A
Real Time/Societal Research Project Report
On

"AUTOMATIC ACCIDENT DETECTION SYSTEM"

Submitted in Partial Fulfillment of the
Academic Requirement for the Award
of Degree of

BACHELOR OF TECHNOLOGY

in

Electronics and Communication Engineering

Submitted by

K. MANASWINI	**(22R01A0493)**
D. KARTHIK	**(22R01A0480)**
M. SHARANYA	**(22R01A0497)**
CH. RAHUL	**(22R01A0477)**

Under the esteemed guidance of

Mr. K. Haripal
(Assistant Professor)
Department of Electronics and Communication Engineering

CMR INSTITUTE OF TECHNOLOGY

(UGCAUTONOMOUS)
Approved by AICTE, Permanent Affiliation to JNTUH, Accredited by NBA and NAAC
Kandlakoya(V), Medchal Dist-501401
www.cmrithyderabad.edu.in

2023-24

CMR INSTITUTE OF TECHNOLOGY

(UGCAUTONOMOUS)

Approved by AICTE, New Delhi, Permanently Affiliated to JNTUH, Hyderabad

Accredited by NBA and NAAC

Kandlakoya (V), Medchal Dist-501401

www.cmrithyderabad.edu.in

CERTIFICATE

This is to certify that a Real time Project entitled with "**AUTOMATIC ACCIDENT DETECTION SYSTEM**" is being submitted by:

K. MANASWINI	**(22R01A0493)**
D. KARTHIK	**(22R01A0480)**
M. SHARANYA	**(22R01A0497)**
CH. RAHUL	**(22R01A0477)**

To JNTUH, Hyderabad, in partial fulfillment of the requirement for award of the degree of B.Tech in Electronics &Communication Engineering and is a record of a bonafide work carried out under our guidance and supervision. The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University forward of another degree or diploma.

Project Supervisor
(Mr. K. Haripal)
Assistant professer

Signature of HOD,
(Dr. K. Niranjan Reddy)
Head of department

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M Janga Reddy**, Director, **Dr. G. MadhusudhanaRao**, Principal and **Dr. K. Niranjan Reddy**, Head of Department, Dept of Electronics & Communication Engineering, CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We would like to thank our project supervisor**,** **Mr. K. Haripal**, Assistant Professor, Department of ECE for the guidance and support, especially for the valuable ideas and knowledge shared to us throughout the Project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of their references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or in directly for every step towards success.

K. MANASWINI **(22R01A0493)**

D. KARTHIK **(22R01A0480)**

M. SHARANYA **(22R01A0497)**

CH. RAHUL **(22R01A0477)**

Declaration

We **MANASWINI (22R01A04093), KARTHIK (22R01A0480), SHARANYA (22R01A0497), RAHUL (22R01A0477)** of the Real time Project entitled as **"AUTOMATIC ACCIDENT DETECTION SYSTEM"** hereby declared that the matter embodied in this project is the genuine work done by us only and has not been submitted either to the university or to any university/institute for the fulfillment of the requirement of any course of study.

K. MANASWINI	**(22R01A0493)**
D. KARTHIK	**(22R01A0480)**
M. SHARANYA	**(22R01A0497)**
CH. RAHUL	**(22R01A0477)**

INDEX PAGE NO.

ABSTRACT

As the number of vehicles are increasing day by day accidents are also increasing rapidly. When an individual riding his/her bike or car, meets with an accident , there is a chance to suffer from serious injury or expire instantaneously and there is no one around to help him. This project focuses to protect the victims from accidents for two or four wheeler. Titled as "COLLISION DETECTOR''.

When an accident is detected, our system starts working due to Piezo electric sensor based on certain pressure/stress applied. We use GSM and GPS Modules, to send an SMS and location to nearest hospitals, control room and family member. It consists of Arduino UNO where we interface the ADXL335 Electric sensor, GSM and GPS modules with it. This system also consists of 10-15 sec time delay circuit to stop functioning & prevent frequent operation whenever it operates accidentally or unnecessarily.

LIST OF FIGURES

1. CHAPTER

INTRODUCTION

1.1. INTRODUCTION TO THE PROJECT

The frequency of traffic collisions in India is the highest in the world. A National Crime Records Bureau (NCRB) report revealed that every year, more than 135,000 traffic collisions related deaths occur in India. The extent of Accidents in Tamil Nadu records the highest road accidents for a decade and its capital Chennai has more accidents than any other city in India. Traffic collision related deaths increased from 13 per hour in 2008 to 14 per hour in 2009.

Vehicle Collison detection is a car safety technology which helps prevent accidents caused by the driver getting Medical Emergency help. Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads. Some of the current systems learn driver patterns and can detect when a driver is becoming drowsy .The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. The aim of this project is to develop a prototype Vehicle Collision detection system. The focus will be placed on designing a system that will accurately locate the vehicle collide area. In this project we use sensors to measure all these factors. The values measured will be sent to the microcontroller where the measured values will be compared with the reference values. If the values measured do not match with the reference values, then the microcontroller will send a warning signal in the LCD display thereby preventing accidents.

2. CHAPTER

DESIGN OF HARDWARE:

This chapter briefly explains about the Hardware Collision Detection system. It discusses the circuit diagram of each module in detail.

1.1. ARDUINO UNO

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

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode. Arduino board has the following new features:

- 1.0 pin out: added SDA and SCL pins that are 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 Arduino Due 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.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

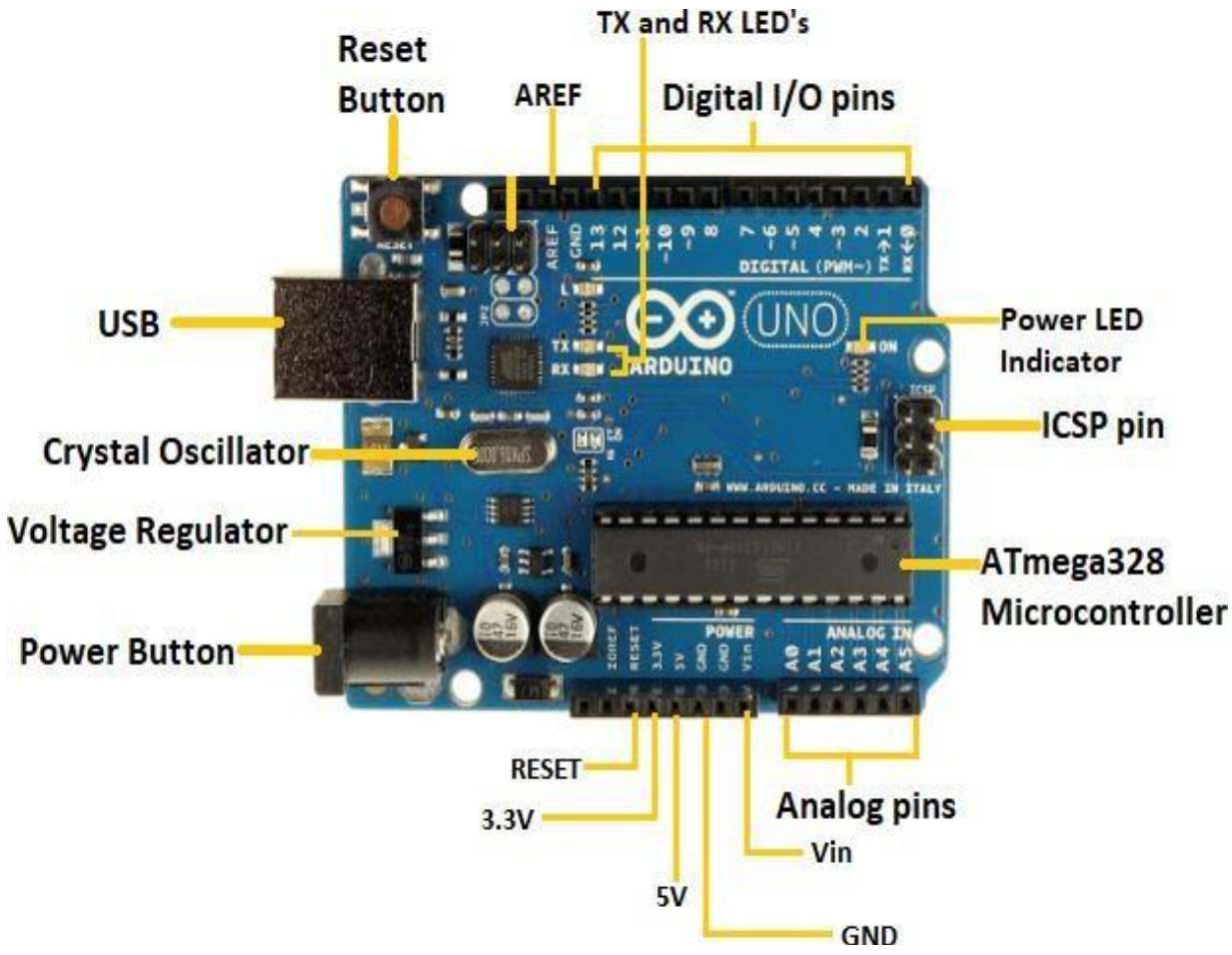


Fig 2.1(a): ARDUINO UNO

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the Atmel family. The processor code inside it is of 8-bit. It combines Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.

- **TX and RX LED's-** The successful flow of data is represented by the lighting of these LED's.
- **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

-

Programming:

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes pre burned 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; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . 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 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.

Software Reset:

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno 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 ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 Nano farad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the boot loader 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 Uno 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 boot loader is running on the Uno. 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 Uno 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 labelled "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.

USB Over current Protection:

The Arduino Uno has a resettable poly fuse that protects your computer's USB ports from shorts and over current. 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.

Physical Characteristics:

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four 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.

UART (Universal Asynchronous Receiver/Transmitter):

FEATURES:

1) 16 byte Receive and Transmit FIFOs
2) Register locations conform to ‘550 industry standard.
3) Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.

4) Built-in fractional baud rate generator with auto bauding capabilities.

5) Mechanism that enables software and hardware flow control implementation

PIN DESCRIPTION:

UART0 pin description

Pin	Type	Description
RXD0	Input	**Serial Input.** Serial receive data.
TXD0	Output	**Serial Output.** Serial transmit data.

REGISTER DESCRIPTION:

UART0 contains registers organized as shown in Table. The Divisor Latch Access Bit (DLAB) is contained in U0LCR[7] and enables access to the Divisor Latches.

1.2. POWER SUPPLY:

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

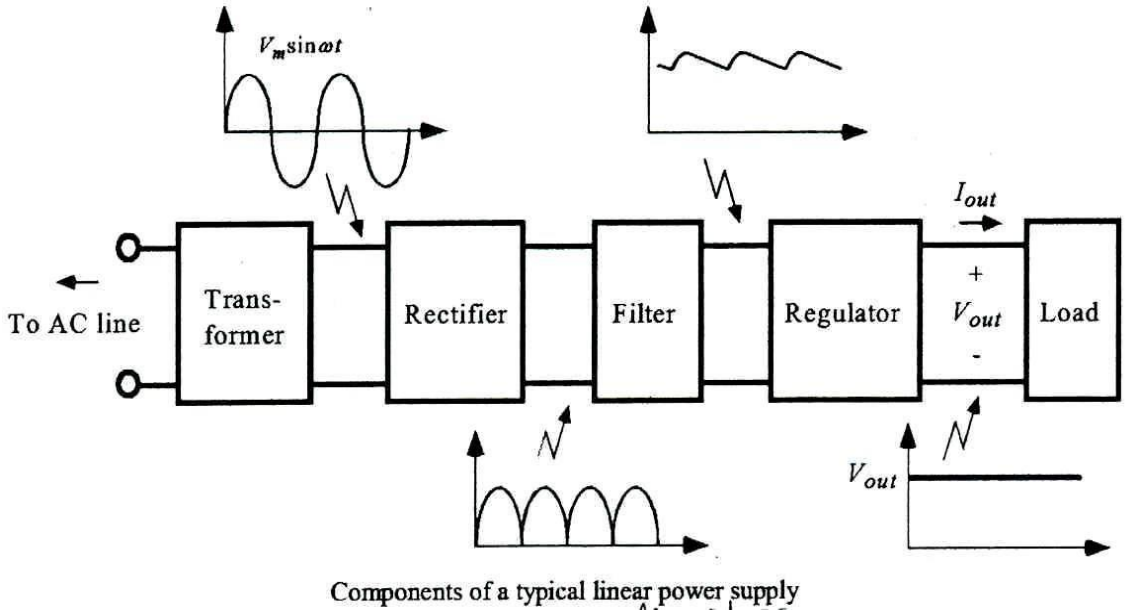


Fig:2.2(a). Block Diagram of Power Supply

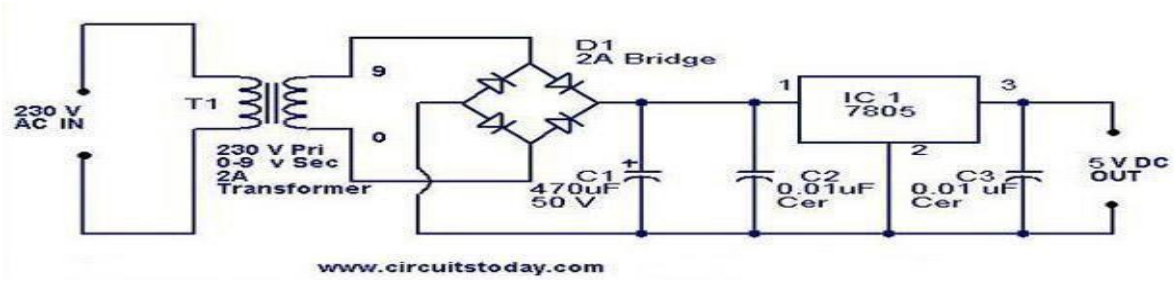


Fig:2.2(b). Schematic Diagram of Power Supply

2.2.1. TRANSFORMER:

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

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

2.2.2. RECTIFIER:

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

Bridge Rectifier:

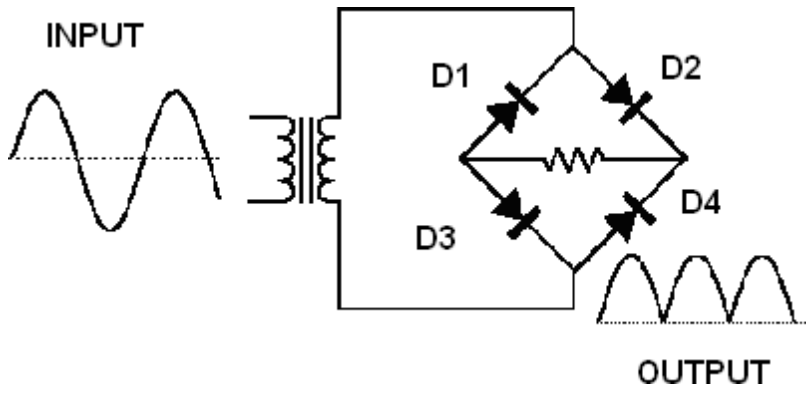


Fig: 2.2(c) Bridge Rectifier

OPERATION:

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

2.2.3. FILTER:

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

2.2.4. VOLTAGE REGULATOR:

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power

supply of 5V and 12V are required. In order to obtain these voltage levels,7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers 05,12 represent the required output voltage.

1.3. LCD:

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

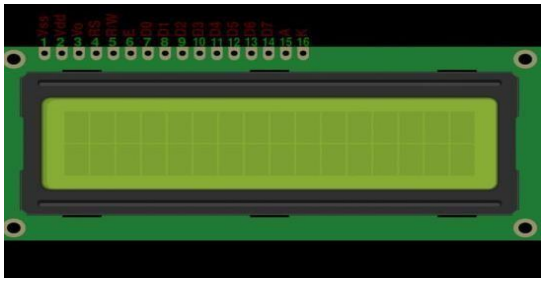


Fig: 2.3(a) LCD

2.3.1. PINS FUNCTIONS:

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

Function	Pin Number	Name	LogicState	Description
Ground	1	Vss	-	0V
Power supply	2	Vdd	-	+5V
Contrast	3	Vee	-	0 –Vdd

Control of operating	4	RS	0 1	D0 – D7 are interpreted as commands D0 – D7 are interpreted as data
	5	R/W	0 1	Write data (from controller to LCD) Read data (from LCD to controller)
	6	E	0 1 From 1 to 0	Access to LCD disabled Normal operating Data/commands are transferred to LCD
Data / commands	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1
	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	13	D6	0/1	Bit 6
	14	D7	0/1	Bit 7 MSB

Table 2:LCD Pin Functions

2.3.2. LCD SCREEN:

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

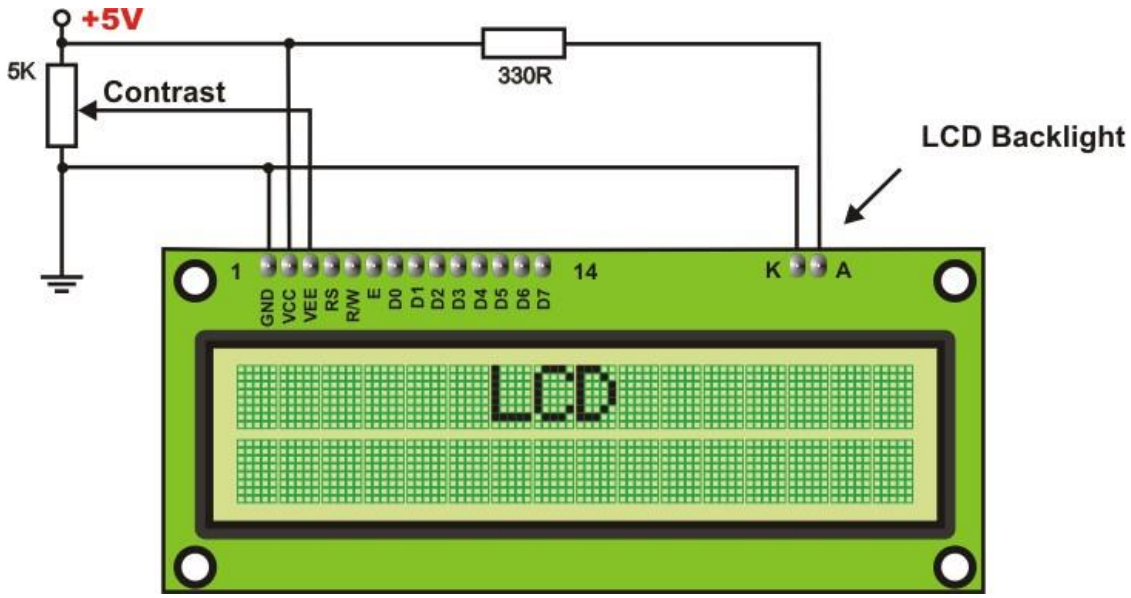


Fig: 2.3(b)LCD Screen Circuit Diagram

- **LCD Basic Commands:**

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

incremented. RS = 0 - Bits D0 - D7 are commands which determine display mode. List of commands which LCD recognizes are given in the table below:

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

Table 3 Basic Commands Of LCD

1.4. GSM MODEM

GSM is a mobile communication modem; it is stands for global system for mobile communication (GSM). The idea of GSM was developed at Bell Laboratories in 1970. It is widely used mobile communication system in the world. GSM is an open and digital cellular technology used for transmitting mobile voice and data services operates at the 850MHz, 900MHz, 1800MHz and 1900MHz frequency bands.

GSM system was developed as a digital system using time division multiple access (TDMA) technique for communication purpose. The digital system has an ability to carry 64 kbps to 120 Mbps of data rates.

There are various cell sizes in a GSM system such as macro, micro, Pico and umbrella cells. Each cell varies as per the implementation domain. There are five different cell sizes in a GSM network macro, micro, Pico and umbrella cells. The coverage area of each cell varies according to the implementation environment.

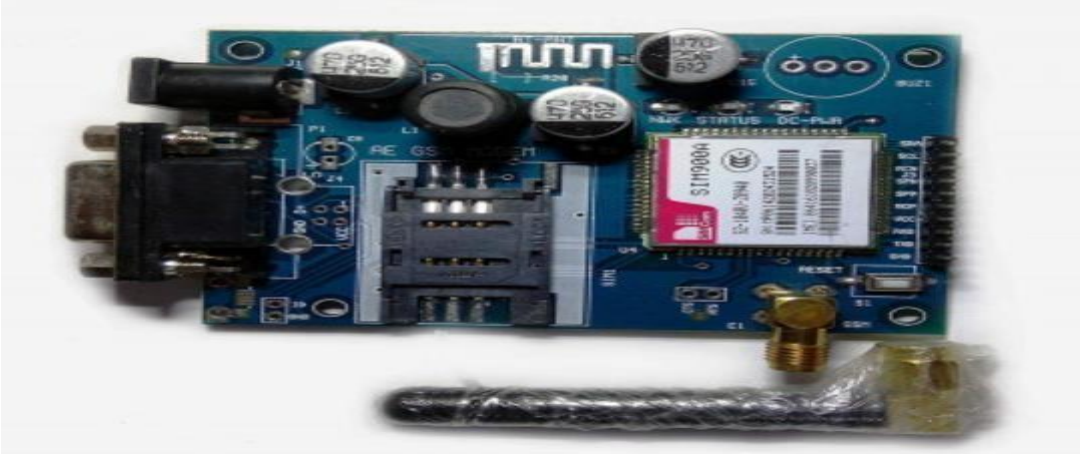


Fig: 2.4(a) GSM Module

Time Division Multiple Access

TDMA technique relies on assigning different time slots to each user on the same frequency. It can easily adapt to data transmission and voice communication.

GSM Architecture

A GSM network consists of the following components:

- **A Mobile Station:** It is the mobile phone which consists of the transceiver, the display and the processor and is controlled by a SIM card operating over the network.
- **Base Station Subsystem:** It acts as an interface between the mobile station and the network subsystem. It consists of the Base Transceiver Station which contains the radio transceivers and handles the protocols for communication with mobiles. It also consists of the Base Station

Controller which controls the Base Transceiver station and acts as a interface between the mobile station and mobile switching centre.

- **Network Subsystem:** It provides the basic network connection to the mobile stations. The basic part of the Network Subsystem is the Mobile Service Switching Centre which provides access to different networks like ISDN, PSTN etc. It also consists of the Home Location Register and the Visitor Location Register which provides the call routing and roaming capabilities of GSM. It also contains the Equipment Identity Register which maintains an account of all the mobile equipments wherein each mobile is identified by its own IMEI number. IMEI stands for International Mobile Equipment Identity.

Features of GSM Module:

- Support wide range of frequencies (from 850 MHZ to 1900 MHZ for different classification of GSM networks).
- Improved spectrum efficiency
- International roaming
- SIM phonebook management
- High-quality speech
- Uses encryption to make phone calls more secure
- Short message service (SMS)

GSM Modem

A GSM modem is a device which can be either a mobile phone or a modem device which can be used to make a computer or any other processor communicate over a network. It can be connected to a computer through serial, USB or Bluetooth connection.A GSM modem can also be a standard GSM mobile phone with the appropriate cable and software driver to connect to a serial port or USB port on your computer.

GSM/GPRS module is used to establish communication between a computer and a **GSM-GPRS system**. **Global System for Mobile communication (GSM)** is an architecture used for mobile communication in most of the countries. **Global Packet Radio Service (GPRS)** is an extension of GSM that enables higher data transmission rate. **GSM/GPRS module consists of a**

GSM/GPRS modem assembled together with power supply circuit and communication interfaces (like RS-232, USB, etc) for computer. The MODEM is the soul of such modules.

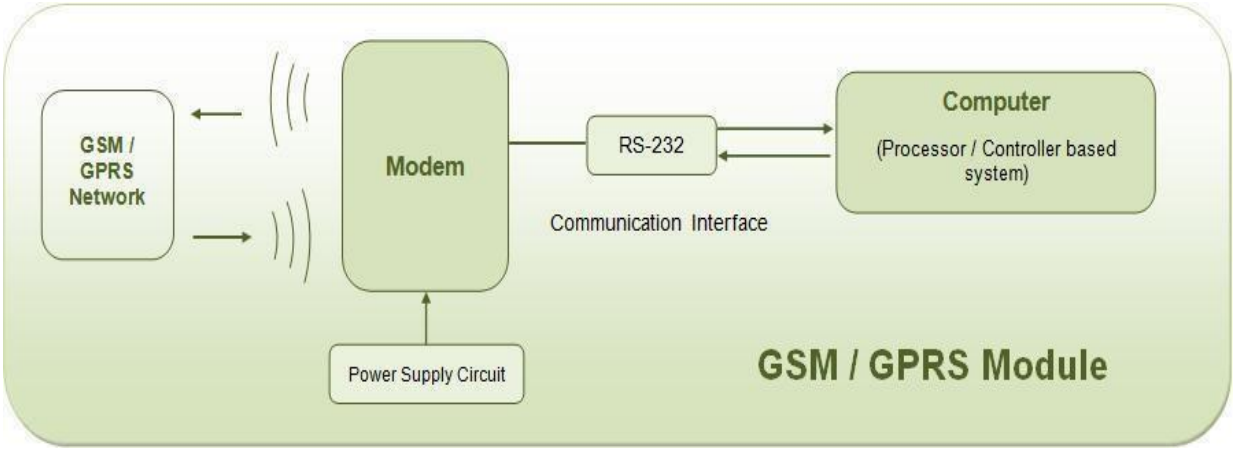


Fig:2.4(b) Communication to Controller

Wireless MODEMs

Wireless MODEMs are the MODEM devices that generate, transmit or decode data from a cellular network, for establishing communication between the cellular network and the computer. These are manufactured for specific cellular network (GSM/UMTS/CDMA) or specific cellular data standard (GSM/UMTS/GPRS/EDGE/HSDPA) or technology (GPS/SIM). Wireless MODEMs like other MODEM devices **use serial communication** to interface with and need **Hayes compatible AT commands** for communication with the computer (any microprocessor or microcontroller system).

GSM/GPRS MODEM is a class of wireless MODEM devices that are designed for communication of a computer with the GSM and GPRS network. It requires a **SIM (Subscriber Identity Module)** card just like mobile phones to activate communication with the network. Also they have **IMEI**(International Mobile Equipment Identity) number similar to mobile phones for their identification. A GSM/GPRS MODEM can perform the following operations:

1. Receive, send or delete SMS messages in a SIM.
2. Read, add, search phonebook entries of the SIM.
3. Make, Receive, or reject a voice call.

The MODEM needs **AT commands**, for interacting with processor or controller, which are communicated through serial communication.

These commands are sent by the controller/processor. The MODEM sends back a result after it receives a command. Different AT commands supported by the MODEM can be sent by the processor/controller/computer to interact with the **GSM and GPRS cellular network**.

A GSM modem is one of the wireless modem that is devised to work with a GSM wireless network. It works with the same frequency of GSM wireless network. It is an important part of the GSM network. Now a days GSM based cell phones are more preferred than cdma phones, hence let us see its operation and its features. The GSM wireless modem works in the way like a dial-up modem. The main difference between the GSM modem and dial up modem is that a dial-up modem sends and receives data through a fixed telephone line while a GSM wireless modem sends and receives data through radio wave propagation.

Operations that can be performed using GSM modem:

1. We can read, write and delete SMS messages.
2. We can start Sending SMS messages.
3. We can reply to a SMS message.
4. We can monitor the signal strength in particular locality.
5. We can monitor the charging status and also the charge level in the battery.
6. We can read, write and search phone book entries.
7. We can use it in various projects for different purposes.

Like a GSM mobile phone, a GSM modem requires a SIM card from a wireless carrier in order to operate. As mentioned in earlier sections of this SMS tutorial, computers use AT commands to control modems. Both GSM modems and dial-up modems support a common set of standard AT commands. You can use a GSM modem just like a dial-up modem.

Setting up the modem

If the modem contains a SIM card with is secured with a PIN code, we have to enter this pin code first:

AT+CPIN="0000" <ENTER> (replace 0000 with your PIN code).

Please not that in most cases you have only 3 attemps to set the correct PIN code. After setting the PIN code, wait some seconds before issueing the next command to give the modem some time to register with the GSM network.

In order to send a SMS, the modem has to be put in SMS text mode first using the following command:

To send the SMS message, type the following command:

AT+CMGS="+31638740161" <ENTER>

Replace the above phone number with your own cell phone number. The modem will respond with:

You can now type the message text and send the message using the <CTRL>-<Z> key combination:

Hello World ! <CTRL-Z>

After some seconds the modem will respond with the message ID of the message, indicating that the message was sent correctly:

+CMGS: 62

The message will arrive on the mobile phone shortly.

1.5. GPS MODULE

A GPS tracking unit is a device, normally carried by a moving vehicle or person, that uses the Global Positioning System to determine and track its precise location, and hence that of its carrier, at intervals. The recorded location data can be stored within the tracking unit, or it may be transmitted to a central location database, or Internet-connected computer, using a cellular(GPRS or SMS), radio, or satellite modem embedded in the unit. This allows the asset's

location to be displayed against a map backdrop either in real time or when analysing the track later, using GPS tracking software. Data tracking software is available for smart phones with GPS capability. The GPS was originally developed for use by the United States military, but in the 1980s, the United States government allowed the system to be used for civilian purposes.

Though the GPS satellite data is free and works anywhere in the world, the GPS device and the associated software must be bought or rented.

A GPS device can retrieve from the GPS system location and time information in all weather conditions, anywhere on or near the Earth. A GPS reception requires an unobstructed line of sight to four or more GPS satellites,[2] and is subject to poor satellite signal conditions. In exceptionally poor signal conditions, for example in urban areas, satellite signals may exhibit multipath propagation where signals bounce off structures, or are weakened by meteorological conditions. Obstructed lines of sight may arise from a tree canopy or inside a structure, such as in a building, garage or tunnel. Today, most standalone GPS receivers are used in automobiles.

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

The Russian Global Navigation Satellite System (GLONASS) was developed contemporaneously with GPS, but suffered from incomplete coverage of the globe until the mid-2000s.[3] GLONASS can be added to GPS devices to make more satellites available and enabling positions to be fixed more quickly and accurately, to within 2 meters.

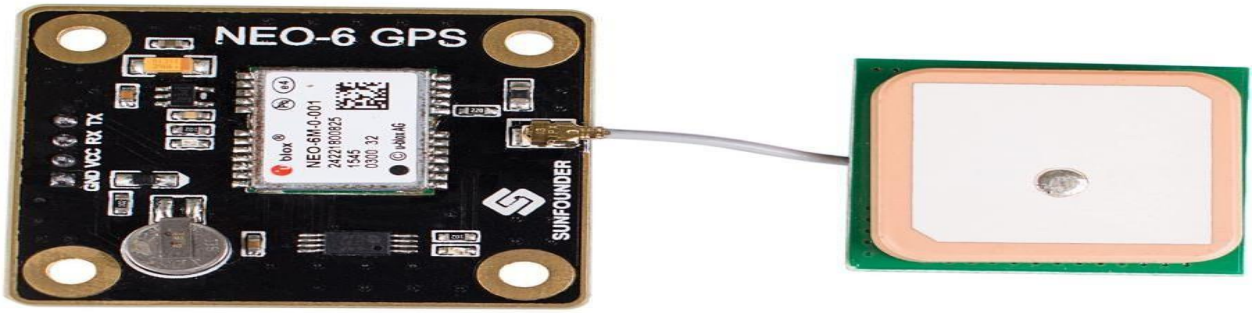


Fig:2.5(a) GPS NEO-6M

1.6. SWITCH

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal.[1] The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, although many un-biased buttons (due to their physical nature) still require a spring to return to their un-pushed state. Terms for the "pushing" of a button include pressing, depressing, mashing, hitting, and punching.

Fig:2.6(a) PUSH BUTTON

1.7. ADXL 335

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimumfull-scale range of ±3 g. It can measure the static acceleration of gravity in tiltsensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

Fig:2.7(a) adxl 335

APPLICATIONS

- Cost sensitive, low power, motion- and tilt-sensing applications
- Mobile devices
- Gaming systems
- Disk drive protection
- Image stabilization
- Sports and health devices

3. CHAPTER

DESIGN OF SOFTWARE

1.2. INTRODUCTION TO ARDUINO IDE:

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

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

1.3. SOFTWARE STEPS:

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

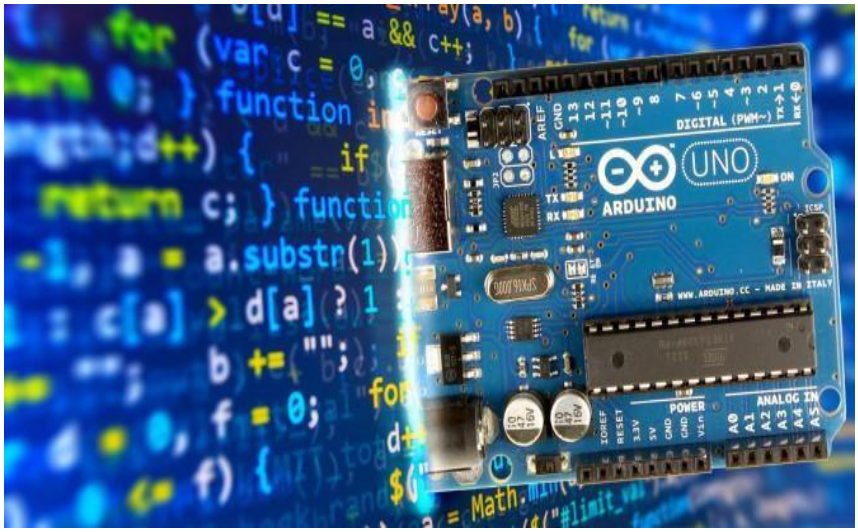
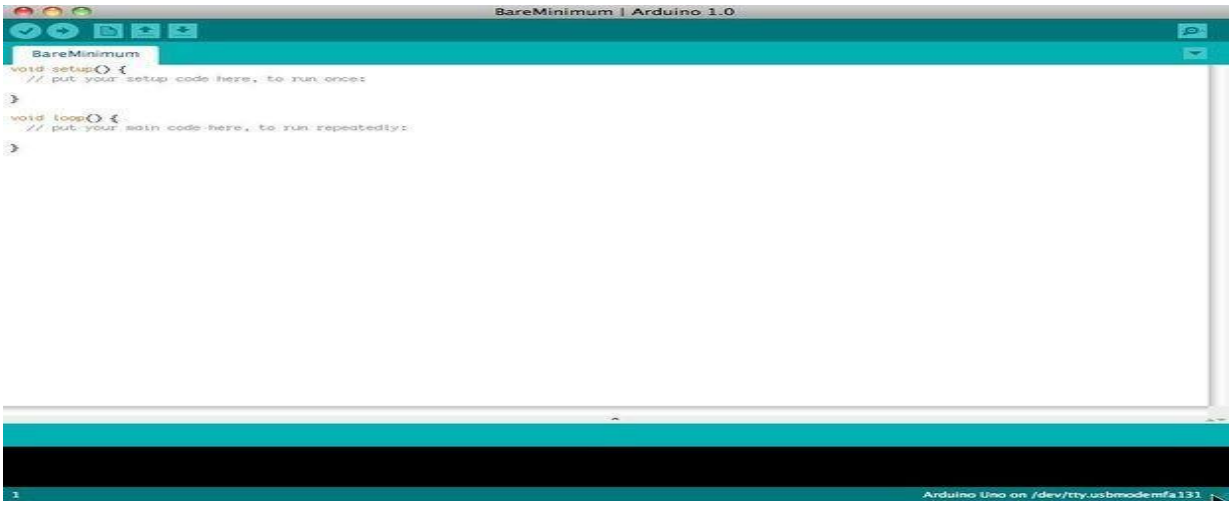
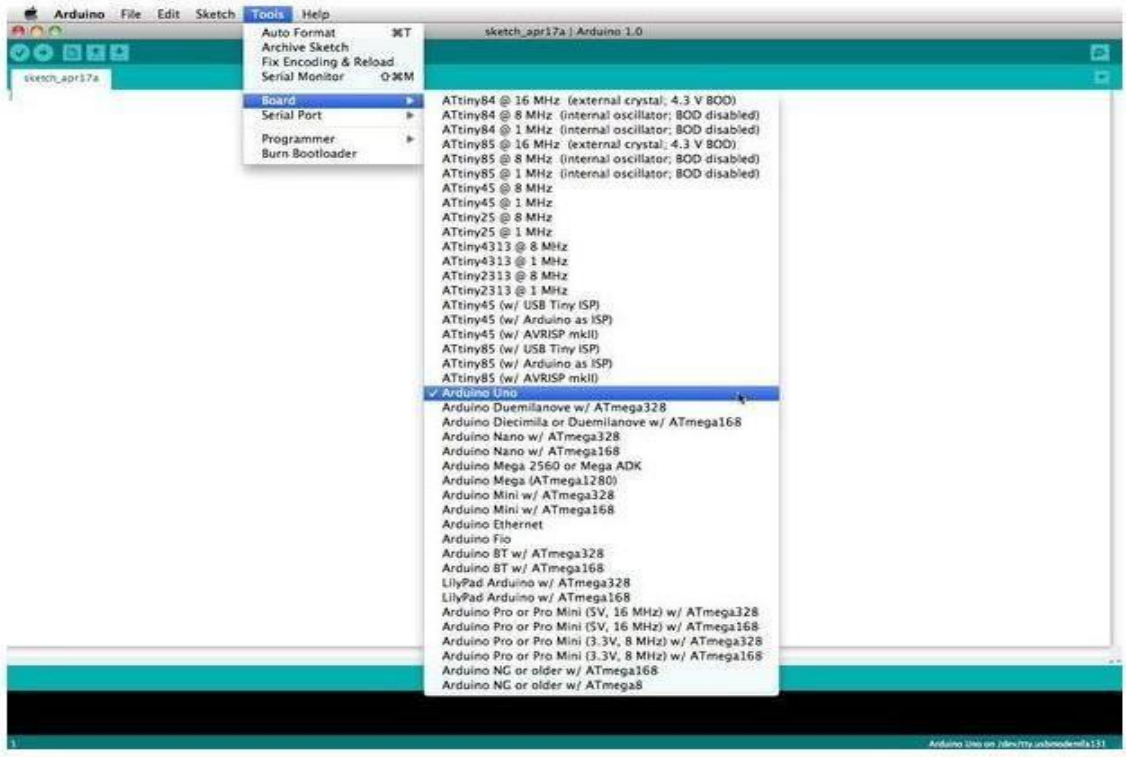


Fig:3.1(a) Aurdino ide



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



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

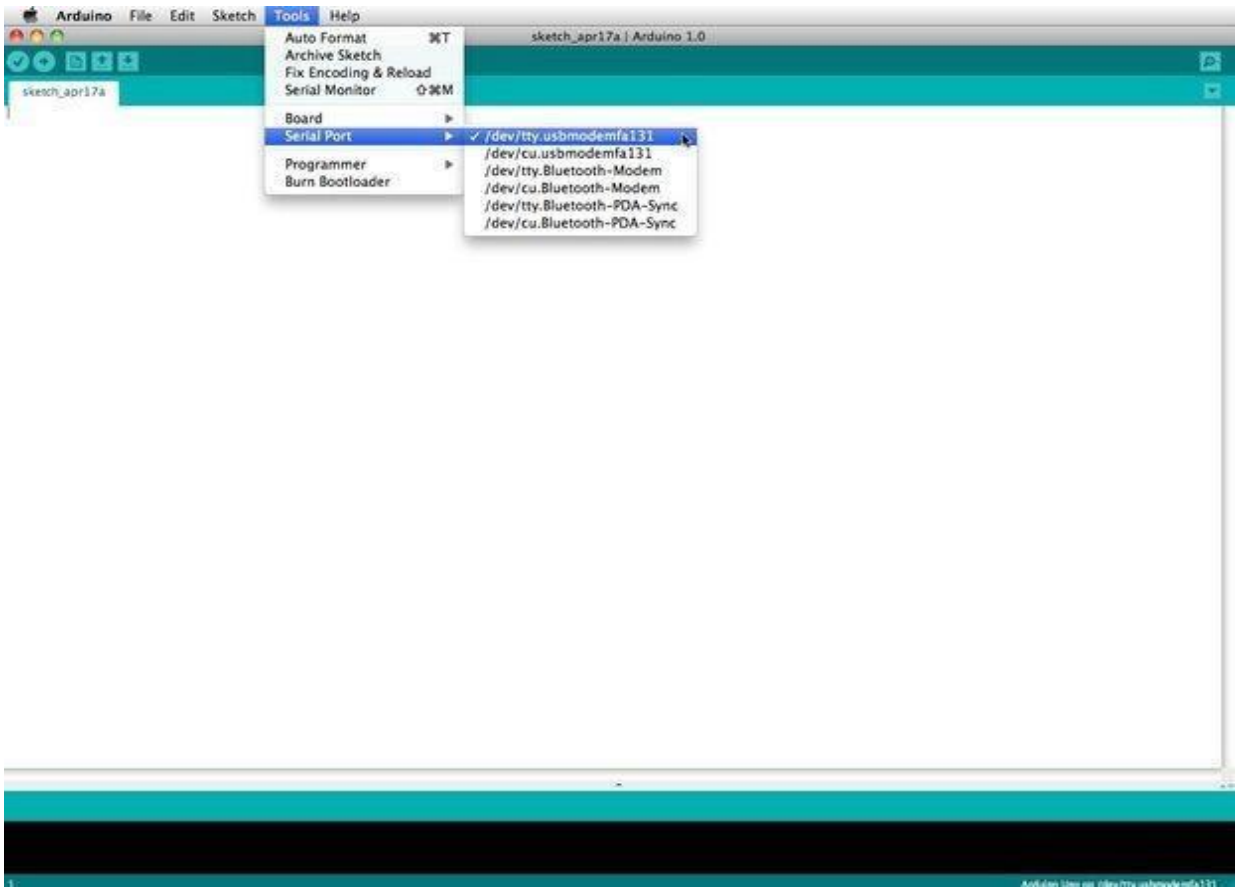
To set the board, go to the following:

Tools --> Boards

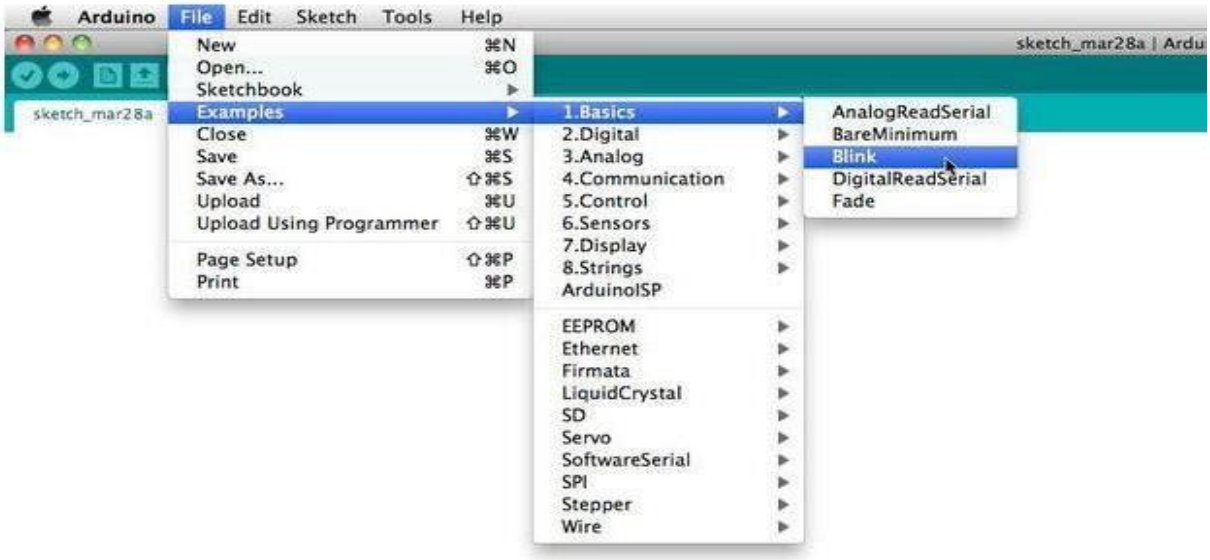
Select the version of board that you are using. Since I have an Arduino Uno plugged in, I obviously selected "Arduino Uno."

To set the serial port, go to the following:

Tools --> Serial Port

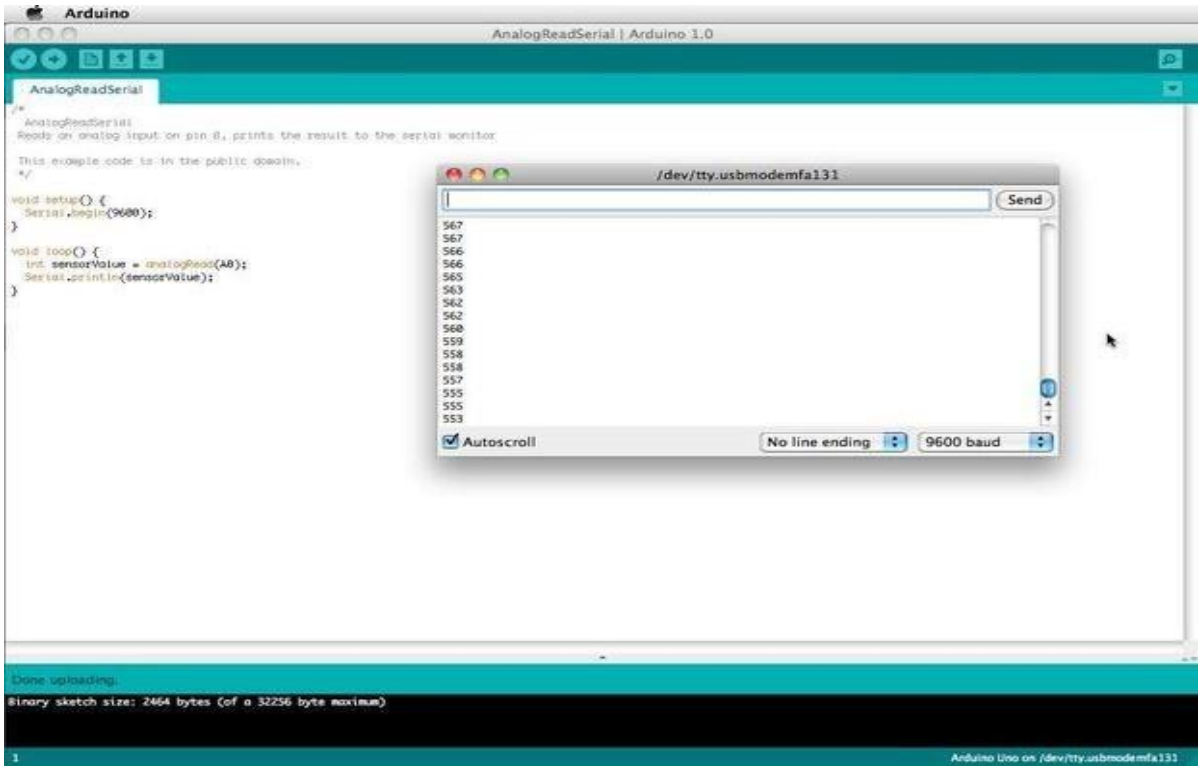


Arduino programs are called sketches. The Arduino programmer comes with a ton of example sketches preloaded. This is great because even if you have never programmed anything in your life, you can load one of these sketches and get the Arduino to do something.



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

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

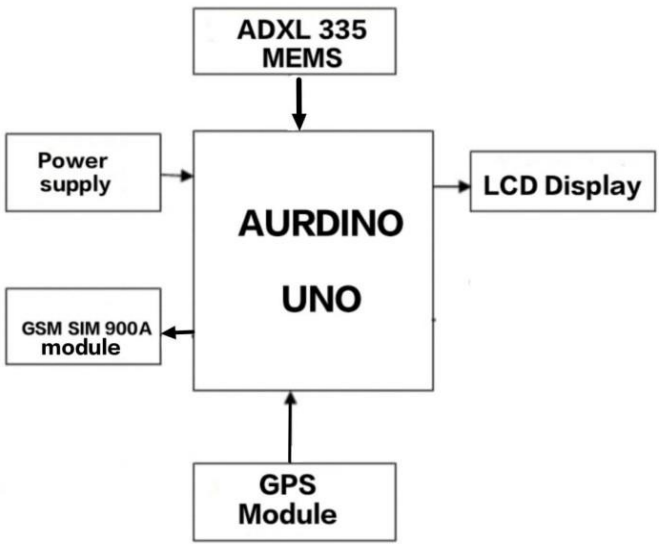


4. CHAPTER

PROJECT DESCRIPTION

This proposed work is an attempt to design an advance vehicle Collision Detection system that uses GPS and GSM system.

1.4. BLOCK DIAGRAM:



1.5. SOFTWARE REQUIREMENTS:

- Embedded c language
- Arduino Uno

1.6. HARDWARE REQUIREMENTS:

- GPS MODULE
- GSM MODEM
- AURDINO UNO
- LCD DISPLAY
- ADXL335 SENSOR
- SWITCH

4.2. WORKING:

The Main Aim of this project is to implement a low cost Vehicle collision detection system with GSM & GPS Modules, for immediate medical emergency help in remote area too. In this project microcontroller communicates with LCD, GPS module and GSM modem. This system will be placed in a moving vehicle. The microcontroller will poll GPS module in prefixed intervals and sends the vehicle location information (Latitude & Longitude) to central station over GSM network. Whenever any collision occurs Piezo sensor detects the vibration of the vehicle and sends mechanical force, to Arduino, by using GPS, we will get particular location where collision occurs, then GSM sends message to authorized members & Medical Teams. One more best feature is when press any emergency button gives message through GSM at current location of Latitude & Longitude values.

5. LAYOUT DIAGRAM

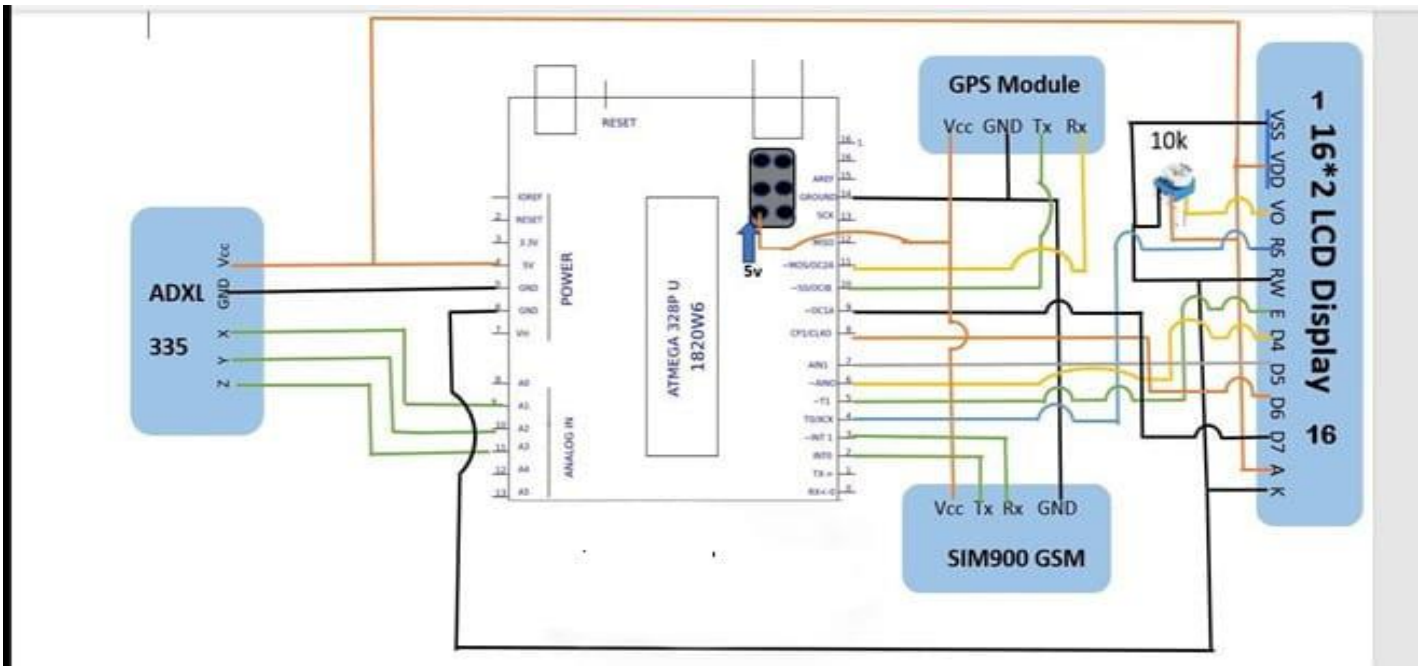


Fig.5.1 (a)

6. CODING:

```
#include<LiquidCrystal_I2C.h>

#include <AltSoftSerial.h>
#include <TinyGPS++.h>

#include <SoftwareSerial.h>
#include <math.h>

#include<Wire.h>
//must add i2c lcd address use i2c-scanner.ino file
LiquidCrystal_I2C lcd(0x27, 16, 2);
//-------------------------------------------------------------
//emergency phone number with country code
const String EMERGENCY_PHONE = "ENTER_EMERGENCY_PHONE_NUMBER";
//-------------------------------------------------------------
//GSM Module RX pin to Arduino 3
//GSM Module TX pin to Arduino 2
#define rxPin 2
#define txPin 3
SoftwareSerial sim900(rxPin,txPin);
//-------------------------------------------------------------
//GPS Module RX pin to Arduino 9
//GPS Module TX pin to Arduino 8
AltSoftSerial neogps;
TinyGPSPlus gps;
//-------------------------------------------------------------
String sms_status,sender_number,received_date,msg;
String latitude, longitude;
//-------------------------------------------------------------
#define BUZZER 12
#define BUTTON 11
//-------------------------------------------------------------
#define xPin A1
#define yPin A2
#define zPin A3
//-------------------------------------------------------------

byte updateflag;

int xaxis = 0, yaxis = 0, zaxis = 0;
int deltx = 0, delty = 0, deltz = 0;
int vibration = 2, devibrate = 75;
int magnitude = 0;
int sensitivity = 20;
double angle;
boolean impact_detected = false;
//Used to run impact routine every 2mS.
unsigned long time1;
unsigned long impact_time;
unsigned long alert_delay = 30000; //30 seconds
//-------------------------------------------------------------

/*****************************************************************************
 * setup() function
*****************************************************************************/
void setup()
{
  //-------------------------------------------------------------
 //Serial.println("Arduino serial initialize");
 Serial.begin(9600);
 //-------------------------------------------------------------
```

```
  //Serial.println("SIM900L serial initialize");
  sim800.begin(9600);
  //------------------------------------------------------------
  //Serial.println("NEO6M serial initialize");
  neogps.begin(9600);
  //------------------------------------------------------------
  pinMode(BUZZER, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  //------------------------------------------------------------
  //initialize lcd screen
  lcd.begin();
  // turn on the backlight
  lcd.backlight();
  lcd.clear();
  //------------------------------------------------------------
  sms_status = "";
  sender_number="";
  received_date="";
  msg="";
  //------------------------------------------------------------
  sim800.println("AT"); //Check GSM Module
  delay(1000);
  //SendAT("AT", "OK", 2000); //Check GSM Module
  sim800.println("ATE1"); //Echo ON
  delay(1000);
   //SendAT("ATE1", "OK", 2000); //Echo ON
  sim800.println("AT+CPIN?"); //Check SIM ready
  delay(1000);
  //SendAT("AT+CPIN?", "READY", 2000); //Check SIM ready
  sim800.println("AT+CMGF=1"); //SMS text mode
  delay(1000);
  //SendAT("AT+CMGF=1", "OK", 2000); //SMS text mode
  sim800.println("AT+CNMI=1,1,0,0,0"); /// Decides how newly arrived SMS should be handled
  delay(1000);
  //SendAT("AT+CNMI=1,1,0,0,0", "OK", 2000); //set sms received format
  //AT +CNMI = 2,1,0,0,0 - AT +CNMI = 2,2,0,0,0 (both are same)
  //------------------------------------------------------------
  time1 = micros();
  //Serial.print("time1 = "); Serial.println(time1);
  //------------------------------------------------------------
  //read calibrated values. otherwise false impact will trigger
  //when you reset your Arduino. (By pressing reset button)
  xaxis = analogRead(xPin);
  yaxis = analogRead(yPin);
  zaxis = analogRead(zPin);
  //------------------------------------------------------------
}

/*****************************************************************************************
 * loop() function
 *****************************************************************************************/
void loop()
{
  //------------------------------------------------------------
  //call impact routine every 2mS
  if (micros() - time1 > 1999) Impact();
  //------------------------------------------------------------
  if(updateflag > 0)
   {
    updateflag=0;
    Serial.println("Impact detected!!");
    Serial.print("Magnitude:"); Serial.println(magnitude);

    getGps();
    digitalWrite(BUZZER, HIGH);
    impact_detected = true;
    impact_time = millis();

    lcd.clear();
```

```
    lcd.setCursor(0,0); //col=0 row=0
    lcd.print("Crash Detected");
    lcd.setCursor(0,1); //col=0 row=1
    lcd.print("Magnitude:"+String(magnitude));
   }
   //---------------------------------------------------------
   if(impact_detected == true)
   {
    if(millis() - impact_time >= alert_delay) {
      digitalWrite(BUZZER, LOW);
      makeCall();
      delay(1000);
      sendAlert();
      impact_detected = false;
      impact_time = 0;
    }
   }

   if(digitalRead(BUTTON) == LOW){
    delay(200);
    digitalWrite(BUZZER, LOW);
    impact_detected = false;
    impact_time = 0;
   }
   //---------------------------------------------------------
   while(sim800.available()){
    parseData(sim800.readString());
   }
   //---------------------------------------------------------
   while(Serial.available())  {
    sim900.println(Serial.readString());
   }
   //---------------------------------------------------------


}




/*****************************************************************************************
 * Impact() function
 *****************************************************************************************/
void Impact()
{
   //---------------------------------------------------------
   time1 = micros(); // resets time value
   //---------------------------------------------------------
   int oldx = xaxis; //store previous axis readings for comparison
   int oldy = yaxis;
   int oldz = zaxis;

   xaxis = analogRead(xPin);
   yaxis = analogRead(yPin);
   zaxis = analogRead(zPin);

   //---------------------------------------------------------
   //loop counter prevents false triggering. Vibration resets if there is an impact. Don't detect new changes until that "time" has passed.
   vibration--;
   //Serial.print("Vibration = "); Serial.println(vibration);
   if(vibration < 0) vibration = 0;
   //Serial.println("Vibration Reset!");

   if(vibration > 0) return;
   //---------------------------------------------------------
   deltx = xaxis - oldx;
   delty = yaxis - oldy;
   deltz = zaxis - oldz;
```

```
  //Magnitude to calculate force of impact.
  magnitude = sqrt(sq(deltx) + sq(delty) + sq(deltz));
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  if (magnitude >= sensitivity) //impact detected
  {
    updateflag=1;
    // reset anti-vibration counter
    vibration = devibrate;
  }
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  else
  {
    //if (magnitude > 15)
      //Serial.println(magnitude);
    //reset magnitude of impact to 0
    magnitude=0;
  }
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
}




/*****************************************************************************************
 * parseData() function
 *****************************************************************************************/
void parseData(String buff){
  Serial.println(buff);

  unsigned int len, index;
  //--------------------------------------------------------------
  //Remove sent "AT Command" from the response string.
  index = buff.indexOf("\r");
  buff.remove(0, index+2);
  buff.trim();
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  if(buff != "OK"){
    //--------------------------------------------------------------
    index = buff.indexOf(":");
    String cmd = buff.substring(0, index);
    cmd.trim();

    buff.remove(0, index+2);
    //Serial.println(buff);
    //--------------------------------------------------------------
    if(cmd == "+CMTI"){
      //get newly arrived memory location and store it in temp
      //temp = 4
      index = buff.indexOf(",");
      String temp = buff.substring(index+1, buff.length());
      temp = "AT+CMGR=" + temp + "\r";
      //AT+CMGR=4 i.e. get message stored at memory location 4
      sim800.println(temp);
    }
    //--------------------------------------------------------------
    else if(cmd == "+CMGR"){
      //extractSms(buff);
      //Serial.println(buff.indexOf(EMERGENCY_PHONE));
      if(buff.indexOf(EMERGENCY_PHONE) > 1){
        buff.toLowerCase();
        //Serial.println(buff.indexOf("get gps"));
        if(buff.indexOf("get gps") > 1){
          getGps();
          String sms_data;
          sms_data = "GPS Location Data\r";
          sms_data += "http://maps.google.com/maps?q=loc:";
          sms_data += latitude + "," + longitude;
```

```
        sendSms(sms_data);
       }
     }
   }
   //-------------------------------------------------------------
  }
  else{
  //The result of AT Command is "OK"
  }
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
}


/*****************************************************************************************
 * getGps() Function
*****************************************************************************************/
void getGps()
{
  // Can take up to 60 seconds
  boolean newData = false;
  for (unsigned long start = millis(); millis() - start < 2000;){
    while (neogps.available()){
      if (gps.encode(neogps.read())){
        newData = true;
        break;
      }
    }
  }

  if (newData) //If newData is true
  {
    latitude = String(gps.location.lat(), 6);
    longitude = String(gps.location.lng(), 6);
    newData = false;
  }
  else {
    Serial.println("No GPS data is available");
    latitude = "";
    longitude = "";
  }

  Serial.print("Latitude= "); Serial.println(latitude);
  Serial.print("Lngitude= "); Serial.println(longitude);
}



/*****************************************************************************************
* sendAlert() function
*****************************************************************************************/
void sendAlert()
{
  String sms_data;
  sms_data = "Accident Alert!!\r";
  sms_data += "http://maps.google.com/maps?q=loc:";
  sms_data += latitude + "," + longitude;

  sendSms(sms_data);
}



/*****************************************************************************************
* makeCall() function
*****************************************************************************************/
void makeCall()
```

```
{
  Serial.println("calling....");
  Sim900.println("ATD"+EMERGENCY_PHONE+";");
  delay(20000); //20 sec delay
  sim900.println("ATH");
  delay(1000); //1 sec delay
}



/********************************************************************************************
 * sendSms() function
 ********************************************************************************************/
 void sendSms(String text)
{
  //return;
  Sim900.print("AT+CMGF=1\r");
  delay(1000);
  sim900.print("AT+CMGS=\""+EMERGENCY_PHONE+"\"\r");
  delay(1000);
  sim900.print(text);
  delay(100);
  sim900.write(0x1A); //ascii code for ctrl-26 //sim900.println((char)26); //ascii code for ctrl-26
  delay(1000);
  Serial.println("SMS Sent Successfully.");
}




/********************************************************************************************
 * SendAT() function
 ********************************************************************************************/
boolean SendAT(String at_command, String expected_answer, unsigned int timeout){

    uint8_t x=0;
    boolean answer=0;
    String response;
    unsigned long previous;

    //Clean the input buffer
    while( sim900.available() > 0) sim900.read();

    sim900.println(at_command);

    x = 0;
    previous = millis();

    //this loop waits for the answer with time out
    do{
        //if there are data in the UART input buffer, reads it and checks for the asnwer
        if(sim900.available() != 0){
            response += sim800.read();
            x++;
            // check if the desired answer (OK) is in the response of the module
            if(response.indexOf(expected_answer) > 0){
                answer = 1;
                break;
            }
        }
    }while((answer == 0) && ((millis() - previous) < timeout));

  Serial.println(response);
  return answer;
      }
```

7. CONCLUSION

Based on the research and experimentation conducted for this year project on collision detection using GSM and GPS modules, it can be concluded that the proposed system is effective in detecting collisions and notifying relevant parties in real-time. The system utilizes GPS technology to accurately determine the location of the vehicle and GSM technology to send SMS alerts to emergency contacts or relevant authorities.

The system was tested in various scenarios, including simulated collisions and actual test runs on roads, and it consistently demonstrated reliable and accurate collision detection capabilities. The system can also be easily integrated with existing vehicle security systems and can be customized to suit different vehicle types and sizes.

Overall, this project has successfully demonstrated the potential of combining GSM and GPS technologies for collision detection and highlights the importance of using technology to improve road safety. Further research can be conducted to optimize the system and improve its functionality, including adding additional sensors for more comprehensive collision detection and incorporating machine learning algorithms for more accurate collision prediction.

8. FUTURE SCOPE

The collision detector can be improved to instantly detect an accident and send alerts to emergency services, nearby hospitals, and family members of the victim. With the increasing popularity of autonomous vehicles, the collision detector can be integrated with them to prevent accidents and ensure the safety of passengers and other road users. The collision detector can be used to manage fleets of vehicles by providing real-time information about the location, speed, and driving behavior of the drivers. Overall, the future scope of collision detector using GSM and GPS technology is vast, and with the advancements in technology, we can expect to see more sophisticated systems that can save lives

9. REFERENCE

1. R. R. Knipling et al., “Assessment of IVHS countermeasures for collision avoidance systems,” National Highway Traffic Safety Administration, Washington, DC, USA, Tech. Rep. DOT HS 807 995, May 1993.

2. T. Kim and H. Jeong, “A Novel Algorithm for Crash Detection Under General Road Scenes Using Crash Probabilities and an Interactive Multiple Model Particle Filter,” in IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 6, pp. 2480-2490, Dec. 2014.

3. V. Naumov, “Analysis of Time and Distance Delays in Car Following Models,” 2010 International Conference on Intelligent Systems, Modelling and Simulation, Liverpool, 2010, pp. 296-299.

4. Geiger, P. Lenz and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, 2012, pp. 3354- 3361.

5. A. Cismas, M. Ioana, C. Vlad and G. Casu, “Crash Detection Using IMU Sensors,” 2017 21st International Conference on Control Systems and Computer Science (CSCS), Bucharest, 2017, pp. 672-676.