

**DEVELOPMENT OF ALCOHOL DETECTION SYSTEM WITH ENGINE
LOCKING USING GSM AND GPS TECHNOLOGIES**

Major Project Report submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Electronics and Communication Engineering

By

A. Avinash Chandra - 2210416201

K Saiteja-2210416223

P Amulya-2210416245

Under the Guidance of

Dr. P Trinatha Rao, Ph.D

Professor



Department of Electrical, Electronics and Communication Engineering

GITAM School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

April 2020

DECLARATION

We submit this Major Project work entitled "**DEVELOPMENT OF ALCOHOL DETECTION SYSTEM WITH ENGINE LOCKING USING GSM AND GPS TECHNOLOGIES**" to GITAM (Deemed to be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology**" in "**Electronics and Communication Engineering**". We declare that it was carried out independently by us under the guidance of **Dr. P Trinatha Rao**, Professor, GITAM (Deemed to be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD

A. Avinash Chandra-2210416201

Date: 10.04.2020

k. Saiteja-2210416223

P.Amulya-2210416245



GITAM (Deemed to be University)

Hyderabad-502329, India

Dated: 10th April, 2020

CERTIFICATE

This is to certify that the Major Project Report entitled “**DEVELOPMENT OF ALCOHOL DETECTION SYSTEM WITH ENGINE LOCKING USING GSM AND GPS TECHNOLOGIES**” is being submitted by A. Avinash Chandra (2210416201), k. Saiteja (2210416226) and P. Amulya (2210416245) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Electronics & Communication Engineering**, Department of Electrical, Electronic and Communication Engineering (EECE) GITAM (Deemed to be University), Hyderabad.

It is faithful record work carried out by them at the **Electrical, Electronics & Communication Engineering Department**, GITAM (Deemed to be University) Hyderabad Campus under my guidance and supervision.

Dr. P Trinatha Rao

Professor

Department of EECE

Dr. K. Manjunathachari

Professor and HOD

Department of EECE

ACKNOWLEDGMENT

Apart from our effort, the success of this major project largely depends on the encouragement and guidance of no. of people. We take this opportunity to express our gratitude to the people who have helped us in the successful competition of the major project.

We would like to thank **Prof. N. Siva Prasad**, Pro Vice Chancellor, GITAM (Deemed to be University), Hyderabad and **Prof. N. Seetharamaiah**, Principal, GITAM School of Technology, GITAM (Deemed to be University), Hyderabad.

We would like to thank respected **Dr. K. Manjunathachari**, Head of the Department of Electrical, Electronics and Communication Engineering, GITAM (Deemed to be University), Hyderabad for giving us such a wonderful opportunity to expand our knowledge for our own branch and giving us guidelines to present the major project report. It helped us a lot to realize of what we study.

We would like to thank our guide, **Dr. P Trinatha Rao**, Professor, GITAM (Deemed to be University), Hyderabad who helped us to make this major project a successful accomplishment.

We would like to thank our friends who helped us to make our work more organized and well-stacked till the end.

A. Avinash Chandra-2210416201

k. Saiteja-2210416223

P.Amulya-2210416245

ABSTRACT

Drunk and drive is one of the biggest threats to road accidents. Driving under the influence of alcohol has affected and killed countless public. People who drink and drive, not only risks their life but also of others. Thus drunk and drive is a major reason of accidents all over the world.

The proposed project is designed for the safety of the people. It is an idea to prevent drunk and drive accidents by using automatic engine locking system with the help of alcohol sensors. This project should be fitted or installed inside the vehicle.

In this project we are developing a prototype alcohol detection and engine locking system by using an Arduino Uno microcontroller interfaced with an alcohol sensor along with a DC motor to demonstrate the concept. This project also includes both GSM and GPS technologies. This system uses MQ-3 alcohol sensor to continuously monitor the blood alcohol content to detect the existence of liquor in the exhalation of a driver. By placing the sensor on the steering wheel, our system has the capacity to continuously check alcohol level from the driver's breath. The ignition will fail to start if the sensors detects content of alcohol in the driver's breath by giving a alarm. At the same time the micro controller reads the data from the GPS module and locates the position of the vehicle, it then sends the message to the user's registered mobile number.

Table of Contents

1. Introduction	1
1.1 Embedded Systems.....	1
1.2 Characteristics	1
1.3 Classification	2
1.4 Applications.....	3
1.5 Introduction to Microcontroller based automatic engine locking system for drunken drivers	3
2. Methodology.....	4
2.1 Block diagram	4
2.2 Description of block diagram.....	4
2.2.1 Hardware Components	4
2.2.2 Software Components.....	5
3. Hardware Components	6
3.1 Arduino	6
3.1.1 Introduction	6
3.1.2 Arduino uno Technical Specifications	6
3.1.3 PIN configurations.....	7
3.1.4 ATmega328	9
3.2 GSM module	12
3.2.1 GSM Architecture.....	12
3.2.2 Features of GSM module.....	13
3.2.3 GSM Modem	13
3.2.4 SIM800L.....	14
3.3 GPS module.....	15

3.3.1 Working of GPS	16
3.3.2 Determining the position by using GPS	17
3.4 Alcohol Sensor	18
3.4.1 Description.....	18
3.4.2 Features.....	19
3.4.3 Sensitivity Adjustment	19
3.5 DC motor	19
3.5.1 Types of DC motors	20
3.6 LCD	21
3.6.1 PIN configuration of LCD 2x16.....	22
3.7 Resistor.....	23
3.8 Power supply	24
3.8.1 DC power supply	25
3.8.2 AC Power Supplies.....	28
4. Software.....	33
4.1 Arduino IDE	33
4.2 Writing Sketches.....	34
4.3 Uploading Sketches	34
4.4 Libraries	35
4.5 Serial monitor and Serial plotter.....	36
4.6 Pin Description.....	37
4.7 Declarations	37
4.7.1 Variables	37
4.7.2 Instances	38
4.8 How to program Arduino board.....	39

4.9 Arduino code libraries.....	39
4.9.1 Library Structure.....	39
4.9.2 Importing Libraries.....	39
4.9.3 From Software to Hardware	40
4.9.4 Setting Up Your IDE	42
4.10 Boards	43
4.11 List of Boards.....	43
5. Flowcharts and Connections.....	46
5.1 Interfacing GSM module with Arduino UNO	47
5.2 Interfacing GPS module to Arduino Uno	48
5.3 Interfacing of MQ3 Alcohol Sensor to Arduino Uno	49
5.4 Interfacing of DC motor to Arduino Uno	50
5.5 Interfacing of LCD with Arduino Uno	50
5.5 Output of the Alcohol detection system	51
6. Source Code.....	53
7. Conclusion.....	64
References.....	65

List of Figures

Figure 2.1 Block diagram of alcohol detection system.....	4
Figure 2.2 Circuit diagram.....	5
Figure 3.1 ATmega328 DIP pinout	10
Figure 3.2 Arduino uno.....	11
Figure 3.3 Arduino uno pinout.....	11
Figure 3.4 Architecture of GSM module	13
Figure 3.5 SIM800L.....	14
Figure 3.6 pinout of SIM800L	15
Figure 3.7 Segments of GPS	17
Figure 3.8 Determining position by using GPS	17
Figure 3.9 Alcohol sensor	18
Figure 3.10 Schematic of alcohol sensor	18
Figure 3.11 Schematic of DC motor cooling fan.....	21
Figure 3.12 2X16 LCD display.....	21
Figure 3.13 pinout diagram of LCD 2x16	22
Figure 3.14 Resistor	24
Figure 3.15 Schematic of basic AC-to-DC power supply	25
Figure 3.16 Switched mode power supply.....	27
Figure 3.17 AC adapter.....	29
Figure 3.18 Programmable power supply	30
Figure 3.19A 30 kV high-voltage power supply	31
Figure 3.20 A bipolar power supply	32

Figure 4.1 Screenshot of arduino IDE	33
Figure 4.2 Screenshot of libraries	35
Figure 4.3 Screenshot of serial monitor	36
Figure 4.4 screenshot of serial plotter	37
Figure 4.5 Screenshot of loop and setup	38
Figure 4.6 screenshot to compile code	40
Figure 4.7 Screenshot to show completion message	41
Figure 4.8 Screenshot to setup IDE	42
Figure 4.9 Screenshot to show boards	43
Figure 5.1 Flowchart of alcohol detection system	46
Figure 5.2 Interfacing GSM module with Arduino	48
Figure 5.3 Interfacing of GPS module to Arduino Uno	48
Figure 5.4 Interfacing of MQ3 sensor with Arduino Uno	49
Figure 5.5 Interfacing of DC motor to Arduino Uno	50
Figure 5.6 Interfacing of LCD with Arduino Uno	51
Figure 5.7 Circuit connections of alcohol detection system using GSM and GPS technologies	52
Figure 5.8 Output	52

List of Tables

Table 3.1 Arduino UNO specifications	9
Table 5.1 Connections of GSM module	47
Table 5.2 Connections GPS module to Arduino Uno.....	48
Table 5.3 Connections of MQ3 sensor with Arduino Uno	49

1. Introduction

1.1 Embedded Systems

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. An embedded system is a microcontroller-based, software driven, reliable, real-time control system, autonomous, or human or network interactive, operating on diverse physical variables and in diverse environments and sold into a competitive and cost conscious market.

An embedded system is not a computer system that is used primarily for processing, not a software system on PC or UNIX, not a traditional business or scientific application. High-end embedded & lower end embedded systems. High-end embedded system - Generally 32, 64 Bit Controllers used with OS. Examples Personal Digital Assistant and Mobile phones etc .Lower end embedded systems - Generally 8,16 Bit Controllers used with an minimal operating systems and hardware layout designed for the specific purpose.

1.2 Characteristics

1. An embedded system is any computer system hidden inside a product other than a computer.
2. Embedded systems have a microprocessor/ microcontroller and a memory. Some have a serial port or a network connection. They usually do not have keyboards, screens or disk drives.
3. They will encounter a number of difficulties when writing embedded system software in addition to those we encounter when we write applications.
 - (a) Throughput: Our system may need to handle a lot of data in a short period of time.
 - (b) Response: Our system may need to react to events quickly.

- (c) Testability: Setting up equipment to test embedded software can be difficult.
- (d) Debugability: Without a screen or a keyboard, finding out what the software is doing wrong (other than not working) is a troublesome problem.
- (e) Reliability: Embedded systems must be able to handle any situation without human intervention.
- (f) Memory space: Memory is limited on embedded systems, and you must make the software and the data fit into whatever memory exists.
- (g) Program installation: You will need special tools to get your software into embedded systems.
- (h) Power consumption: Portable systems must run on battery power, and the software in these systems must conserve power.
- (i) Processor hogs: Computing that requires large amounts of CPU time can complicate the response problem.
- (j) Cost: Reducing the cost of the hardware is a concern in many embedded system projects; software often operates on hardware that is barely adequate for the job.

1.3 Classification

1. Real Time Systems (RTS):
 - a. RTS is one which has to respond to events within a specified deadline.
 - b. A right answer after the dead line is a wrong answer.
2. RTS Classification:
 - a. Hard real time system: Hard real time systems have very narrow response time.
Example: Nuclear power system, Cardiac pacemaker.
 - b. Soft real time system: Soft real time systems have reduced constraints on "lateness" but still must operate very quickly and repeatably.
Example: Railway reservation system – takes a few extra seconds the data remains valid.

1.4 Applications

The following are the applications of the embedded systems:

1. Military and aerospace embedded software applications.
2. Communication Applications .
3. Industrial automation and process control software .
4. Mastering the complexity of applications.
5. Reduction of product design time.
6. Real time processing of ever increasing amounts of data.
7. Intelligent, autonomous sensor.

1.5 Introduction to Microcontroller based automatic engine locking system for drunken drivers

Most of these days, we hear lot of accidents due to drunken driving. Drunken drivers will not be in stable condition and so the rash driving is the inconvenience for other road users and also question of life and death for the drunken driver and for others.

The system uses a compact circuitry built around Flash version of AT89S52 microcontroller with a non-volatile memory capable of retaining the password data for over ten years. Programs are developed in embedded C. ISP is used to dump the code into the microcontroller.

The main purpose behind this project is “Drunken driving detection”. Now-a-days, many accidents are happening because of the alcohol consumption of the driver or the person who is driving the vehicle. Thus drunk driving is a major reason of accidents in almost all countries all over the world. Alcohol Detector in Car project is designed for the safety of the people seating inside the car. This project should be fitted or installed inside the vehicle.

2. Methodology

2.1 Block diagram

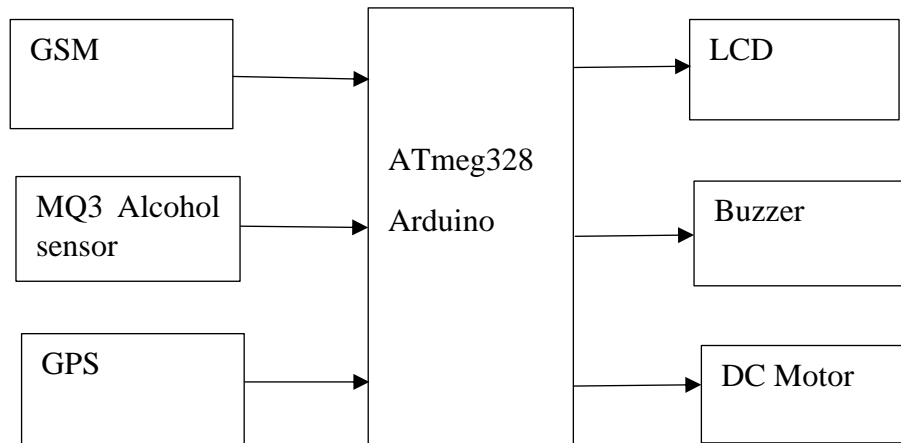


Figure 2.2.1 Block diagram of alcohol detection system

2.2 Description of block diagram

The following hardware and software is used in this system. Detailed explanations of these components follow in the later sections.

2.2.1 Hardware Components

1. Arduino ATmega328
2. GSM module
3. GPS module
4. Alcohol detection sensor
5. DC motor
6. LCD
7. Resistor
8. Leds
9. Power supply

2.2.2 Software Components

1. Arduino software
2. Arduino software programming

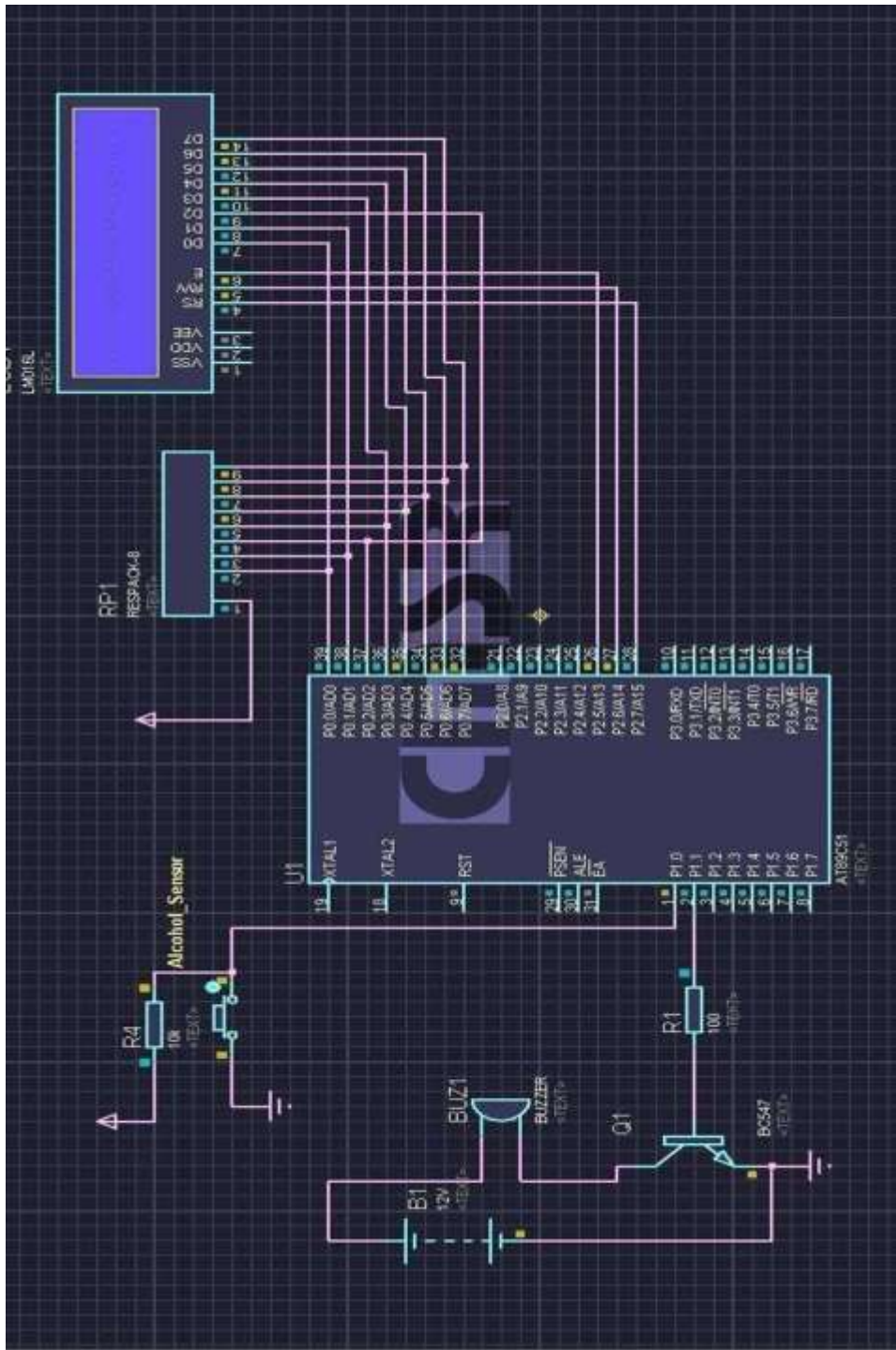


Figure 2.2 Circuit diagram

3. Hardware Components

3.1 Arduino

3.1.1 Introduction

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its products are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form or as do-it-yourself (DIY) kits.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers can be programmed using C and C++ programming languages. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2005 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats and motion detectors.

3.1.2 Arduino uno Technical Specifications

1. Microcontroller: Microchip ATmega328P
2. Operating Voltage: 5 Volts
3. Input Voltage: 7 to 20 Volts

4. Digital I/O Pins: 14 (of which 6 provide PWM output)
5. Analog Input Pins: 6
6. DC Current per I/O Pin: 20 mA
7. DC Current for 3.3V Pin: 50 mA
8. Flash Memory: 32 KB of which 0.5 KB used by bootloader
9. SRAM: 2 KB
10. EEPROM: 1 KB
11. Clock Speed: 16 MHz
12. Length: 68.6 mm
13. Width: 53.4 mm
14. Weight: 25 g

3.1.3 PIN configurations

3.1.3.1 General PIN configurations

1. LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it's off.
2. VIN: The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
3. 5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator and can damage the board.
4. 3V3: A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
5. GND: Ground pins.

6. IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.
7. Reset: Typically used to add a reset button to shields which block the one on the board.

3.1.3.2 Special PIN functions

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the `analogReference()` function.

In addition, some pins have specialized functions:

1. Serial / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.
2. External interrupts: 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.
3. PWM (pulse-width modulation): 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the `analogWrite()` function.
4. SPI (Serial Peripheral Interface): 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
5. TWI (two-wire interface) / I²C: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
6. AREF (analog reference): Reference voltage for the analog inputs.

3.1.4 ATmega328

The ATmega328 is a single-chip microcontroller created by Atmel in the megaAVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core.

The Atmel 8-bit AVR RISC-based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHz.

The following are the specifications of arduino uno:

Table 3.1 Arduino UNO specifications

Parameter	Value
CPU type	8-bit AVR
Performance	20 MIPS at 20 MHz
Flash memory	32 KB
SRAM	2 KB
EEPROM	KB
Pin count	28 or 32 pins: PDIP-28, MLF-28, TQFP-32, MLF-32

Maximum operating frequency	20 MHz
Number of touch channels	16
Hardware QTouch Acquisition	No
Maximum I/O pins	23
External interrupts	2
USB Interface	No
USB Speed	—

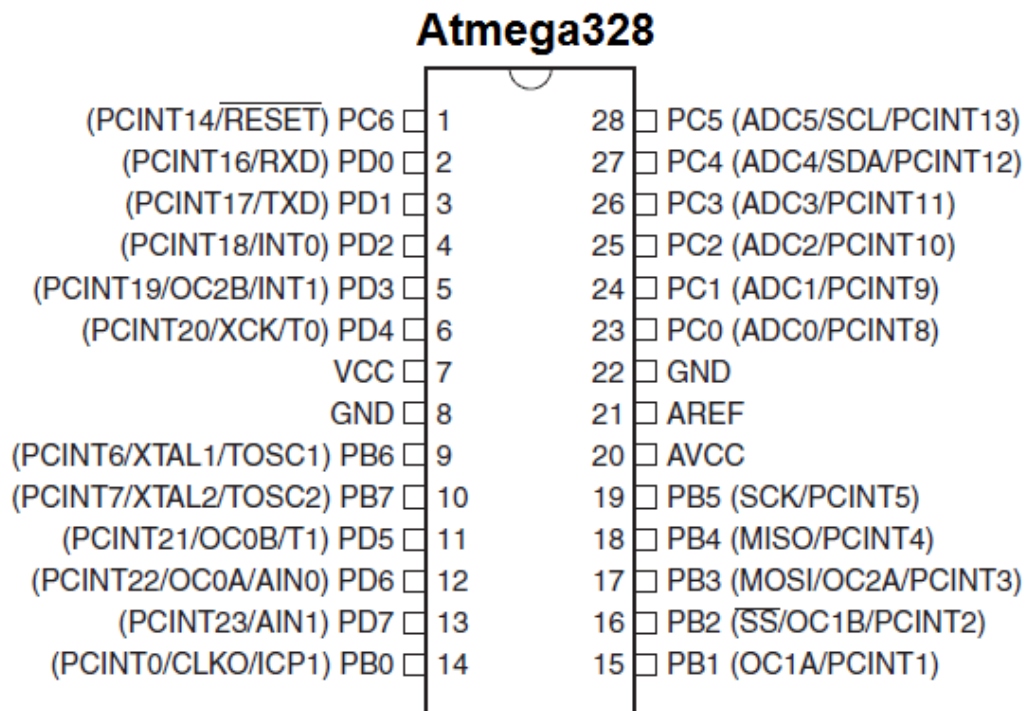


Figure 3.1 ATmega328 DIP pinout



Figure 3.2 Arduino uno

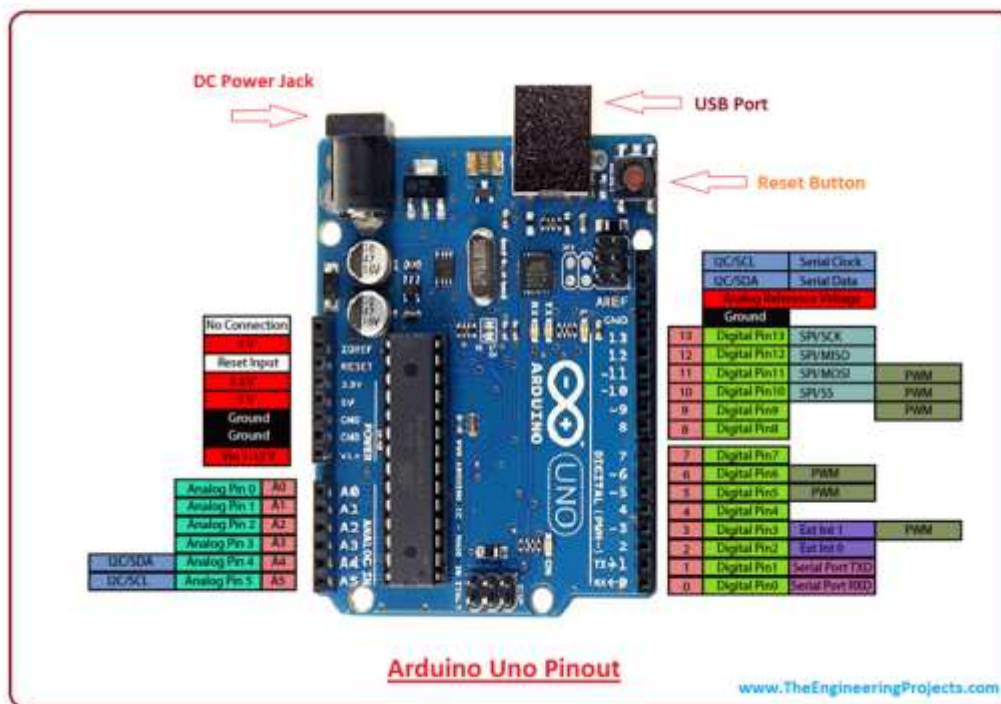


Figure 3.3 Arduino uno pinout

3.2 GSM module

GSM is a mobile communication modem, it 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. A GSM digitizes and reduces the data, then sends it down through a channel with two different streams of client data, each in its own particular time slot. 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.

3.2.1 GSM Architecture

A GSM network consists of the following components:

1. **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.
2. **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.
3. **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.

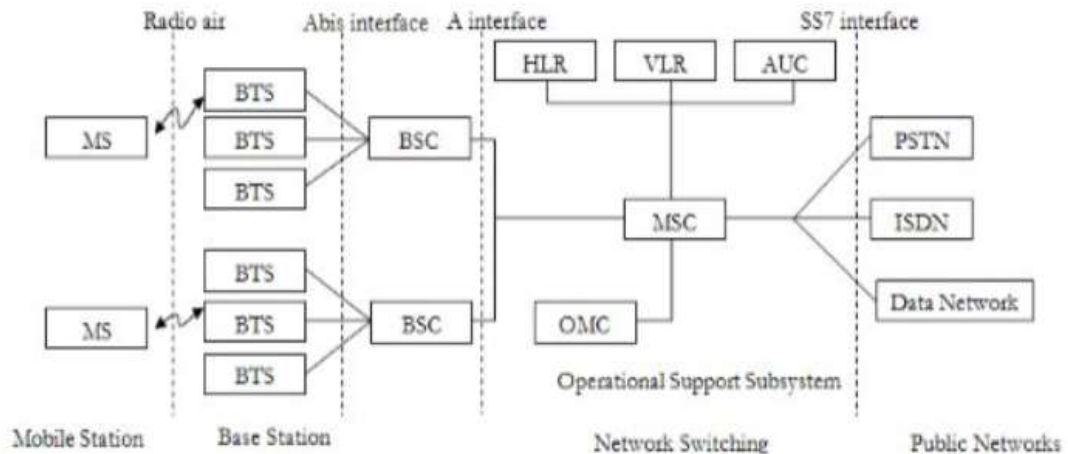


Figure 3.4 Architecture of GSM module

3.2.2 Features of GSM module

1. Improved spectrum efficiency
2. International roaming
3. Compatibility with integrated services digital network (ISDN)
4. Support for new services.
5. SIM phonebook management
6. Fixed dialing number (FDN)
7. Real time clock with alarm management
8. High-quality speech
9. Uses encryption to make phone calls more secure
10. Short message service (SMS)

3.2.3 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. A GSM modem requires a SIM card to be operated and operates over a

network range subscribed by the network operator. 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 modem is usually preferable to a GSM mobile phone. The GSM modem has wide range of applications in transaction terminals, supply chain management, security applications, weather stations and GPRS mode remote data logging.

3.2.4 SIM800L

SIM800L is a miniature cellular module which allows for GPRS transmission, sending and receiving SMS and making and receiving voice calls. Low cost and small footprint and quad band frequency support make this module perfect solution for any project that require long range connectivity. After connecting power module boots up, searches for cellular network and login automatically. On board LED displays connection state.



Figure 3.5 SIM800L

3.2.4.1 Specifications of SIM800L

1. Supply voltage: 3.8V - 4.2V
2. Recommended supply voltage: 4V
3. Power consumption:
4. sleep mode < 2.0mA
5. idle mode < 7.0mA

6. GSM transmission (avg): 350 mA
7. GSM transmission (peek): 2000mA
8. Module size: 25 x 23 mm
9. Interface: UART (max. 2.8V) and AT commands
10. SIM card socket: microSIM (bottom side)
11. Supported frequencies: Quad Band (850 / 950 / 1800 /1900 MHz)
12. Antenna connector: IPX
13. Status signaling: LED
14. Working temperature range: -40 do + 85 ° C

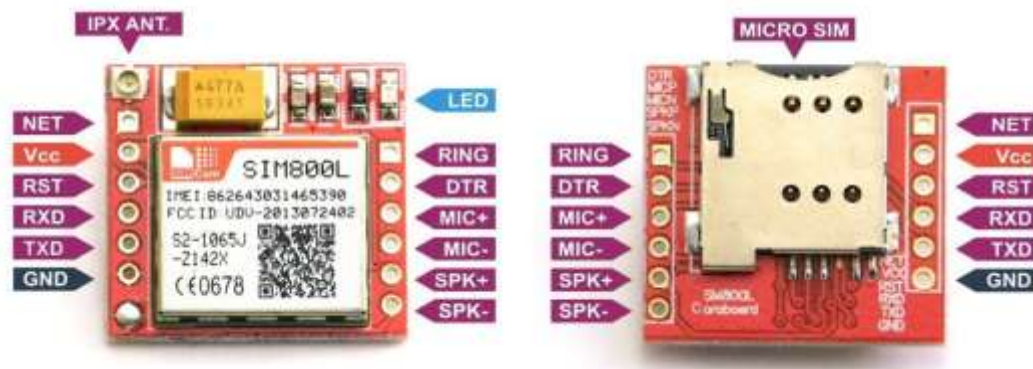


Figure 3.6 pinout of SIM800L

3.3 GPS module

Global Positioning System (GPS) is a satellite based navigation system that can provide people who use it with their exact position on Earth, tell them how to get to another location, how fast they are moving, where they have been, how far they have gone, what time it is, and many more.

GPS was originally designed to help the U.S. military with finding the accurate location of their soldiers, vehicles, planes and ships around the world. Now, GPS is used in cellular phones, outdoor recreation, emergency services, navigation and map making. It is also used a lot in scientific research. For example, meteorologists use GPS to forecast the weather and geologists use it to measure earthquake movement.

3.3.1 Working of GPS

The GPS system consists of three segments:

1. The space segment: the GPS satellites
2. The control system, operated by the U.S. military
3. The user segment, which includes both military and civilian users and their GPS equipment.

3.3.1.1 Space segment

The space segment is the number of satellites in the constellation. It comprises of 29 satellites circling the earth every 12 hours at 12,000 miles in altitude. The function of the space segment is utilized to route/navigation signals and to store and retransmit the route/navigation message sent by the control segment. These transmissions are controlled by highly stable atomic clocks on the satellites. The GPS Space Segment is formed by a satellite constellation with enough satellites to ensure that the users will have, at least, 4 simultaneous satellites in view from any point at the Earth surface at any time.

3.3.1.2 Control Segment

The control segment comprises of a master control station and five monitor stations outfitted with atomic clocks that are spread around the globe. The five monitor stations monitor the GPS satellite signals and then send that qualified information to the master control station where abnormalities are revised and sent back to the GPS satellites through ground antennas. Control segment also referred as monitor station.

3.3.1.3 User Segment

The user segment comprises of the GPS receiver, which receives the signals from the GPS satellites and determine how far away it is from each satellite. Mainly this segment is used for the U.S military, missile guidance systems, civilian applications for GPS in almost every field. Most of the civilian uses this from survey to transportation to natural resources and from there to agriculture purpose and mapping too.

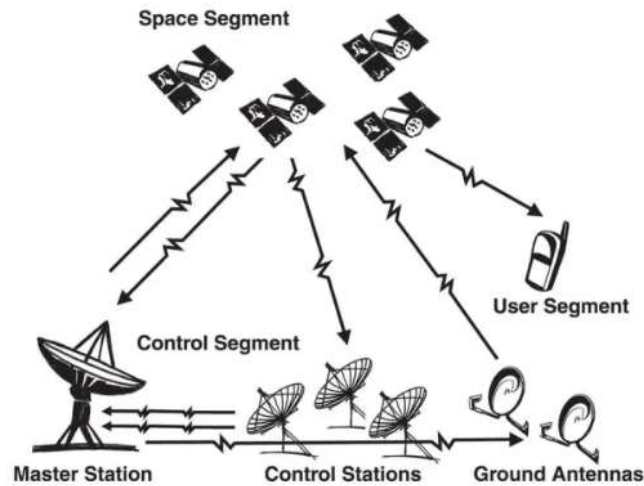


Figure 3.7 Segments of GPS

3.3.2 Determining the position by using GPS

The working of Global positioning system is based on the ‘trilateration’ mathematical principle. The position is determined from the distance measurements to satellites. From the figure, the four satellites are used to determine the position of the receiver on the earth. The target location is confirmed by the 4th satellite. And three satellites are used to trace the location place. A fourth satellite is used to confirm the target location of each of those space vehicles. Global positioning system consists of satellite, control station and monitor station and receiver. The GPS receiver takes the information from the satellite and uses the method of triangulation to determine a user’s exact position.

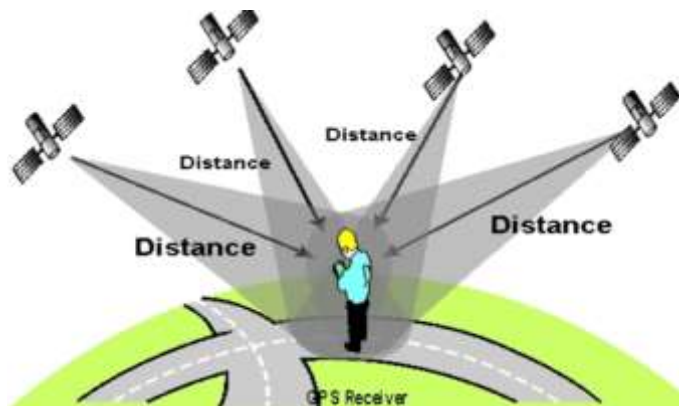


Figure 3.8 Determining position by using GPS

3.4 Alcohol Sensor

Sensitive material of MQ-3 gas sensor is SnO_2 , which with lower conductivity in clean air. When the target alcohol gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electro circuit, Convert change of conductivity to correspond output signal of gas concentration. MQ-3 gas sensor has high sensitivity to Alcohol, and has good resistance to disturb of gasoline, smoke and vapor. The sensor could be used to detect alcohol with different concentration, it is with low cost and suitable for different application.



Figure 3.9 Alcohol sensor

3.4.1 Description

This alcohol sensor is suitable for detecting alcohol concentration on your breath, just like your common breath analyser. It has a high sensitivity and fast response time. Sensor provides an analog resistive output based on alcohol concentration. The drive circuit is very simple, all it needs is one resistor. A simple interface could be a 0-3.3V ADC.

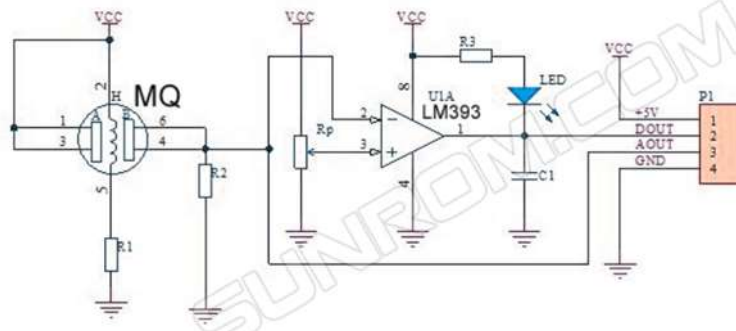


Figure 3.10 Schematic of alcohol sensor

3.4.2 Features

1. High sensitivity to alcohol and small sensitivity to benzene.
2. Fast response and high sensitivity.
3. Stable and long life.

3.4.3 Sensitivity Adjustment

Resistance value of MQ-3 is difference to various kinds and various concentration gases. So, when using this components, sensitivity adjustment is very necessary. we recommend that you calibrate the detector for 0.4mg/L (approximately 200ppm) of Alcohol concentration in air and use value of Load resistance that(RL) about 200 K Ω (100K Ω to 470 K Ω). When accurately measuring, the proper alarm point for the gas detector should be determined after considering the temperature and humidity influence.

3.5 DC motor

A DC motor is a motor that transforms electrical energy into mechanical energy by drawing in direct current. DC motors induce a rotation movement in the machine by electromagnetism. DC motors have inductors (electromagnet) within them that create a magnetic field that aids the rotation of the motor.

The electromagnet is a piece of iron with wire coil windings around it. This coil has current running through its terminals. This alignment has two stationary magnets on both the sides of the electromagnet. The opposing and attractive forces of these magnets create a torque.

In a simple DC motor, there is a fixed set of magnets in the stator. It has an armature with one or more windings of insulated wire wrapped around a soft iron core. This iron core concentrates the magnetic field produced. The ends of the wire winding are connected to a commutator that powers up and energizes each armature coil. The strength of the electromagnetic field thus created, depends upon the amount of current passing through the coil, its size and the material around which it is wrapped.

A rotating magnetic field can be created by merely turning these coils on and off in a sequence. Now, when this rotating magnetic field interacts with the magnetic fields of the magnets (permanent or electromagnet), it exerts a force on the armature which sets it in a rotation motion. By altering the direction and magnitude of the current flowing through the coils, you can change the direction and magnitude of the magnetic field produced by it.

3.5.1 Types of DC motors

DC motors can be classified into four basic categories:

3.5.1.1 Permanent Magnet DC Motors

In a permanent magnet DC motor, a permanent magnet is used to produce the magnetic field flux. It does not have a field winding on the stator frame and relies on permanent magnets to create the magnetic field against the opposing force of the rotor field to produce torque. It generates a good starting torque and provides a decent speed. These motors are usually found in applications running on low horsepower.

3.5.1.2 Series DC Motors

In a series, DC motor, the field windings, and the armature are connected in series to a common D.C. power source. This motor has a very high starting torque and is used for starting high inertia loads (e.g. trains, elevators).

3.5.1.3 Shunt DC Motors

In a shunt, DC motor, the field windings, and the armature are connected parallel (shunt) to each other with a common D.C. power source. These motors provide excellent speed regulation because the parallel field can be excited separately from the armature windings. Shunt DC motors are mostly used for industrial, adjustable speed applications (e.g. machine tools, winding/unwinding machines)

3.5.1.4 Compound DC Motors

In a compound DC Motor, the field windings, and the armature are connected in a combination of both shunt and series, thus, imparting to it the certain features of

both a shunt and series DC motor. This motor can be connected in two different arrangements: cumulatively or differentially. While cumulative compound motors connect the series field to aid the shunt field (higher torque, less speed regulation), the differential compound DC motors provide good speed regulation while operating at constant speed.

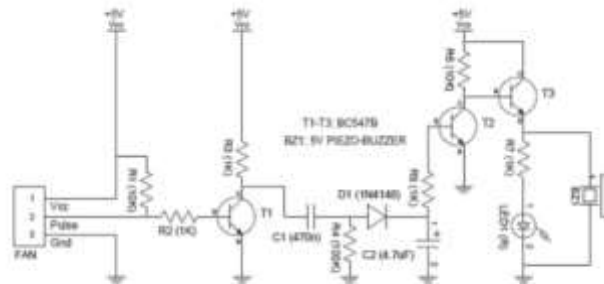


Figure 3.11 Schematic of DC motor cooling fan

3.6 LCD

It is combination of two states of matter, the solid and the liquid. LCD uses a liquid crystal to produce a visible image. Liquid crystal displays are super-thin technology display screen that are generally used in laptop computer screen, TVs, cell phones and portable video games. LCD's technologies allow displays to be much thinner when compared to cathode ray tube (CRT) technology.

An LCD is an electronic display module which uses liquid crystal to produce a visible image. The 16×2 LCD display is a very basic module commonly used in DIYs and circuits. The 16×2 translates o a display 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5×7 pixel matrix.



Figure 3.12 2X16 LCD display

3.6.1 PIN configuration of LCD 2x16

1. Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
2. Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
3. Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
4. Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1 (0 = data mode, and 1 = command mode).
5. Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
6. Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.

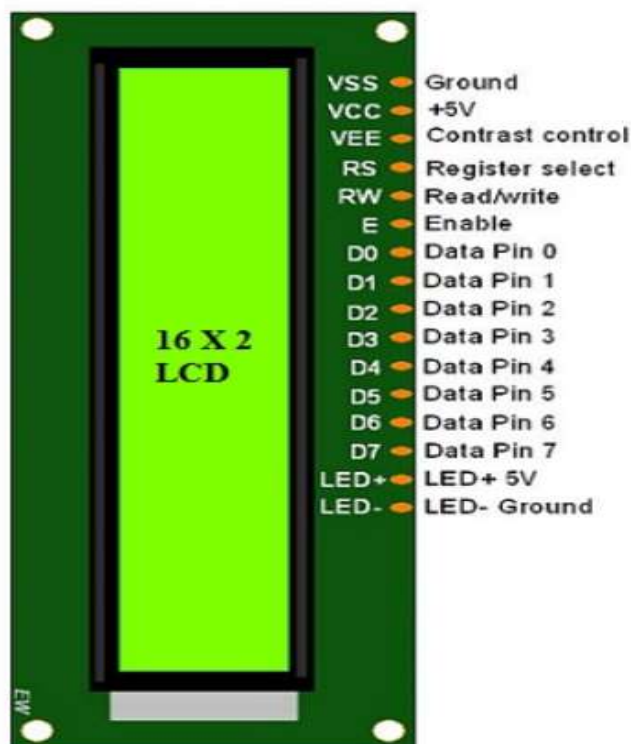


Figure 3.13 pinout diagram of LCD 2x16

7. Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only 4 pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
8. Pin15 (+ve pin of the LED): This pin is connected to +5V
9. Pin 16 (-ve pin of the LED): This pin is connected to GND.

3.7 Resistor

A resistor is a passive component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat, may be used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage. Variable resistors can be used to adjust circuit elements (such as a volume control or a lamp dimmer), or as sensing devices for heat, light, humidity, force, or chemical activity.

Resistors are common elements of electrical networks and electronic circuits and are ubiquitous in electronic equipment. Practical resistors as discrete components can be composed of various compounds and forms. Resistors are also implemented within integrated circuits.

The electrical function of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than nine orders of magnitude. The nominal value of the resistance falls within the manufacturing tolerance, indicated on the component.

The behaviour of an ideal resistor is dictated by the relationship specified by Ohm's law: $V=IR$

Ohm's law states that the voltage (V) across a resistor is proportional to the current (I), where the constant of proportionality is the resistance (R). For example, if a 300 ohm resistor is attached across the terminals of a 12 volt battery, then a current of $12 / 300 = 0.04$ amperes flows through that resistor.

Practical resistors also have some inductance and capacitance which affect the relation between voltage and current in alternating current circuits.



Figure 3.14 Resistor

3.8 Power supply

A power supply is an electrical device that supplies electric power to an electrical load. The primary function of a power supply is to convert electric current from a source to the correct voltage, current, and frequency to power the load. As a result, power supplies are sometimes referred to as electric power converters. Some power supplies are separate standalone pieces of equipment, while others are built into the load appliances that they power. Examples of the latter include power supplies found in desktop computers and consumer electronics devices. Other functions that power supplies may perform include limiting the current drawn by the load to safe levels, shutting off the current in the event of an electrical fault, power conditioning to prevent electronic noise or voltage surges on the input from reaching the load, power-factor correction, and storing energy so it can continue to power the load in the event of a temporary interruption in the source power.

All power supplies have a power input connection, which receives energy in the form of electric current from a source, and one or more power output connections that deliver current to the load. The source power may come from the electric power grid, such as an electrical outlet, energy storage devices such as batteries or fuel cells, generators or alternators, solar power converters, or another power supply. The input

and output are usually hardwired circuit connections, though some power supplies employ wireless energy transfer to power their loads without wired connections. Some power supplies have other types of inputs and outputs as well, for functions such as external monitoring and control.

3.8.1 DC power supply

A DC power supply is one that supplies a constant DC voltage to its load. Depending on its design, a DC power supply may be powered from a DC source or from an AC source such as the power mains.

3.8.1.1 AC-to-DC Power supply

DC power supplies use AC mains electricity as an energy source. Such power supplies will employ a transformer to convert the input voltage to a higher or lower AC voltage. A rectifier is used to convert the transformer output voltage to a varying DC voltage, which in turn is passed through an electronic filter to convert it to an unregulated DC voltage.

The filter removes most, but not all of the AC voltage variations; the remaining AC voltage is known as ripple. The electric load's tolerance of ripple dictates the minimum amount of filtering that must be provided by a power supply. In some applications, high ripple is tolerated and therefore no filtering is required. For example, in some battery charging applications it is possible to implement a mains-powered DC power supply with nothing more than a transformer and a single rectifier diode, with a resistor in series with the output to limit charging current.

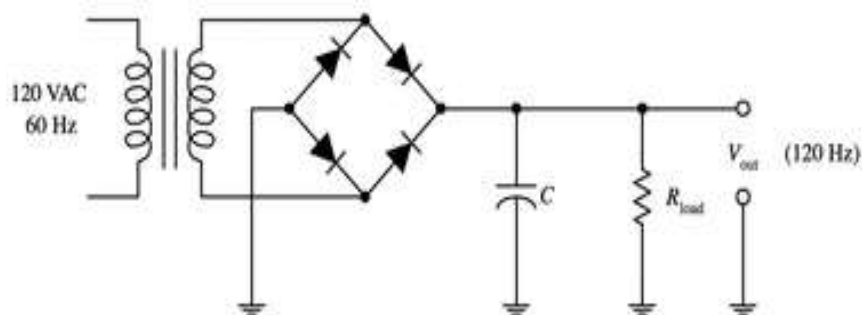


Figure 3.15 Schematic of basic AC-to-DC power supply

3.8.1.2 Switched-mode power supply

In a switched-mode power supply (SMPS), the AC mains input is directly rectified and then filtered to obtain a DC voltage. The resulting DC voltage is then switched on and off at a high frequency by electronic switching circuitry, thus producing an AC current that will pass through a high-frequency transformer or inductor. Switching occurs at a very high frequency (typically 10 kHz — 1 MHz), thereby enabling the use of transformers and filter capacitors that are much smaller, lighter, and less expensive than those found in linear power supplies operating at mains frequency. After the inductor or transformer secondary, the high frequency AC is rectified and filtered to produce the DC output voltage. If the SMPS uses an adequately insulated high-frequency transformer, the output will be electrically isolated from the mains, this feature is often essential for safety.

Switched-mode power supplies are usually regulated, and to keep the output voltage constant, the power supply employs a feedback controller that monitors current drawn by the load. The switching duty cycle increases as power output requirements increase.

SMPSs often include safety features such as current limiting or a crowbar circuit to help protect the device and the user from harm.[1] In the event that an abnormal high-current power draw is detected, the switched-mode supply can assume this is a direct short and will shut itself down before damage is done. PC power supplies often provide a power good signal to the motherboard; the absence of this signal prevents operation when abnormal supply voltages are present.

Some SMPSs have an absolute limit on their minimum current output.[2] They are only able to output above a certain power level and cannot function below that point. In a no-load condition the frequency of the power slicing circuit increases to great speed, causing the isolated transformer to act as a Tesla coil, causing damage due to the resulting very high voltage power spikes. Switched-mode supplies with protection circuits may briefly turn on but then shut down when no load has been detected. A very small low-power dummy load such as a ceramic power resistor or 10-watt light bulb can be attached to the supply to allow it to run with no primary load attached.

The switch-mode power supplies used in computers have historically had low power factors and have also been significant sources of line interference (due to induced power line harmonics and transients). In simple switch-mode power supplies, the input stage may distort the line voltage waveform, which can adversely affect other loads (and result in poor power quality for other utility customers), and cause unnecessary heating in wires and distribution equipment. Furthermore, customers incur higher electric bills when operating lower power factor loads. To circumvent these problems, some computer switch-mode power supplies perform power factor correction, and may employ input filters or additional switching stages to reduce line interference.

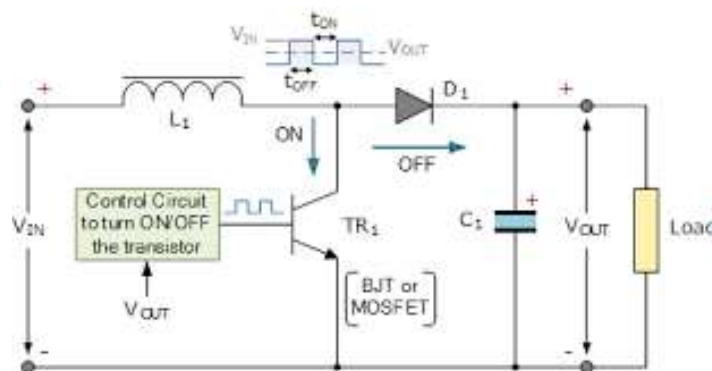


Figure 3.16 Switched mode power supply

3.8.1.3 Line Regulator

The function of a linear voltage regulator is to convert a varying DC voltage to a constant, often specific, lower DC voltage. In addition, they often provide a current limiting function to protect the power supply and load from overcurrent (excessive, potentially destructive current).

A constant output voltage is required in many power supply applications, but the voltage provided by many energy sources will vary with changes in load impedance. Furthermore, when an unregulated DC power supply is the energy source, its output voltage will also vary with changing input voltage. To circumvent this, some power supplies use a linear voltage regulator to maintain the output voltage at a steady value, independent of fluctuations in input voltage and load impedance. Linear regulators can also reduce the magnitude of ripple and noise on the output voltage.

3.8.2 AC Power Supplies

An AC power supply typically takes the voltage from a wall outlet (mains supply) and uses a transformer to step up or step down the voltage to the desired voltage. Some filtering may take place as well. In some cases, the source voltage is the same as the output voltage; this is called an isolation transformer. Other AC power supply transformers do not provide mains isolation; these are called autotransformers; a variable output autotransformer is known as a variac. Other kinds of AC power supplies are designed to provide a nearly constant current, and output voltage may vary depending on impedance of the load. In cases when the power source is direct current, (like an automobile storage battery), an inverter and step-up transformer may be used to convert it to AC power. Portable AC power may be provided by an alternator powered by a diesel or gasoline engine (for example, at a construction site, in an automobile or boat, or backup power generation for emergency services) whose current is passed to a regulator circuit to provide a constant voltage at the output. Some kinds of AC power conversion do not use a transformer. If the output voltage and input voltage are the same, and primary purpose of the device is to filter AC power, it may be called a line conditioner. If the device is designed to provide backup power, it may be called an uninterruptable power supply. A circuit may be designed with a voltage multiplier topology to directly step-up AC power; formerly, such an application was a vacuum tube AC/DC receiver.

In modern use, AC power supplies can be divided into single phase and three phase systems. "The primary difference between single phase and three phase AC power is the constancy of delivery." [3] AC power Supplies can also be used to change the frequency as well as the voltage, they are often used by manufacturers to check the suitability of their products for use in other countries. 230 V 50 Hz or 115 60 Hz or even 400 Hz for avionics testing.

3.8.2.1 AC Adapter

An AC adapter is a power supply built into an AC mains power plug. AC adapters are also known by various other names such as "plug pack" or "plug-in adapter", or by slang terms such as "wall wart". AC adapters typically have a single AC or DC output that is conveyed over a hardwired cable to a connector, but some adapters

have multiple outputs that may be conveyed over one or more cables. "Universal" AC adapters have interchangeable input connectors to accommodate different AC mains voltages.

Adapters with AC outputs may consist only of a passive transformer (plus a few diodes in DC-output adapters), or they may employ switch-mode circuitry. AC adapters consume power (and produce electric and magnetic fields) even when not connected to a load; for this reason they are sometimes known as "electricity vampires", and may be plugged into power strips to allow them to be conveniently turned on and off.



Figure 3.17 AC adapter

3.8.2.2 Programmable power supply

A programmable power supply is one that allows remote control of its operation through an analog input or digital interface such as RS232 or GPIB. Controlled properties may include voltage, current, and in the case of AC output power supplies, frequency. They are used in a wide variety of applications, including automated equipment testing, crystal growth monitoring, semiconductor fabrication, and x-ray generators.

power supplies typically employ an integral microcomputer to control and monitor power supply operation. Power supplies equipped with a computer interface may use proprietary communication protocols or standard protocols and device control languages such as SCPI.



Figure 3.18 Programmable power supply

3.8.2.3 Uninterruptible power supply

An uninterruptible power supply (UPS) takes its power from two or more sources simultaneously. It is usually powered directly from the AC mains, while simultaneously charging a storage battery. Should there be a dropout or failure of the mains, the battery instantly takes over so that the load never experiences an interruption. Instantly here should be defined as the speed of electricity within conductors which is somewhat near the speed of light. That definition is important because transmission of high speed data and communications service must have continuity/NO break of that service. Some manufacturers use a quasi-standard of 4 milliseconds. However, with high speed data even 4 ms of time in transitioning from one source to another is not fast enough. The transition must be made in a break before make method. The UPS meeting that requirement is referred to as a True UPS or a Hybrid UPS. How much time the UPS will provide is most often based on batteries and in conjunction with generators. That time can range from a quasi-minimum 5 to 15 minutes to literally hours or even days. In many computer installations, only enough time on batteries to give the operators time to shut down the system in an orderly way. Other UPS schemes may use an internal combustion engine or turbine to supply power during a utility power outage and the amount of battery time is then dependent upon how long it takes the generator to be on line and the criticality of the equipment served. Such a scheme is found in hospitals, data centres, call centres, cell sites and telephone central offices.

3.8.2.4 High-voltage power supply

A high-voltage power supply is one that outputs hundreds or thousands of volts. A special output connector is used that prevents arcing, insulation breakdown and accidental human contact. Federal Standard connectors are typically used for applications above 20 kV, though other types of connectors (e.g., SHV connector) may be used at lower voltages. Some high-voltage power supplies provide an analog input or digital communication interface that can be used to control the output voltage. High-voltage power supplies are commonly used to accelerate and manipulate electron and ion beams in equipment such as x-ray generators, electron microscopes, and focused ion beam columns, and in a variety of other applications, including electrophoresis and electrostatics.

High-voltage power supplies typically apply the bulk of their input energy to a power inverter, which in turn drives a voltage multiplier or a high turns ratio, high-voltage transformer, or both (usually a transformer followed by a multiplier) to produce high voltage. The high voltage is passed out of the power supply through the special connector and is also applied to a voltage divider that converts it to a low-voltage metering signal compatible with low-voltage circuitry. The metering signal is used by a closed-loop controller that regulates the high voltage by controlling inverter input power, and it may also be conveyed out of the power supply to allow external circuitry to monitor the high-voltage output.



Figure 3.19A 30 kV high-voltage power supply

3.8.2.5 Bipolar power supply

A bipolar power supply operates in all four quadrants of the voltage/current Cartesian plane, meaning that it will generate positive and negative voltages and currents as required to maintain regulation.[4] When its output is controlled by a low-level analog signal, it is effectively a low-bandwidth operational amplifier with high output power and seamless zero-crossings. This type of power supply is commonly used to power magnetic devices in scientific applications.



Figure 3.20 A bipolar power supply

4. Software

4.1 Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub `main()` into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program `avrdude` to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

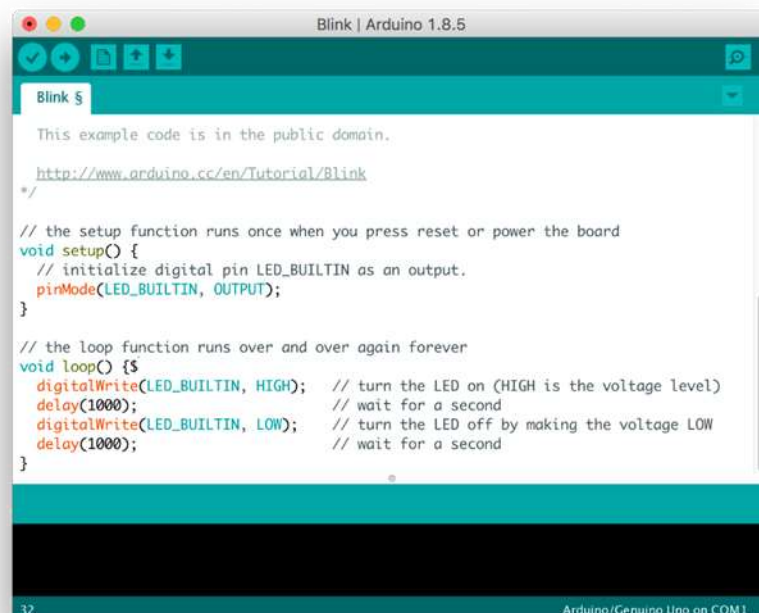


Figure 4.1 Screenshot of arduino IDE

4.2 Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension `.ino`. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

4.3 Uploading Sketches

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

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

4.4 Libraries

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

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

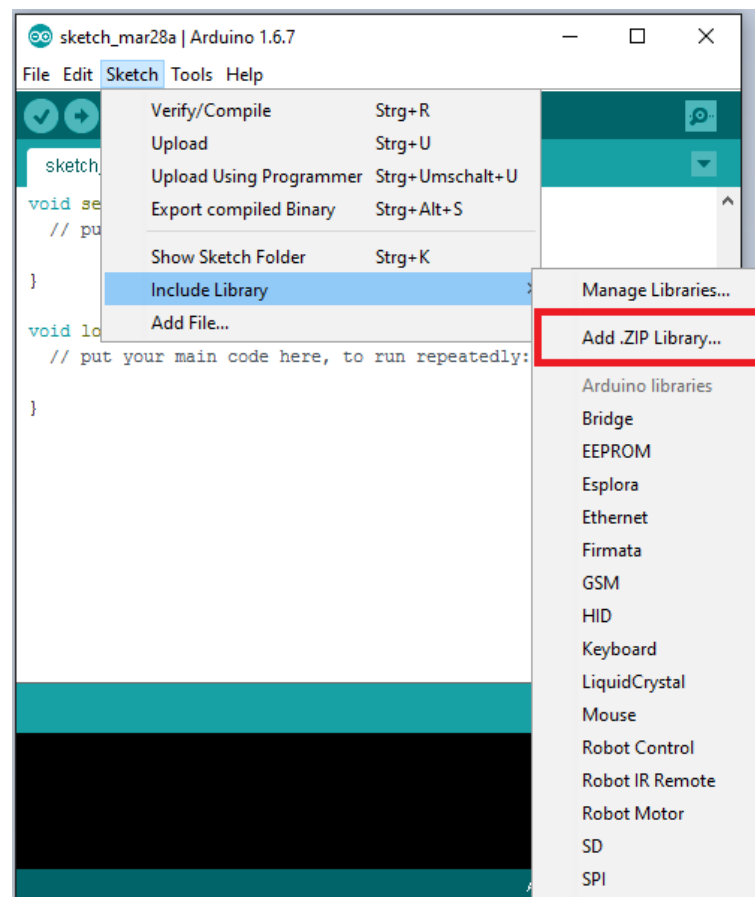


Figure 4.2 Screenshot of libraries

4.5 Serial monitor and Serial plotter

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

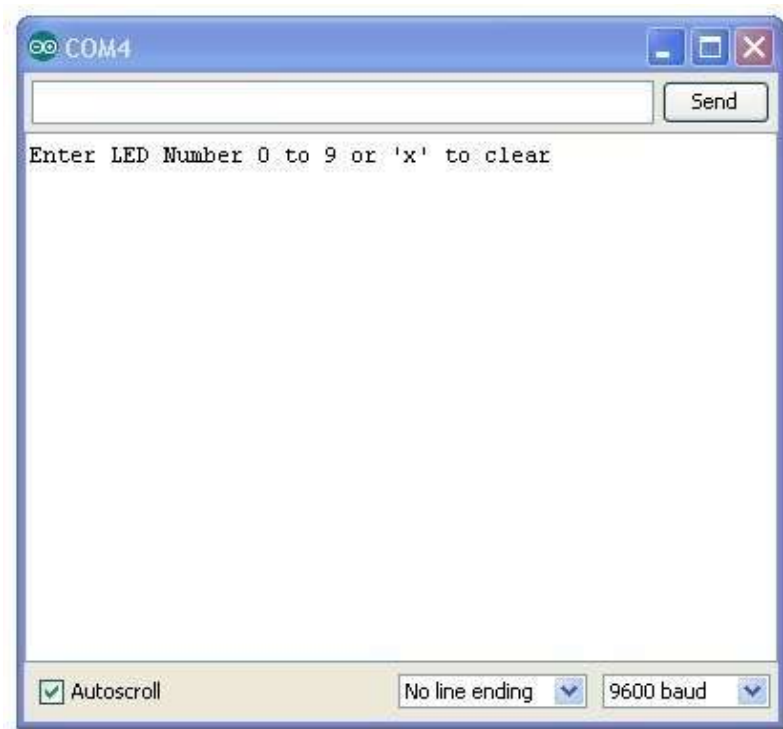


Figure 4.3 Screenshot of serial monitor

Arduino serial plotter is another component of the Arduino IDE, which allows you to generate a real-time graph of your serial data. The serial plotter makes it much easier for you to analyze your data through a visual display. You're able to create graphs, negative value graphs, and conduct waveform analysis.

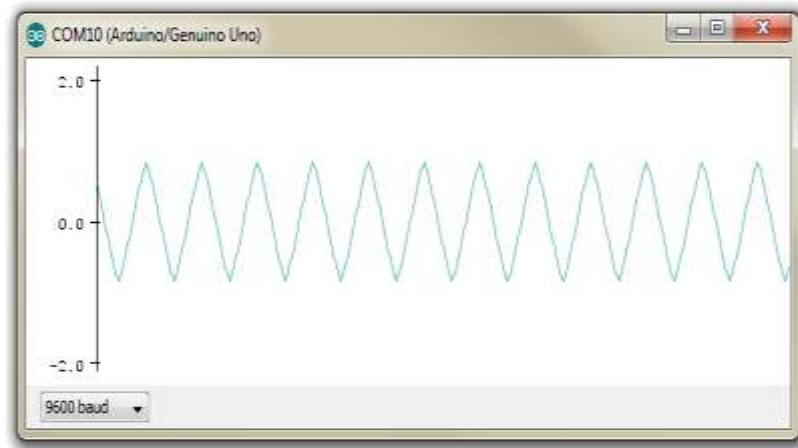


Figure 4.4 screenshot of serial plotter

4.6 Pin Description

To use the Arduino pins, you need to define which pin is being used and its functionality. A convenient way to define the used pins is by using:

```
'#define pinName pinNumber'.
```

The functionality is either input or output and is defined by using the `pinMode ()` method in the setup section.

4.7 Declarations

4.7.1 Variables

Whenever you're using Arduino, you need to declare global variables and instances to be used later on. In a nutshell, a variable allows you to name and store a value to be used in the future. For example, you would store data acquired from a sensor in order to use it later. To declare a variable you simply define its type, name and initial value.

It's worth mentioning that declaring global variables isn't an absolute necessity. However, it's advisable that you declare your variables to make it easy to utilize your values further down the line.

4.7.2 Instances

In software programming, a class is a collection of functions and variables that are kept together in one place. Each class has a special function known as a constructor, which is used to create an instance of the class. In order to use the functions of the class, we need to declare an instance for it.

Setup()

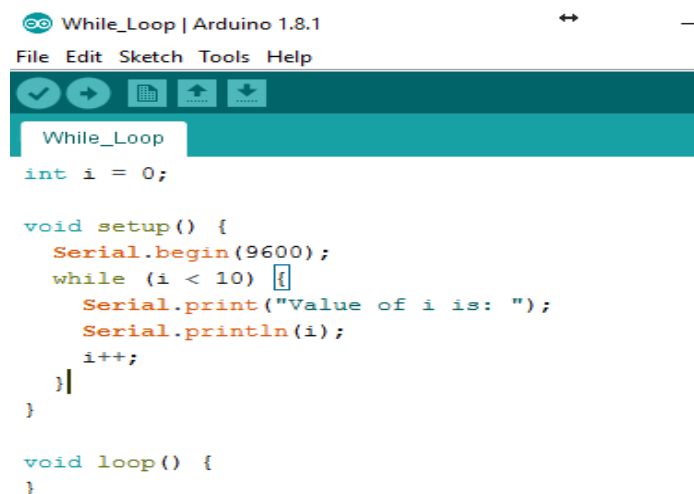
Every Arduino sketch must have a setup function. This function defines the initial state of the Arduino upon boot and runs only once.

Here we'll define the following:

1. Pin functionality using the pinMode function
2. Initial state of pins
3. Initialize classes
4. Initialize variables
5. Code logic

Loop()

The loop function is also a must for every Arduino sketch and executes once setup() is complete. It is the main function and as its name hints, it runs in a loop over and over again. The loop describes the main logic of your circuit.



```
While_Loop | Arduino 1.8.1
File Edit Sketch Tools Help
While_Loop
int i = 0;

void setup() {
  Serial.begin(9600);
  while (i < 10) {
    Serial.print("Value of i is: ");
    Serial.println(i);
    i++;
  }
}

void loop() {
}
```

Figure 4.5 Screenshot of loop and setup

4.8 How to program Arduino board

The basic Arduino code logic is an “if-then” structure and can be divided into 4 blocks:

Setup - will usually be written in the setup section of the Arduino code, and performs things that need to be done only once, such as sensor calibration.

Input - at the beginning of the loop, read the inputs. These values will be used as conditions (“if”) such as the ambient light reading from an LDR using `analogRead()`.

Manipulate Data - this section is used to transform the data into a more convenient form or perform calculations. For instance, the `AnalogRead()` gives a reading of 0-1023 which can be mapped to a range of 0-255 to be used for PWM.(see `analogWrite()`)

Output - this section defines the final outcome of the logic (“then”) according to the data calculated in the previous step. Looking at our example of the LDR and PWM, turn on an LED only when the ambient light level goes below a certain threshold.

4.9 Arduino code libraries

4.9.1 Library Structure

A library is a folder comprised of files with C++ (.cpp) code files and C++ (.h) header files.

The **.h** file describes the structure of the library and declares all its variables and functions.

The **.cpp** file holds the function implementation.

4.9.2 Importing Libraries

The first thing you need to do is find the library you want to use out of the many libraries available online. After downloading it to your computer, you just need to open Arduino IDE and click on Sketch > Include Library > Manage Libraries. You can then select the library that you want to import into the IDE. Once the process is complete the library will be available in the sketch menu.

In the code provided by circuito.io instead of adding external libraries like mentioned before, we provide them with the firmware folder. In this case, the IDE knows how to find them when using `#include`.

4.9.3 From Software to Hardware

There is a lot to be said of Arduino's software capabilities, but it's important to remember that the platform is comprised of both software and hardware. The two work in tandem to run a complex operating system.

Code → Compile → Upload → Run

At the core of Arduino, is the ability to compile and run the code.

After writing the code in the IDE you need to upload it to the Arduino. Clicking the upload button (the right-facing arrow icon), will compile the code and upload it if it passed compilation. Once your upload is complete, the program will start running automatically.

You can also do this step by step:

First, compile the code. To do this simply click the check icon (or click on sketch > Verify / Compile in the menu bar.

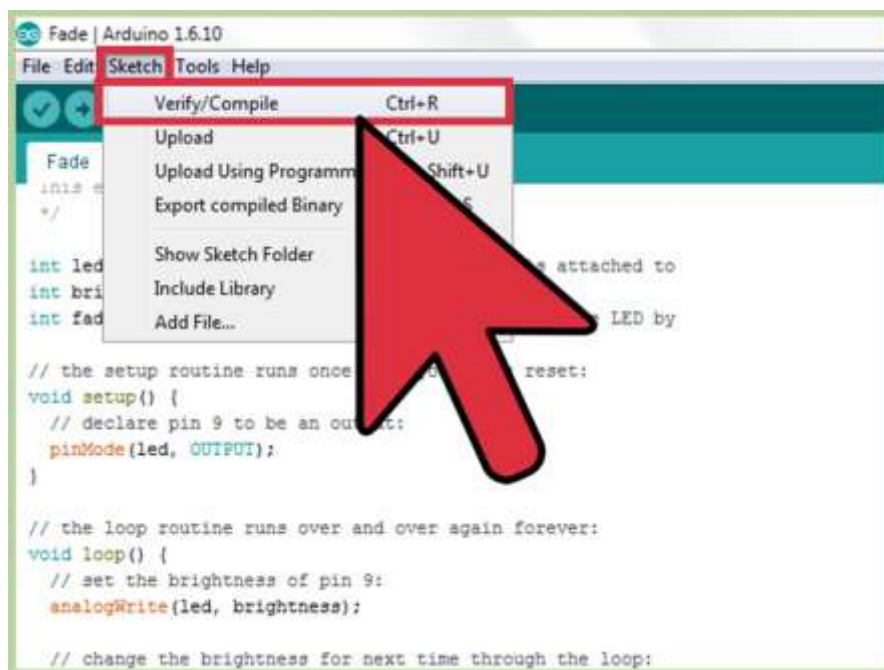


Figure 4.6 screenshot to compile code

As you can see, the check icon is located in the top left underneath the “File” tag in the menu section.

Once you’ve done this, Arduino will start to compile. Once it’s finished, you’ll receive a completion message that looks like this:

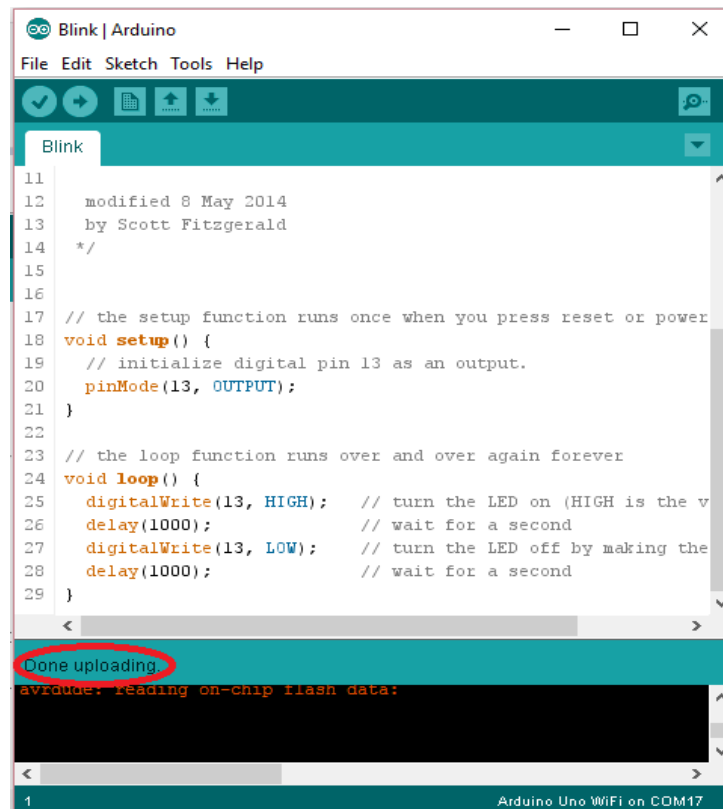


Figure 4.7 Screenshot to show completion message

As you can see, the green line at the bottom of the page tells you that you’re “done compiling”. If your code fails to run, you’ll be notified in the same section, and the problematic code will be highlighted for editing.

Once you’ve compiled your sketch, it’s time to upload it.

1. Choose the serial port your Arduino is currently connected to. To do this click on Tools > Serial port in the menu to designate your chosen serial port (as shown earlier above). You can then upload the compiled sketch.

2. To upload the sketch, click on the upload icon next to the tick. Alternatively you can go to the menu and click File > upload. Your Arduino LEDs will flicker once the data is being transferred.

Once complete, you'll be greeted with a completion message that tells you Arduino has finished uploading.

4.9.4 Setting Up Your IDE

In order to connect an Arduino board to your computer you need a USB cable. When using the Arduino UNO, the USB transfers the data in the program directly to your board. The USB cable is used to power your arduino. You can also run your Arduino through an external power source.

Before you can upload the code, there are some settings that you need to configure.

Choose your board - You need to designate which Arduino board you're going to be using. Do this by click Tools > Board > Your Board.

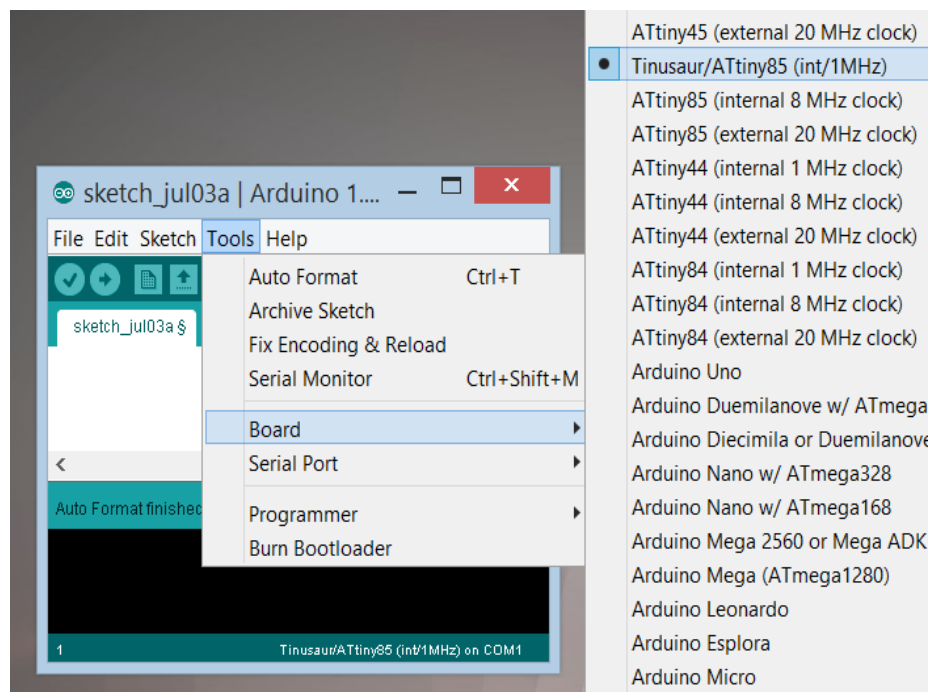


Figure 4.8 Screenshot to setup IDE

4.10 Boards

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

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

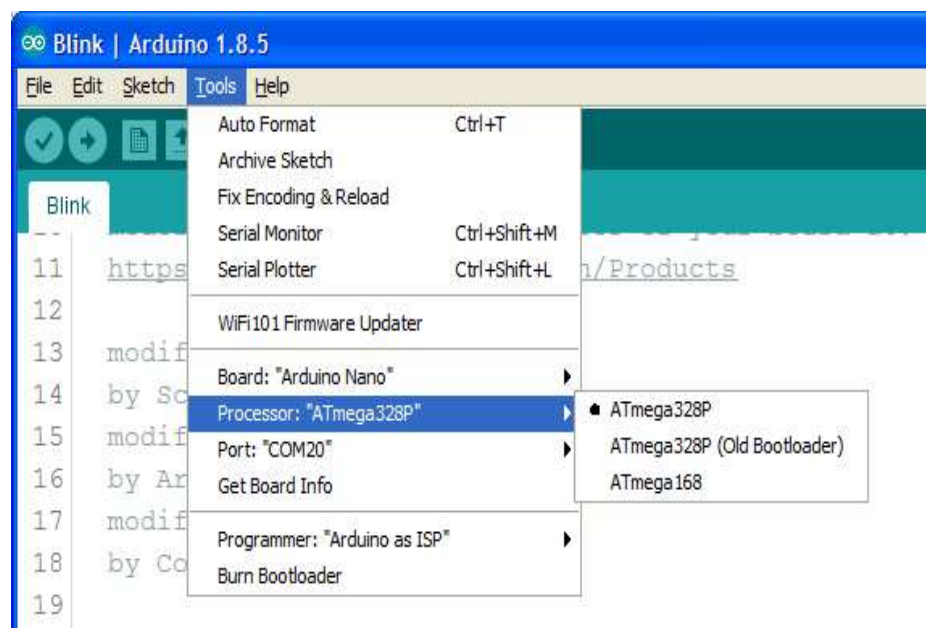


Figure 4.9 Screenshot to show boards

4.11 List of Boards

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

2. Arduino/Genuino Uno - An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
3. Arduino Diecimila or Duemilanove w/ ATmega168 - An ATmega168 running at 16 MHz with auto-reset.
4. Arduino Nano w/ ATmega328P - An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.
5. Arduino/Genuino Mega 2560 - An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
6. Arduino Mega - An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
7. Arduino Mega ADK - An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
8. Arduino Leonardo - An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
9. Arduino/Genuino Micro - An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
10. Arduino Esplora - An ATmega32u4 running at 16 MHz with auto-reset.
11. Arduino Mini w/ ATmega328P - An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
12. Arduino Ethernet - Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
13. Arduino Fio - An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.
14. Arduino BT w/ ATmega328P - ATmega328P running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM.

15. LilyPad Arduino USB - An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
16. LilyPad Arduino - An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
17. Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P - An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.
18. Arduino NG or older w/ ATmega168 - An ATmega168 running at 16 MHz without auto-reset. Compilation and upload are equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.
19. Arduino Robot Control - An ATmega328P running at 16 MHz with auto-reset.
20. Arduino Robot Motor - An ATmega328P running at 16 MHz with auto-reset.
21. Arduino Gemma - An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

5. Flowcharts and Connections

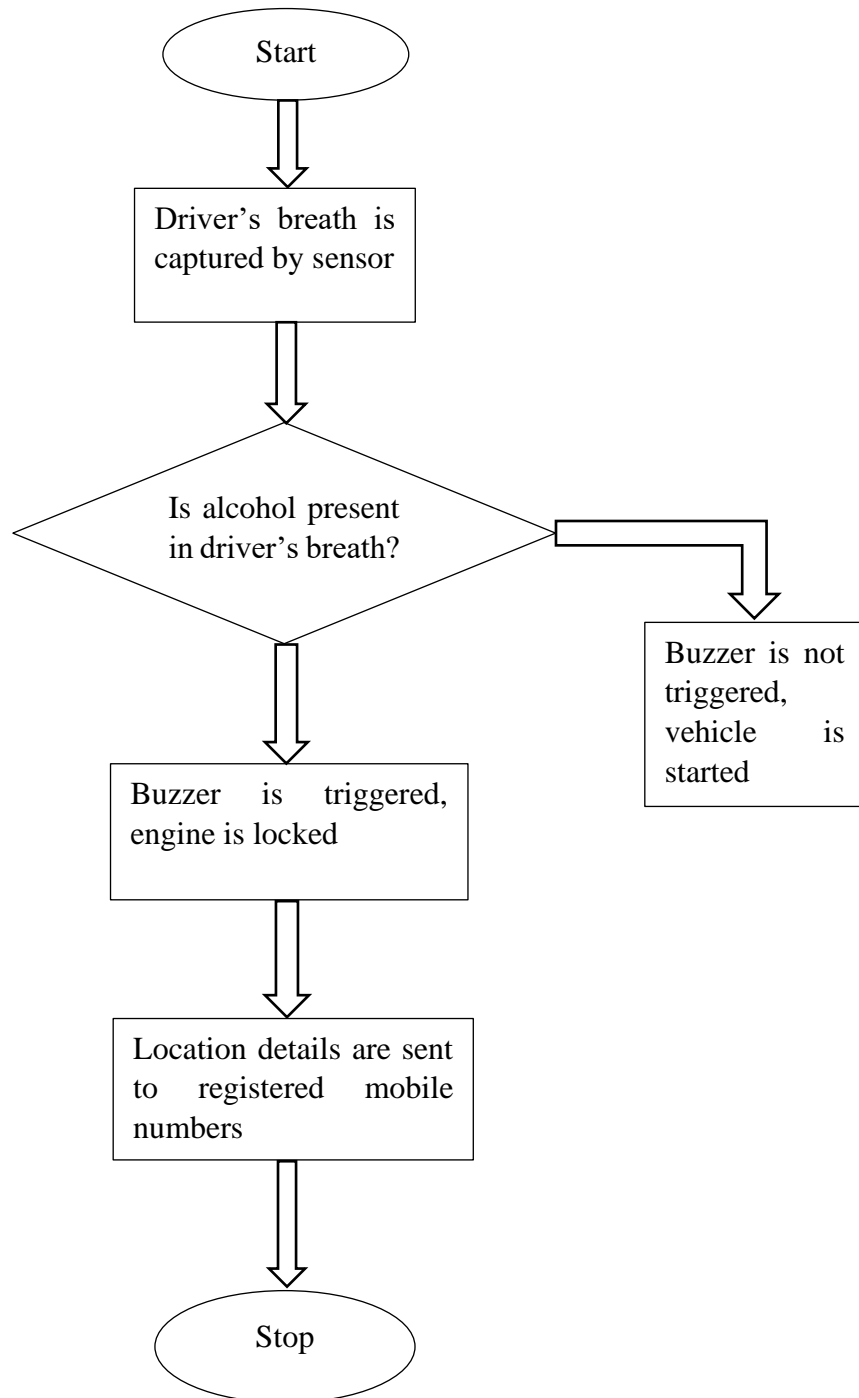


Figure 5.1 Flowchart of alcohol detection system

The steps involved the working process of the proposed system are :

1. The input for the Microcontroller is identified by the alcohol sensor through the breath of a human.
2. If the driver has consumed alcohol, then exhaled carbon dioxide which contains ethanol concentration is measured by the sensor.
3. In the next scenario the levels of alcohol measured by the sensor are compared with the threshold.
4. If the threshold of consumption of alcohol is less than the alcohol consumed by the person, the system of activating relay is initiated which in turn activates the automatic lock on the vehicle, the system will lock the Engine at the same time.
5. The micro controller reads the data from the GPS unit which gives the position of the vehicle.
6. Then it sends the message to the user registered mobile number with the help of GSM modem.

5.1 Interfacing GSM module with Arduino Uno

Connect the Arduino with the SIM800L module, according to the below table 5.1

Table 5.1 Connections of GSM module

Arduino Uno	SIM800L module
+5V	VCC pin
GND	GND pin
PIN 10	TXD pin
PIN 11	RXD pin

If all the wiring connections are checked and confirmed as correct, go to the Arduino program and do the coding. Before coding, you must identify the correct direction of the SIM card. It is normally printed on the surface of the SIM slot.

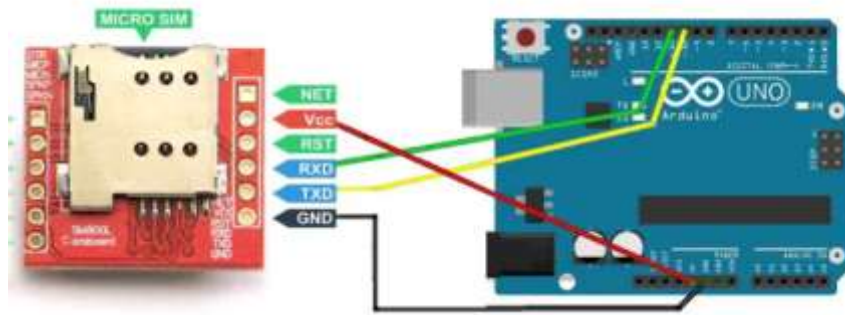


Figure 5.2 Interfacing GSM module with Arduino

5.2 Interfacing GPS module to Arduino Uno

To connect your GPS module to Arduino, use a +5V from the power side of the Arduino and any ground pin. Any two pins will work for the serial communication

Table 5.2 Connections GPS module to Arduino Uno

Arduino Uno	GPS module
5V	VCC
GND	GND
Pin 8	RX
Pin 9	TX

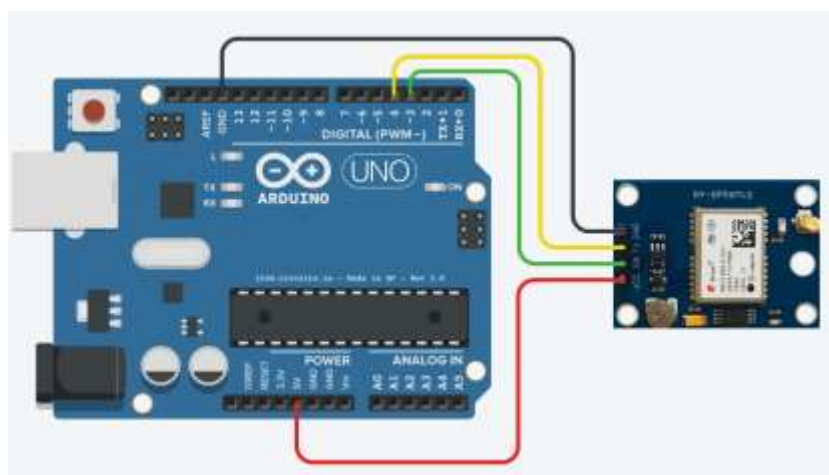


Figure 5.3 Interfacing of GPS module to Arduino Uno

5.3 Interfacing of MQ3 Alcohol Sensor to Arduino Uno

To connect the sensor, there are 4 leads. 2 of them are for power. The +5V terminal of the sensor connects into the 5V terminal of the arduino board. The GND terminal of the sensor connects into the GND terminal of the arduino. This establishes power for the sensor. The other 2 connections are the analog and digital outputs of the sensor. The connected to digital pin D11, respectively.

Table 5.3 Connections of MQ3 sensor with Arduino Uno

Arduino Uno	Alcohol Sensor
+5V	VCC
GND	GND
D11	DOUT
-	AOUT

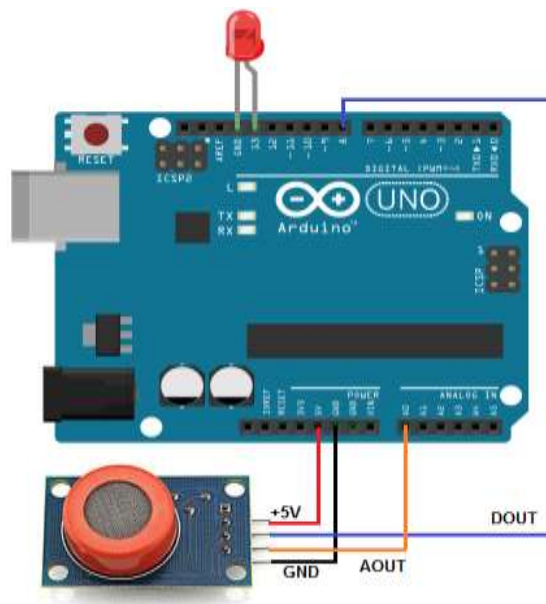


Figure 5.4 Interfacing of MQ3 sensor with Arduino Uno

5.4 Interfacing of DC motor to Arduino Uno

The DC motor cooling fan internally contains a small Hall-Effect sensor to aware how fast the fan blades are rotating. Here we can use this sensor output and Interface with Arduino to serially print RPM value. It has Red wire for +Vcc, and black for Ground supply finally the Yellow gives signal output. Pull up resistor gives strength to the signal and may connected to external micro-controller.

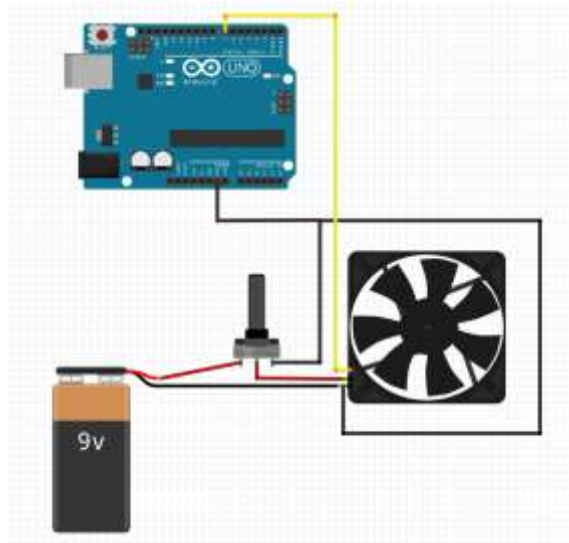


Figure 5.5 Interfacing of DC motor to Arduino Uno

5.5 Interfacing of LCD with Arduino Uno

LCDs (Liquid Crystal Displays) are used in embedded system applications for displaying various parameters and status of the system. LCD 16x2 is a 16-pin device that has 2 rows that can accommodate 16 characters each. LCD 16x2 can be used in 4-bit mode or 8-bit mode. It is also possible to create custom characters. It has 8 data lines and 3 control lines that can be used for control purposes. For more information about LCD 16x2 and how to use it, refer the topic LCD 16x2 module in the sensors and modules section.

To wire your LCD screen to your board, connect the following pins:

1. LCD RS pin to digital pin 12
2. LCD Enable pin to digital pin 11

3. LCD D4 pin to digital pin 5
4. LCD D5 pin to digital pin 4
5. LCD D6 pin to digital pin 3
6. LCD D7 pin to digital pin 2

Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector

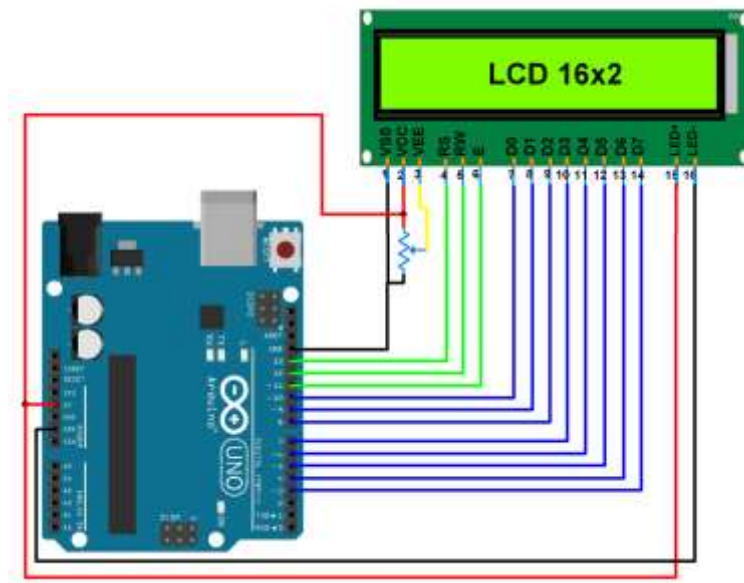


Figure 5.6 Interfacing of LCD with Arduino Uno

5.5 Output of the Alcohol detection system

The alcohol sensor placed inside the vehicle monitors the driver's body and checks the alcohol content. If driver is not under the influence of the alcohol, the user's registered number will get a message as "Vehicle started at location: latitude value, longitude value," and if the driver is under the influence of the alcohol, the buzzer will be on and the vehicle is stopped automatically. At the same time the micro controller reads the data from GPS module and sends the location details to user's registered mobile number with the help of GSM module. The user's registered mobile number will get a message as "Alcohol detected vehicle stopped at location: latitude value, longitude value".



Figure 5.7 Circuit connections of alcohol detection system using GSM and GPS technologies

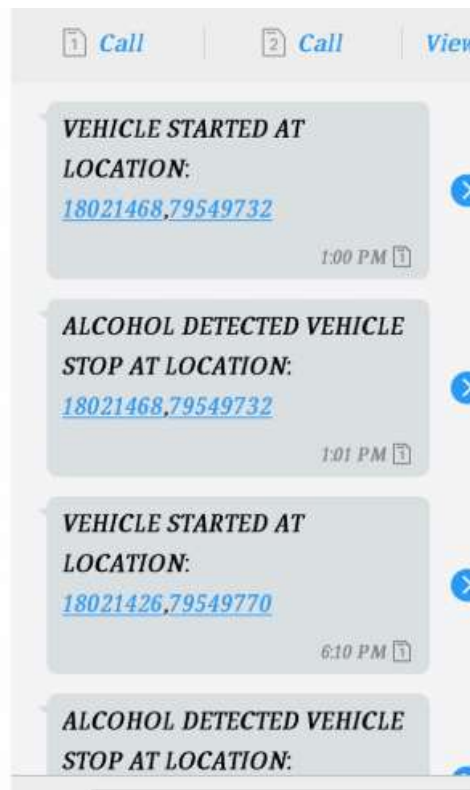


Figure 5.8 Output

6. Source Code

```
#include <SoftwareSerial.h>

#include<LiquidCrystal.h>

#include <TinyGPS.h>

TinyGPS gps; // create gps object

SoftwareSerial ser(8,9); ///rx tx

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

char msg[50],cmd[20];

long latt,lon; // create variable for latitude and longitude object

long z,y;

long x1_d,x1_r,y1_d,y1_r;

int x,i,g,m,v;

int x_v,t_v,sw_v,s_v,alch_v;

int buz=10;

int alch=11;

int mot=12;

int sw_start=13;

void setup()

{

m=1;

Serial.begin(9600);

delay(1000);

ser.begin(9600);

pinMode(alch,INPUT);
```



```

pinMode(alch,INPUT);

pinMode(sw_start,INPUT_PULLUP);

pinMode(mot,OUTPUT);

pinMode(buz,OUTPUT);

digitalWrite(mot,LOW);

digitalWrite(buz,LOW);

lcd.begin(16,2);

lcd.print("  WELCOME ");

delay(1000);

ser.println("AT+CMGF=1");

delay(1000);

ser.println("AT+CNMI=2,2,0,0,0");

delay(1000);

Serial.begin(9600);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("INITILIZING GPS...");

lcd.clear();

lcd.setCursor(0,0);

lcd.clear();

lcd.setCursor(0,0);

lcd.print("LAT:");

lcd.setCursor(0,1);

lcd.print("LON:");

```

```

}

void loop()

{
  /*

  Serial.print("MEMS_X:");

  Serial.println(analogRead(mems_x));

  Serial.print("MEMS_Y:");

  Serial.println(analogRead(mems_y));

  Serial.print("ALCH:");

  Serial.println(digitalRead(alch));

  Serial.print("RF:");

  Serial.println(digitalRead(rf));

  Serial.print("SW_START:");

  Serial.println(digitalRead(sw_start));

  Serial.print("sw_stop:");

  Serial.println(digitalRead(sw_stop));

  Serial.println();

  Serial.println();

  delay(1000);

  */

  xx:

  //while(Serial.available())

  //{

  // check for gps data

```

```

if(gps.encode(Serial.read()))

{ // encode gps data

gps.get_position(&latt,&lon); // get latitude and longitude

//display position

//Serial.print("Position: ");

//Serial.print("lat: ");Serial.print(latt);Serial.print(" ");// print latitude

//Serial.print("lon: ");Serial.println(lon); // print longitude

}

//}

lcd.setCursor(4,0);

lcd.print(latt);

lcd.setCursor(4,1);

lcd.print(lon);

if( (!digitalRead(sw_start)) && (digitalRead(alch)) )

{

sw_v=1;

digitalWrite(mot,HIGH);

sendsms1();

sw_v=0;

while(1)

{

if(Serial.available())

{

x=Serial.read();

```

```

i=0;

if(x=='*')

{

while(x!='#')

{

cmd[i]=x=ser.read();

i++;

}

cmd[i]='\0';

for(i=0;cmd[i]!='\0';i++)

{

Serial.write(cmd[i]);

}

if(cmd[0]=='T')

{

lcd.setCursor(15, 1);

lcd.print("T");

t_v=1;

sendsms1();

delay(500);

lcd.setCursor(15, 1);

lcd.print(" ");

t_v=0;

}

```

```

if(cmd[0]=='S')
{
  lcd.setCursor(15, 1);

  lcd.print("S");

  analogWrite(mot,255);

  s_v=1;

  sendsms1();

  delay(500);

  lcd.setCursor(15, 1);

  lcd.print(" ");

  s_v=0;
}

if(cmd[0]=='X')
{
  lcd.setCursor(15, 1);

  lcd.print("X");

  x_v=1;

  analogWrite(mot,0);

  sendsms1();

  delay(500);

  lcd.setCursor(15, 1);

  lcd.print(" ");

  x_v=0;
}

```

```

    }

    }

    if(!digitalRead(alch))

    {

        alch_v=1;

        digitalWrite(mot,LOW);

        digitalWrite(buz,HIGH);

        lcd.setCursor(15,0);

        lcd.print("A");

        delay(1000);

        sendsms1();

        lcd.setCursor(15,0);

        lcd.print(" ");

        digitalWrite(buz,LOW);

        alch_v=0;

    }

    }

    }

    if( (!digitalRead(sw_start)) && !(digitalRead(alch)) )

    {

        if(!digitalRead(alch))

        {

            lcd.setCursor(15,0);

            lcd.print("A");

```

```

    }

    digitalWrite(buz,HIGH); /////OON ON ON

    digitalWrite(mot,LOW);

    delay(2000);

    digitalWrite(buz,LOW);

    lcd.setCursor(15,0);

    lcd.print(" ");

}

if(ser.available())

{

    x=ser.read();

    i=0;

    if(x=='*')

    {

        while(x!='#')

        {

            cmd[i]=x=ser.read();

            i++;

        }

        cmd[i]='\0';

        for(i=0;cmd[i]!='\0';i++)

        {

            Serial.write(cmd[i]);

        }

```

```

if(cmd[0]=='T')

{

lcd.setCursor(15, 1);

lcd.print("T");

t_v=1;

sendsms1();

delay(500);

lcd.setCursor(15, 1);

lcd.print(" ");

t_v=0;

}

if(cmd[0]=='S')

{

lcd.setCursor(15, 1);

lcd.print("S");

analogWrite(mot,255);

s_v=1;

sendsms1();

delay(500);

lcd.setCursor(15, 1);

lcd.print(" ");

s_v=0;

}

if(cmd[0]=='X')

```



```

{
  lcd.setCursor(15, 1);

  lcd.print("X");

  x_v=1;

  analogWrite(mot,0);

  sendsms1();

  delay(500);

  lcd.setCursor(15, 1);

  lcd.print(" ");

  x_v=0;
}

}

}

}

void sendsms1()

{
  ser.println("AT");

  delay(500);

  ser.println("AT+CMGF=1");

  delay(500);

  ser.print("AT+CMGS=");

  ser.write("");

  ser.print("9705100881");

  ser.write("");
}

```

```

ser.println();

delay(500);

//ser.print("https://maps.google.com/maps?q=");

//delay(1000);

if(sw_v==1)

ser.println("VEHICLE STARTED AT LOCATION:");

if(s_v==1)

ser.println("VEHICLE STARTED AT LOCATION:");

if(x_v==1)

ser.println("VEHICLE STOPPED AT LOCATION:");

if(t_v==1)

ser.println("VEHICLE IDENTIFIED AT LOCATION:");

if(alch_v==1)

ser.println("ALCOHOL DETECTED VEHICLE STOP AT LOCATION:");

delay(1000);

//ser.print("https://maps.google.com/maps/@");

ser.print(latt);

ser.print(",");

ser.print(lon);

delay(500);

ser.write(0x1a);

ser.println();

delay(1000);

}

```

7. Conclusion

In this project we have developed a real time model that can automatically lock the engine when a drunken driver tries to drive a car. Now-a-days car accidents are mostly happening because of drunk and driving. As by fitting this alcohol sensor into the car, we can safeguard the life of the driver. The life time of the project is high. It has low or zero maintenance cost and of course low power consumption.

In this project GSM and GPS technologies have been used, with the help of these technologies it is easy to locate the person who is drunk and send the information of the vehicle to the user's registered mobile number.

This is a developed design to efficiently check drunken driving. By implementing this design a safe car journey is possible decreasing the accident rate due to drinking. By implementing this design, drunken drivers can be controlled so are the accidents due to drunken driving. Government must enforce laws to install such circuit in every car and must regulate all car companies to preinstall such mechanisms while manufacturing the car itself. If this is achieved the deaths due to drunken drivers can be brought to minimum level. In this type of system, future scope can be safely landing of car aside without disturbing other vehicles.

References

- [1] I Ahmad, MF Suhaimi, NAN Yusri, “ development of alcohol sensor detector with engine locking system for accident prevention,” AIP Conference Proceedings, 2019.
- [2] Monisha V, priyanga M, Yamini C, Sobiya P, “ Automatic Engine Locking System through alcohol detection in Arduino using IOT,” International Research Journal of Engineering and Technology (IRJET), 2018.
- [3] Priyanka Sahu, Sakshi Dixit, Shruti Mishra, Smriti Srivastava, “Alcohol Detection based Engine Locking System using MQ-3 Sensor,” International Research Journal of Engineering and Technology (IRJET),2017.
- [4] Amityakumartripathy, sejachopra, Samantha Bosco, SrinidhiShetty, FirdosSayyad, “Travolution-An Embedded System in passenger car for road safety,” International journal and magazine of engineering technology, management and research (IJMETMR), 2016.
- [5] R.Ramani, S.Valarmathy, Dr. N.SuthanthiraVanitha, S.Selvaraju, M.Thirupathi, R.Thangam, “Vehicle Tracking and Locking System Based on GSM and GPS”, I.J. Intelligent Systems and Applications, 2013.