+CGNSSEQ=RMC",1000,DEBUG);

delay(150);

sim808.print("AT+CMGF=1\r");

delay(100);

// Set module to send SMS data to serial out upon receipt

sim808.print("AT+CNMI=2,2,0,0,0\r");

digitalWrite(relay, LOW);

//Setup LCD display with welcome screen

for (int thisReading = 0; thisReading < numReadings; thisReading++)

{

  readings[thisReading] = 0;

}

delay(2500);

lcd.clear();

//gpssms();
```

```
}
 void loop()
{
  lcd.begin(16,2);
 digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;
//distanceInch = duration*0.0133/2;
safetyDistance = distanceCm;
if (safetyDistance <= 20){
 digitalWrite(ledPin, HIGH);
 lcd.setCursor(0,1);
lcd.print("Object detected ");
}
else{
digitalWrite(ledPin, LOW);
}
lcd.setCursor(1,0);
```

```
Serial.print("Distance: ");lcd.print("Distance: ");

Serial.print(distanceCm);

Serial.print("Cm");

Serial.print("\t");

lcd.print(distanceCm);

lcd.print("Cm");

delay(3000);

lcd.clear();

//lcd.clear();

getgps();

sendData( "AT+CGNSINF",1000,DEBUG);

delay(1000);

button1State = digitalRead(buttonPin);

button2State = digitalRead(buttonPin1);

button3State = digitalRead(buttonPin2);

sensorValue = analogRead(sensorPin);

if (button1State == HIGH)

{

  Serial.println("button1");

//lcd.begin(0, 0); lcd.print("");

lcd.clear();

lcd.setCursor(0, 1);
lcd.print("help me");

gpssms();

gpssms3();

delay(2000);

lcd.clear();

}

if (button2State == HIGH)

{

  Serial.println("button2");

lcd.clear();

lcd.setCursor(0, 1);

lcd.print("emergency");

gpssms1();

gpssms4();

delay(2000);

lcd.clear();

}

if (button3State == HIGH)

{

  Serial.println("button3");

lcd.clear();

digitalWrite(relay, HIGH);
```

```
lcd.setCursor(0, 1);

lcd.print("Boat Over Load");

gpssms2();

gpssms5();

delay(2000);

lcd.clear();

}

 if(sim808.available()>0)

{ textMessage = sim808.readString();

Serial.print(textMessage);

delay(10);

}

if(textMessage.indexOf("HI")>=0){

// Turn on relay and save current state gpssms();

delay(200);

digitalWrite(relay, HIGH);

motorState = "ON";

Serial.println("MOTOR IS ON");

textMessage = "";

lcd.clear();
```

```
}

else{

digitalWrite(relay, LOW);

digitalWrite(buzzer, LOW);

totalWind = totalWind - readings[readIndex];

// read from the sensor:

readings[readIndex] = sensorValue;

// add the reading to the total:

totalWind = totalWind + readings[readIndex];

// advance to the next position in the array:

readIndex = readIndex + 1;

sensorVoltage2 = sensorValue * voltageConversionConstant; //Convert sensor value to actual
voltage

// if we're at the end of the array...

if (readIndex >= numReadings)

{

 // ...wrap around to the beginning:

readIndex = 0;

// calculate the average:

averageWind = totalWind / numReadings;

sensorVoltage = averageWind * voltageConversionConstant; //Convert sensor value to actual
voltage
//Convert voltage value to wind speed using range of max and min voltages and wind speed for
the anemometer

if (sensorVoltage <= voltageMin) {

windSpeed = 0; //Check if voltage is below minimum value. If so, set wind speed to zero.

}

else {

windSpeed = ((sensorVoltage - voltageMin) * windSpeedMax / (voltageMax -
voltageMin))*2.232694;

//For voltages above minimum value, use the linear relationship to calculate wind speed.

}

}

//Print voltage and windspeed to serial

Serial.print("Voltage: ");

Serial.print(sensorVoltage);

Serial.print("Average: ");

Serial.print(averageWind);

Serial.print("\t");

Serial.print("Wind speed: ");

Serial.println(windSpeed);

//Display Wind Speed results to LCD with Max wind speed

lcd.setCursor(0, 0);

lcd.print("Wind Speed:");

lcd.print(windSpeed);
lcd.print("km/h");

//lcd.setCursor(11, 0);

//lcd.print("windSpeed");

delay(1000);

lcd.clear();

//lcd.setCursor(0, 1);

 //lcd.print(b);

//lcd.setCursor(5, 1);

//lcd.print(readIndex);

// lcd.setCursor(7, 1);

// lcd.print("Max=");

//lcd.setCursor(11, 1);

// lcd.print(y); delay(sensorDelay); lcd.clear();

}

}

void gpssms()

{

 sendTabData("AT+CGNSINF",1000,DEBUG);

if (state !=0)

{

  Serial.println("State :"+state);

Serial.println("Time :"+timegps);
Serial.println("Latitude :"+latitude);

Serial.println("Longitude :"+longitude);

sim808.print("AT+CMGS=\"");

sim808.print(phone_no);

sim808.println("\"");

delay(300);

sim808.print("Need Help ");

sim808.print("http://maps.google.com/maps?q=loc:");

sim808.print(latitude);

sim808.print(",");

sim808.print (longitude);

 delay(200);

sim808.println((char)26); // End AT command with a ^Z, ASCII code 26 delay(200);

sim808.println();

delay(20000);

sim808.flush();

} else {

Serial.println("GPS Initialising...");


}

}
```

```
void gpssms1()


{
  sendTabData("AT+CGNSINF",1000,DEBUG);
if (state !=0)
{
  Serial.println("State :"+state);
Serial.println("Time :"+timegps);
Serial.println("Latitude :"+latitude);
Serial.println("Longitude :"+longitude);
sim808.print("AT+CMGS=\"");
sim808.print(phone_no);
sim808.println("\"");
delay(300);
sim808.print("emergency");
sim808.print("http://maps.google.com/maps?q=loc:");
sim808.print(latitude);
sim808.print(",");
sim808.print (longitude);
delay(200);
 sim808.println((char)26); // End AT command with a ^Z, ASCII code 26 delay(200);
sim808.println();
```

```
delay(20000);
sim808.flush();
} else {
Serial.println("GPS Initialising...");
}
}
void gpssms2()
{ sendTabData("AT+CGNSINF",1000,DEBUG);
if (state !=0)
{
  Serial.println("State :"+state);
Serial.println("Time :"+timegps);
Serial.println("Latitude :"+latitude);
Serial.println("Longitude :"+longitude);
sim808.print("AT+CMGS=\"");
sim808.print(phone_no);
sim808.println("\"");
delay(300);
sim808.print("Boat Overload");
sim808.print("http://maps.google.com/maps?q=loc:");
sim808.print(latitude);
sim808.print(",");
```

```
sim808.print (longitude);

delay(200);

sim808.println((char)26); // End AT command with a ^Z, ASCII code 26

 delay(200);

sim808.println();

delay(20000);

sim808.flush();

} else {

Serial.println("GPS Initialising...");



}

}

void gpssms3()

{

  sendTabData("AT+CGNSINF",1000,DEBUG);

if (state !=0)

{ Serial.println("State :"+state);

Serial.println("Time :"+timegps);

Serial.println("Latitude :"+latitude);

Serial.println("Longitude :"+longitude);

sim808.print("AT+CMGS=\"");

sim808.print(phone_no1);
```

```
sim808.println("\"");
delay(300);
sim808.print("Need Help ");
sim808.print("http://maps.google.com/maps?q=loc:");
sim808.print(latitude);
sim808.print(",");
sim808.print (longitude); delay(200);
sim808.println((char)26); // End AT command with a ^Z, ASCII code 26 delay(200);
 sim808.println();
delay(20000);
sim808.flush();
} else {
Serial.println("GPS Initialising...");
}
}
void gpssms4()
{
  sendTabData("AT+CGNSINF",1000,DEBUG);
if (state !=0)
{
  Serial.println("State :"+state);
Serial.println("Time :"+timegps);
Serial.println("Latitude :"+latitude);

Serial.println("Longitude :"+longitude);

sim808.print("AT+CMGS=\"");

sim808.print(phone_no1);

sim808.println("\"");

delay(300);

sim808.print("emergency");

sim808.print("http://maps.google.com/maps?q=loc:");

sim808.print(latitude);

sim808.print(",");

sim808.print (longitude); delay(200);

sim808.println((char)26); // End AT command with a ^Z, ASCII code 26 delay(200);

sim808.println();

 delay(20000);

sim808.flush();

} else {

Serial.println("GPS Initialising...");

}

}

void gpssms5()


{
  sendTabData("AT+CGNSINF",1000,DEBUG);
if (state !=0)
{
  Serial.println("State :"+state);
Serial.println("Time :"+timegps);
Serial.println("Latitude :"+latitude);
Serial.println("Longitude :"+longitude);
sim808.print("AT+CMGS=\"");
sim808.print(phone_no1);
sim808.println("\"");
delay(300);
sim808.print("Boat Overload");
sim808.print("http://maps.google.com/maps?q=loc:");
sim808.print(latitude);
sim808.print(",");
sim808.print (longitude);
delay(200);
sim808.println((char)26); // End AT command with a ^Z, ASCII code 26 delay(200);
sim808.println();
delay(20000);
 sim808.flush();
} else {
```

```
Serial.println("GPS Initialising...");


}

}

void sendTabData(String command , const int timeout , boolean debug)

{

  sim808.println(command);

long int time = millis();

int i = 0;

while((time+timeout) > millis())

{

  while(sim808.available()){

char c = sim808.read();

if (c != ',')

{

data[i] +=c;

delay(100);

}

else { i++;

}


if (i == 5)
{
  delay(100);
  goto exitL;
}
}
}
exitL:
if (debug)
{
  state = data[1];
timegps = data[2];
latitude = data[3];
 longitude =data[4];
}
}
void getgps(void)
{
sendData( "AT+CGNSPWR=1",1000,DEBUG);
sendData( "AT+CGNSSEQ=RMC",1000,DEBUG);
}
String sendData(String command, const int timeout, boolean debug)
```

```
{

String response = "";

sim808.println(command);

long int time = millis();

while( (time+timeout) > millis())

{

while(sim808.available())

{

char c = sim808.read(); response+=c;

}

}

if(debug)

{

Serial.print(response);

}

return response;

}
```

## 4.2 WIRING DIAGRAM OF THE CIRCUIT

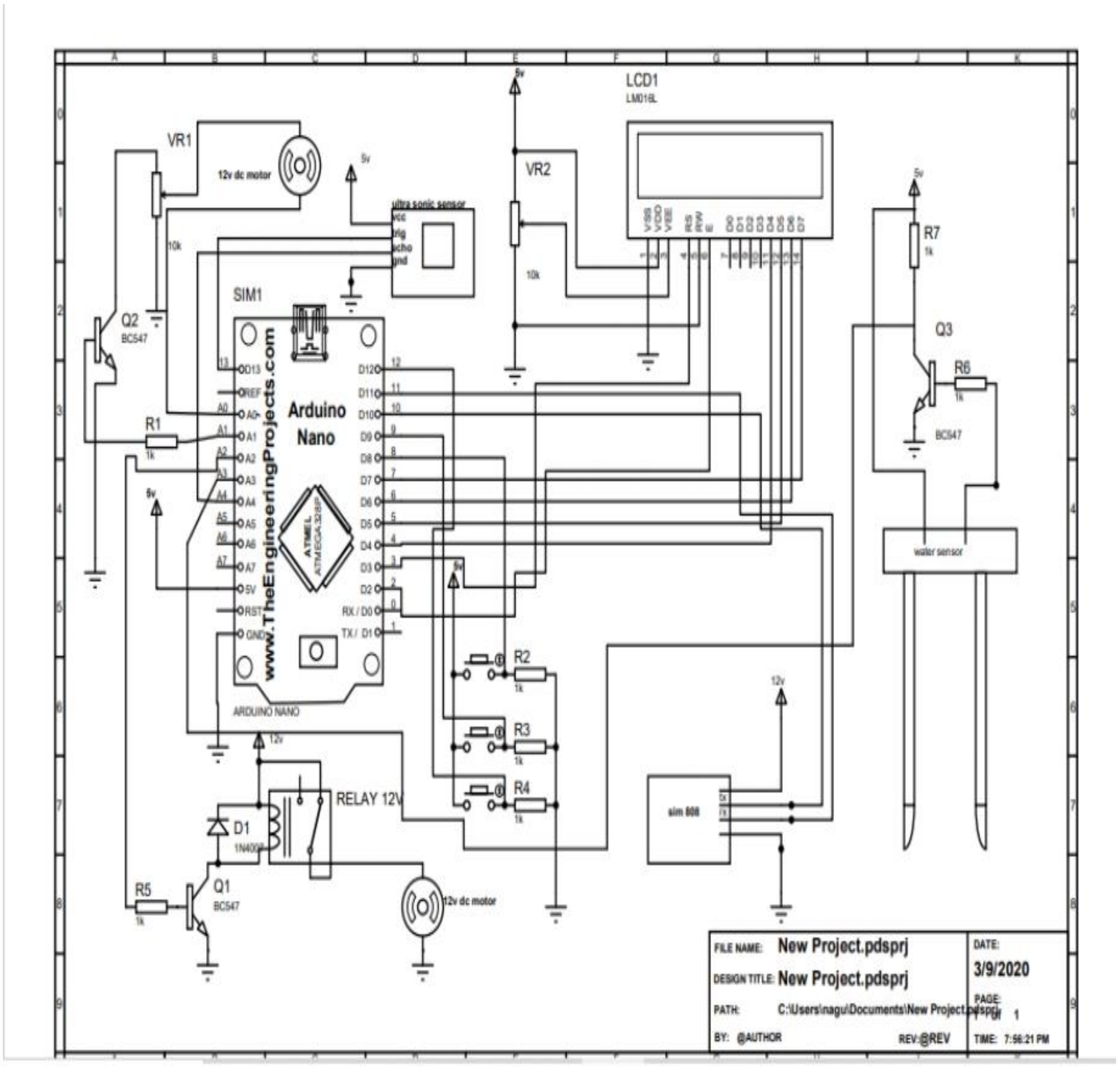


FIG 4.2 WIRING DIAGRAM OF THE CIRCUIT

The objective of this paper is to present the status of the current trends and implementation of Agricultural and autonomous systems and outline the potential for future applications. Different applications of autonomous vehicles in agriculture are examined and compared with typical systems, wherever 3 main teams of field operations are known to be the primary potential sensible applications: crop institution, plant care and selective harvesting. The vehicle is controlled by Android application interfaced with hardware through Bluetooth. The language input permits a user to move with the automaton that is acquainted to most of the individuals. The advantages of those robots are hands-free and quick information input operations. In the field of agricultural autonomous vehicle, a concept is been developed to investigate if multiple small autonomous machine could be more efficient than traditional large tractors and human forces. Keeping the above ideology in mind, a unit with the following feature is designed Ploughing is one of the first steps in farming. During this method we have a tendency to until the land and create it prepared for the seed sowing. By cultivation we have a tendency to mean that a plough is used ready to} have teeth's like structure at the tip and can be able to flip the highest layer of soil down and vice-versa. Seed sowing comes next wherever the seeds got to be place in ground at regular intervals and these have to be controlled mechanically. Limiting the flow of seeds from the seeds chamber is usually doing this. Mud radical is fitted to shut the seeds to the soil.

## 4.3 RADOR SYSTEM

Radars send out electromagnetic waves similar to wireless computer networks and mobile phones. The signals are sent out as short pulses which may be reflected by objects in their path, in part reflecting back to the radar. ... In that same way, the pulse reflects off precipitation and sends a signal back to the radar. At the same time, radar has found an increasing number of important civilian applications, notably air traffic control, weather observation, remote sensing of the environment, aircraft and ship navigation, speed measurement for industrial applications and for law enforcement, space surveillance, and planetary Radar is a detection system that uses radio waves to determine the distance, angle, or velocity of objects. It can be used to detect aircraft, ships, spacecraft, guided missiles, motor vehicles, weather formations, and terrain

## 4.3.1 BASIC DESIGN OF RADOR SYSTEM

The following figure shows the operating principle of a primary radar set. The radar antenna illuminates the target with a microwave signal, which is then reflected and picked up by a receiving device. The electrical signal picked up by the receiving antenna is called echo or return. The radar signal is generated by a powerful transmitter and received by a highly sensitive receiver

### TRANSMITTER

The radar transmitter produces the short duration high-power rf pulses of energy that are into space by the antenna.

### DUPLEXER

The duplexer alternately switches the antenna between the transmitter and receiver so that only one antenna need be used. This switching is necessary because the high-power pulses of the transmitter would destroy the receiver if energy were allowed to enter the

### RECEIVER.

The receivers amplify and demodulate the received RF-signals. The receiver provides video signals on the output.

### RADAR ANTENNA

The Antenna transfers the transmitter energy to signals in space with the required distribution and efficiency. This process is applied in an identical way on reception

# CHAPTER-5

# EXPERIMENTAL RESULT

## 5.1 RESULT COMPREHENSION

Maritime industry, especially in the transportation Sector is a dangerous industry. Many risks can occur During the voyage. Therefore, the marine safety is Important in national shipping and international shipping. Especially Indonesia, which is an archipelago, making Sea transportation a vital means that can support and Facilitate access in reaching the region in Indonesia. Marine Transportation has the potential to be developed Because it is able to carry large quantities of cargo so as To encourage economic improvement and public welfare. Therefore, improving the safety performance of shipping Is a very important thing to be realized. However, the Safety of shipping in Indonesia is not optimal. This can be Seen with the number of ship accidents in 2010 until 2016 That is 54 cases. With the number of 337fatalities and 474 Injured .

**STEP-1: WHEN THE BOAT IS OVERLOADED**

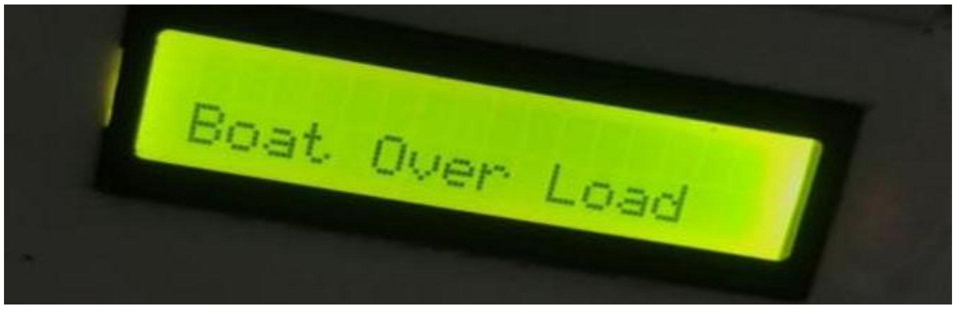

Boat Over Load

**Fig 5.1 boat is overloaded**

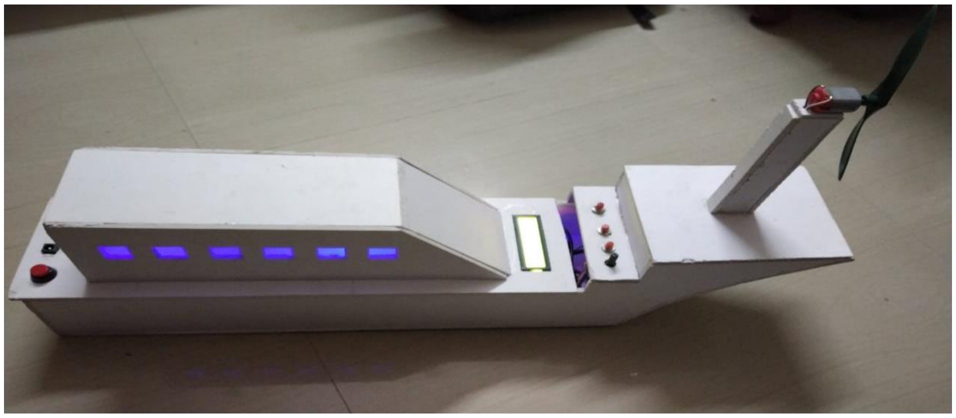


**Fig 5.2 over view**

When the boat is overload based on the fishes weight then the boats is unstable position in that time overload sensor is activated then the single is passed to the arduino then the relay is switched to the normal closed to normal open in that time motor of the boat is off and it displayed the boat is overload and also monitor the whether condition and it is also have the wind sensor and it is displayed in display Identification of missing boat using GPS can be used to trace the exact location of boat, the message will be sent to the nearest load center the rescue team will be alerted and will protect the humans

**STEP-2:- WHEN THE UNDER WATER LEVEL STONES OR ICE BERGS HAPPENS**

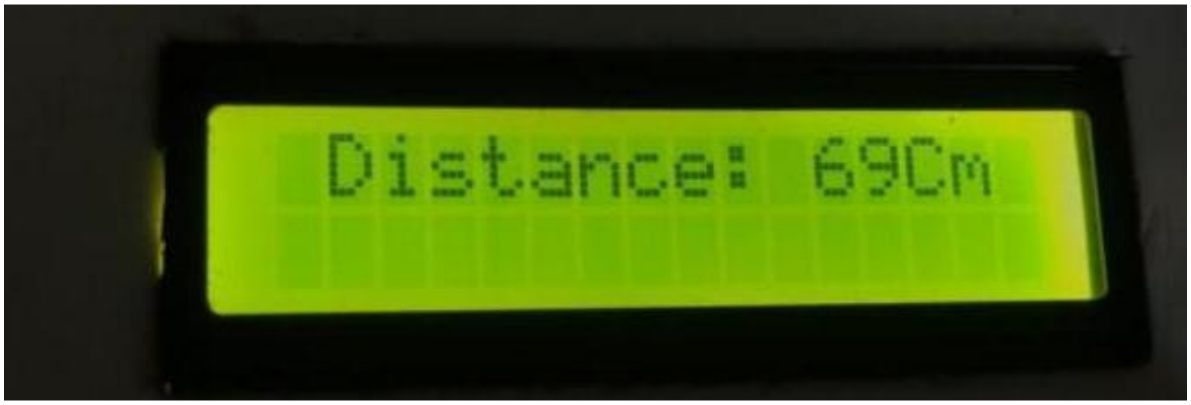


**FIG 5.3 RADOR SYSTEM**

During the boat running condition the ice bergs or stones are any kind of situation like fishes etc the boat can locates the some distance the alarm will be immediately rings by the sensor .

## STEP -3:- WHEN THE BOAT IS EMERGENCY CONDITION

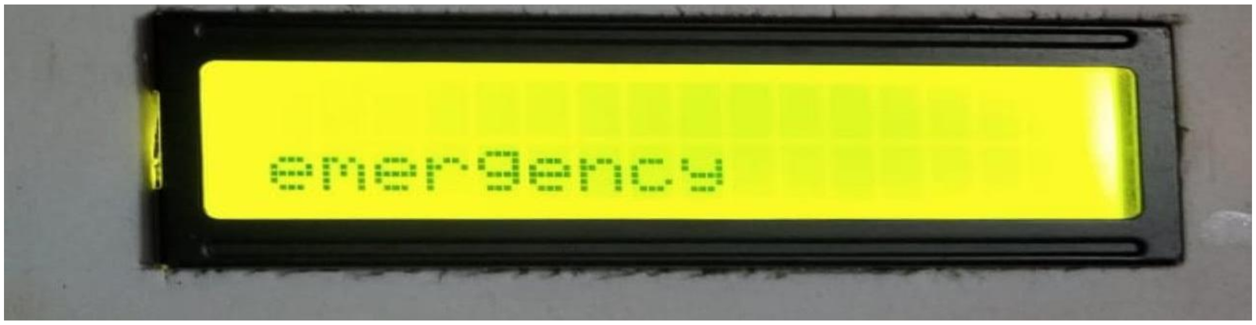


**FIG 5.4 EMERGENCY CONDITION**

The object detection is performed inside an angle of 180 degrees. The motion sensor that covers this angle uses a stepper motor. Data are presented on a LCD display, the motor movement being synchronized with the browsing of the radar screen. Synchronization is made by programming the timers of the microcontroller. These timers generate interruptions at preset moments, correlated with the completion of a specific area of the graphic display

## 5.2 APPLICATIONS

- A boat is designed to float on water and can be used for
- Travel,recreation, sports, fishing, transportation, military use and for rescue operations.
- Boats have been used by humans since the earliest civilizations when they were simply made of logs and reeds tied together to make rafts

# CHAPTER-6
# (CONCLUSION  AND FUTURE SCOPE)

Shipping accidents of today have become more “environmental” and the issue has been though than ever for all parties concerned, as those may end up withhuge financial losses. Shipping accidents are also a threat to smooth flow of shipping trade and damage to the environment.

Ships are ex posed to various ex ternal hazards such as darkness, different visibility conditions, bad weather, and currents, which one w ay or another, may contribute to shipping accidents like collisions, standings or groundings. Bad look-outs, not taking the proper action until a very late stage, close presence of a third ship which prevents taking early action and a late proper maneuver as against the crossing, overtaking and meeting end-on rules, etc. also constitute the internal threats of such incidents.

A large oil spill resulting from a shipping activity, or a series of activities, does not hurt only the people directly involved. Neighboring economies may also be badly affected as a result of a pollution incident occurred in other state’s territorial water.

Almost every new ship built today and very many others as existing ships are fitted with sophisticated shipboard equipment to reduce navigational risks, sustain and enhance safety of life and property, and preserve the environment

Increasing the dimensions of ships to an incredibly larger size for the good sake of economies of scale how even, has brought in higher risks and ultimately more costly actions in case

# CHAPTER-7

# (APPENDIX & REFERENCES)

## REFERENCES

1 A. Grasiano, A.P. Teixeria, abd C. Guedes Soares. Classification of human errors in grounding and collusion accidents using the tracer taxonomy. Safety science, Volume 84, (2016), pp. 245-257.

2 Asa Ek, Marcus Runefors and Jonas Borel.Relationship between safety culture aspects-A work process to enable interpretation. Marine Policy, Volume 44, (2014), pp. 79-186.

3 Aven. T. On how to define, understand and describe risk. Reliable Eng. Syst, Volume 95, (2010), pp. 623-631.

4 Barry Strauch.. Investigating fatigue in marine accident investigations. Procedia manufacturing, Volume 3, (2015), pp. 3115-3122.

5 Barry Strauch. Can we examine safety culture in accident investigations, or should we? Safety science, Volume 77, (2015), pp. 102-111.

6. Bhagwati. K. Managing Safety. Wiley-VCH,2006)

7. Celik. M. Cebi. Analytical HFACS for investigating human errors in shipping accidents, Anal. Pref,Volume41,(2009),pp66-75.