IOT BASED INDUSTRIAL DATA ACQUISITION SYSTEM USING ARDUINO

A Project Report Submitted

In Partial Fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

INSTRUMENTATION ENGINEERING

Under the Esteemed Guidance of

Prof. M. Ramesh Patnaik

By

Ch. DEVI (316106816003)

G. SWETHA ROSHNI (316106816007)

K. KISHORE (316106816010)

K. KRISHNA VIJAY (316106816011)

S. FATIMA (316106816020)

T. VINEETH (316106816021)

DEPT. OF INSTRUMENTATION ENGINEERING
ANDHRA UNIVERSITY
COLLEGE OF ENGINEERING
VISAKHAPTNAM

2019-2020

DEPARTEMENT OF INSTRUMENTATION ENGINEERING

COLLEGE OF ENGINEERING

ANDHRA UNIVERSITY

VISAKHAPTNAM

CERTIFICATE

This is to certify that this project entitled "**IOT based Industrial Data Acquisition System using Arduino**" is Bonafide work of

Ch. DEVI (316106816003)

G. SWETHA ROSHNI (316106816007)

K. KISHORE (316106816010)

K. KRISHNA VIJAY (316106816011)

S. FATIMA (316106816020)

T. VINEETH (316106816021)

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Instrumentation Engineering, Andhra University , Visakhapatnam for the academic year 2019-2020.

Prof. M. Ramesh Patnaik
Project Guide
Dept. Of Instrumentation Technology
Andhra University,
Visakhapatnam-530003

Prof. Y. Srinivasa Rao
Head of the Department
Dept. Of Instrumentation Technology
Andhra University,
Visakhapatnam-530003

ACKNOWLEDGEMENTS

We express our deep sense of gratitude and indebt to our guide **Prof. M. Ramesh Patnaik**, Department of Instrumentation Engineering, Andhra University, who suggested this fine work and has been a source of inspiration all along for the success of venture.

We also express our thanks to **Prof. Y. Srinivasa Rao**, Head of the Department of Instrumentation Engineering his timely suggestions.

We also wish to convey our gratitude to all teaching and non-teaching Staff of the department of instrumentation engineering who had extended all their co-operation towards the completion of our project work.

Ch. Devi
G. Swetha Roshni
K. Kishore
K. Krishna Vijay
S. Fatima
T. Vineeth

ABSTRACT

Data Acquisition is the process of measuring and analyzing various electrical and physical entities like voltage, current, temperature, pressure etc. A DAQ system consists of sensors, signal conditioning circuitry, analog to digital converter, and application software. DAQ system has a wide range of applications including Research and Analysis, Control and automation, Design validation and Verification. DAQ's applications are not only limited to medical instruments, industrial equipment and other home appliances but are used for variety of products.

Generally DAQ system consists of Sensors, acquired data measurement hardware and computer software. In contrast our System will be equipped with Arduino Uno Rev3. It will be equipped with WI-FI for long range transmission. The system will be using thus various Internet of Things protocol options. We will also provide support for Android based application so that acquired data can be analyzed by our application using various means of communication. In this project we will be explaining working and acquisition of data using various sensors of several Industrial parameters like Temperature, Level, Gas Intensity, Light Intensity, and Infrared Rays. With the number of connectivity options and various connectivity ranges, the system is generalized for being used for several applications and also it is portable to use.

Keywords — Data Acquisition System, Arduino Uno Rev3, Internet of Things, Wi-Fi and Sensors.

Contents

CHAPTER-1

CHAPTER-2

CHAPTER-3

CHAPTER-4

CHAPTER-5

CHAPTER-6

CHAPTER-7

CHAPTER-8

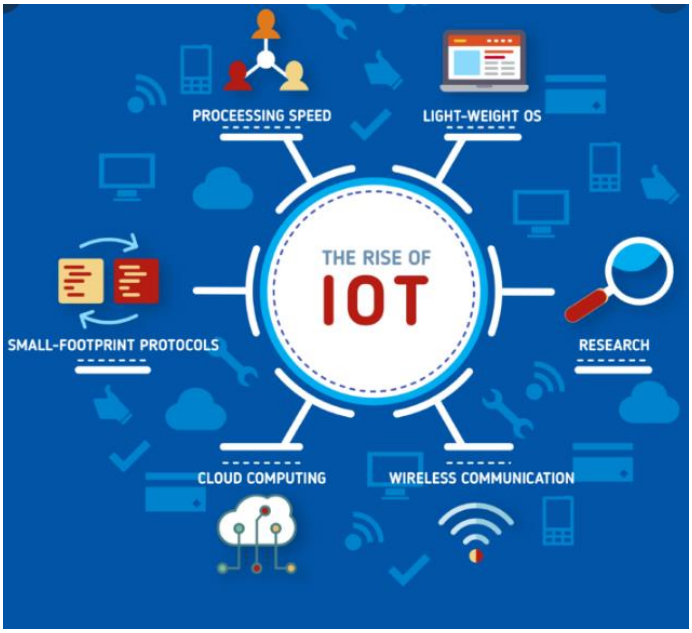
CHAPTER 1
INTRODUCTION

1.1 Internet of Things

1.1.1 IoT Overview

IoT (Internet of Things) is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system. IoT systems have applications across industries through their unique flexibility and ability to be suitable in any environment. They enhance data collection, automation, operations, and much more through smart devices and powerful enabling technology. IoT systems allow users to achieve deeper automation, analysis, and integration within a system.

They improve the reach of these areas and their accuracy. IoT utilizes existing and emerging technology for sensing, networking, and robotics. IoT exploits recent advances in software, falling hardware prices, and modern attitudes towards technology. Its new and advanced elements bring major changes in the delivery of products, goods, and services; and the social, economic, and political impact of those changes.

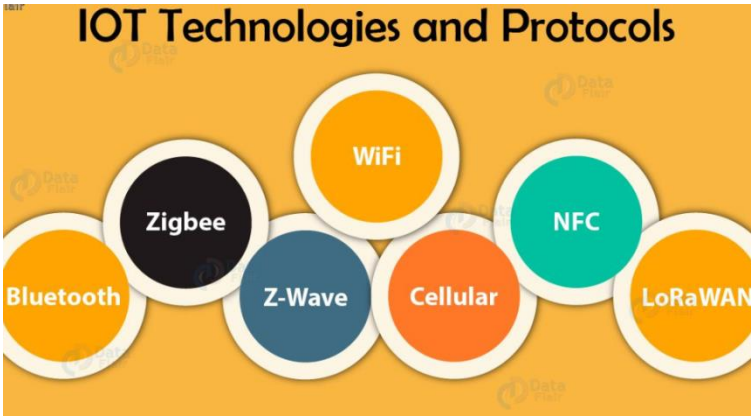


IoT − Key Feature

The most important features of IoT include artificial intelligence, connectivity, sensors, active engagement, and small device use. A brief review of these features is given below:

- **AI** – IoT essentially makes virtually anything "smart", meaning it enhances every aspect of life with the power of data collection, artificial intelligence algorithms, and networks. This can mean something as simple as enhancing your refrigerator and cabinets to detect when milk and your favorite cereal run low, and to then place an order with your preferred grocer.
- **Connectivity** – New enabling technologies for networking, and specifically IoT networking, mean networks are no longer exclusively tied to major providers. Networks can exist on a much smaller and cheaper scale while still being practical. IoT creates these small networks between its system devices.
- **Sensors** – IoT loses its distinction without sensors. They act as defining instruments which transform IoT from a standard passive network of devices into an active system capable of real-world integration.
- **Active Engagement** – Much of today's interaction with connected technology happens through passive engagement. IoT introduces a new paradigm for active content, product, or service engagement.
- **Small Devices** – Devices, as predicted, have become smaller, cheaper, and more powerful over time. IoT exploits purpose-built small devices to deliver its precision, scalability, and versatility.

1.1.2 IoT Technologies and Protocols



IoT primarily exploits standard protocols and networking technologies. However, the major enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and Wi-Fi-Direct. These technologies support the specific networking functionality needed in an IoT system in contrast to a standard uniform network of common systems.

NFC and RFID
RFID (radio-frequency identification) and NFC (near-field communication) provide simple, low energy, and versatile options for identity and access tokens, connection bootstrapping, and
payments.
- RFID technology employs 2-way radio transmitter-receivers to identify and track tags associated with objects.
- NFC consists of communication protocols for electronic devices, typically a mobile device and a standard device.

Low-Energy Bluetooth
This technology supports the low-power, long-use need of IoT function while exploiting a standard technology with native support across systems.

Low-Energy Wireless
This technology replaces the most power hungry aspect of an IoT system. Though sensors and other elements can power down over long periods, communication links (i.e., wireless) must remain in listening mode. Low-energy wireless not only reduces consumption, but also extends the life of the device through less use.

Radio Protocols
ZigBee, Z-Wave, and Thread are radio protocols for creating low-rate private area networks. These technologies are low-power, but offer high throughput unlike many similar options. This increases the power of small local device networks without the typical costs.

LTE-A
LTE-A, or LTE Advanced, delivers an important upgrade to LTE technology by increasing not only its coverage, but also reducing its latency and raising its throughput. It gives IoT a tremendous power through expanding its range, with its most significant applications being vehicle, UAV, and similar communication.

Wi-Fi-Direct
Wi-Fi-Direct eliminates the need for an access point. It allows P2P (peer-to-peer) connections with the speed of Wi-Fi, but with lower latency. Wi-Fi-Direct eliminates an element of a network that often bogs it down, and it does not compromise on speed or throughput.

1.1.3 IOT Common Uses

IoT has applications across all industries and markets. It spans user groups from those who want to reduce energy use in their home to large organizations who want to streamline their operations. It proves not just useful, but nearly critical in many industries as technology advances and we move towards the advanced automation imagined in the distant future.

Engineering, Industry, and Infrastructure

Applications of IoT in these areas include improving production, marketing, service delivery, and safety. IoT provides a strong means of monitoring various processes; and real transparency creates greater visibility for improvement opportunities.

The deep level of control afforded by IoT allows rapid and more action on those opportunities, which include events like obvious customer needs, nonconforming product, malfunctions in equipment, problems in the distribution network, and more.

Government and Safety

IoT applied to government and safety allows improved law enforcement, defense, city planning, and economic management. The technology fills in the current gaps, corrects many current flaws, and expands the reach of these efforts.

Home and Office

In our daily lives, IoT provides a personalized experience from the home to the office to the organizations we frequently do business with. This improves our overall satisfaction, enhances productivity, and improves our health and safety.

1.1.4 IoT Advantages

The advantages of IoT span across every area of lifestyle and business. Here is a list of some of the advantages that IoT has to offer:

- **Improved Customer Engagement** – Current analytics suffer from blind-spots and significant flaws in accuracy; and as noted, engagement remains passive. IoT completely transforms this to achieve richer and more effective engagement with audiences.
- **Technology Optimization** – The same technologies and data which improve the customer experience also improve device use, and aid in more potent improvements to technology. IoT unlocks a world of critical functional and field data.
- **Reduced Waste** – IoT makes areas of improvement clear. Current analytics give us superficial insight, but IoT provides real-world information leading to more effective management of resources.
- **Enhanced Data Collection** – Modern data collection suffers from its limitations and its design for passive use. IoT breaks it out of those spaces, and places it exactly where humans really want to go to analyze our world. It allows an accurate picture of everything.

1.1.5 IoT Disadvantages

Though IoT delivers an impressive set of benefits; it also presents a significant set of challenges.

Here is a list of some its major issues:

- **Security** – IoT creates an ecosystem of constantly connected devices communicating over networks. The system offers little control despite any security measures. This leaves users

exposed to various kinds of attackers.

- **Privacy** – The sophistication of IoT provides substantial personal data in extreme detail without the user's active participation.
- **Complexity** – Some find IoT systems complicated in terms of design, deployment, and maintenance given their use of multiple technologies and a large set of new enabling technologies.
- **Flexibility** – Many are concerned about the flexibility of an IoT system to integrate easily with another. They worry about finding themselves with several conflicting or locked systems.
- **Compliance** – IoT, like any other technology in the realm of business, must comply with regulations. Its complexity makes the issue of compliance seem incredibly challenging when many consider standard software compliance a battle.

1.2 Embedded Systems

1.2.1 Introduction to **Embedded Systems**

An embedded system can be defined as a computing device that does a specific focused job. Appliances such as the air-conditioner, VCD player, DVD player, printer, fax machine, mobile phone etc. are examples of embedded systems. Each of these appliances will have a processor and special hardware to meet the specific requirement of the application along with the embedded software that is executed by the processor for meeting that specific requirement. The embedded software is also called “firm ware”. The desktop/laptop computer is a general purpose computer. You can use it for a variety of applications such as playing games, word processing, accounting, software development and so on. In contrast, the software in the embedded systems is always fixed listed below: Embedded systems do a very specific task; they cannot be programmed to do different things. Embedded systems have very limited resources, particularly the memory. Generally, they do not have secondary storage devices such as the CDROM or the floppy disk.

Embedded systems have to work against some deadlines. A specific job has to be completed within a specific time. In some embedded systems, called real-time systems, the deadlines are stringent. Missing a deadline may cause a catastrophe-loss of life or damage

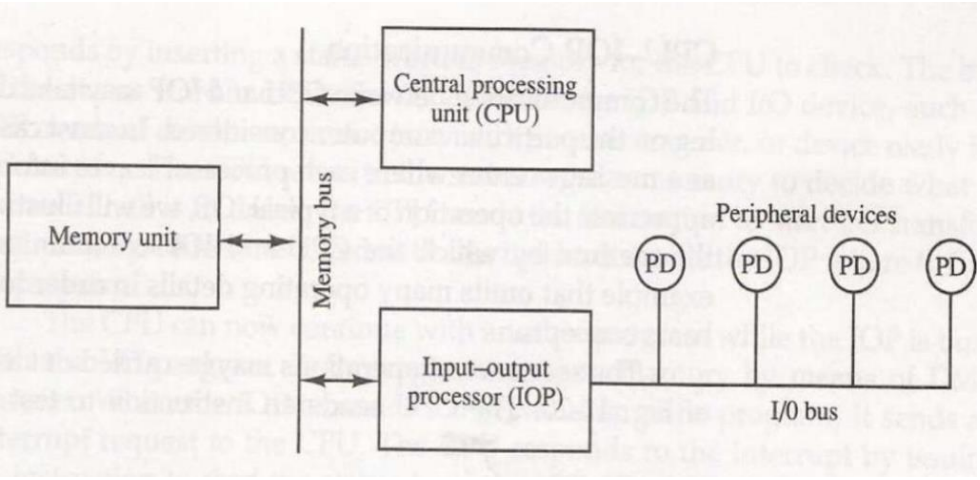
to property. Embedded systems are constrained for power. As many embedded systems operate through a battery, the power consumption has to be very low.
Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity.

1.2.2 Overview of Embedded System Architecture

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

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run for a long time you don't need to reload new software.
Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in the figure, embedded system consists of the following block

· Central Processing Unit (CPU)
· Memory (Read-only Memory and Random Access Memory)
· Input Devices
· Communication interfaces
· Application-specific circuitry



Central Processing Unit (CPU)

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. So, for small applications, a micro-controller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. DSP is used mainly for applications in which signal processing is involved such as audio and video processing.

Memory

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

Input devices

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

Output devices

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

Fig: 16X2 Liquid Crystal Display

Communication interfaces

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

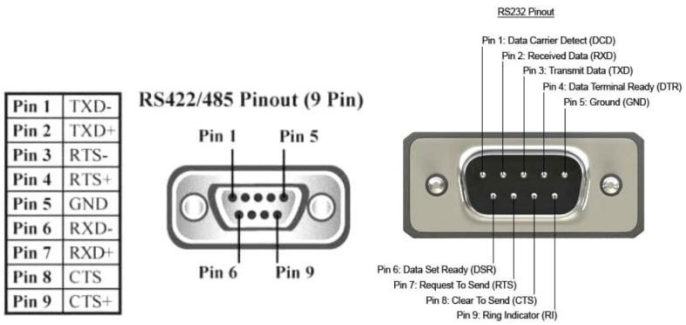


Fig: Pin out of RS422 and RS232 Communication Interfaces

Peripherals

Peripherals are the various devices that are connected to the CPU, for performing various functions. Embedded systems talk with the outside world via peripherals, such as:

- Serial communication interfaces (SCI): RS-232, RS-422, RS-458 etc.
- Synchronous Serial communication interfaces (SSCI): I2C, SPI, SSC and ESSI
- Universal Serial Bus (USB)
- Networks: Controller Area Network, etc.
- Timers: PLL(s), Capture/Compare and Time Processing units.
- Discrete I/O: General Purpose Input/output (GPIO).

Processors

Processors are the key elements in any embedded system. They interact with the memory, where the various instructions of useful functions into a single IC package. These functions are:

- The ability to execute a stored set of instructions to carry out user defined tasks.
- The ability to be able to access external memory chips to both read and writes data from and to the memory.

CHAPTER 2
ANALYSIS OF THE PROJECT

2.1 Project Overview

Data acquisition (DAQ) system today represents one of the main elements in engineering and natural science studying process. This system is essential for collecting data from environments and provides adequate information about natural phenomena and their time dependently changes to teacher and students. Due to high demands and constant upgrades of information and communication technologies, it is necessary to build a sensor system which is low cost and relies on new concepts like Internet of Things (IoT).

Today, commercial DAQ systems used in education purpose are usually general-purpose multi-functional systems and they are equipped with sensor nodes and software. In order to provide a global access to information, a new technology and paradigm like IoT must be applied and integrated in sensor nodes as parts of DAQ system. In addition, a price of sensor nodes can significantly reduce application of whole systems in educational purpose.

Today, natural science relies on large sets of data gathered by monitoring real phenomena and thus requires a strong DAQ system which can provide an orchestration of sensor nodes and characteristics like:
• Capable of managing large amounts of data;
• High-speed connection for DAQ;
• Digital recording; and
• Full reconfigure possibility.

In general, DAQ systems are used as an interface between the real world of physical parameters, which are usually fully analog, and the digital world which today represents standard for computation and control. Relying on digital systems which are widely used because of low cost, accurate and relatively simple implementation and devices that perform the interfacing function between analog and digital worlds (A/D, D/A converters), DAQ systems are largely used today, almost in all processes of collecting information, documenting and analyzing the phenomenon.

The main building elements of DAQ system can be divided in four groups
1**. Sensors**: to convert the physical phenomena in electrical signal
2. **Analog-to-digital converter**: to convert analog signal to digital signal
3. **Multiplexer and amplifier**: to switch and amplify the input analog signals with an

analog–digital converter in digital form
4. **Display/computer:** to visualize/manage the data.
In the proposed solution of a DAQ system, the first three building elements are included in IoT sensor nodes, while the last element is separated on system, which is usually called a "broker" service and holds a multiple role in the system.

With the advent of open source Arduino boards along with cheap sensors, it is viable to create devices that can monitor Industrial Parameters like Temperature, Level, Intensity of Gas, Intensity of Light, and IR Rays when needed. The proposed system makes use of microcontroller ATMEGA328P on Arduino Uno platform and IOT which enable to remotely monitor the status of Industrial Data by knowing the sensor values.

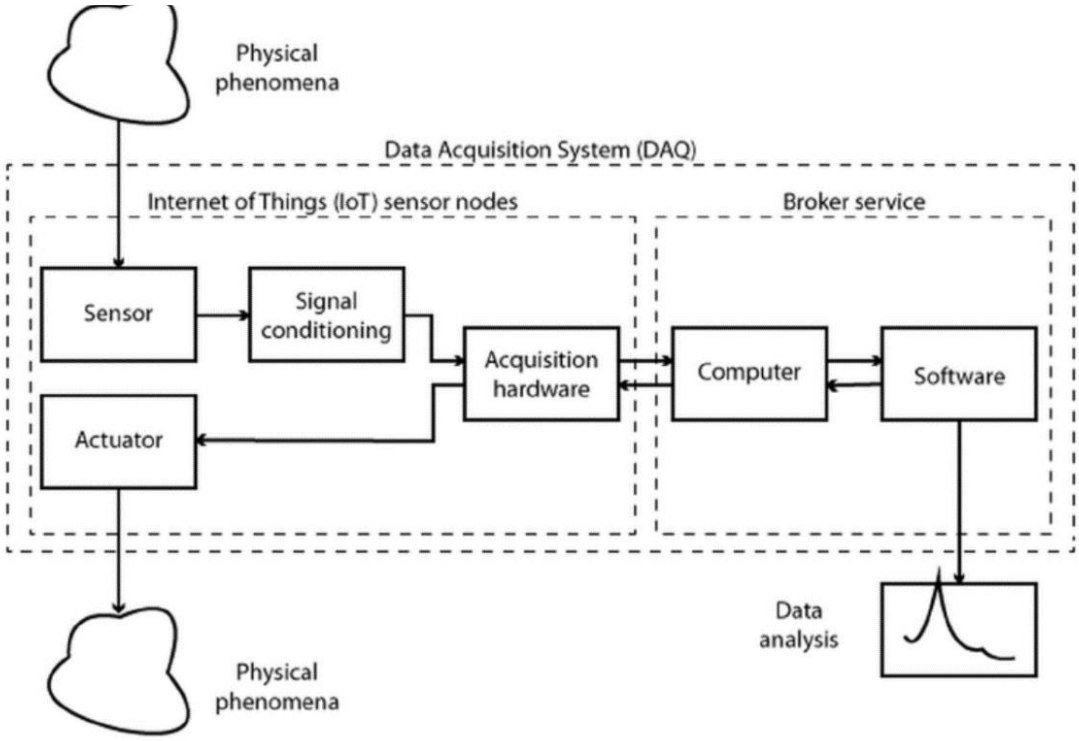


Fig: An Internet of Things based Data Acquisition System

2.2 Block Diagram

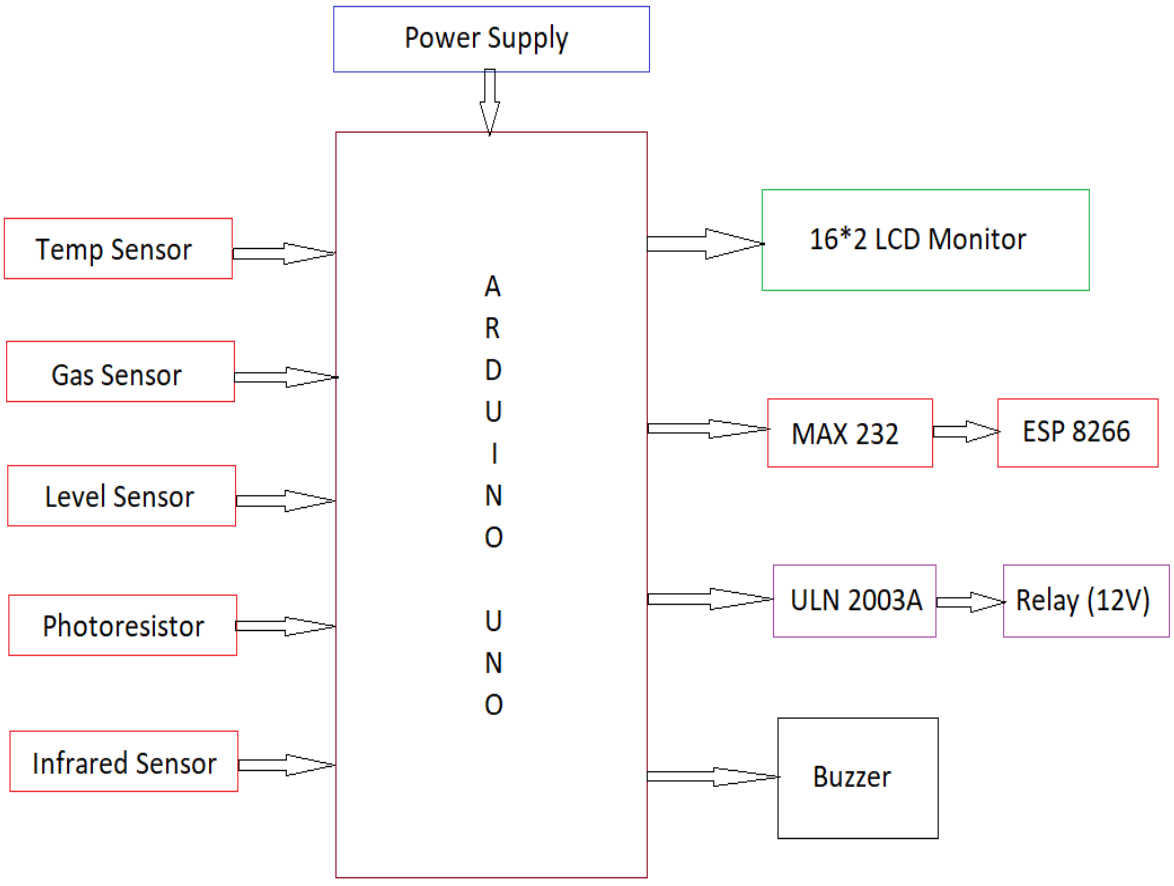


Fig: Project Block Diagram

2.2.1 Block Diagram Explanation

Power Supply: 9V Power to Arduino Uno is supplied from the power supply circuit which will be discussed later.

Arduino Uno: Arduino refers to an open-source electronics platform or board and the software used to program it. Arduino Uno plays the role of a controller by reading digital and analog reading from the sensors connected. All the devices are connected to the Arduino Board. It sends and receives data from MAX232 serial interface module and also takes the required control action according to the program.

Sensors: Sensors senses the changes in the parameters and convert them into an electrical signal. These signals either Digital or Analog are fed to the Arduino Board which reads the signal and take necessary action. Different sensors used in this project are

1. Temperature Sensor- LM35
2. Gas Sensor- MQ6
3. Level Sensor- HC SR04
4. Photo Resistor
5. Passive Infrared Sensor

MAX 232: The MAX232 is an integrated circuit first that converts signals from a TIA-232 (RS-232) serial port to signals suitable for use in TTL-compatible digital logic circuits. The MAX232 is a dual transmitter / dual receiver that typically is used to convert the RX, TX, CTS, RTS signals. In this project, it is used for communication with ESP8266 Wi-Fi Module.

ESP 8266: The ESP8266 is a low-cost Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability ESP8266EX is capable of functioning consistently in industrial environments, due to its wide operating temperature range. With highly-integrated on-chip features and minimal external discrete component count, the chip offers reliability, compactness and robustness. This module sends the data from sensors to Thing speak.

ULN 2003A: ULN2003a is a relay driver circuit. We can use seven relays with relay driver circuit using it. It is a 16 pin IC consisting of 7 Darlington pairs. In this project, 2 ULN2003 IC's are used to drive 3 12V relays according to the control signal from the Arduino Board.

Relay: Relays are the switches which aim at closing and opening the circuits electronically as well as electromechanically. It controls the opening and closing of the circuit contacts of an electronic circuit. When the relay contact is open (NO), the relay isn't energizing with the open contact. However, if it is closed (NC), the relay isn't energize given the closed contact.

Buzzer: Buzzer is used as an alarm when parameters exceed their predefined limit. The frequency and duration of the buzz can be controlled by programming it in the Arduino.

2.3 Circuit Diagram

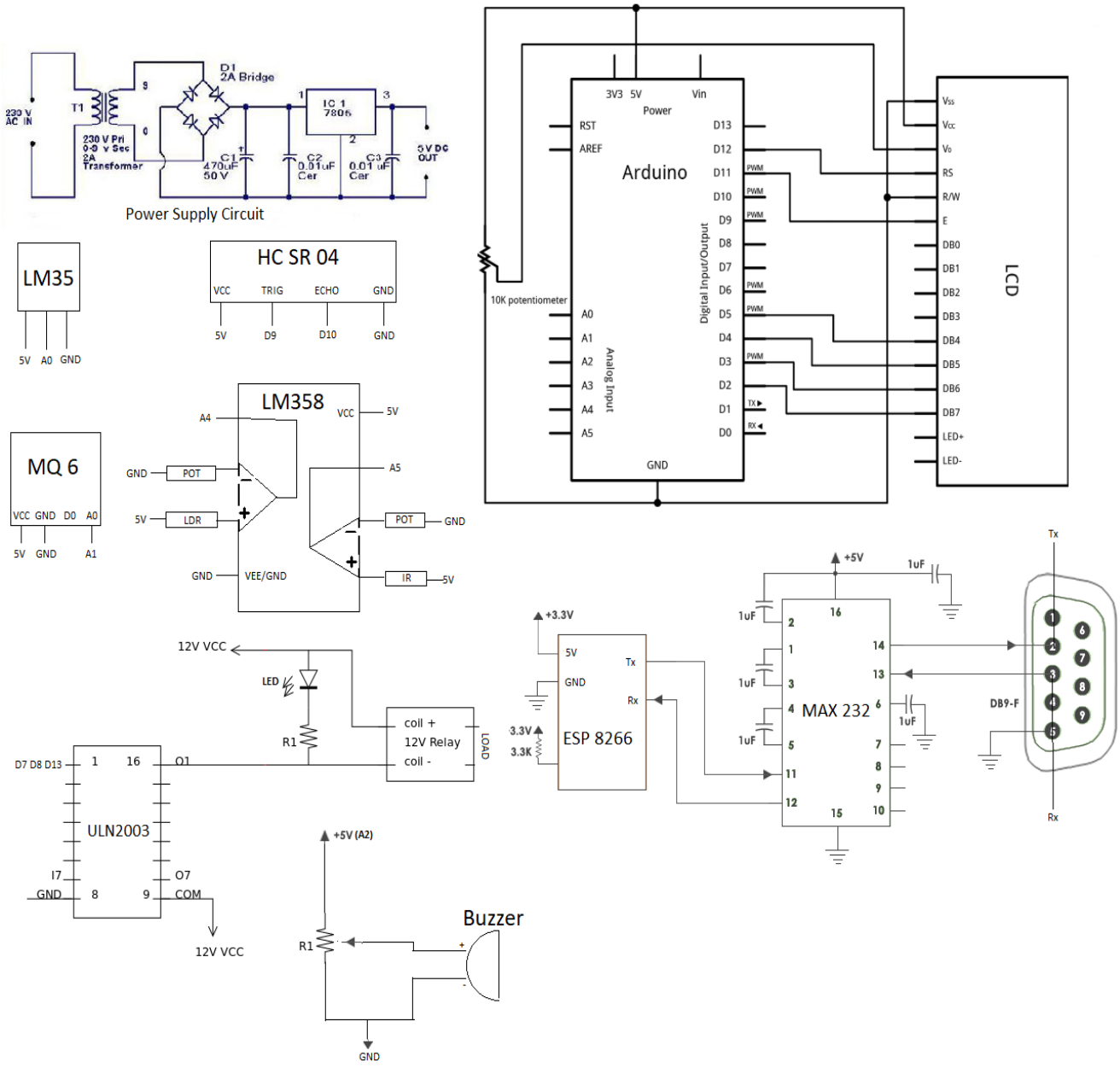


Fig: Circuit Diagram of the Project

2.3.1 Description of the Circuit

In the circuit description we will be discussing about how the interfacings of the modules are done and connections established between modules and the microcontroller. The heart of the project is the microcontroller and program in it will be the key for its working. Some of the important components used in this project are Relays, MAX232, ESP8266, LCD, Buzzer and Sensors.

- The microcontroller used in Arduino is ATMEGA 328P. The basic connections to the microcontroller are that pin 40 connected to Vcc, 20th pin for ground and pin 9 for Reset. Crystal oscillator is connected between18 and 19th pins and it is the timer for the controller to work in a correct pattern. Without crystal oscillator the program code doesn't execute and there will be no synchronization between input and output signals.

- In 16*2 Liquid Crystal Display, RS and Enable pins are connected to 12 and 11 pins of the Arduino respectively.
  4 data pins are connected to 2, 3, 4, 5 pins.
  5V power supply is given from the power supply circuit.
  A 10kohm resistor is connected between Vcc and Vo pins for contrast.
  Vss and R/W pins are also connected. Vcc and LED+ pin, Vss and LED- are connected for background light.

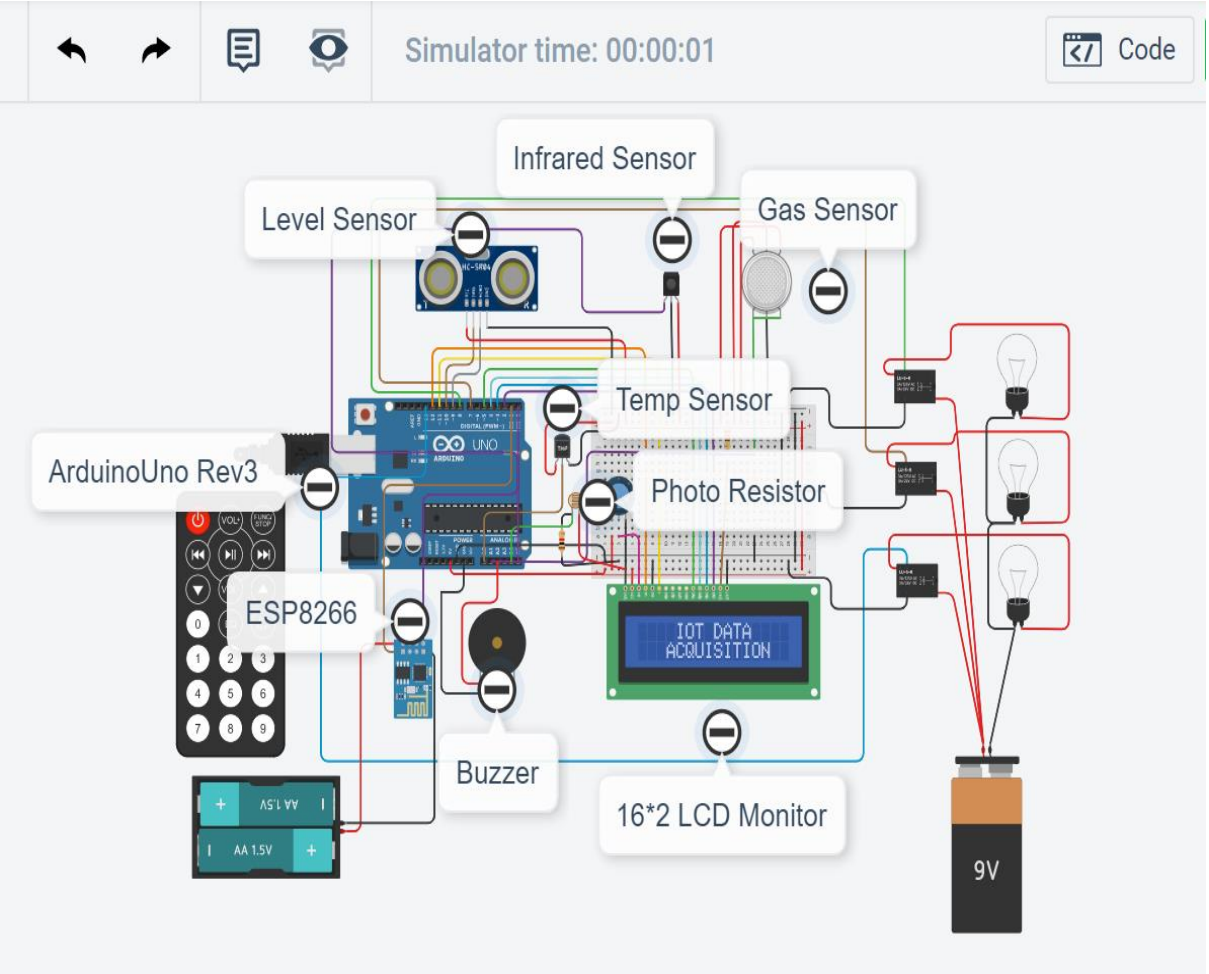
- Temperature sensor LM35 output is given to A0 pin, Gas sensor MQ6 analog output is given to A1, LDR output from LM358 is given to A4, IR output is given to A5 and D9 and D10 pins are connected to Trig and Echo pin of Level sensor HC SR04.

- 6, 7 and 8^{th} pins are given to ULN2003 IC to drive relays and 13^{th} pin is directly given to 5V relay.

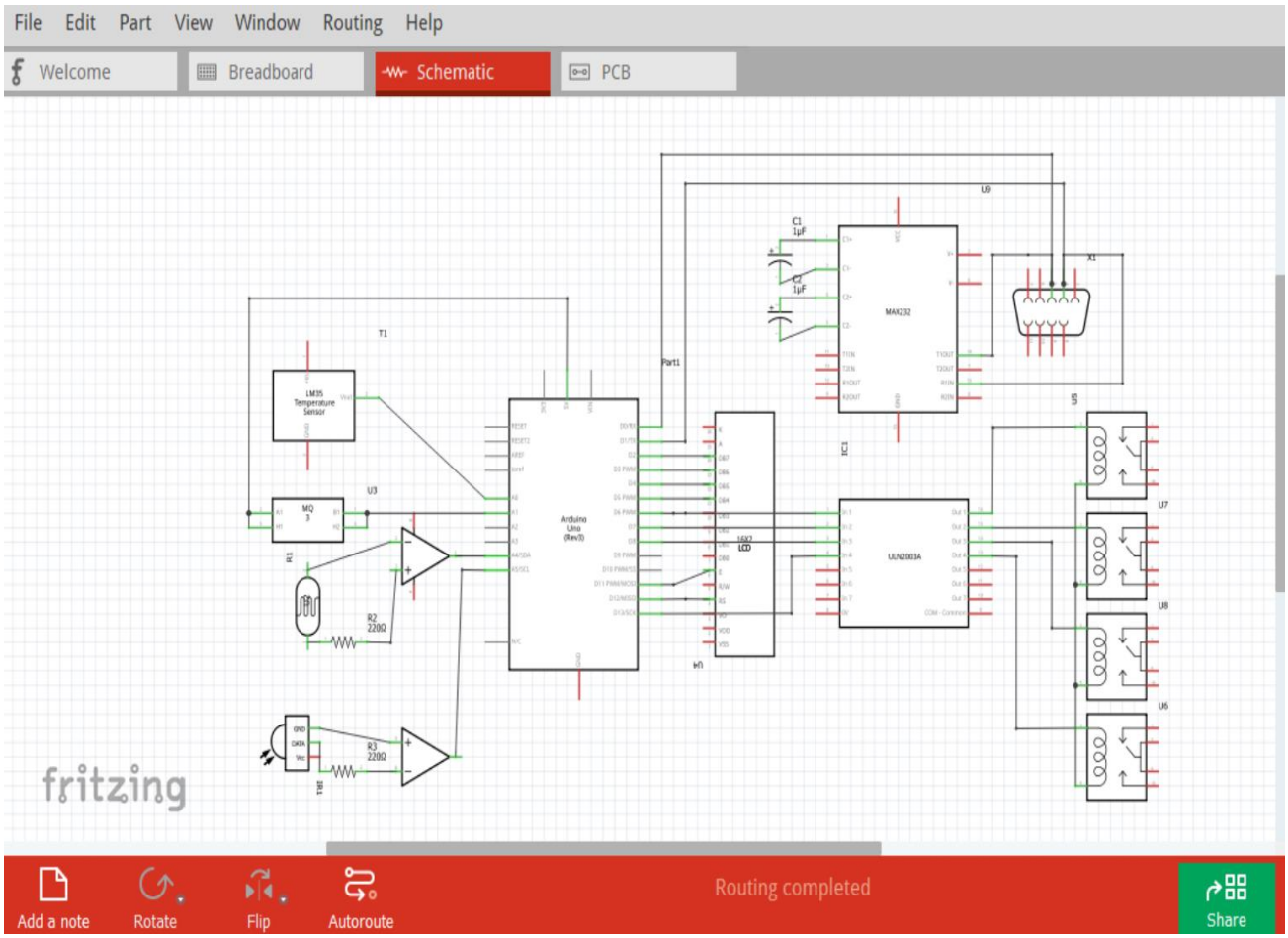
- Tx and Rx pins of Esp8266 module is connected to 11^{th} and 12^{th} pins of MAX232 IC. 14^{th} and 13^{th} pins of MAX 232 is given to 2^{nd} and 3^{rd} pins of RS232 cable which are connected to Tx and Rx pins of Arduino.

- A2 pin of Arduino is given as input to the buzzer.

2.4 Online Simulation of Project in Tinkercad



2.5 Schematic view of Project in Fritzing



CHAPTER 3
HARDWARE DESCRIPTION

3.1 Arduino Uno

3.1.1 ATMEGA328P

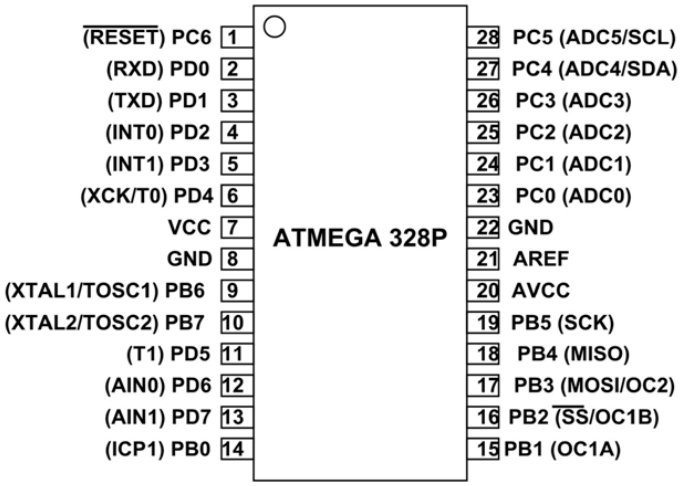
Fig: ATMEGA328P IC

Microcontroller can be described as a computer embedded on a rather small circuit board. To describe the function of a microcontroller more precisely, it is a single chip that can perform various calculations and tasks, and send/receive signals from other devices via the available pins. Precisely what tasks and communication with the world it does, is what is governed by what instructions we give to the Microcontroller. It is this job of telling the chip what to do, is what we refer to as programming on it.

However, the uC by itself cannot accomplish much; it needs several external inputs: power, for one; a steady clock signal, for another. Also, the job of programming it has to be accomplished by an external circuit. So typically, a microcontroller is used along with a circuit which provides these things to it; this combination is called a microcontroller board. The Arduino Uno that you have received is one such microcontroller board. The actual microcontroller at its heart is the chip called Atmega328. The advantages that Arduino offers over other microcontroller boards are largely in terms of reliability of the circuit hardware as well as the ease of programming and using it.

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

Pin Out Diagram



Features

Parameter	**Value**
CPU Type	8-Bit AVR
Performance	20 MIPS AT 20MHz
Flash Memory	32kb
SRAM	2kb
EEPROM	1kb
Pin Count	28 or 32 pin: PDIP-28, MLF-28, TQFP-32, MLF-32
Max Operating Frequency	20 MHz
No of touch channels	16
Max I/O pins	23
External Interrupts	2
USB Interface	No
USB Speed	-
Hardware QTouch Acquisition	No

3.1.2 Crystal Oscillator

Fig: 16MHz Crystal Oscillator

A crystal oscillator is an electronic oscillator circuit that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a precise frequency. This frequency is often used to keep track of time, as in quartz wristwatches, to provide a stable clock signal for digital integrated circuits, and to stabilize frequencies for radio transmitters and receivers. The most common type of piezoelectric resonator used is the quartz crystal, so oscillator circuits incorporating them became known as crystal oscillators, but other piezoelectric materials including polycrystalline ceramics are used in similar circuits.

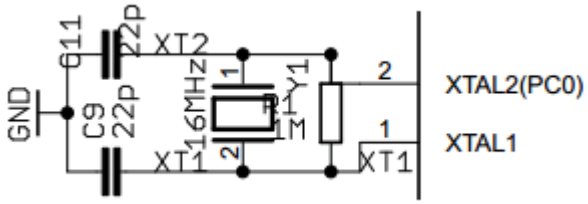


Fig: Schematic of Crystal Oscillator

Crystal oscillators operate on the principle of inverse piezoelectric effect in which an alternating voltage applied across the crystal surfaces causes it to vibrate at its natural frequency. It is these vibrations which eventually get converted into oscillations. These oscillators are usually made of Quartz crystal, even though other substances like Rochelle salt and Tourmaline exhibit the piezoelectric effect because, quartz is inexpensive, naturally-available and mechanically-strong when compared to others.

In crystal oscillators, the crystal is suitably cut and mounted between two metallic plates. In reality, the crystal behaves like a series RLC circuit, formed by the components

1. A low-valued resistor R_S
2. A large-valued inductor L_S
3. A small-valued capacitor C_S

This will be in parallel with the capacitance of its electrodes Cp.

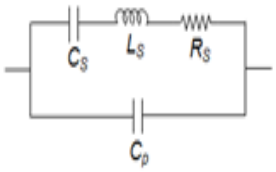


Fig: Quartz Crystal **Fig: Equivalent Electric Circuit**

1. Series Resonant Frequency, f_s which occurs when the series capacitance C_S resonates with the series inductance L_S. At this stage, the crystal impedance will be the least and hence the amount of feedback will be the largest. Mathematical expression for the same is given as

$$f_s = \frac{1}{2\pi\sqrt{L_s C_s}}$$

2. Parallel Resonant frequency, f_p which is exhibited when the reactance of the L_SC_S leg equals the reactance of the parallel capacitor C_p i.e. L_S and C_S resonate with C_p. At this instant, the crystal impedance will be the highest and thus the feedback will be the least. Mathematically it can be given as

$$f_p = \frac{1}{2\pi\sqrt{L_S \frac{C_P C_S}{C_P + C_S}}}$$

The behavior of the capacitor will be capacitive both below f_S and above f_p. However for the frequencies which lie in-between f_S and above f_p, the crystal's behavior will be inductive. Further when the frequency becomes equal to parallel resonant frequency f_p, then the interaction between L_S and C_p would form a parallel tuned LC tank circuit. Hence, a crystal can be viewed as a combination of series and parallel tuned resonance circuits due to which one needs to tune the circuit for any one among these two. Moreover it is to be noted that f_p will be higher than f_s and the closeness between the two will be decided by the cut and the dimensions of the crystal in-use.

Crystal oscillators can be designed by connecting the crystal into the circuit such that it offers low impedance when operated in series-resonant mode and high impedance when operated in anti-resonant or parallel resonant mode.

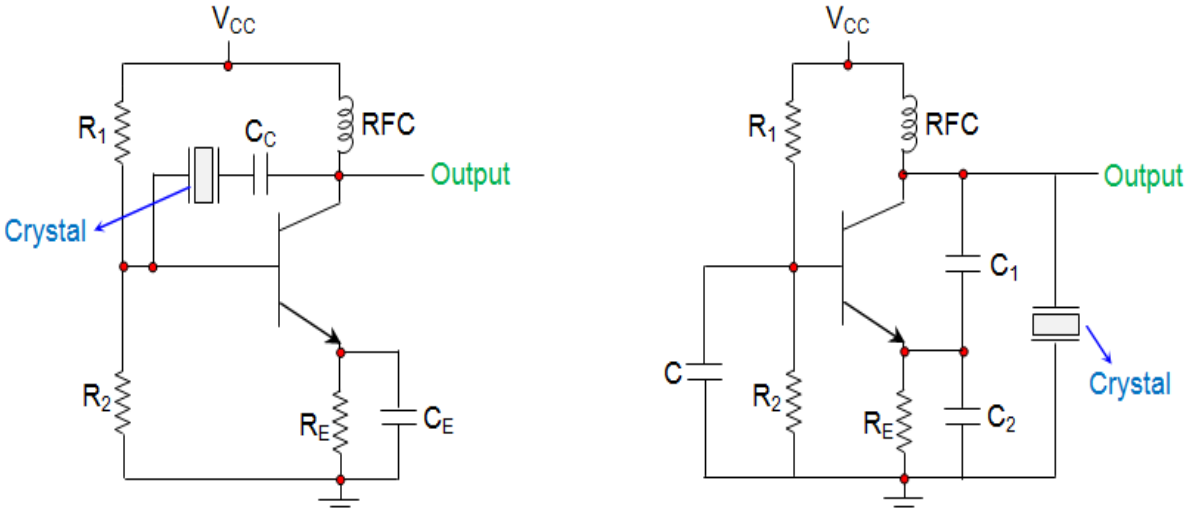


Fig. Crystal Oscillator working in Series Resonance **Parallel Resonance**

In the circuits shown, the resistors R_1 and R_2 form the voltage divider network while the emitter resistor R_E stabilizes the circuit. Further, C_E acts as an AC bypass capacitor while the coupling capacitor C_C is used to block DC signal propagation between the collector and the base terminals. Next, the capacitors C_1 and C_2 form the capacitive voltage divider network in the case of Figure 2b. In addition, there is also a Radio Frequency Coil (RFC) in the circuits which offers dual advantage as it provides even the DC bias as well as frees the circuit-output from being affected by the AC signal on the power lines.

On supplying the power to the oscillator, the amplitude of the oscillations in the circuit increases until a point is reached wherein the nonlinearities in the amplifier reduce the loop gain to unity. Next, on reaching the steady-state, the crystal in the feedback loop highly influences the frequency of the operating circuit. Further, here, the frequency will self-adjust so as to facilitate the crystal to present a reactance to the circuit such that the Barkhausen phase requirement is fulfilled.

The typical operating range of the crystal oscillators is from 40 KHz to 100 MHz wherein the low frequency oscillators are designed using OpAmps while the high frequency-ones are designed using the transistors (BJTs or FETs). The frequency of oscillations generated by the circuit is decided by the series resonant frequency of the crystal and will be unaffected by the variations in supply voltage, transistor parameters, etc. As a result, crystal oscillators exhibit high Q-factor with excellent frequency stability, making them most suitable for high-frequency applications. However care should be taken so as to drive the crystal with optimum power only. This is because, if too much of power is delivered to the crystal, then the parasitic resonances might be excited in the crystal which leads to unstable resonant frequency. Further even its output waveform might be distorted due to the degradation in its phase noise performance. Moreover it can even result in the destruction of the device (crystal) due to overheat

3.1.3 Arduino Uno Rev3

The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts.

- Microcontroller: Microchip ATmega328P
- Operating Voltage: 5 Volts
- Input Voltage: 7 to 20 Volts
- Digital I/O Pins: 14 (of which 6 can provide PWM output)
- UART: 1
- I2C: 1
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by boot loader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz
- Length: 68.6 mm
- Width: 53.4 mm
- Weight: 25 g

General pin functions

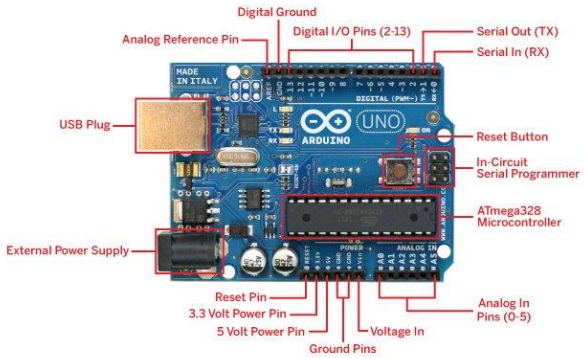


Fig: Arduino Uno Pin Diagram

- **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 is off.
- **VIN**: The input voltage to the Arduino/Genuino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V**: 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.
- **3V3**: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND**: Ground pins.
- **IOREF**: This pin on the 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.
- **Reset**: Typically used to add a reset button to shields that block the one on the board.

Special pin functions

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software control (using pinMode(), digitalWrite(), and digitalRead() functions). They operate at 5 volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50K ohm. A maximum of 40mA 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 provides 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upper end of the range using the AREF pin and the analogReference() function.

- **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.
- **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.
- **PWM** (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite() function.
- **SPI** (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.
- **TWI** (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.
- **AREF** (analog reference): Reference voltage for the analog inputs.

Communication

The Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, 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. Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows serial communication on any of the Uno's digital pins.

Automatic (software) reset

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board 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.

3.2 Power Supply

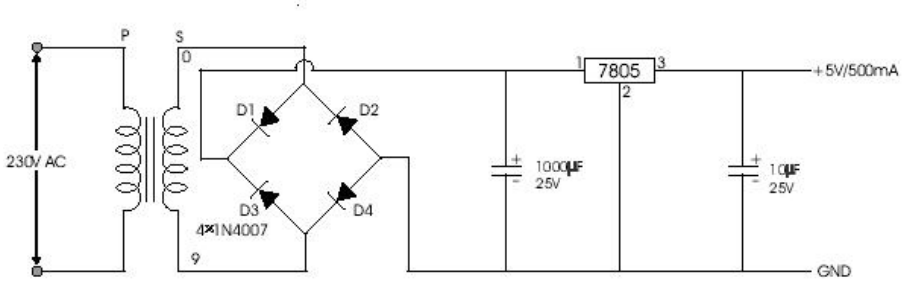


Fig: Power Supply Circuit

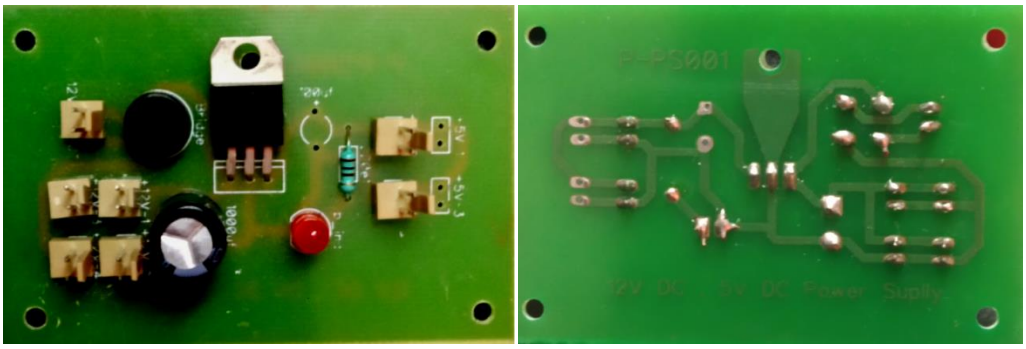


Fig: Power Supply Circuit on a PCB

The components used in Power supply circuit are

1. 230/12 Step Down Transformer

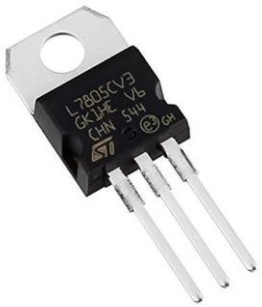
2. Bridge Rectifier
3. 7805 Voltage Regulator

230/12 Step Down Transformer

Bridge Rectifier (DC W06M)

DC W06M is a Single phase Silicon Bridge Rectifier. A diode bridge is an arrangement of four (or more) diodes in a bridge circuit configuration that provides the same polarity of output for either polarity of input. When used in its most common application, for conversion of an alternating-current (AC) input into a direct-current (DC) output, it is known as a bridge rectifier. A bridge rectifier provides full-wave rectification from a two-wire AC input, resulting in lower cost and weight as compared to a rectifier with a 3-wire input from a transformer with a center-tapped secondary winding.

Voltage Regulator (7805)



Voltage regulators are very common in electronic circuits. They provide a constant output voltage for a varied input voltage. In our case the 7805 IC is an iconic regulator IC that finds its application in most of the projects. The name 7805 signifies two meaning, “78” means that it is a positive voltage regulator and “05” means that it provides 5V as output. So our 7805 will provide a +5V output voltage. The output current of this IC can go up to 1.5A. But, the IC suffers from heavy heat loss hence a Heat sink is recommended for projects that consume more current.

3.3 Liquid Crystal Display

3.3.1 Introduction

Liquid Crystal Displays are created by sandwiching a thin (10-12 micro mm) layer of a liquid crystal fluid between two glass plates. A transparent, electrically conductive film or back plane is put up on the rear glass sheet. The transparent sections of the conductive film in the shape of the desired characters are coated on the front glass plate. When a voltage is applied between a segment and the back plane, an electric field is created in the region under the segment. This electric field changes the transmission of light through the region under the segment film.

3.3.2 Liquid Crystal Display Description

In this project, JHD 162A Liquid Crystal Display (16x2), which is shown in Figure 3.9, is interfaced with the CPU.

Fig: 16*2 Liquid Crystal Display

The pin description of the JHD 162A LCD without backlight is as shown in Table. If the LCD is having Backlight, then it will have two more pins with pin numbers 15 & 16 connected to VCC and GND respectively.

Pin number	Symbol	Level	I/O	Function
1	Vss	-	-	Power supply (GND)
2	Vcc	-	-	Power supply (+5V)
3	Vee	-	-	Contrast adjust
4	RS	0/1	I	0= Instruction input 1 = Data input
5	R/W	0/1	I	0 = Write to LCD module 1 = Read from LCD module
6	E	1, 1->0	I	Enable signal
7	DB0	0/1	I/O	Data bus line 0 (LSB)
8	DB1	0/1	I/O	Data bus line 1
9	DB2	0/1	I/O	Data bus line 2
10	DB3	0/1	I/O	Data bus line 3
11	DB4	0/1	I/O	Data bus line 4
12	DB5	0/1	I/O	Data bus line 5
13	DB6	0/1	I/O	Data bus line 6
14	DB7	0/1	I/O	Data bus line 7 (MSB)

An LCD allows an application to output a very specific message (or prompt) to the user, making the application much more user friendly and impressive. LCD's are invaluable for displaying status messages and information during application debug. ASCII-input LCDs even though they have these advantages, they have a reputation of being difficult to hook up

and get to work. Most alphanumeric LCD's use a common controller chip and a common connector interface. Both of these actions have resulted in alphanumeric LCDs that range in size from 8 characters to 80 (arranged as 40 b 2 or 20 b 4) and are interchangeable, without requiring hardware or software changes.

The ASCII code to be displayed is 8 bits long and is sent to the LCD either 4 or 8 bits at time. If the 4-bit mode is used, two nibbles of data (sent high 4 bits then low 4 bits with an E clock pulse with each nibble) are sent to make up a full 8-bit transfer. The "E" clock is used to initiate the data transfer within the LCD. In the LCD there is a cursor,

This specifies where the next data character is to be written. This cursor can be moved or be made invisible to blink. The blinking function is very rarely used because it is pretty obnoxious.

Sending parallel data either as 4 or 8 bits are the two primary modes of operation. While there are secondary considerations and modes deciding how to send the data to the LCD is more critical decision to be made for an LCD interface application. 4-bit mode is best used when the speed required in an application and at least 10 I/O pins are available. 4-bit mode requires minimum 6 bits. To wire a Microcontroller to an LCD 4-bit mode, just the top 4-bits (DB4-7) are written as:

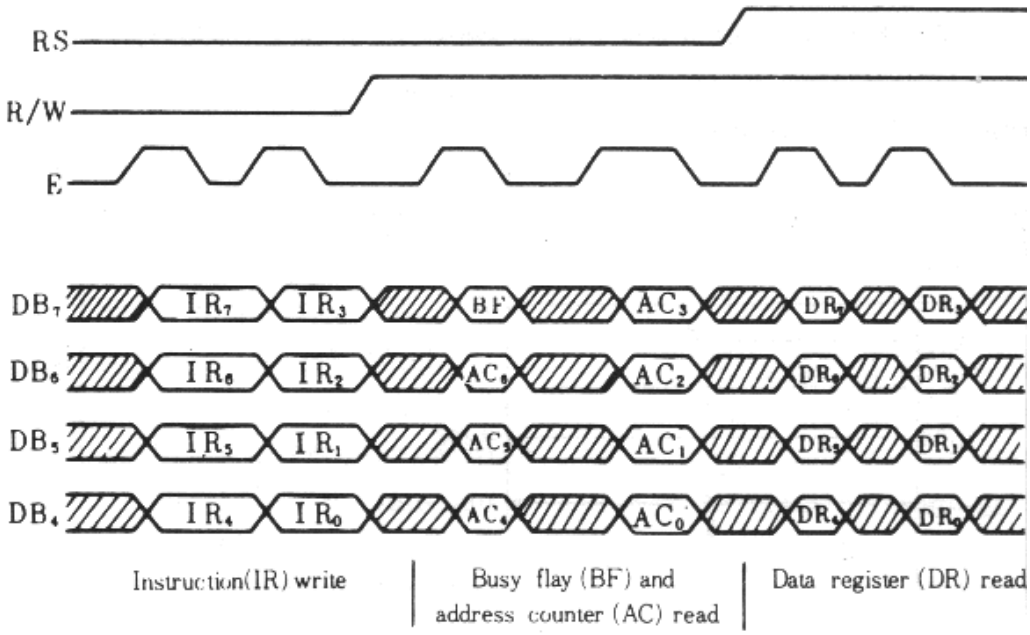


Fig: Data Transfer using 4-bit Interface

Using a shift register so that a minimum of three I/O pins is required can further reduce this. 8-bit mode could be used with a shift register, but a ninth bit (which will be used as R/S) will be required.

The display contains two internal byte-wide registers, one for command (RS=0) and the second for the characters to be displayed (RS=1). The R/S bit is used to select whether data or an instruction is being transferred between the Microcontroller and the LCD. If the bit is

set, the byte at the current LCD cursor position can be read or written. When the bit is reset, either an instruction is being sent to the LCD or the execution status of the last instruction is read back (whether it has completed or not).

The display contains two internal byte-wide registers, one for command (RS=0) and the second for characters to be displayed (RS=1). It also contains a user programmed RAM area (the character RAM) that can be programmed to generate a desired character that can be formed using a dot matrix.

To distinguish between these two data areas, the hex command byte 80 will be used to signify that the display RAM address 00h is chosen.

Port 1 is used to furnish the commands or data byte and ports 3.2 to 3.4 furnish register select and read/write levels. The display takes varying amounts of time to accomplish the functions. LCD bit 7 is monitored for a long high (bus) to ensure the display is not over written. A slightly more complicated LCD display (4 lines* 40 characters) is currently being used in medical diagnostic systems to run a very familiar program.

Getting The LCD To Display Text: -

After successfully initializing the LCD and turning the display ON, one can begin to display messages on the LCD by sending the correct instructions to it. Getting the LCD to display text is a two-step process. First, the LCD's cursor must be moved to the LCD address where the character is to be displayed. This is done with the "DDRAM Address Set" command. Second, the actual character must be written to the cursor in order to store it in the DDRAM at the cursor's location. This is performed with the "CGRAM/DDRAM Data Write" command.

3.3.3 LCD Commands

The library used for the commands is **#include <LiquidCrystal.h>**

LiquidCrystal()

Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, omit the pin numbers for d0 to d3 and leave those lines unconnected. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function's parameters.

Syntax

LiquidCrystal(rs, enable, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)

LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)
Parameters
rs: the number of the Arduino pin that is connected to the RS pin on the LCD
rw: the number of the Arduino pin that is connected to the RW pin on the LCD (optional)
enable: the number of the Arduino pin that is connected to the enable pin on the LCD
d0, d1, d2, d3, d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD. d0, d1, d2, and d3 are optional; if omitted, the LCD will be controlled using only the four data lines (d4, d5, d6, d7).
Example

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);
void setup()
{
lcd.begin(16,1);
lcd.print("hello, world!");
}
void loop() {}
```

begin()

Description
Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. begin() needs to be called before any other LCD library commands.
Syntax
lcd.begin(cols, rows)
Parameters
lcd: a variable of type LiquidCrystal
cols: the number of columns that the display has
rows: the number of rows that the display has

clear()

Description
Clears the LCD screen and positions the cursor in the upper-left corner.
Syntax
lcd.clear()
Parameters
lcd: a variable of type LiquidCrystal

setCursor()

Description
Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.
Syntax
lcd.setCursor(col, row)
Parameters
lcd: a variable of type LiquidCrystal
col: the column at which to position the cursor (with 0 being the first column)
row: the row at which to position the cursor (with 0 being the first row)

3.4 MAX232

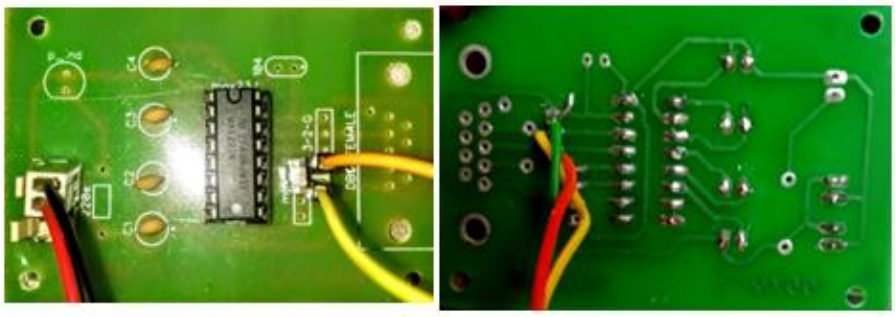


Fig: MAX232 on a PCB

3.4.1 Description

IC MAX232 is used for TTL/CMOS to RS232 conversion. Meaning most of our Microcontrollers (PIC/ARM/Atmel) operates on TTL/CMOS logic that is it communicates through either 0V or +5V, but our computers work with the help of RS232 which operates at logic level -24V or +24V. So, if we have to interface these microcontrollers with Computer we need to convert the TTL/CMOS logic to RS232 logic. Hence if you are looking for an IC to perform this conversion and interface a Microcontroller with your computer then MAX232 IC is the right one for you

3.4.2 MAX232 Pin Configuration

Pin Number	Pin Name	Description
1	C1 +	Connects to the positive end of First capacitor
2	V+	Connects to one end of the capacitor & other end is grounded
3	C1 -	Connects to the negative end of First Capacitor
4	C2+	Connects to positive end of second capacitor
5	C2-	Connects to negative end of second capacitor
6	V-	Connects to one end of the capacitor & other end is grounded
7	T2 OUT	Transmission pin of second converter module for RS232 cable
8	R2 IN	Reception pin of second converter module for RS232 cable
9	R2 OUT	Reception pin of second converter module for Microcontroller(Rx)
10	T2 IN	Transmission pin of second converter module for Microcontroller(Tx)
11	T1 IN	Transmission pin of first converter module for Microcontroller(Tx)
12	R1 OUT	Reception pin of first converter module for Microcontroller(Rx)
13	R1 IN	Reception pin of first converter module for RS232 cable
14	T1 OUT	Transmission pin of first converter module for RS232 cable
15	Ground	Connects to the ground of the circuit
16	Vcc	Connects to the supply voltage typically +5V

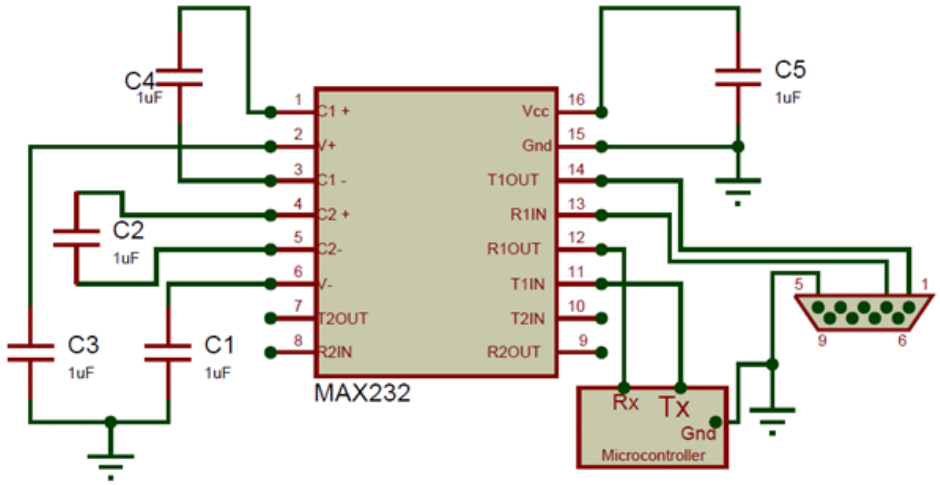
3.4.3 Features

- TTL/CMOS to RS232 Converter IC
- Can support two conversion at the same time (Dual drivers and Receivers)
- Easy to set-up and initialize
- Operating speed: 120kbit/s
- ±30-V Input Levels
- Operating current: 8mA
- Available in 16-pin PDIP, SO, SOIC packages

3.4.4 RS232 Converter IC

This IC is easy to set-up and can be used easily. The IC work with the help of +5V, hence power the Vcc with +5V and the ground pin to circuit ground. The IC also needs four capacitors to work; these capacitors can be of any value from 1uF to 22uF.

MAX232 IC can help you to convert two logic level conversion that is you can use two Microcontrollers, but to get started we will use only one MCU and connect it to a computer, the complete circuit is to do the same is shown below. As you can see the pins (T2 out, R2 in, T2 in, R2 out) are left free since we are not using the second module.



Every microcontroller that has Serial communication capability will have a Tx pin and a Rx pin. These two pins should be connected to the T1 in (pin 10) and the R1 out (pin 13) pin respectively. This is the TTL/CMOS logic inputs, and then the converter RS232 logic signals can be obtained from the pins R1 in (pin 13) and T1 out (pin 14). You might wonder how a 5V signal is being converter to +25V signal when the IC itself is powered with +5V. This is made possible by a method called charge pump**,** which consists of a capacitor that charges up to provide 25V.

3.4.5 Applications

- Used to Connect Microcontroller with Computers
- TTL/CMOS logic to RS232 converters
- Used in RS232 cables

3.5 ESP8266 WiFi Module

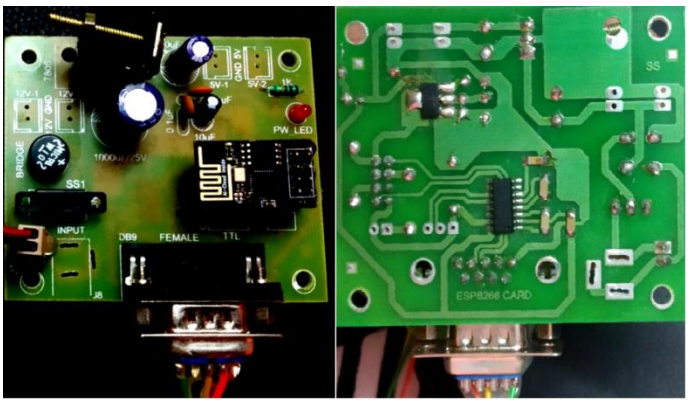


Fig: ESP8266 on a PCB

3.5.1 Description

The ESP8266 is a very user friendly and low cost device to provide internet connectivity to your projects. The module can work both as a Access point (can create hotspot) and as a station (can connect to Wi-Fi), hence it can easily fetch data and upload it to the internet making Internet of Things as easy as possible. It can also fetch data from internet using API's hence your project could access any information that is available in the internet, thus making it smarter. Another exciting feature of this module is that it can be programmed using the Arduino IDE which makes it a lot more user friendly. However this version of the module has only 2 GPIO pins (you can hack it to use upto 4) so you have to use it along with another microcontroller like Arduino, else you can look onto the more standalone ESP-12 or ESP-32 versions. So if you are looking for a module to get started with IOT or to provide internet connectivity to your project then this module is the right choice for you.

3.5.2 ESP8266 Pin Configuration

Pin Number	Pin Name	Alternate Name	Normally used for	Alternate purpose
1	**Ground**	-	Connected to the ground of the circuit	-
2	**TX**	GPIO – 1	Connected to Rx pin of programmer/uC to upload program	Can act as a General purpose Input/output pin when not used as TX
3	**GPIO-2**	-	General purpose	-

			Input/output pin	
4	**CH_EN**	-	Chip Enable – Active high	-
5	**GPIO - 0**	Flash	General purpose Input/output pin	Takes module into serial programming when held low during start up
6	**Reset**	-	Resets the module	-
7	**RX**	GPIO - 3	General purpose Input/output pin	Can act as a General purpose Input/output pin when not used as RX
8	**Vcc**	-	Connect to +3.3V only	

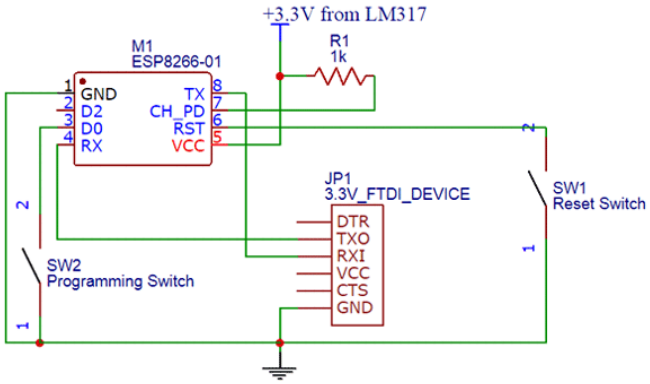
3.5.3 ESP8266-01 Features

- Low cost, compact and powerful Wi-Fi Module
- Power Supply: +3.3V only
- Current Consumption: 100mA
- I/O Voltage: 3.6V (max)
- I/O source current: 12mA (max)
- Built-in low power 32-bit MCU @ 80MHz
- 512kB Flash Memory
- Can be used as Station or Access Point or both combined
- Supports Deep sleep (<10uA)
- Supports serial communication hence compatible with many development platform like Arduino
- Can be programmed using Arduino IDE or AT-commands or Lua Script

ESP8266-01 Boot Option

GPIO – 0	GPIO – 2	Mode	Used For
High	High	Flash Mode	Run the program that is already uploaded to the module
Low	High	UART Mode	Programming mode- to program using Arduino or any serial communication

The ESP8266 module works with 3.3V only, anything more than 3.7V would kill the module hence be cautions with your circuits. The best way to program an ESP-01 is by using the FTDI board that supports 3.3V programming. If you don't have one it is recommended to buy one or for time being you can also use an Arduino board. One commonly problem that every one faces with ESP-01 is the powering up problem. The module is a bit power hungry while programming and hence you can power it with a 3.3V pin on Arduino or just use a potential divider. So it is important to make a small voltage regulator for 3.31v that could supply a minimum of 500mA. One recommended regulator is the LM317 which could handle the job easily. A simplified circuit diagram for using the ESP8266-01 module is given below



The switch SW2 (Programming Switch) should be held pressed to hold the GPIO-0 pin to ground. This way we can enter into the programming mode and upload the code. Once the code is released the switch can be released.

3.5.4 Applications

- IOT Projects
- Access Point Portals
- Wireless Data logging
- Smart Home Automation
- Learn basics of networking
- Portable Electronics
- Smart bulbs and Sockets

3.6 LM35 Temperature Sensor

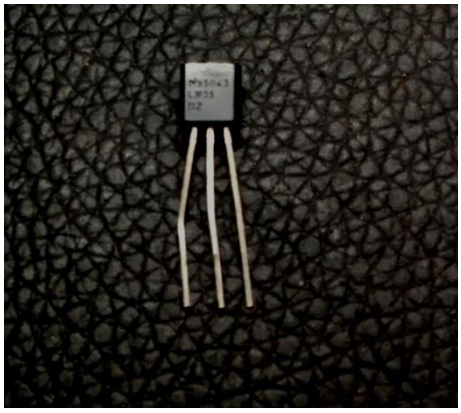
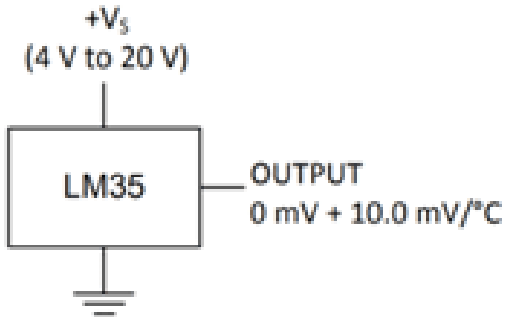


Fig: LM35 Temperature Sensor

LM35 is a precession Integrated circuit Temperature sensor, whose output voltage varies, based on the temperature around it. It is a small and cheap IC which can be used to measure temperature anywhere between -55°C to 150°C. It can easily be interfaced with any Microcontroller that has ADC function or any development platform like Arduino.

Power the IC by applying a regulated voltage like +5V (V_S) to the input pin and connected the ground pin to the ground of the circuit. Now, you can measure the temperate in form of voltage as shown below.



If the temperature is 0°C, then the output voltage will also be 0V. There will be rise of 0.01V (10mV) for every degree Celsius rise in temperature. The voltage can converted into temperature using the below formulae.

$V_{OUT} = 10\ mv/°C \times T$

where

- V_{OUT} is the LM35 output voltage
- T is the temperature in °C

3.6.1 Description

- Transducers convert physical data such as temp, light intensity, speed etc to electrical signal.

- Depending on the transducer, the output produced in the form of voltage, current, resistance or capacitance.
- For example: Temperature is converted in to electrical signals using a transducer called Thermister. A thermister responds to the temperature change by changing resistance, but its response is not linear.
- That's why we are using linear temperature sensors like LM34 or LM35 series.
- The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature.
- Its output is 10MilliVolts per degree centigrade.
- So if the output is 310 mV then temperature is 31 degree C.
- LM35 is a popular and low cost temperature sensor.
- It has three pins as follows.

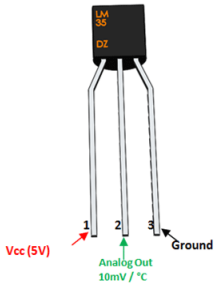


Fig: LM35 Pin Out

- To use the sensor simply connect the Vcc to 5V, GND to Gnd and the Out to one of the ADC (analog to digital converter channel). The output linearly varies with temperature.

3.6.2 Pin Configuration

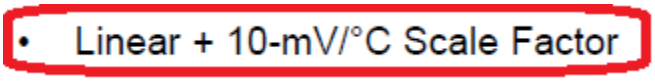
Pin Number	Pin Name	Description
1	Vcc	Input voltage is +5V for typical applications
2	Analog Out	There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
3	Ground	Connected to ground of circuit

3.6.3 LM35 Regulator Features

- Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically 5V.

- Can measure temperature ranging from -55°C to 150°C .
- Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
- ±0.5°C Accuracy
- Drain current is less than 60uA.
- Low cost temperature sensor.
- Small and hence suitable for remote applications.
- Available in TO-92, TO-220, TO-CAN and SOIC package.

3.6.4 LM35 Working Principle



LM35 scale factor

In order to understand the working principle of LM35 temperature sensor we have to understand the linear scale factor. In the features of LM35 it is given to be **+10 mills volt per degree centigrade**. It means that with increase in output of 10 mills volt by the sensor Vout pin the temperature value increases by one. For example, if the sensor is outputting 100 mills volt at vout pin the temperature in centigrade will be 10-degree centigrade.

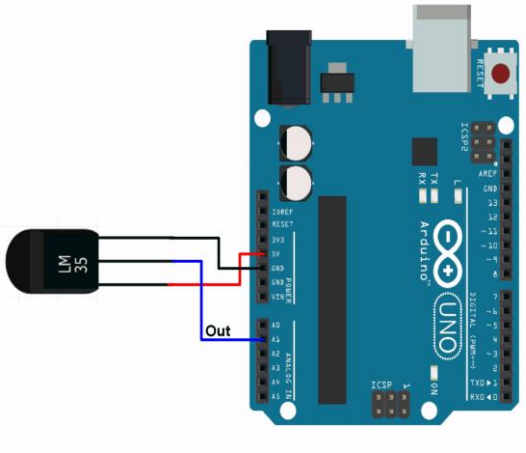


Fig:LM35 Temparature Sensor connections to Arduino Board

3.6.5 LM35 Temperature Sensor Applications:

- Measuring temperature of a particular environment
- Providing thermal shutdown for a circuit/component
- Monitoring Battery Temperature
- Measuring Temperatures for HVAC applications.

3.7 MQ6 Gas Sensor

Fig: MQ-6 Gas Sensor

The MQ-6 Gas sensor can detect or measure gases like LPG and butane. The MQ-6 sensor module comes with a Digital Pin which makes this sensor to operate even without a microcontroller and that comes in handy when you are only trying to detect one particular gas. When it comes to measuring the gas in ppm the analog pin has to be used, the analog pin also TTL driven and works on 5V and hence can be used with most common microcontrollers.

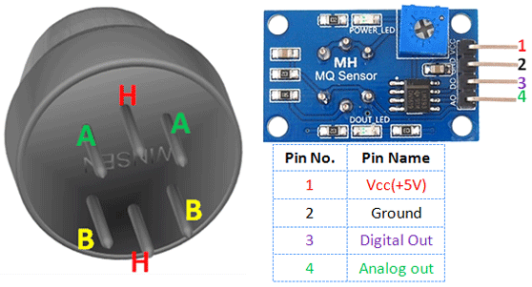
3.7.1 Description

- The MQ-6 module is used in gas leakage detecting equipment in family and industry, This module has high sensitivity to LPG, iso-butane, propane and LNG. It can also be used to detect the presence of alcohol, cooking fumes, and cigarette smoke.
- The module gives out the concentration of the gases as a analog voltage equivalent to the concentration of the gases. The module also has an onboard comparator for comparing against an adjustable preset value and giving out a digital high or low. It can be easily interfaced with your Arduino or Raspberry Pi.
- This is a simple-to-use MQ-6 Liquefied Petroleum, iso-butane, propane gas Sensor module, suitable for sensing LPG (composed of mostly propane and butane) concentrations in the air. The MQ-6 can detect gas concentrations anywhere from 200 to 10000ppm.This sensor has a high sensitivity and fast response time. The sensor's output is an analog resistance. The drive circuit is very simple; all you need to do is power the heater coil with 5V, add a load resistance, and connect the output to an ADC.
- Sensitive material of MQ-6 gas sensor is SnO2, which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.MQ-6 gas sensor has high sensitity to Propane, Butane and LPG, also response to Natural gas. The sensor

could be used to detect different combustible gas, especially Methane, it is with low cost and suitable for different application.

3.7.2 Pin Configuration

Pin No:	Pin Name:	Description
For MQ6 Sensor Module		
1	Vcc	This pin powers the module, typically the operating voltage is +5V
2	Ground	Used to connect the module to system ground
3	Digital Out	You can also use this sensor to get digital output from this pin, by setting a threshold value using the potentiometer
4	Analog Out	This pin outputs 0-5V analog voltage based on the intensity of the gas
For MQ6 Sensor		
1	H -Pins	Out of the two H pins, one pin is connected to supply and the other to ground
2	A-Pins	The A pins and B pins are interchangeable. These pins will be tied to Supply voltage.
3	B-Pins	The A pins and B pins are interchangeable. One pin will act as output while the other will be pulled to ground.



Pin No.	Pin Name
1	Vcc(+5V)
2	Ground
3	Digital Out
4	Analog out

Fig: MQ-6 Pin Out

3.7.3 Features of MQ6 Gas sensor

- Operating Voltage is +5V
- Can be used to detect LPG or Butane gas.
- Analog output voltage: 0V to 5V
- Digital Output Voltage: 0V or 5V (TTL Logic)
- Preheat duration 20 seconds.
- Can be used as a Digital or analog sensor. The Sensitivity of Digital pin can be varied using the potentiometer.
- High sensitivity to LPG, iso-butane, propane.
- Small sensitivity to alcohol, smoke.
- Load resistance: 20KΩ
- Heater resistance: 33Ω±5%
- Heating consumption: less than 750mw
- Using Temperature: -10°C-50°C
- Storage Temperature: -20°C-70°C
- Related humidity: less than 95%Rh
- Oxygen concentration: 21%(standard condition)Oxygen concentration can affect sensitivity.

3.7.4 Working Principle

The MQ6 sensor is the main sensing component which is used in a gas detection appliance. The sensor consist of a sensing material which ionize the gases which comes in its contact. As a result, the ionization process of the gases changes the resistance across the circuit.

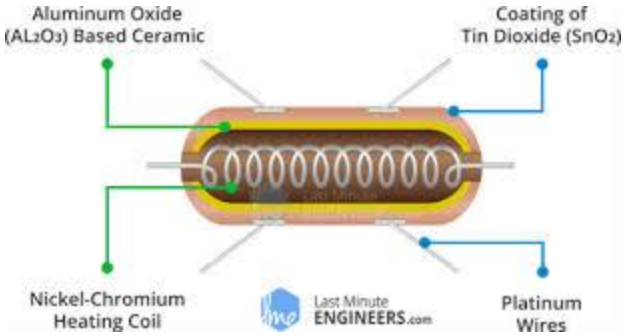


Fig: MQ-6 working principle

Using a MQ sensor it detect a gas is very easy. You can either use the digital pin or the analog pin to accomplish this. Simply power the module with 5V and you should notice the power LED on the module to glow and when no gas it detected the output LED will remain turned off meaning the digital output pin will be 0V. Remember that these sensors have to be kept on for pre-heating time (mentioned in features above) before you can actually work with it. Now, introduce the sensor to the gas you want to detect and you should see the output LED to go high along with the digital pin, if not use the potentiometer until the output gets high. Now every time your sensor gets introduced to this gas at this particular concentration the digital pin will go high (5V) else will remain low (0V).

You can also use the analog pin to achieve the same thing. Read the analog values (0-5V) using a microcontroller, this value will be directly proportional to the concentration of the gas to which the sensor detects. You can experiment with this values and check how the sensor reacts to different concentration of gas and develop your program accordingly.

3.7.5 Measuring ppm using MQ6

If you are looking for some accuracy with your readings then measuring the PPM would be the best way to go with it. It can also help you to distinguish one gas from another. So to measure PPM you can directly use a module. A basic wiring for the sensor from datasheet is shown below.

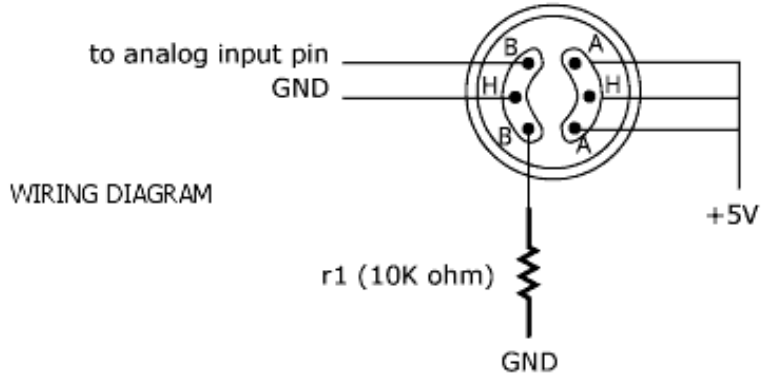


Fig: MQ-6 Pin Out

3.7.6 Deriving Gas concentration from Output Voltage

Here is a the equation which convert analog output to PPM gas concentration PPM = Analog Voltage in mV x 2

Example: Gas sensor voltage is giving output as 2500mV(2.5V)
So the gas contentration in PPM = 2500x2 = 5000 PPM
Here is the table for your reference

PPM:	**200**	**300**	**500**	**1000**	**2000**	**3000**	**5000**	**9000**	**10000**
Voltage(mV)	100	150	250	500	1000	1500	2500	4500	5000

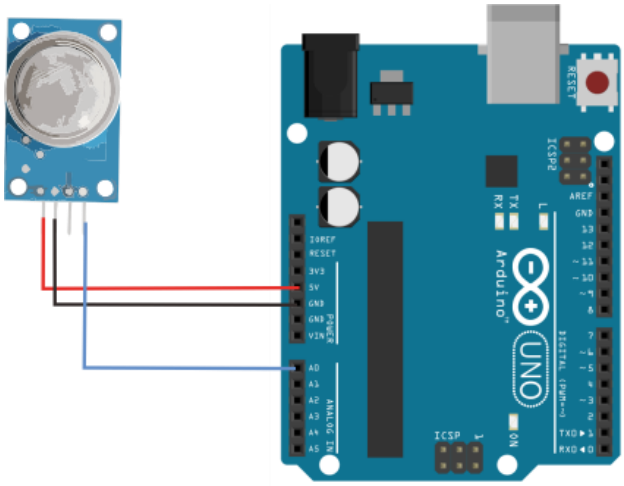
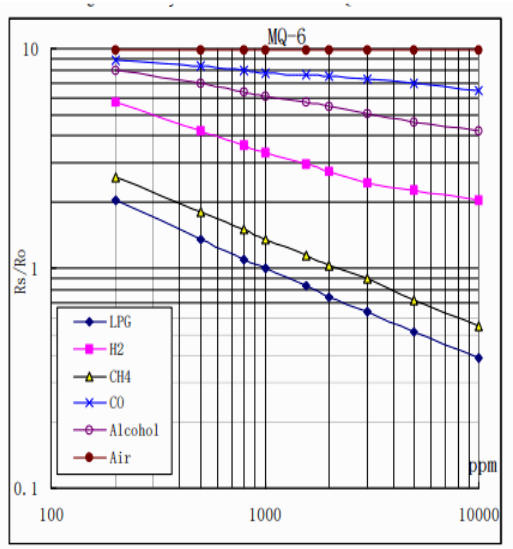


Fig: MQ-6 connections to Arduino Board

The procedure to measure PPM using MQ sensor is the same but few constant values will vary based on the type of MQ sensor used. Basically we need to look into the (Rs/Ro) VS PPM graph given in the **MQ-6 datasheet**, and also shown below:



The value of Ro is the value of resistance in fresh air and the value of Rs is the value of resistance in Gas concentration. First you should calibrate the sensor by finding the values of Ro in fresh air and then use that value to find Rs using the formulae:

Resistance of sensor(Rs): $Rs=(Vc/VRL-1)\times RL$

Once we calculate Rs and Ro we can find the ratio and then using the graph shown above we can calculate the equivalent value of PPM for that particular gas.

3.7.7 Applications

- Detect or measure Gases like LPG, and butane
- Air quality monitor
- Gas leak alarm
- Safety standard maintenance
- Maintaining environment standards in hospitals

3.8 HC-SR04 Ultrasonic Level Sensor

Fig: HC-SR04 Ultrasonic Level Sensor

3.8.1 Description

- HC-SR04 is an ultrasonic sensor mainly used to determine the distance of the target object.
- It measures accurate distance using a non-contact technology – A technology that involves no physical contact between sensor and object.
- Transmitter and receiver are two main parts of the sensor where former converts an electrical signal to ultrasonic waves while later converts that ultrasonic signals back to electrical signals.
- These ultrasonic waves are nothing but sound signals that can be measured and displayed at the receiving end.

3.8.2 Pin Configuration

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

4	Ground	This pin is connected to the Ground of the system.

Fig: HC-SR04 Pinout

3.8.3 HC-SR04 Sensor Features:

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA
- Operating Frequency: 40Hz

3.8.4 HC-SR04 Ultrasonic Sensor – Working

As shown above the HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below

Fig: Working principle of HC-SR04 Ultrasonic Level Sensor

Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

3.8.5 Using HC-SR04

HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used.

Power the Sensor using a regulated +5V through the Vcc ad Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.

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

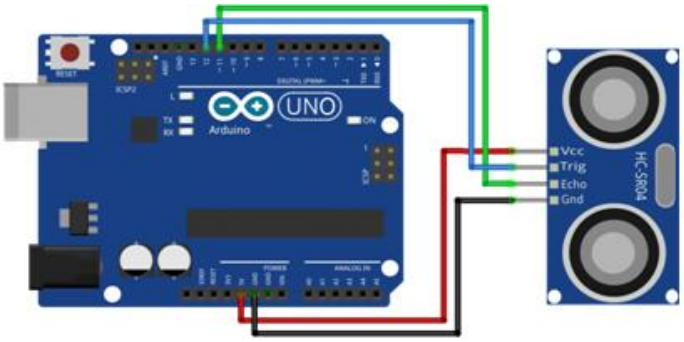


Fig: HC-SR04 Ultrasonic Level Sensor connections to Arduino Board.

3.8.6 Applications

- Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
- Used to measure the distance within a wide range of 2cm to 400cm
- Can be used to map the objects surrounding the sensor by rotating it
- Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water
- Speed and direction measurement
- Wireless charging
- Humidifiers
- Medical ultrasonography
- Burglar alarms
- Embedded system
- Depth measurement
- Non-destructive testing

3.9 Photodiode (Infrared Receiver) Flame Sensor

3.9.1 Description

A **flame detector** is a sensor designed to detect and respond to the presence of a flame or fire. Responses to a detected flame depend on the installation, but can include sounding an alarm, deactivating a fuel line (such as a propane or a natural gas line), and activating a fire suppression system.

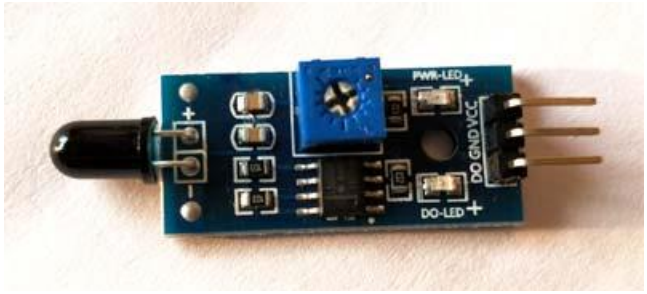


Fig: Photodiode Flame sensor

There are different types of flame detection methods. Some of them are: Ultraviolet detector, near IR array detector, infrared (IR) detector, Infrared thermal cameras, UV/IR detector etc.

When fire burns it emits a small amount of Infra-red light, this light will be received by the Photodiode (IR receiver) on the sensor module. Then we use an Op-Amp to check for change in voltage across the IR Receiver, so that if a fire is detected the output pin (DO) will give 0V(LOW) and if the is no fire the output pin will be 5V(HIGH).

In this project we are using an **IR based flame sensor**. It is based on the YG1006 sensor which is a high speed and high sensitive NPN silicon phototransistor. It can detect infrared light with a wavelength ranging from 700nm to 1000nm and its detection angle is about 60°. Flame sensor module consists of a photodiode (IR receiver), resistor, capacitor, potentiometer, and LM393 comparator in an integrated circuit. The sensitivity can be adjusted by varying the on board potentiometer. Working voltage is between 3.3v and 5v DC, with a digital output. Logic high on the output indicates presence of flame or fire. Logic low on output indicates absence of flame or fire.

3.9.2 Specifications

- Photosensitivity is high
- Response time is fast
- Simple to use
- Sensitivity is adjustable
- Detection angle is 600,
- It is responsive to the flame range.
- Accuracy can be adjustable
- Operating voltage of this sensor is 3.3V to 5V
- Analog voltage o/ps and digital switch o/ps
- The PCB size is 3cm X 1.6cm
- Power indicator & digital switch o/p indicator
- If the flame intensity is lighter within 0.8m then the flame test can be activated, if the flame intensity is high, then the detection of distance will be improved.

3.9.3 Pin Configuration

S.no	Pin	Description
1	Vcc	3.3 – 5V power supply
2	GND	Ground
3	Dout	Digital output

3.9.4 Working

The flame sensor detects the presence of fire or flame based on the Infrared (IR) wavelength emitted by the flame. It gives logic 1 as output if flame is detected, otherwise it gives logic 0 as output. Arduino Uno checks the logic level on the output pin of the sensor and performs further tasks such as activating the buzzer and LED, sending an alert message.

IR sensor consist an IR LED and photodiode, in which IR LED emits IR radiation and photodiode detects the radiation. Photodiode conducts current in reverse direction, whenever light falls on it, and voltage across it changes, this voltage change is sensed by voltage comparator (like LM358) and generates output accordingly.

Photodiode acts as the IR receiver as its conducts when light falls on it. Photodiode is a semiconductor which has a P-N junction, operated in Reverse Bias, means it start conducting the current in reverse direction when Light falls on it, and the amount of current flow is proportional to the amount of Light. This property makes it useful for IR detection. Photodiode looks like a LED, with a black colour coating on its outer side, Black colour absorbs the highest amount of light.

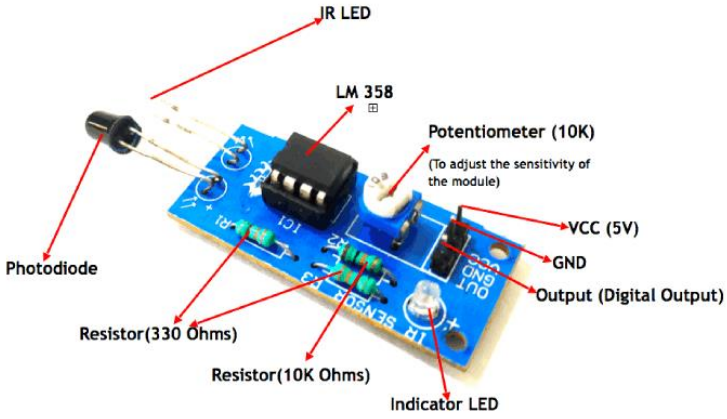


Fig: Photodiode Flame Sensor Module

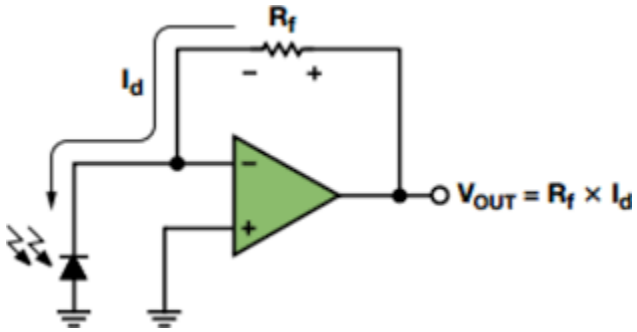


Fig: Circuit Design

3.9.5 Photodiode as Flame sensor

A sensor which is most sensitive to a normal light is known as a flame sensor. That's why this sensor module is used in flame alarms. This sensor detects flame otherwise wavelength within the range of 760 nm – 1100 nm from the light source. This sensor can be easily damaged to high temperature. So this sensor can be placed at a certain distance from the flame. The flame detection can be done from a 100cm distance and the detection angle will be 600. The output of this sensor is an analog signal or digital signal. These sensors are used in fire fighting robots like as a flame alarm.

This sensor/detector can be built with an electronic circuit using a receiver like electromagnetic radiation. This sensor uses the infrared flame flash method, which allows the sensor to work through a coating of oil, dust, water vapor, otherwise ice.

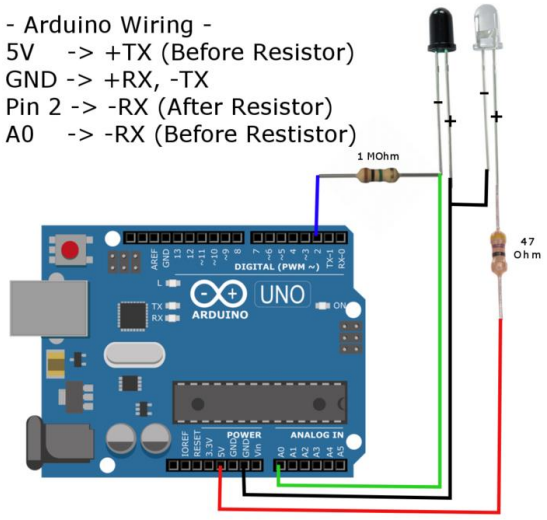


Fig: Photodiode Flame sensor connections to Arduino Board

3.9.6 Applications

- Hydrogen stations
- Industrial heating
- Fire detection
- Fire alarm
- Fire fighting robot
- Drying systems
- Industrial gas turbines
- Domestic heating systems

- Gas-powered cooking devices
- Combustion monitors for burners
- Oil and gas pipelines
- Automotive manufacturing facilities
- Nuclear facilities
- Aircraft hangars
- Turbine enclosures

3.10 Light Dependent Resistor (LDR)/ Photo resistor Light Sensor

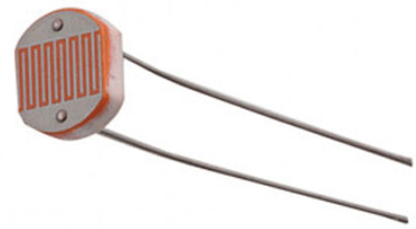


Fig: LDR Light sensor

3.10.1 Description

A Light Dependent Resistor (LDR) is also called a photoresistor or a cadmium sulfide (CdS) cell. It is also called a photoconductor. It is basically a photocell that works on the principle of photoconductivity. The passive component is basically a resistor whose resistance value decreases when the intensity of light decreases. This optoelectronic device is mostly used in light varying sensor circuit, and light and dark activated switching circuits.

LDRs or Light Dependent Resistors are very useful especially in light/dark sensor circuits. As the amount of light falling on this LDR increases, its resistance decreases. Light Dependent Resistor is suitable for use in projects which require a device or circuit to be automatically switched on or off in darkness or light. The light detector itself is just 12mm in diameter with a lens and epoxy sealed metal package.

The **Light Dependent Resistor** (**LDR**) is just another special type of Resistor and hence has no polarity. Meaning they can be connected in any direction. They are breadboard friendly and can be easily used on a perf board also. The symbol for LDR is just as similar to Resistor but adds to inward arrows as shown above. The arrows indicate the light signals.

The photo resistor module detects light intensity. As the intensity of light on the photoresistor increases, so the resistance of the photoresistor decreases. A built in 10k resistor on the module can be used to build a divider circuit for measuring changes in light intensity from

the photoresistor. A device such as an Arduino can be used to read the light intensity using and analog input pin.

3.10.2 Features

- High Reliability
- Heat Proof Construction
- High Power
- Wide spectal response
- Low cost
- Wide ambient temp range.
- Can be used to sense Light.
- Easy to use on Breadboard or Perf Board.
- Easy to use with Microcontrollers or even with normal Digital/Analog IC.
- Small, cheap and easily available.
- Available in PG5 ,PG5-MP, PG12, PG12-MP, PG20 and PG20-MP series.

3.10.3 Pin Configuration

The **Light Dependent Resistor** (**LDR**) is just another special type of Resistor and hence has no polarity. Meaning they can be connected in any direction.

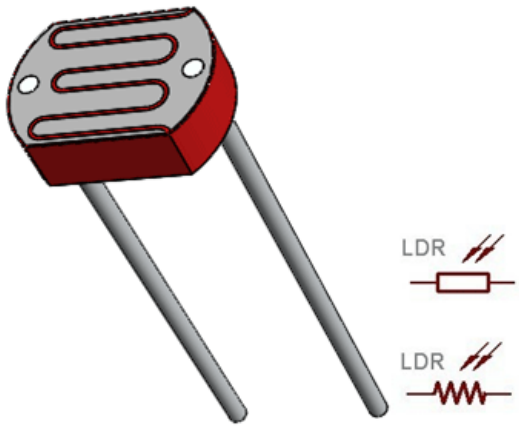


Fig: LDR pin out

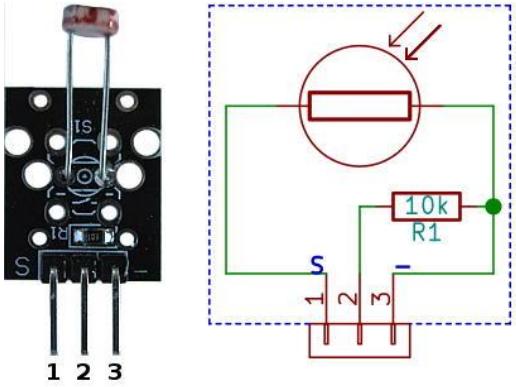
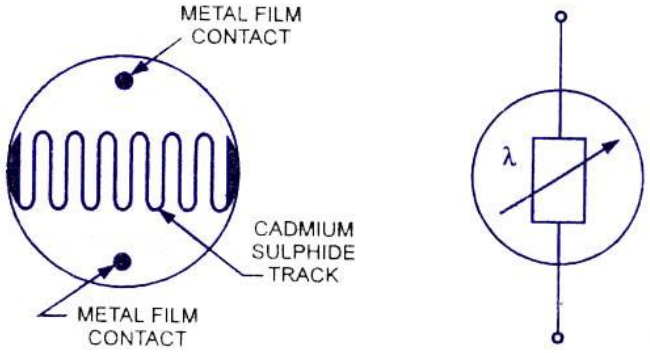


Fig: LDR sensor module Pin out

Resistance of the Photoresistor on the Module:
Resistance of the photoresistor is typically less than 80 Ohms (< 80Ω) in full light.

Dark resistance is typically greater than 20 Meg Ohms (> 20MΩ) – when the photoresistor is in full darkness.

3.10.4 Working



(a) Basic Structure *(b) Symbol*

LDR

The snake like track shown below is the Cadmium Sulphide (CdS) film which also passes through the sides. On the top and bottom are metal films which are connected to the terminal leads. It is designed in such a way as to provide maximum possible contact area with the two metal films. The structure is housed in a clear plastic or resin case, to provide free access to external light. As explained above, the main component for the construction of LDR is cadmium sulphide (CdS), which is used as the photoconductor and contains no or very few electrons when not illuminated. In the absence of light it is designed to have a high resistance inthe range of megaohms. As soon as light falls on the sensor, the electrons are liberated and the conductivity of the material increases. When the light intensity exceeds a certain frequency, the photons absorbed by the semiconductor give band electrons the energy required to jump into the conduction band. This causes the free electrons or holes to conduct electricity and thus dropping the resistance dramatically (< 1 Kiloohm).

The equation to show the relation between resistance and illumination can be written as

R = A.E^a

Where E–Illumination (lux)
R – Resistance (Ohms)
A,a – constants

The value of ‘a’ depends on the CdS used and on the manufacturing process. Values usually range betwee 0.7 and 0.9.

The working principle of an LDR is photoconductivity, that is nothing but an optical phenomenon. When the light is absorbed by the material then the conductivity of the material reduces. When the light falls on the LDR, then the electrons in the valence band of the material are eager to the conduction band. But, the photons in the incident light must have energy superior than the bandgap of the material to make the electrons jump from one band to another band (valance to conduction).

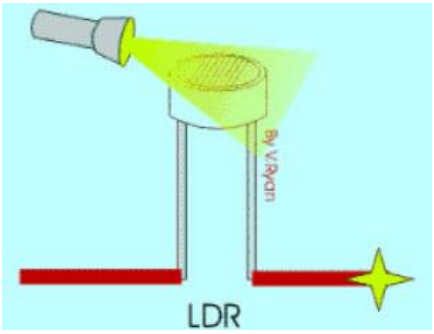


Fig: LDR Working Principle

Hence, when light having ample energy, more electrons are excited to the conduction band which grades in a large number of charge carriers. When the effect of this process and the flow of the current starts flowing more, the resistance of the device decreases.

3.10.5 Using a LDR sensor

As said earlier a LDR is one of the different types of resistors, hence using it is very easy. There are many ways and different circuit in which an LDR can be used. For instance it can be used with Microcontroller Development platforms like Arduino, PIC or even normal Analog IC’s like Op-amps. But, here we will use a very simple circuit like a potential divider so that it can be adapted for most of the projects.

A potential Divider is a circuit which has two resistors in series. A constant voltage will be applied across the both the resistor and the output voltage will be measured from the lower

resistor. In our case, the lower resistor will be a **LDR** and the constant voltage will be +5V. The set-up is shown below

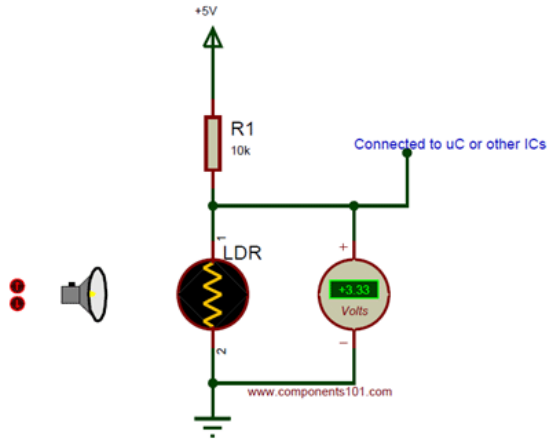


Fig: LDR pin out

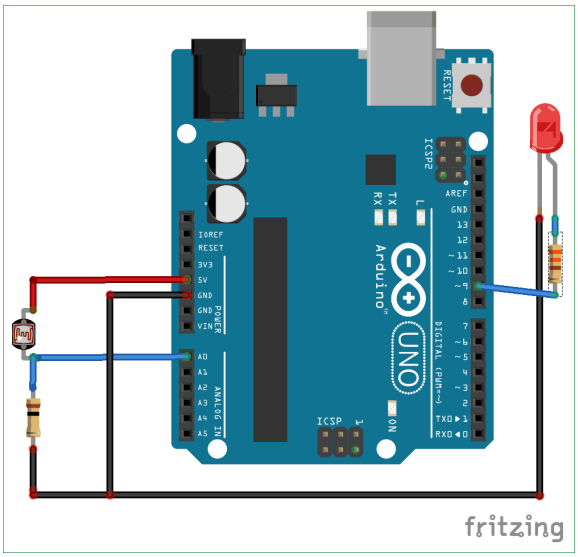


Fig: LDR connections to Arduino board

3.10.6 Applications

- Automatic Street Light
- Detect Day or Night
- Automatic Head Light Dimmer
- Position sensor
- Used along with LED as obstacle detector
- Automatic bedroom Lights
- Automatic Rear view mirror
- Flame monitor (for boiler) Analyzers
- Observation equipment

- Camera (electronic shutter)
- Strobe (colour temperature reading)

3.11 LM358 Operational Amplifier IC

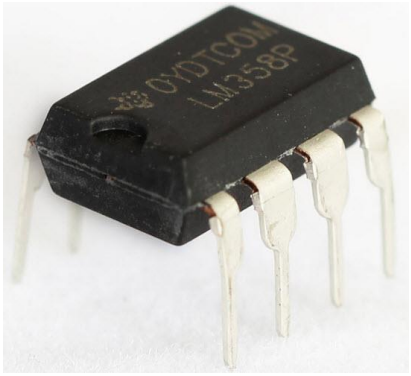


Fig: LM358 Op-Amp IC

3.11.1 Description

The LM358 IC is a great, low power and easy to use dual channel op-amp IC. It is designed and introduced by national semiconductor. It consists of two internally frequency compensated, high gain , independent op-amps. This IC is designed for specially to operate from a single power supply over a wide range of voltages. The LM358 IC is available in a chip sized package and applications of this op amp include conventional op-amp circuits, DC gain blocks and transducer amplifiers. LM358 IC is a good, standard operational amplifier and it is suitable for your needs. It can handle 3-32V DC supply & source up to 20mA per channel. This op-amp is apt, if you want to operate two separate op-amps for a single power supply. It's available in an 8-pin DIP package It is used in detector circuits. The abbreviation **LM358** indicates an 8-pin integrated circuit, comprising two operational amplifiers at low power.

3.11.2 Pin Configuration of LM358 IC

The pin diagram of LM358 IC comprises of 8 pins, where

- Pin-1 and pin-8 are o/p of the comparator
- Pin-2 and pin-6 are inverting i/ps
- Pin-3 and pin-5 are non inverting i/ps

- Pin-4 is GND terminal
- Pin-8 is VCC+

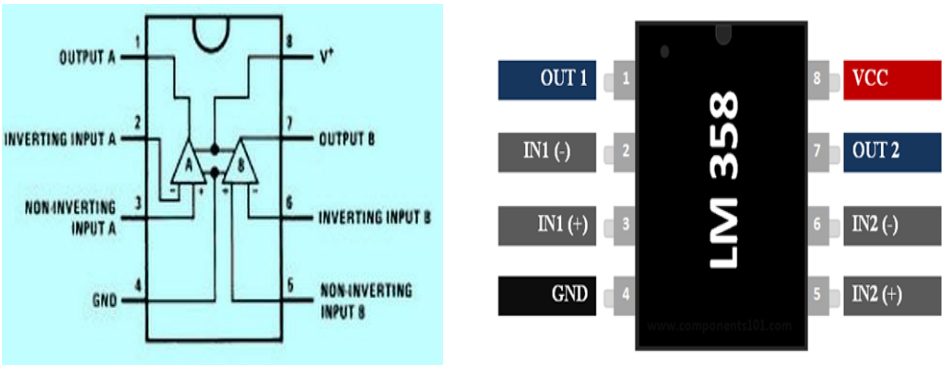


Fig: LM358 Op-Amp IC pin out

Pin Number	**Pin Name**	**Description**
1	OUTPUT1	Output of Op-Amp 1
2	INPUT1-	Inverting Input of Op-Amp 1
3	INPUT1+	Non-Inverting Input of Op-Amp 1
4	V_{EE}, GND	Ground or Negative Supply Voltage
5	INPUT2+	Non-Inverting Input of Op-Amp 2
6	INPUT2-	Inverting Input of Op-Amp 2
7	OUTPUT2	Output of Op-Amp 2
8	V_{CC}	Positive Supply Voltage

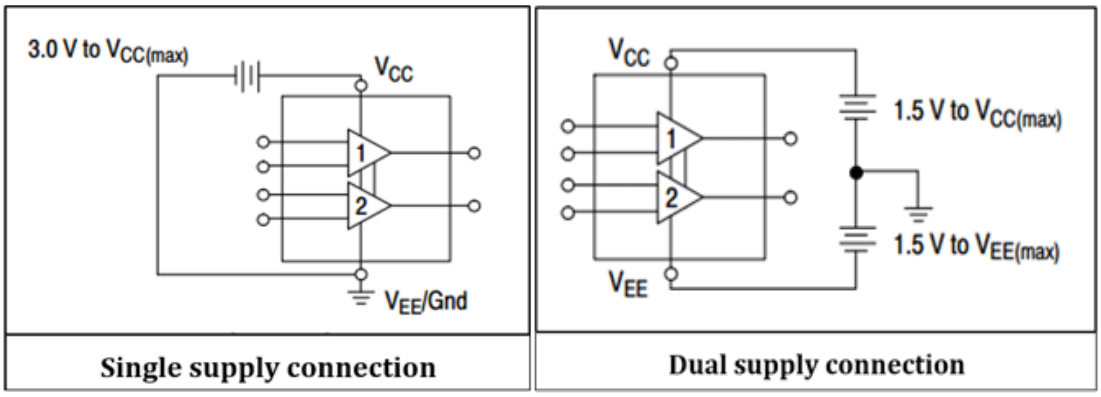
3.11.3 Features of LM358 IC

The features of the LM358 IC are

- It consists of two op-amps internally and frequency compensated for unity gain
- The large voltage gain is 100 dB
- Wide bandwidth is 1MHz
- Range of wide power supplies includes single and dual power supplies
- Range of Single power supply is from 3V to 32V

- Range of dual power supplies is from + or -1.5V to + or -16V
- The supply current drain is very low, i.e., 500 µA
- 2mV low i/p offset voltage
- Common mode i/p voltage range comprises ground
- The power supply voltage and differential i/p voltages are similar
- o/p voltage swing is large. Integrated with two Op-Amps in a single package
- Wide power supply Range
- Low Supply current – 700uA
- Single supply for two op-amps enables reliable operation
- Short circuit protected outputs
- Operating ambient temperature – 0˚C to 70˚C
- Soldering pin temperature – 260 ˚C (for 10 seconds – prescribed)
- Available packages: TO-99, CDIP, DSBGA, SOIC, PDIP, DSBGA

3.11.4 Supply Connection



Single supply connection | **Dual supply connection**

3.11.5 Working of circuit

Working of circuits is very simple as we know LM358 compares the voltage applied at input pin and provide you the output. The voltage level which we want to detect is applied on either of the input pins and the voltage to be detected is applied on the other pin. For dark sensor circuits we are applying voltage on negative pin and voltage to be detected is applied on positive pin. Whenever input voltage applied on positive pin because of light which falls on LDR, photodiode and phototransistor slightly rises above the voltage on negative pin the output suddenly raises to the positive maximum and remains positive until the input voltage falls below the level to be detected. Transistor T1 is used to amplify the signals to drive the LED and resistor R1 is used as current limiter to protect LED and for light sensor circuit just reverse of above phenomenon is occurring. You just change the LDR with photodiode and photo transistor and see circuit is working properly and sensitivity of circuit also increases.

You can also connect different components and devices at the output, so try connecting your own device and see the output.

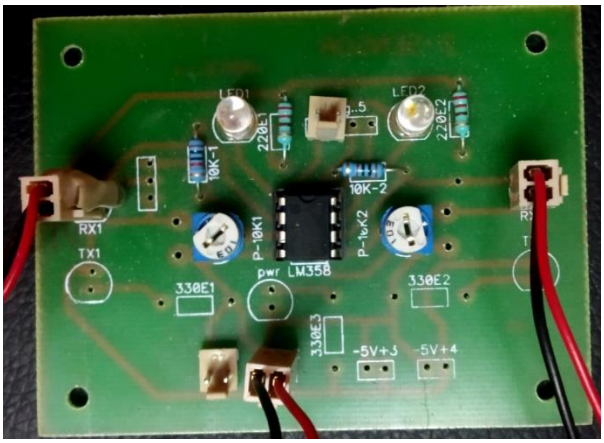


Fig: 2 Sensor card with LM358 Op-Amp IC

3.11.6 Advantages of LM358 IC

- Two operational amplifiers are compensated internally
- Two internally compensated op amps
- Removes the necessity of dual supplies
- Permits direct sensing close to GND & VOUT
- Well-suited with all methods of logic
- Power drains appropriate for the operation of the battery

3.11.7 Applications

- Transducer Amplifiers
- Conventional op-amp circuits
- Integrator, Differentiator, Summer, adder, Voltage follower, etc.,
- DC gain blocks, Digital multimeters, Oscilloscopes
- Comparators (Loop control & regulation) DC gain blocks.
- General signal conditioning.
- Active filters.
- Current loop transmitters for 4 to 20mA.

3.12 ULN2003A Relay Driver IC

Fig: ULN2003A Relay Driver IC

3.12.1 Description

A Relay driver IC is an electro-magnetic switch that will be used whenever we want to use a low voltage circuit to switch a light bulb ON and OFF which is connected to 220V mains supply. The required current to run the relay coil is more than can be supplied by various integrated circuits like Op-Amp, etc.Relays have unique properties and are replaced with solid state switches that are strong than solid-state devices. High current capacities, capability to stand ESD and drive circuit isolation are the unique properties of Relays. There are various ways to drive relays.Some of the Relay Driver ICs are as below:

- High side toggle switch driver
- Low side toggle switch driver
- Bipolar NPN transistor driver
- N-Channel MOSFET driver and
- Darlington transistor driver
- ULN2003 driver

  The relay driver uln2003 ic is a high voltage and current darlington array ic, it comprises of 7-open collector darlington pairs with common emitters. A pair of darlington is an arrangement of two bipolar transistors. This IC belongs to the family of ULN200x ICs and various types of this family interface to various logic families. This ULN2003 IC is for 5V TTL and CMOS logic devices. These ICs are used as relay drivers as well as to drive a wide range of loads, line drivers, display drivers etc. This IC is also normally used while driving Stepper Motors. The pairs of darlington in ULN2003 is esteemed at 500mA and can withstand peak current of 600mA.In the pin layout, the i/ps & o/ps are provided reverse to each other. Each driver also has a suppression diode to dissipate voltage spikes while driving inductive loads.

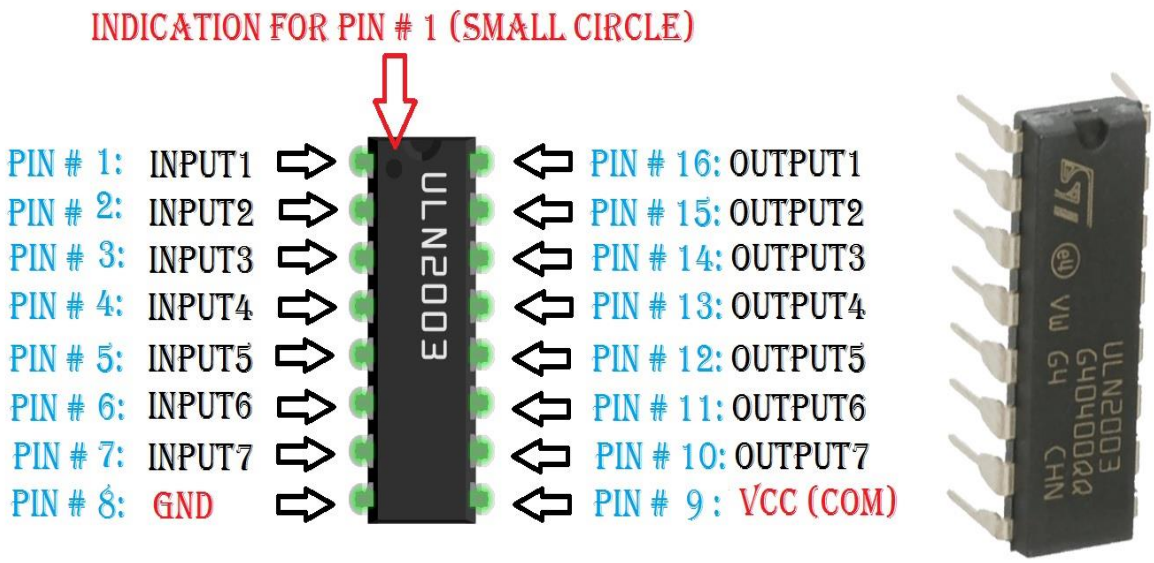
It is basically a relay driver IC and it is a darlington array having high voltages and high currents as well. It is made up of seven open collector darlington pairs having common emitter which shows ULN2003 has a capability of handling seven different relays at a time.
A single darlington pair consists of two bipolar transistors and it operates on the current range of 500mA to 600mA. ULN200X is a well known series of IC's. ULN2003 is also the part of this series. ULN2003 operates on 5V and TTL (Transistor Transistor Logic) and CMOS (Complementary Metal Oxide Semi Conductor). Its pin configuration is designed so that the input pins are at the left side of the IC whereas the the output pins of it are on right side in front of the corresponding input pin. This IC has a very wide range of applications. They are commonly used as relay drivers in order to drive different kinds of loads. ULN2003A can also be used to drive different motors (e.g. DC Motors or Stepper Motors) with Microcontrollers (like Arduino, PIC Microcontroller or 8051 Microcontroller etc.) . Some of the other *applications of ULN2003* include logic buffers, lamp drivers, line drivers, LED display, motor driver circuits etc.

3.12.2 Pin Configuration

ULN2003 has **16** pins in total out of which there are:

- **7 Input** pins (Pin # 1 to Pin # 7)
- **7 Output** pins (Pin # 10 to Pin # 16)
- **1 Ground** pin (Pin # 8)
- **1 COM** pin (Pin # 9)

Pin Number	Pin Name	Description
1 to 7	Input 1 to Input 7	Seven Input pins of Darlington pair, each pin is connected to the base of the transistor and can be triggered by using +5V
8	Ground	Ground Reference Voltage 0V
9	COM	Used as test pin or Voltage suppresser pin (optional to use)
10 to 1	Output 1 to Output 7	Respective outputs of seven input pins. Each output pin will be connected to ground only when its respective input pin is high(+5V)

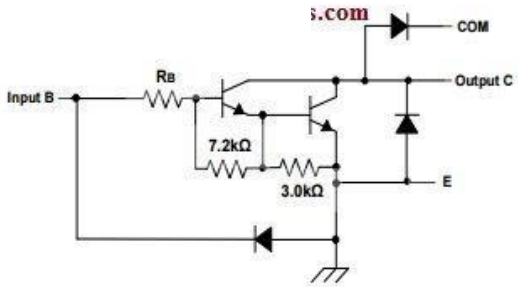


ULN2003 PINOUT

Its pin division on the basis of functions associated with them is shown in above figure. Further description will be given in this tutorial later. In short the pins can be divided into four different categories i.e. input pins, output pins, ground pin and common pin. These pins along with their positions are shown.

3.12.3 Internal Circuit Diagram

Internal circuit diagram of ULN 2003 having different resistors and diodes is shown in the figure below.



3.12.4 Features

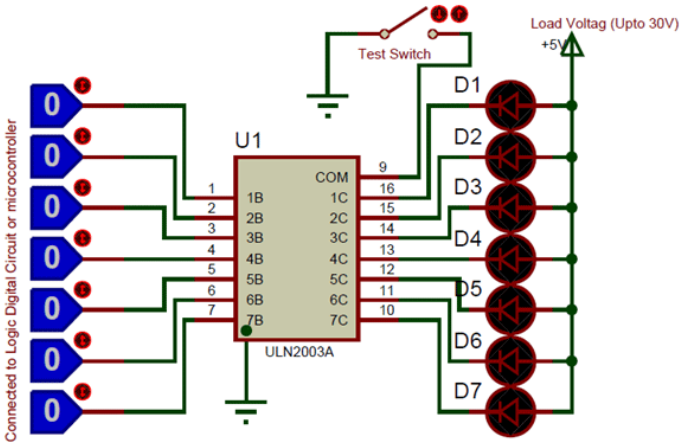
- Contains 7 high-voltage and high current Darlington pairs
- Each pair is rated for 50V and 500mA
- Input pins can be triggered by +5V
- All seven Output pins can be connected to gather to drive loads up to (7×500mA) ~3.5A.
- Can be directly controlled by logic devices like Digital Gates, Arduino, PIC etc

- Available in 16-pin DIP, TSSOP, SOIC packages500mA of rated collector.
- High output voltage of around 50V.
- Relay driver applications.
- Output clamp diodes.
- Compatible input with popular logic types.

3.12.5 Using a ULN2003

ULN2003 IC is one of the most commonly used **Motor driver IC**. This IC comes in handy when we need to drive high current loads using digital logic circuits like Op-maps, Timers, Gates, Arduino, PIC, ARM etc. For example a motor that requires 9V and 300mA to run cannot be powered by an Arduino I/O hence we use this IC to source enough current and voltage for the load. This IC is commonly used to drive Relay modules, Motors, high current LEDs and even Stepper Motors. So if you have anything that anything more than 5V 80mA to work, then this IC would be the right choice for you.

The **ULN2003** is a 16-pin IC. It has seven Darlington Pairs inside, where each can drive loads up to 50V and 500mA. For these seven Darlington Pairs we have seven Input and Output Pins. Adding to that we can a ground and Common pin. The ground pin, as usual is grounded and the usage of Common pin is optional. It might be surprising to note that this IC does not have any Vcc (power) pin; this is because the power required for the transistors to work will be drawn from the input pin itself. The below circuit is a simple circuit that can be used to test the **working of ULN2003 IC**.



In the circuit consider the LED to be the loads and the logic pins (blue colour) as the pins connected to the Digital circuit or Microcontroller like Arduino. Notice that the Positive pin of the LED is connected to the positive load voltage and the negative pin is connected to the output pin of the IC. This is because when the input pin of the IC gets high the respective output pin will get connected to ground. So when the negative terminal of the LED is grounded it completes the circuit and thus glows. The loads connected to the output pin can be maximum of 50C and 500mA each. However you can run higher current loads buy

combining two or more output pins to gather. For example if you combine three pins you can drive up to (3*500mA) ~1.5A.

The COM pin is connected to ground through a switch, this connection is optional. It can be used a test switch, meaning when this pin is grounded all the output pins will be grounded.

3.12.6 Applications

- Used to drive high current loads using Digital Circuits
- High current LED's can be driven
- Relay Driver module (can drive 7 relays)
- Logic Buffers in digital electronics
- Used as Touch sensor for Arduino Logic buffers.
- Line drivers.
- Lamp drivers.
- LED display drivers (display devices).
- Motor (stepper and DC brushed motor) drivers.

3.13 Relay

Fig: 5V Relay

3.13.1 Description

A relay can be defined as a switch. Switches are generally used to close or open the circuit manually .Relay is also a switch that connects or disconnects two circuits. But instead of manual operation a relay is applied with electrical signal, which in turn connects or disconnects another circuit.

Relays can be of different types like electromechanical, solid state. Electromechanical relays are frequently used. Let us see the internal parts of this relay before knowing about it working. Although many different types of relay were present, their working is same.

Every electromechanical relay consists of an consists of an

1. Electromagnet
2. Mechanically movable contact
3. Switching points and
4. Spring

Electromagnet is constructed by wounding a copper coil on a metal core. The two ends of the coil are connected to two pins of the relay as shown. These two are used as DC supply pins.

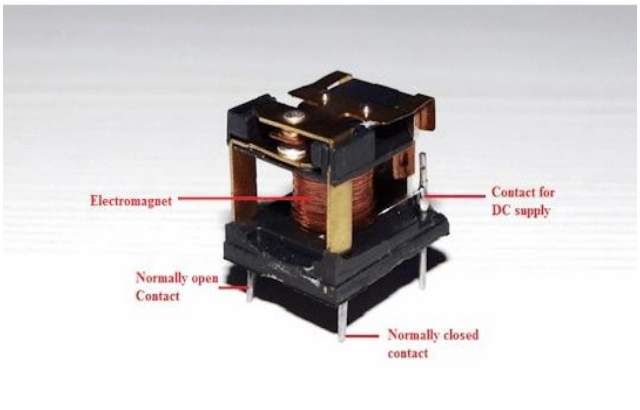


Fig: Relay circuit connection

Generally two more contacts will be present, called as switching points to connect high ampere load. Another contact called common contact is present in order to connect the switching points. These contacts are named as normally open (NO),normally closed(NC) and common(COM) contacts.
Relay can be operated using either AC or DC.
In case of AC relays, for every current zero position, the relay coil gets demagnetized and hence there would be a chance of continues breaking of the circuit.
So, AC relays are constructed with special mechanism such that continues magnetism is provided in order to avoid above problem. Such mechanisms include electronic circuit arrangement or shaded coil mechanism.

3.13.2 Pin Configuration

Pin Number	**Pin Name**	**Description**
1	Coil End 1	Used to trigger(On/Off) the Relay, Normally one end is connected to 5V and the other end to ground

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

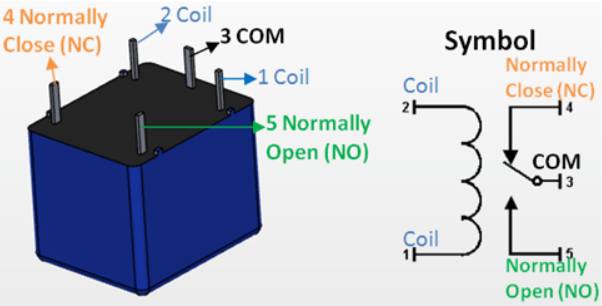


Fig: Relay pin out

3.13.3 Features of 5-Pin 5V Relay:

- Trigger Voltage (Voltage across coil) : 5V DC
- Trigger Current (Nominal current) : 70mA
- Maximum AC load current: 10A @ 250/125V AC
- Maximum DC load current: 10A @ 30/28V DC
- Compact 5-pin configuration with plastic moulding
- Operating time: 10msec Release time: 5msec
- Maximum switching: 300 operating/minute (mechanically)

3.13.4 Working

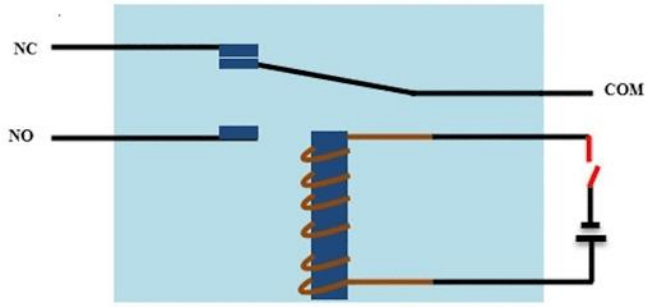


Fig: Working of a Relay

- Relay works on the principle of electromagnetic induction.
- When the electromagnet is applied with some current it induces a magnetic field around it.
- Above image shows working of the relay .A switch is used to apply DC current to the load.
- In the relay Copper coil and the iron core acts as electromagnet.
- When the coil is applied with DC current it starts attracting the contact as shown. This is called energizing of relay.
- When the supply is removed it retrieves back to the original position. This is called De energizing of relay.

There are also such relays, whose contacts are initially closed and opened when there is supply i.e. exactly to opposite to the above shown relay.
Solid state relays will have sensing element to sense the input voltage and switches the output using opto-coupling.

3.13.5 Relay Contact Types

As we have seen that relay is a switch. The terminology “Poles and throws” is also applicable for relay. Depending on the number of contacts and number of circuits it switches relays can be classified.
Before we know about this classification of contacts we have to know the poles and throws of a relay switch.

3.13.6 Poles and Throws

Relays can switch one or more circuits. Each switch in relay is referred as pole. Number of circuits a relay connects is indicated by throws.
Depending on the poles and throws, relays are classified into

- Single pole single throw
- Single pole double throw

- Double pole single throw
- Double pole double throw

Single Pole Single Throw

A single pole single throw relay can control one circuit and can be connected to one output. It is used for the applications which require only ON or OFF state.

Single Pole Double Throw

A single pole double throw relay connects one input circuit to one of the two outputs. This relay is also called as changeover relay.

Though the SPDT has two output positions, it may consist of more than two throws depends on the configuration and requirement of the application.

Double pole single throw

A double pole single throw relay has two poles and single throw and it can be used to connect two terminals of a single circuit at a time. For example, this relay is used for connecting both phase and neutral terminals to the load at a time.

Double pole double throw

A DPDT (double pole double throw) relay has two poles and two throws for each pole. In motor direction control, these are used for phase or polarity reversal.

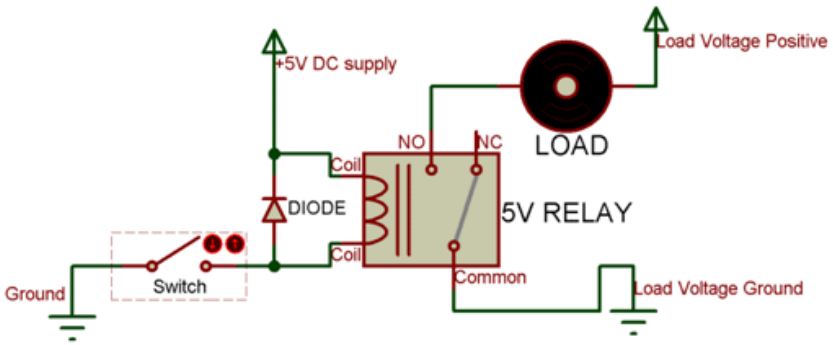
The switching action between contacts for all these relays is performed when the coil get energized as shown in figure below.

Relays can be classified into different types depending on their functionality, structure, application etc.

3.13.7 How to use a Relay?

Relays are most commonly used switching device in electronics. Let us learn how to use one in our circuits based on the requirement of our project.

Before we proceed with the circuit to drive the relay we have to consider two important parameter of the relay. Once is the **Trigger Voltage**, this is the voltage required to turn on the relay that is to change the contact from Common->NC to Common->NO. Our relay here has 5V trigger voltage, but you can also find relays of values 3V, 6V and even 12V so select one based on the available voltage in your project. The other parameter is your **Load Voltage & Current**, this is the amount of voltage or current that the NC,NO or Common terminal of the relay could withstand, in our case for DC it is maximum of 30V and 10A. Make sure the load you are using falls into this range.



The above circuit shows a bare-minimum concept for a relay to operate. Since the relay has 5V trigger voltage we have used a +5V DC supply to one end of the coil and the other end to ground through a switch. This **switch** can be anything from a small transistor to a microcontroller or a microprocessor which can perform switching operating. You can also notice a diode connected across the coil of the relay, this diode is called the **Fly back Diode**. The purpose of the diode is to protect the switch from high voltage spike that can produced by the relay coil. As shown one end of the load can be connected to the Common pin and the other end is either connected to NO or NC. If connected to NO the load remains disconnected before trigger and if connected to NC the load remains connected before trigger.

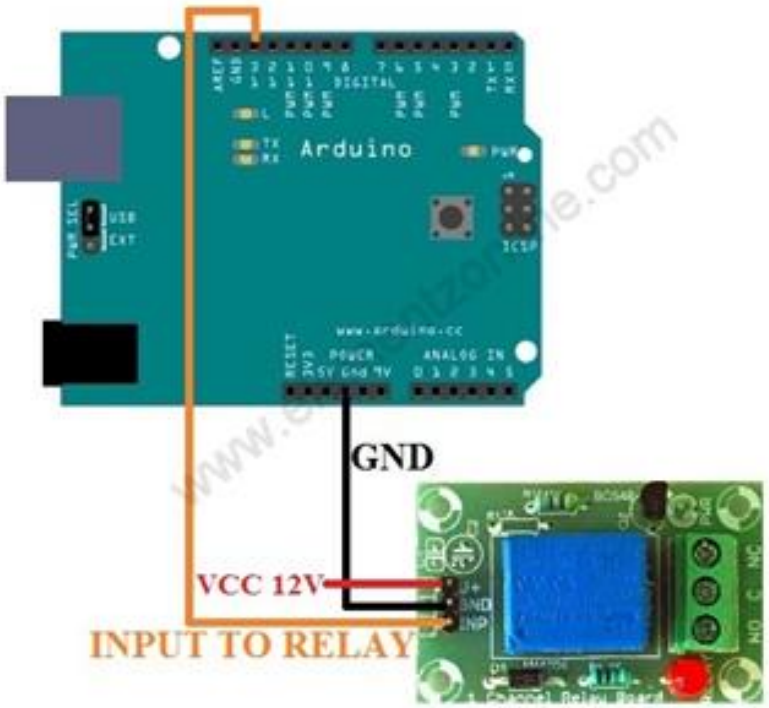


Fig: Relay connection to Arduino

Fig:4 Relay Board with ULN2003A Driver IC

3.13.8 Applications of Relay

Relays are used to protect the electrical system and to minimize the damage to the equipment connected in the system due to over currents/voltages. The relay is used for the purpose of protection of the equipment connected with it. These are used to control the high voltage circuit with low voltage signal in applications audio amplifiers and some types of modems. These are used to control a high current circuit by a low current signal in the applications like starter solenoid in automobile. These can detect and isolate the faults that occurred in power transmission and distribution system. Typical application areas of the relays include

- Lighting control systems
- Telecommunication
- Industrial process controllers
- Traffic control
- Motor drives control
- Protection systems of electrical power system
- Computer interfaces
- Automotive
- Home appliances commonly used in switching circuits.
- For Home Automation projects to switch AC loads
- To Control (On/Off) Heavy loads at a pre-determined time/condition
- Used in safety circuits to disconnect the load from supply in event of failure
- Used in Automobiles electronics for controlling indicators glass motors etc

3.14 Piezoelectric Buzzer

Fig: Piezoelectric Buzzer

3.14.1 Description

Piezo buzzer is an electronic device commonly used to produce sound. Light weight, simple construction and low price make it usable in various applications like car/truck reversing indicator, computers, call bells etc. Piezo buzzer is based on the inverse principle of piezo electricity discovered in 1880 by Jacques and Pierre Curie. It is the phenomena of generating electricity when mechanical pressure is applied to certain materials and the vice versa is also true. Such materials are called piezo electric materials. Piezo electric materials are either naturally available or manmade. Piezoceramic is class of manmade material, which poses piezo electric effect and is widely used to make disc, the heart of piezo buzzer. When subjected to an alternating electric field they stretch or compress, in accordance with the frequency of the signal thereby producing sound.

The **piezo buzzer** produces sound based on reverse of the piezoelectric effect. The generation of pressure variation or strain by the application of electric potential across a piezoelectric material is the underlying principle. These buzzers can be used alert a user of an event corresponding to a switching action, counter signal or sensor input. They are also used in alarm circuits.

The buzzer produces a same noisy sound irrespective of the voltage variation applied to it. It consists of piezo crystals between two conductors. When a potential is applied across these crystals, they push on one conductor and pull on the other. This, push and pull action, results in a sound wave. Most buzzers produce sound in the range of 2 to 4 kHz.

The Red lead is connected to the Input and the Black lead is connected to Ground.

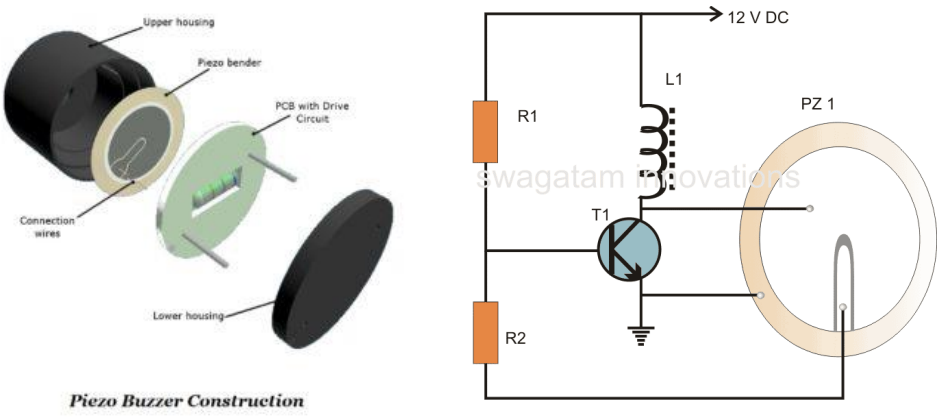


Fig: Inside connections of a piezobuzzer

3.14.2 Buzzer Pin Configuration

Pin Number	Pin Name	Description
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

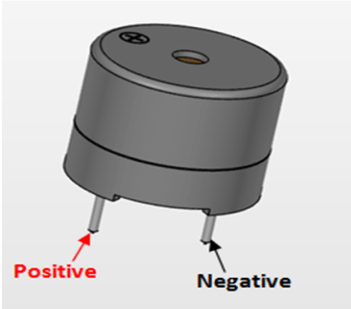


Fig: Piezoelectric buzzer pin out

3.14.3 Buzzer Features and Specifications

- Rated Voltage: 6V DC
- Operating Voltage: 4-8V DC
- Rated current: <30mA
- Sound Type: Continuous Beep
- Resonant Frequency: ~2300 Hz

3.14.4 Working

When a small DC voltage is applied to the input pins, it is first converted to an oscillating signal using the combination of resistor and transistor. These oscillating signals are amplified using the inductor coil. When high voltage alternating signals are applied to the piezo ceramic disc, it causes mechanical expansion and contraction in radial direction. This causes the metal plate to bend in opposite direction. When metal plate bends and shrinks in opposite direction continuously it produces sound waves in the air.

3.14.5 Using a Buzzer

A **buzzer** is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications. There are two types are buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beeeeeeppp.... sound, the other type is called a readymade buzzer which will look bulkier than this and will produce a Beep. Beep. Beep. Sound due to the internal oscillating circuit present inside it. But, the one shown here is most widely used because it can be customised with help of other circuits to fit easily in our application. This buzzer can be used by simply powering it using a DC power supply ranging from 4V to 9V. A simple 9V battery can also be used, but it is recommended to use a regulated +5V or +6V DC supply. The buzzer is normally associated with a switching circuit to turn ON or turn OFF the buzzer at required time and require interval.

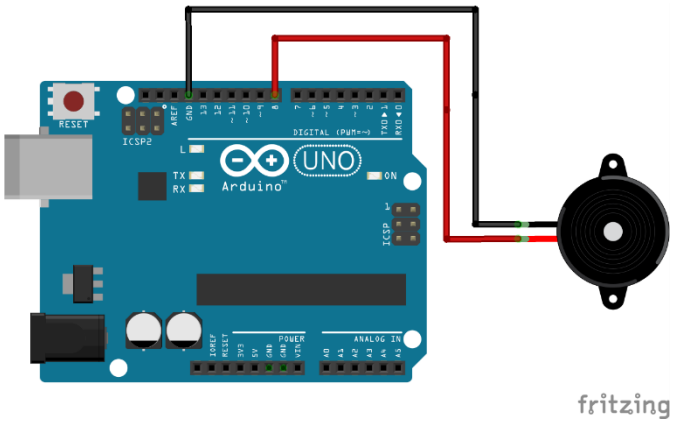


Fig: Buzzer Connection to Arduino

3.14.6 Applications of Buzzer

- Alarming Circuits, where the user has to be alarmed about something
- Communication equipments
- Automobile electronics

CHAPTER 4
SOFTWARE MODULES

4.1 Arduino IDE (Integrated Development Environment)

4.1.1 Introduction to Arduino IDE

- Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
- This environment supports both C and C++ languages.

4.1.2 How to download Arduino IDE?

We can download the Software from Arduino main website. As said earlier, the software is available for common operating systems like Linux, Windows, and MAX, so make sure you are downloading the correct software version that is easily compatible with your operating system.

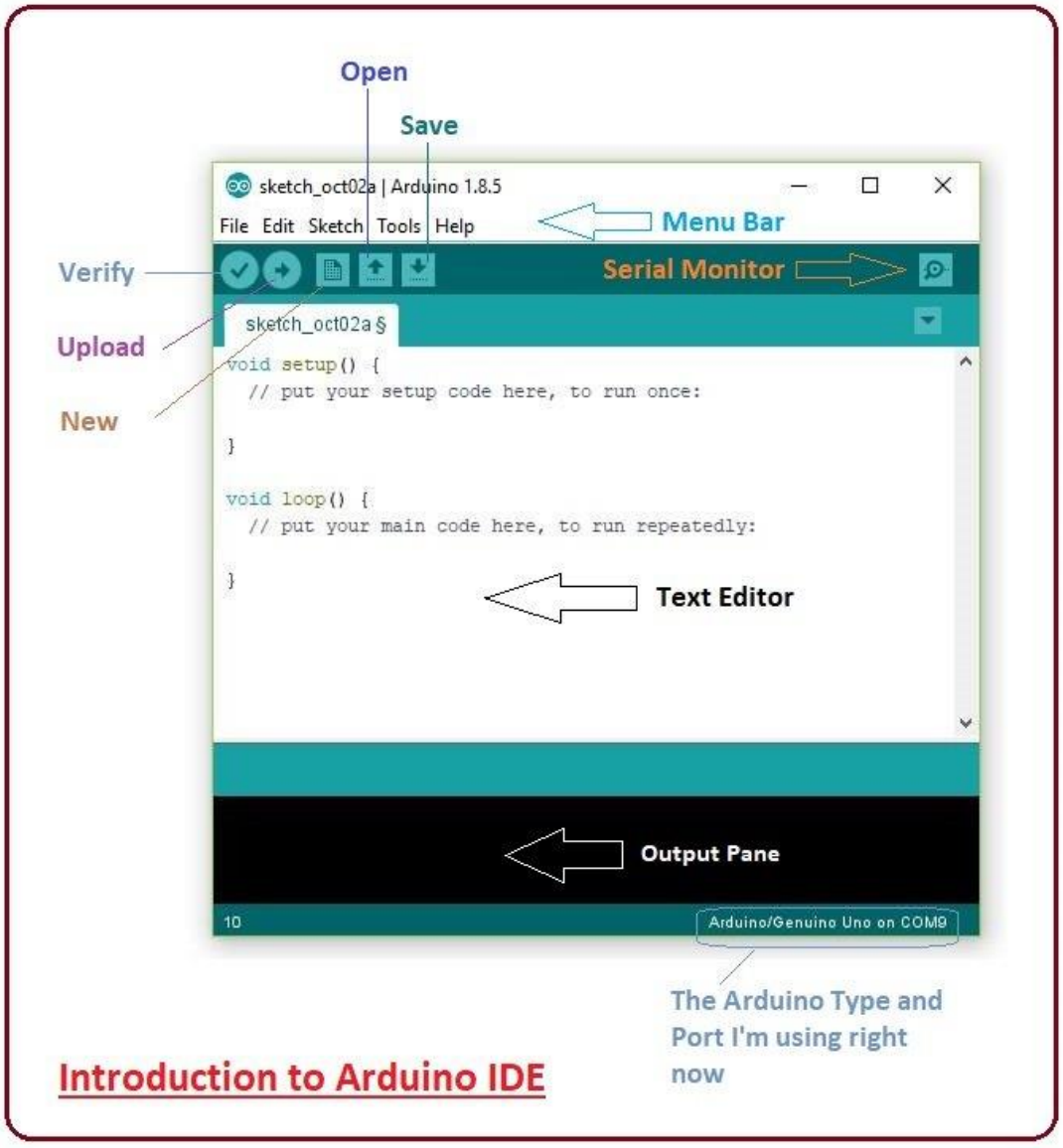
- If you aim to download Windows app version, make sure you have Windows 8.1 or Windows 10, as app version is not compatible with Windows 7 or older version of this operating system.

4.1.3 Details of IDE

The IDE environment is mainly distributed into three sections

- **1. Menu Bar**
- **2. Text Editor**
- **3. Output Pane**

As you download and open the IDE software, it will appear like an image below.

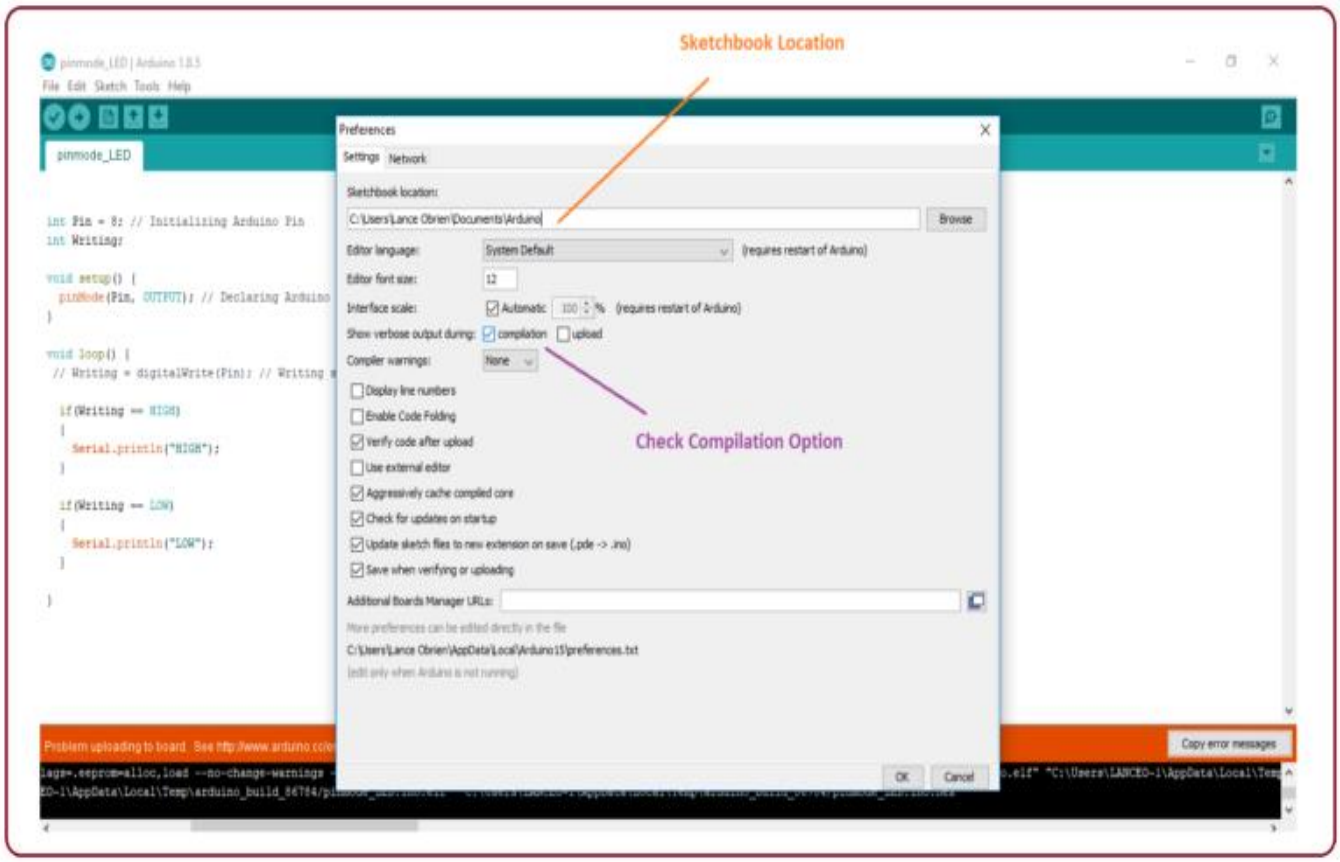


The bar appearing on the top is called **Menu Bar** that comes with five different options as follow

- **File** – You can open a new window for writing the code or open an existing one. Following table shows the number of further subdivisions the file option is categorized into.

File	
New	This is used to open new text editor window to write your code
Open	Used for opening the existing written code
Open Recent	The option reserved for opening recently closed program
Sketchbook	It stores the list of codes you have written for your project
Examples	Default examples already stored in the IDE software
Close	Used for closing the main screen window of recent tab. If two tabs are open, it will ask you again as you aim to close the second tab
Save	It is used for saving the recent program
Save as	It will allow you to save the recent program in your desired folder
Page setup	Page setup is used for modifying the page with portrait and landscape options. Some default page options are already given from which you can select the page you intend to work on
Print	It is used for printing purpose and will send the command to the printer
Preferences	It is page with number of preferences you aim to setup for your text editor page
Quit	It will quit the whole software all at once

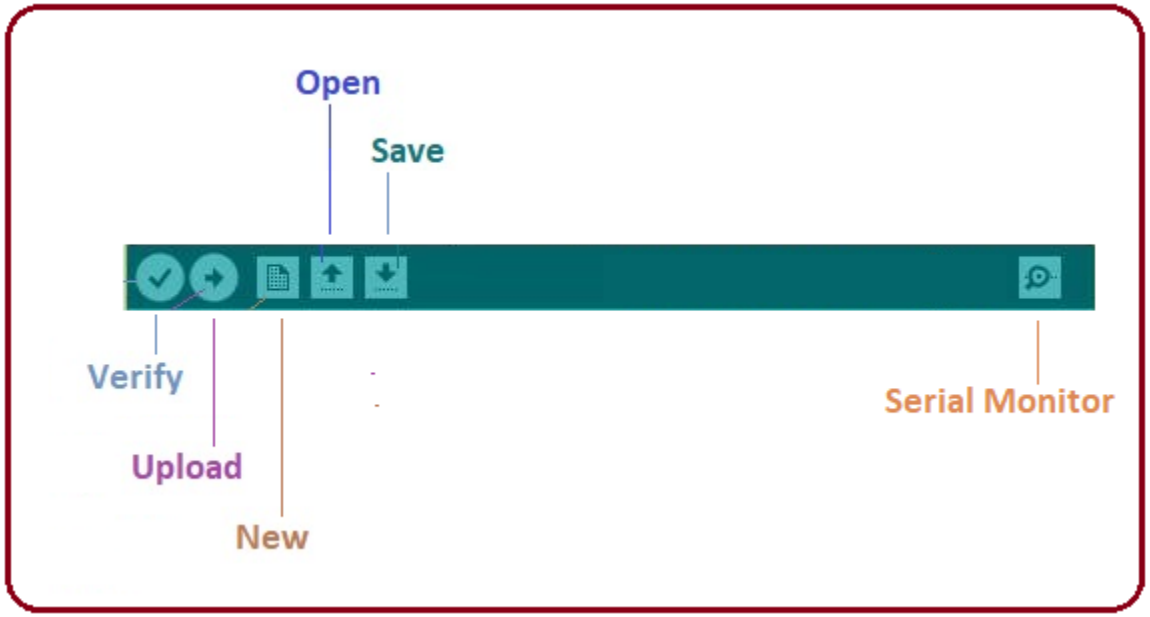
As you go to the preference section and check the compilation section, the Output Pane will show the code compilation as you click the upload button.



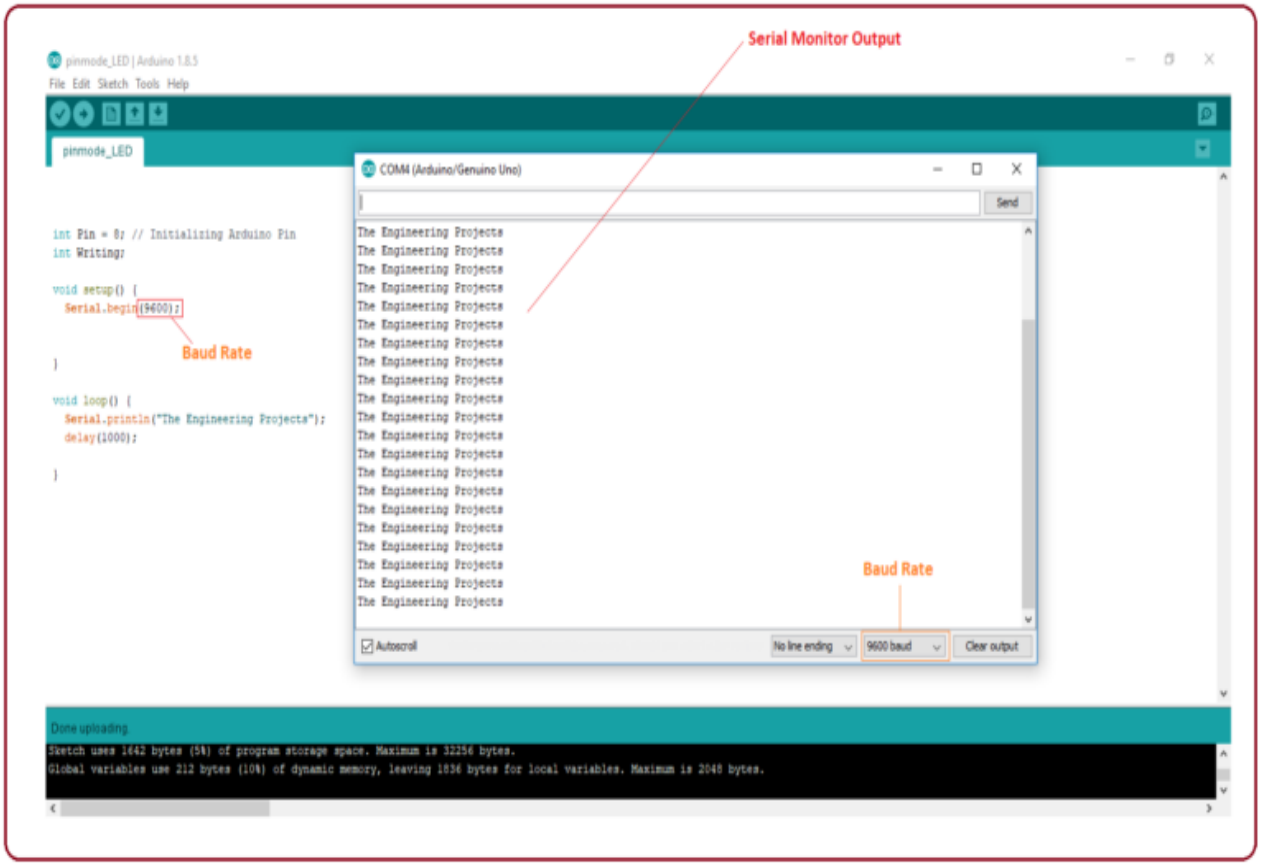
And at the end of compilation, it will show you the hex file it has generated for the recent sketch that will send to the Arduino Board for the specific task you aim to achieve.

- **Edit** – Used for copying and pasting the code with further modification for font
- **Sketch** – For compiling and programming
- **Tools** – Mainly used for testing projects. The Programmer section in this panel is used for burning a bootloader to the new microcontroller.
- **Help** – In case you are feeling skeptical about software, complete help is available from getting started to troubleshooting.

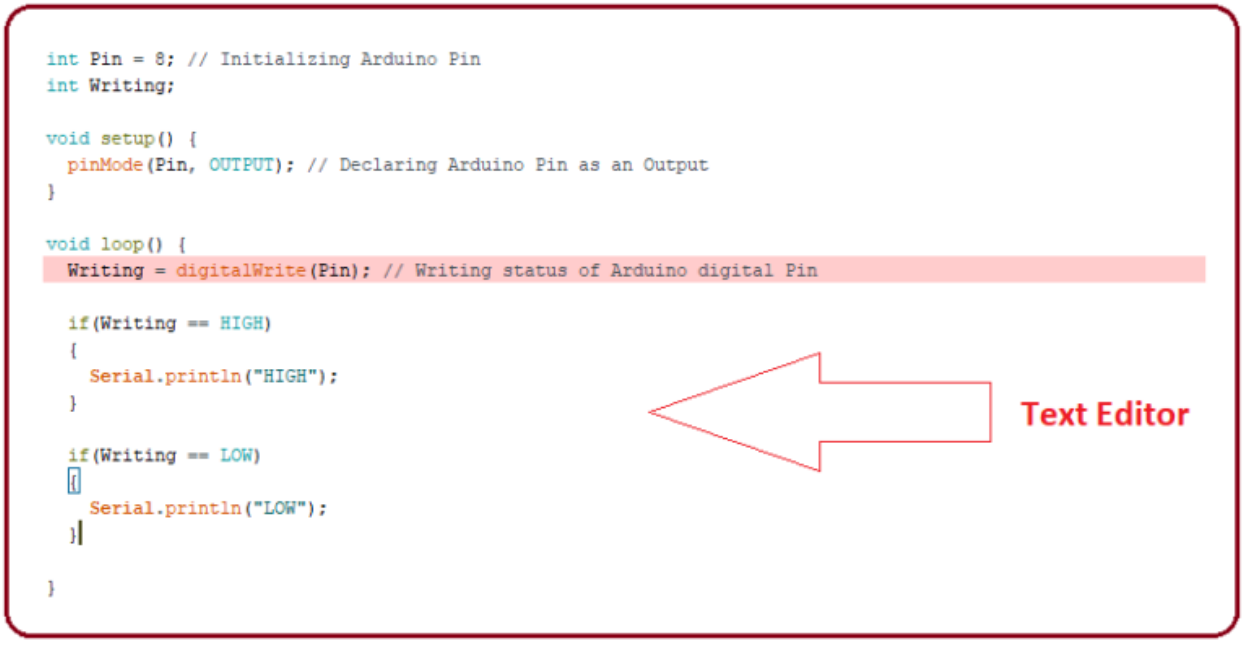
The **Six Buttons** appearing under the Menu tab are connected with the running program as follow.



- The check mark appearing in the circular button is used to verify the code. Click this once you have written your code.
- The arrow key will upload and transfer the required code to the Arduino board.
- The dotted paper is used for creating a new file.
- The upward arrow is reserved for opening an existing Arduino project.
- The downward arrow is used to save the current running code.
- The button appearing on the top right corner is a **Serial Monitor** – A separate pop-up window that acts as an independent terminal and plays a vital role for sending and receiving the Serial Data. You can also go to the Tools panel and select Serial Monitor, or pressing Ctrl+Shift+M all at once will open it instantly. The Serial Monitor will actually help to debug the written Sketches where you can get a hold of how your program is operating. Your Arduino Module should be connected to your computer by USB cable in order to activate the Serial Monitor.
- You need to select the baud rate of the Arduino Board you are using right now. For my Arduino Uno Baud Rate is 9600, as you write the following code and click the Serial Monitor, the output will show as the image below.

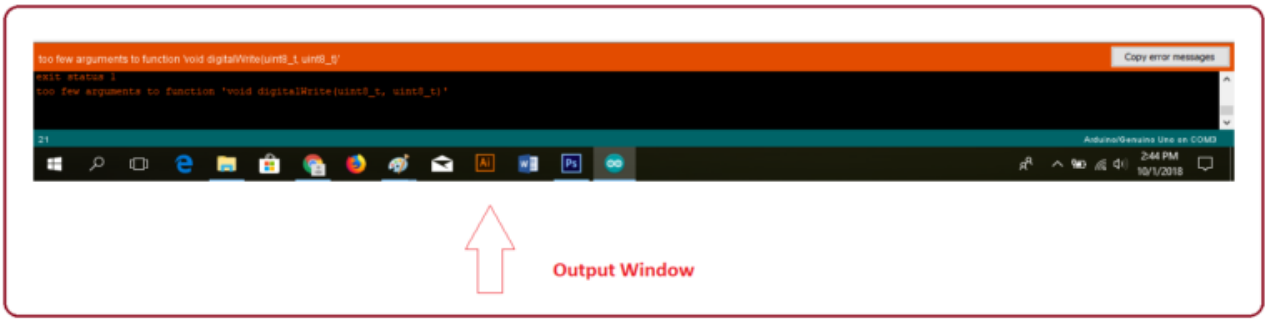


The main screen below the Menu bard is known as a simple text editor used for writing the required code.



The bottom of the main screen is described as an Output Pane that mainly highlights the compilation status of the running code: the memory used by the code, and errors occurred in the

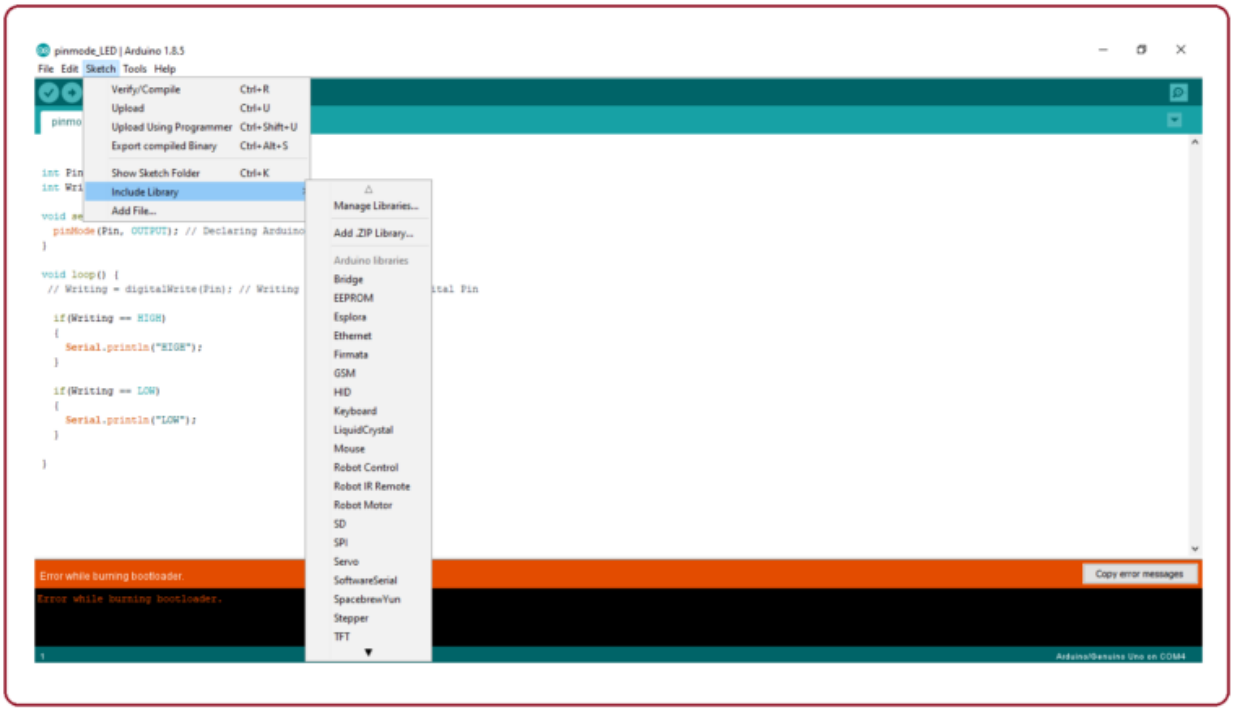
program. You need to fix those errors before you intend to upload the hex file into your Arduino Module.



More or less, Arduino C language works similar to the regular C language used for any embedded system microcontroller, however, there are some dedicated libraries used for calling and executing specific functions on the board.

4.1.4 Libraries

Libraries are very useful for adding the extra functionality into the Arduino Module. There is a list of libraries you can add by clicking the Sketch button in the menu bar and going to Include Library.



As you click the Include Library and Add the respective library it will on the top of the sketch with a #include sign. Suppose, I Include the EEPROM library, it will appear on the text editor as

#include <EEPROM.h>.

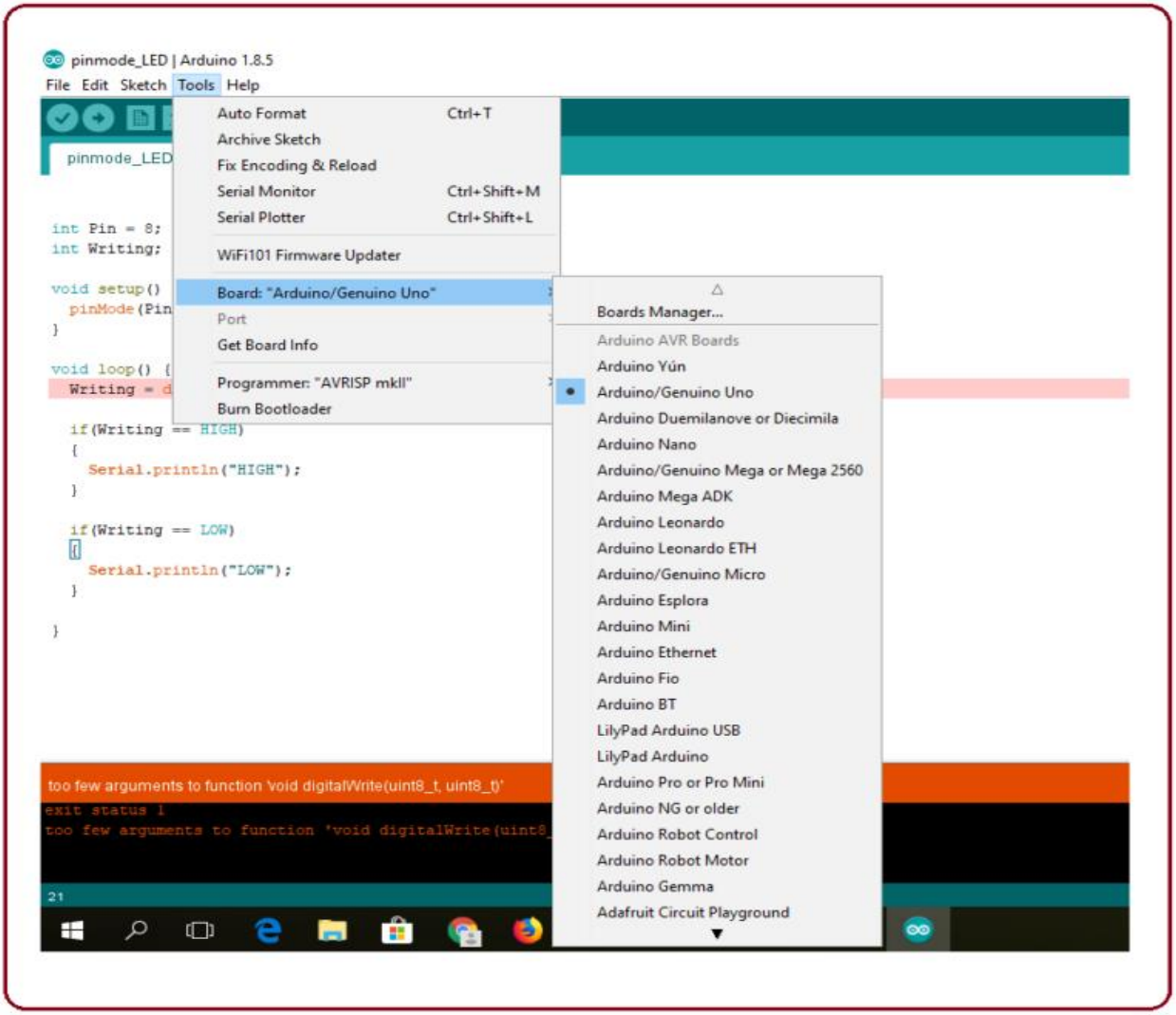
Most of the libraries are preinstalled and come with the Arduino software. However, you can also download them from the external sources.

4.1.5 Making pins INPUT or OUTPUT

The digitalRead and digitalWrite commands are used for addressing and making the Arduino pins as an input and output respectively.
These commands are text sensitive i.e. you need to write them down the exact way they are given like digitalWrite starting with small “d” and write with capital “W”. Writing it down with Digitalwrite or digitalwrite won’t be calling or addressing any function.

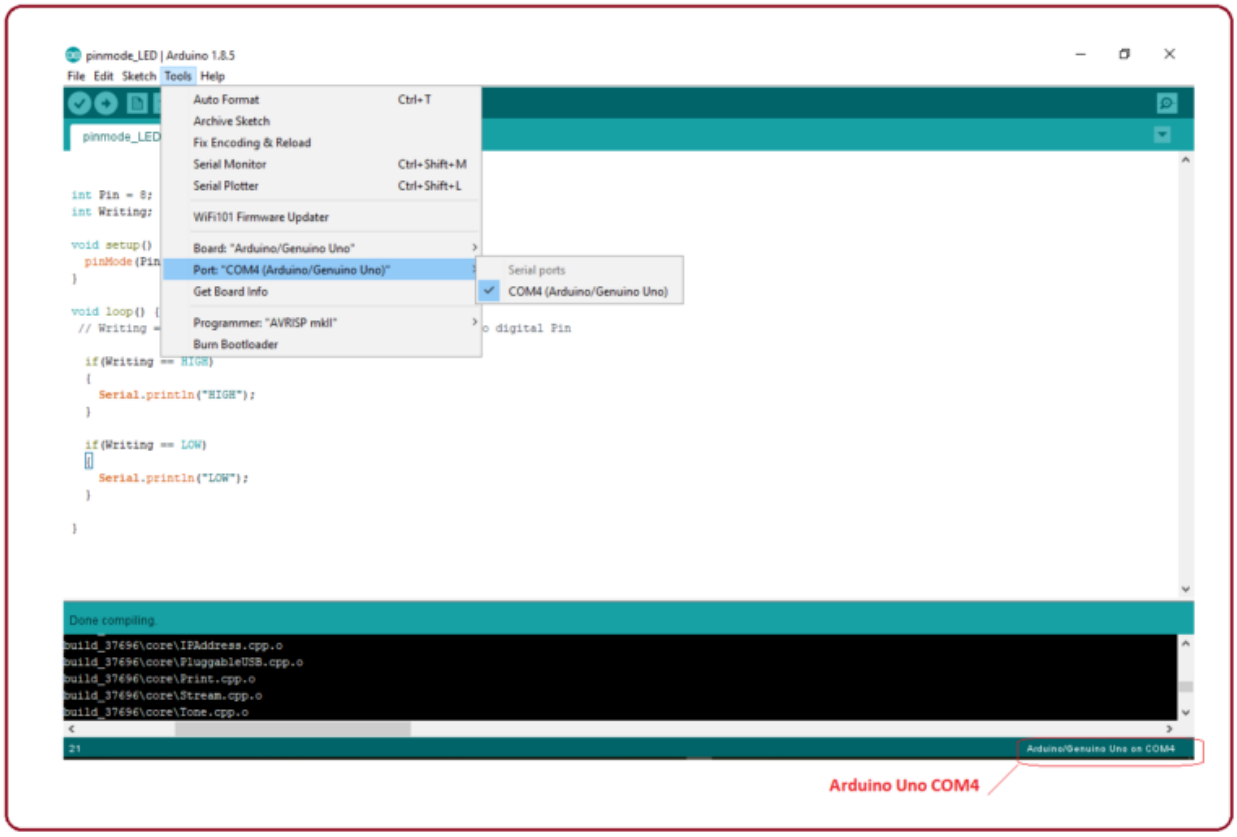
4.1.6 How to select the Board?

In order to upload the sketch, you need to select the relevant board you are using and the ports for that operating system. As you click the Tools on the Menu, it will open like the figure below.



- Just go to the “Board” section and select the board you aim to work on. Similarly, COM1, COM2, COM4, COM5, COM7 or higher are reserved for the serial and USB board. You can look for the USB serial device in the ports section of the Windows Device Manager.

Following figure shows the COM4 that I have used for my project, indicating the Arduino Uno with COM4 port at the right bottom corner of the screen.



- After correct selection of both Board and Serial Port, click the verify and then upload button appearing in the upper left corner of the six button section or you can go to the Sketch section and press verify/compile and then upload.
- The sketch is written in the text editor and is then saved with the file extension .ino.

It is important to note that the recent Arduino Modules will reset automatically as you compile and press the upload button the IDE software, however, older version may require the physical reset on the board.

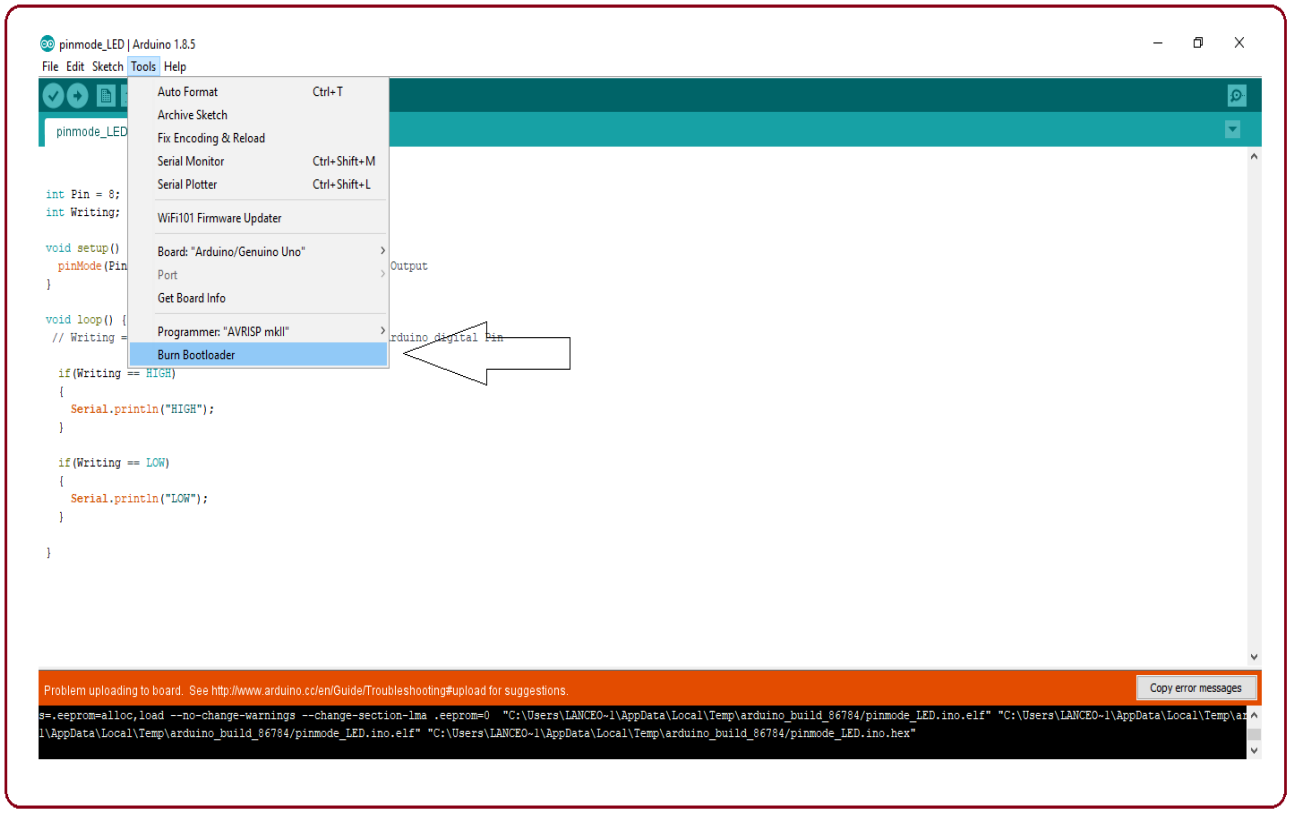
- Once you upload the code, TX and RX LEDs will blink on the board, indicating the desired program is running successfully.

Note: The port selection criteria mentioned above is dedicated for Windows operating system only, you can check this Guide if you are using MAC or Linux.

- The amazing thing about this software is that no prior arrangement or bulk of mess is required to install this software, you will be writing your first program within 2 minutes after the installation of the IDE environment.

4.1.7 Bootloader

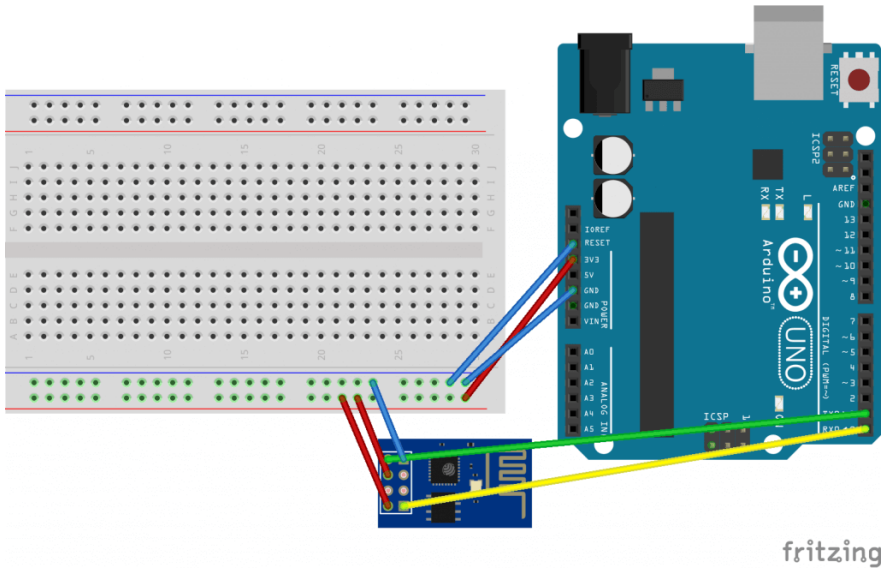
As you go to the Tools section, you will find a bootloader at the end. It is very helpful to burn the code directly into the controller, setting you free from buying the external burner to burn the required code.



When you buy the new Arduino Module, the bootloader is already installed inside the controller. However, if you intend to buy a controller and put in the Arduino module, you need to burn the bootloader again inside the controller by going to the Tools section and selecting the burn bootloader.

4.2 Connecting to Wifi module (AT Commands)

4.2.1 Connecting ESP8266 Wifi module to Arduino



4.2.2 Using the Arduino IDE

In the Arduino IDE, you don't need to choose a board, as we're not uploading anything to the ESP8266. Just choose the right port in the **Tools** menu and go to **Tools → Serial Monitor**. Then simply set your baud rate to **115200** (the default ESP8266 firmware uses it) and your line endings to **Both NL & CR**.

If you type `AT` in the message field and press enter, it should respond with `OK`.

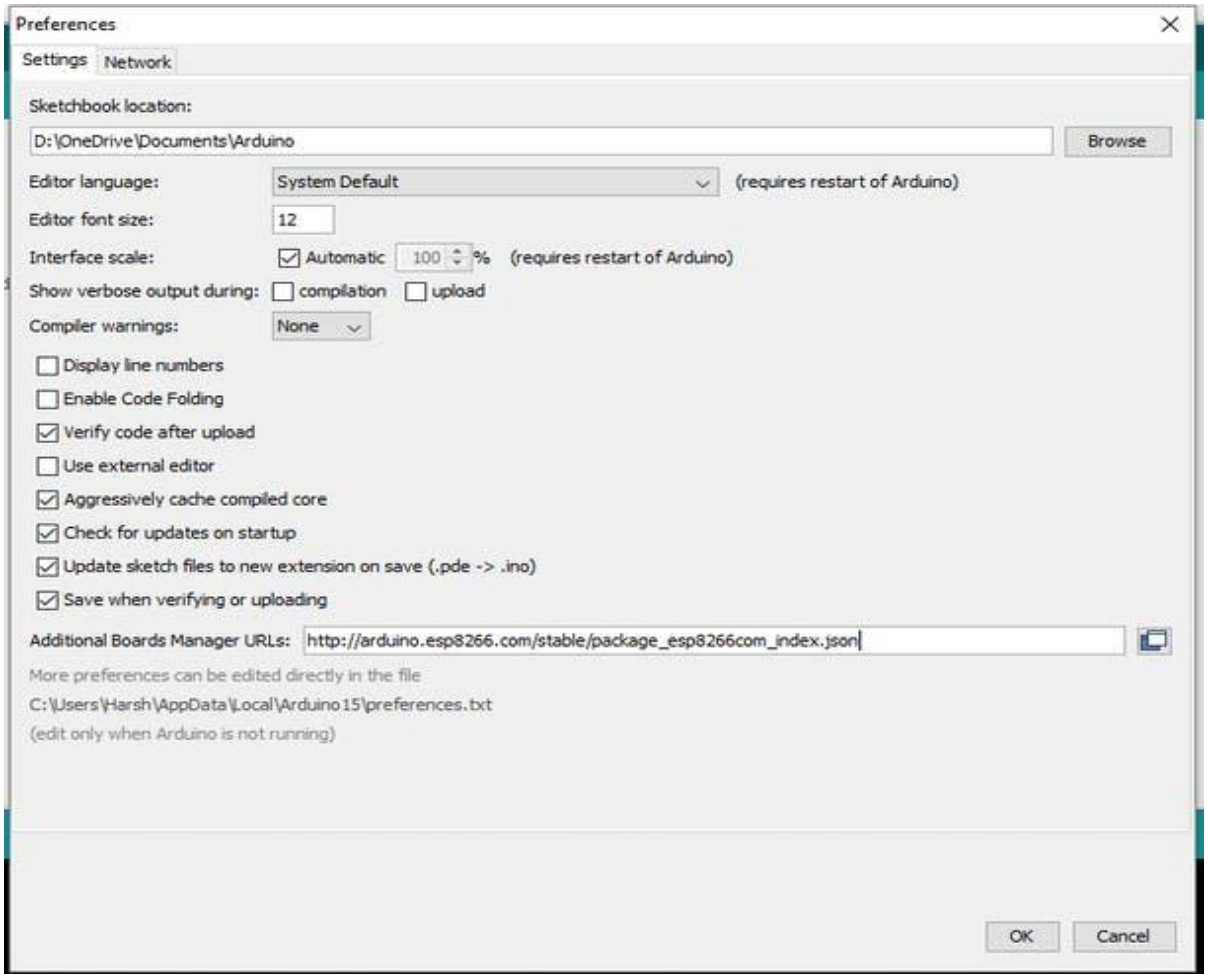
And since we talked about the CH_PD pin, remember that if you want to flash the ESP8266 you should connect the GPIO0 pin to GND (blue line), which puts the ESP into flash mode.

4.2.3 How to program ESP8266 with Arduino Uno?

Step 1: Installing Board to Arduino IDE

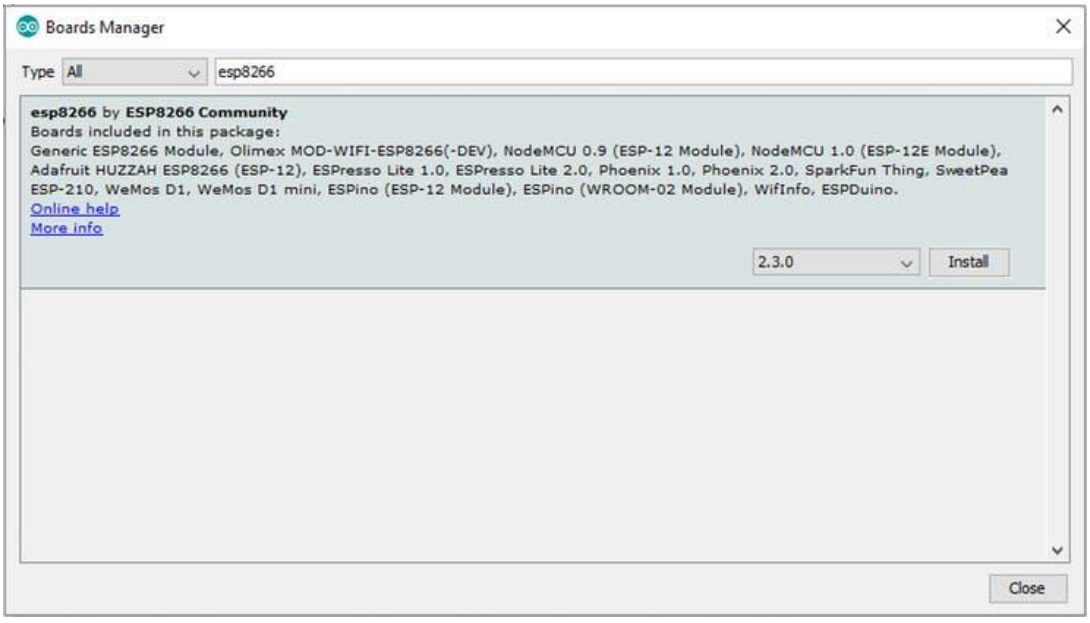
First, install ESP8266 to Arduino IDE. If you have already installed the board to boards manager of Arduino IDE, skip this step else follow the steps

- Start the Arduino IDE
- Go to **File > Preferences**
- Add the below given link to **Additional Boards Manager URLs** http://arduino.esp8266.com/stable/package_esp8266com_index.json

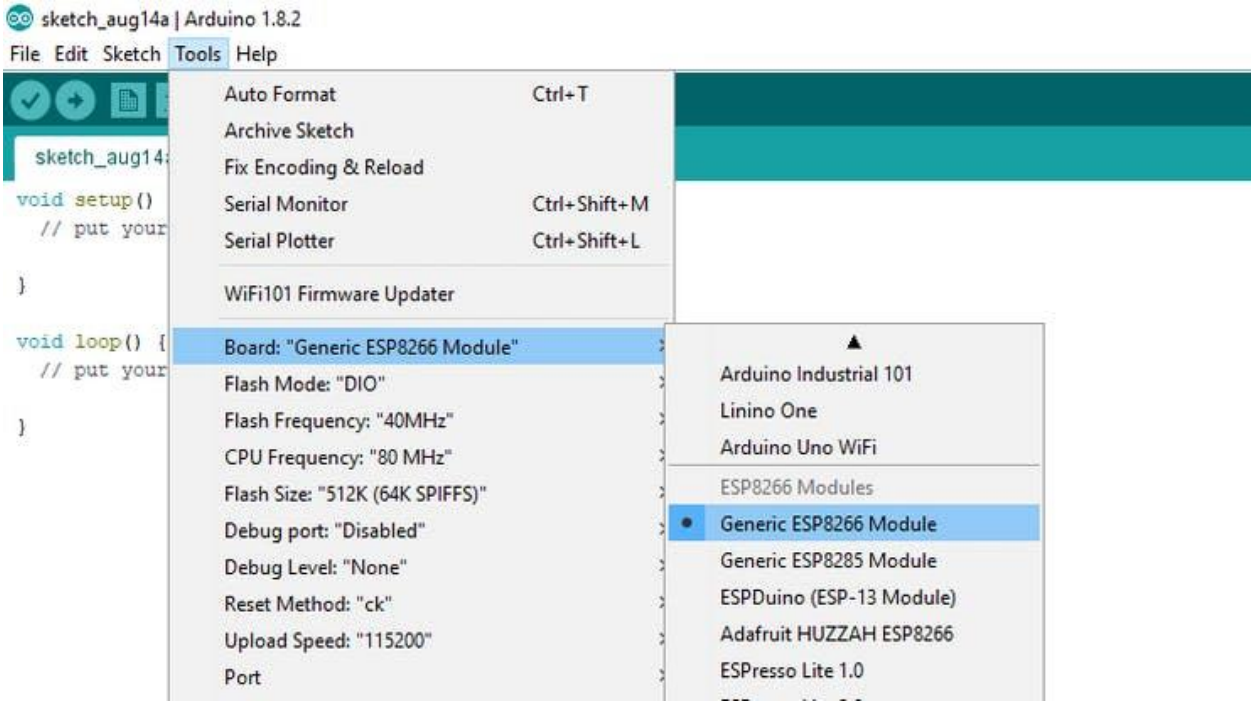


Arduino IDE Preferences

- Go to **Tools > Boards > Boards Manager...**
- Search **ESP8266**



- Click **Install** button to install the ESP8266 Board
- Now close the Boards Manager window and select the **Generic ESP8266 Module** from board selection list



- Installation of ESP8266 in Arduino IDE is done.

Step 2: Program ESP8266 using Arduino

Make the circuit as per the above given diagram. Power up the Arduino UNO board and wait till the Arduino Board boots up successful. (It will take around 5 seconds) Connect the Arduino Reset pin to Ground. Reset pin is grounded to bypass the Arduino. It will disable Arduino Board and upload code directly to the ESP8266. Sample program for Blink LED is as below

After the Arduino IDE shows done uploading of Blink LED program, connect the LED to GPIO_2 Pin of ESP8266. Please do not connect the LED before or at the time of uploading program, it can cause some issue in uploading program.

Note: If the L light is on at the time of before uploading program, disconnect the power supply for one minute, else it will show error in uploading program to ESP8266.

4.2.4 Code for Blink LED in ESP8266 ESP-01

```
#include<ESP8266WiFi.h>

void
setup()
{
          // initialize LED_BUILTIN as an output pin.
          pinMode(LED_BUILTIN, OUTPUT);
        }


        void loop() {
          digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on
          delay(1000);
          digitalWrite(LED_BUILTIN, LOW);    // turn the LED off
          delay(1000);
        }
```

4.2.5 AT commands

Function	**AT Commands**	**Response**
Working	AT	OK
Restart	AT+RST	OK ..... Ready
Firmware Version	AT+GMR	<AT version info> information about AT version <SDK version info> information about SDK version <compile time> time of the bin was compiled OK
List Access Point	AT+CWLAP	+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<ch>,<freq offset> OK

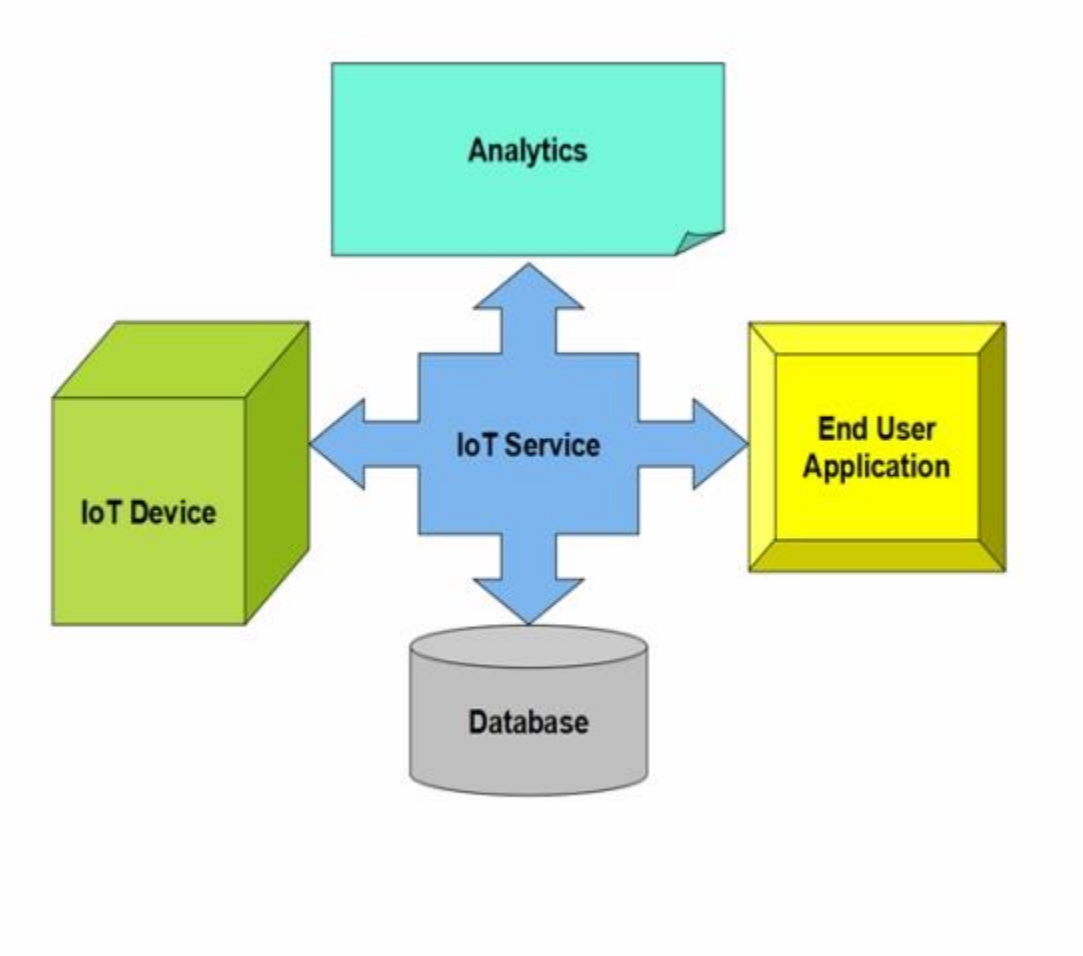
Query Joined Access Point	AT+CWJAP?	+CWJAP:<ssid>,<bssid>,<channel>,<rssi> OK
Join Access Point	AT+CWJAP=”SSID”,”Password”	WIFI CONNECTED WIFI GOT IP OK
Quit Access Point	AT+CWQAP	OK WIFI DISCONNECTED
Get IP Address	AT+CIFSR (Assuming AT+CWMODE=3)	+CIFSR:APIP,<IP address> +CIFSR:APMAC,<mac address> +CIFSR:STAIP,<IP address> +CIFSR:STAMAC,<mac address> OK
Query WiFi Mode	AT+CWMODE?	+CWMODE:<mode>
Set WiFi Mode	AT+CWMODE=<mode> Mode: - 1 = STA (station) 2 = AP (Access Point) 3 = BOTH i.e. STA & AP	OK
Query TCP/UDP Connection	AT+CIPMUX?	+CIPMUX:<mode>
Set TCP/UDP Connection	AT+CIPMUX=<mode> Mode: - 0 = Single Connection 1 = Multiple Connection	OK
TCP/IP Connection status	AT+CIPSTATUS	STATUS:<status> Possible statuses are

		2: Got IP 3: Connected 4: Disconnected
Query TCP transmission mode	AT+CIPMODE?	+CIPMODE:<mode>
Set TCP transmission mode	AT+CIPMODE=<mode> Mode: - 0 = Normal mode 1 = Transparent mode	OK
Set up TCP/UDP connection	(CIPMUX=0) AT+CIPSTART =<type>,<addr>,<port> (CIPMUX=1) AT+CIPSTART=<id>,<type>,<addr>, <port> Example (CIPMUX=0): AT+CIPSTART="TCP","192.168.101.110",80	CONNECT OK
Send Data	(CIPMUX=0) AT+CIPSEND=<data length> (CIPMUX=1) AT+CIPSEND=<id>,<data length>	OK > (Note: write your data after > and enter it to send it will return status like.) Recv <data length> bytes SEND OK (after we receive response from server if any for default auto receive mode) (CIPMUX=0): + IPD, <length>: <data> (CIPMUX=1): + IPD, <id>, <length>: <data>
Close TCP/UDP Connection	AT+CIPCLOSE	CLOSED OK

4.3 Uploading Data to a Server(Thingspeak)

4.3.1 Introduction

The Internet of Things(IoT) is a system of ‘connected things’. The things generally comprise of an embedded operating system and an ability to communicate with the internet or with the neighboring things. One of the key elements of a generic IoT system that bridges the various ‘things’ is an IoT service. An interesting implication from the ‘things’ comprising the IoT systems is that the things by themselves cannot do anything. At a bare minimum, they should have an ability to connect to other ‘things’. But the real power of IoT is harnessed when the things connect to a ‘service’ either directly or via other ‘things’. In such systems, the service plays the role of an invisible manager by providing capabilities ranging from simple data collection and monitoring to complex data analytics. The below diagram illustrates where an IoT service fits in an IoT ecosystem:



One such IoT application platform that offers a wide variety of analysis, monitoring and counter-action capabilities is ‘ThingSpeak’. Let us consider ThingSpeak in detail.

4.3.2 What is ThingSpeak?

ThingSpeak is a platform providing various services exclusively targeted for building IoT applications. It offers the capabilities of real-time data collection, visualizing the collected data in the form of charts, ability to create plugins and apps for collaborating with web

services, social network and other APIs. We will consider each of these features in detail below.

The core element of ThingSpeak is a ‘ThingSpeak Channel’. A channel stores the data that we send to ThingSpeak and comprises of the below elements:

- 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
- 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device.
- 1 status field - A short message to describe the data stored in the channel.

To use ThingSpeak, we need to signup and create a channel. Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same. Let us start exploring ThingSpeak by signing up and setting up a channel.

4.3.3 Uploading of ESP8266 sensor data using Internet

It is three step process.

1. Connect to your WiFi hot spot having internet access.
2. Read Sensor data
3. Upload data to ThingSpeak

Step 1: Sign up ThingSpeak

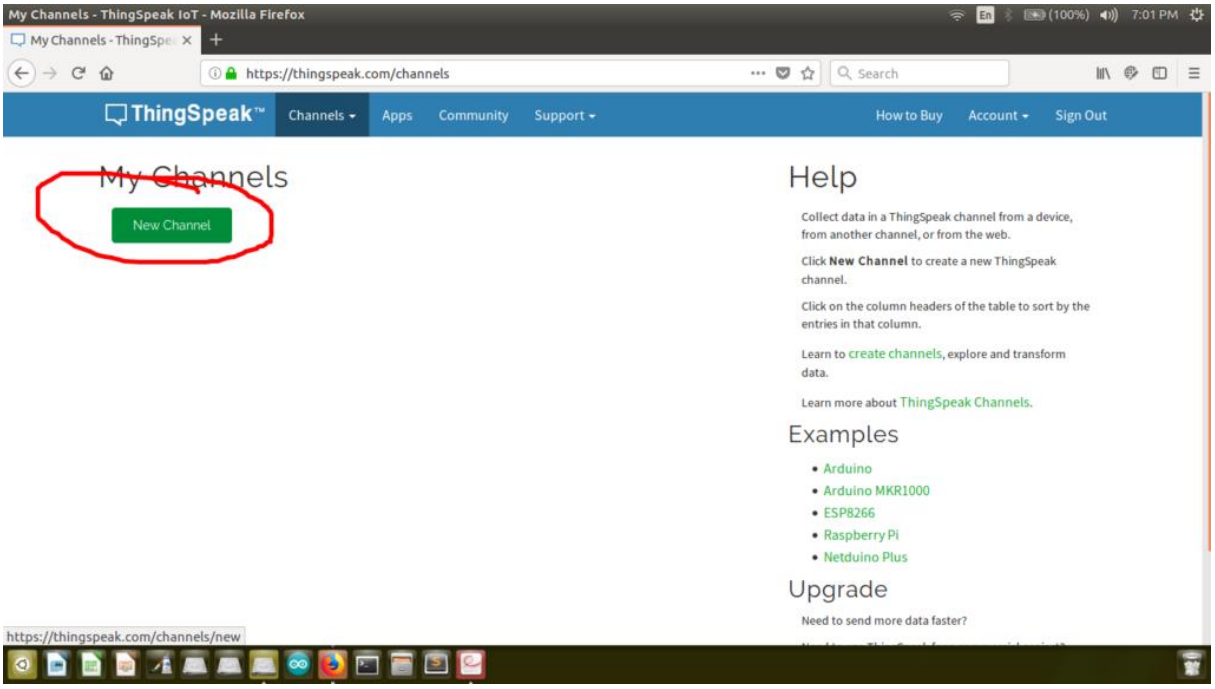
Its simple just enter your email id and verify your account.

Step 2: Configuring ThingSpeak

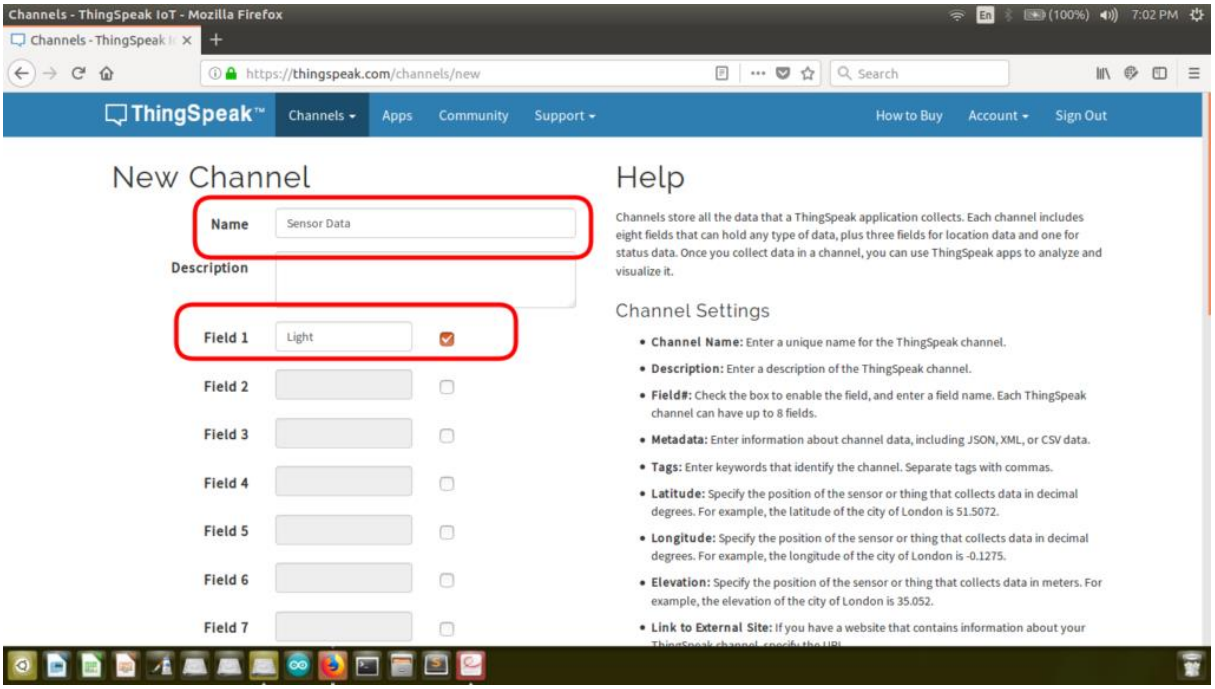
Configuration is just few clicks job

Step 2.1: Create New Channel

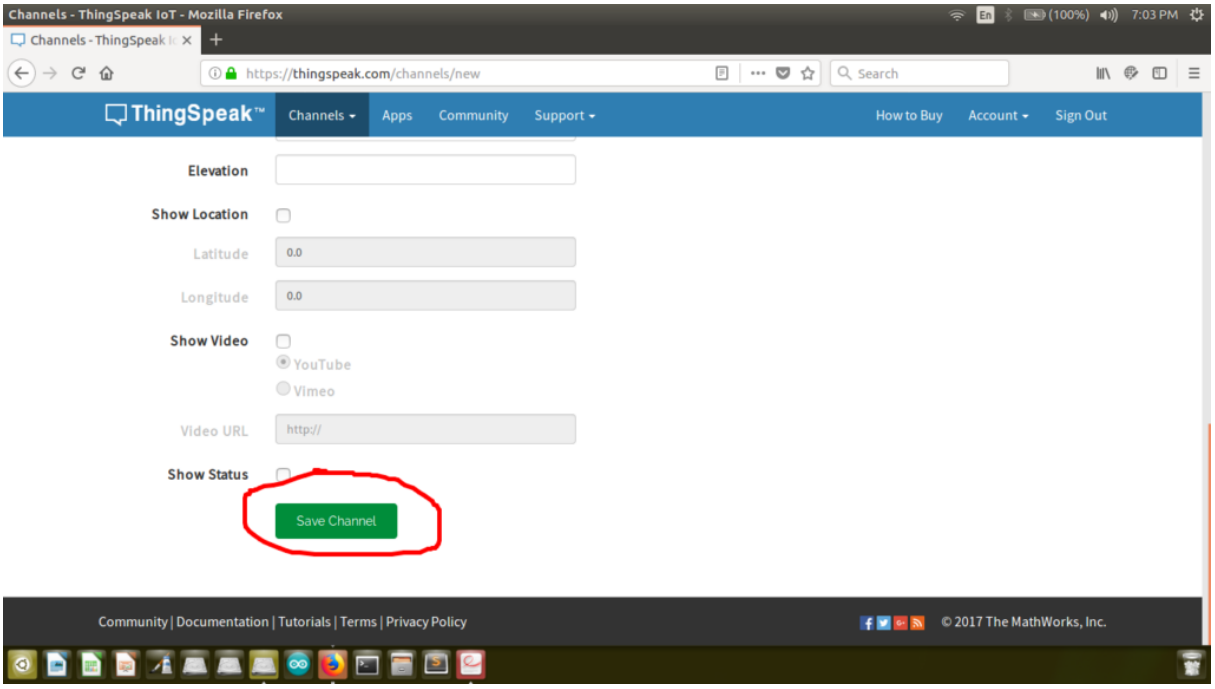
Click on New Channel



Enter Name and Field. You may have multiple Fields depending on number of sensor create multiple fields such as Light, Temperature, Humidity, etc.

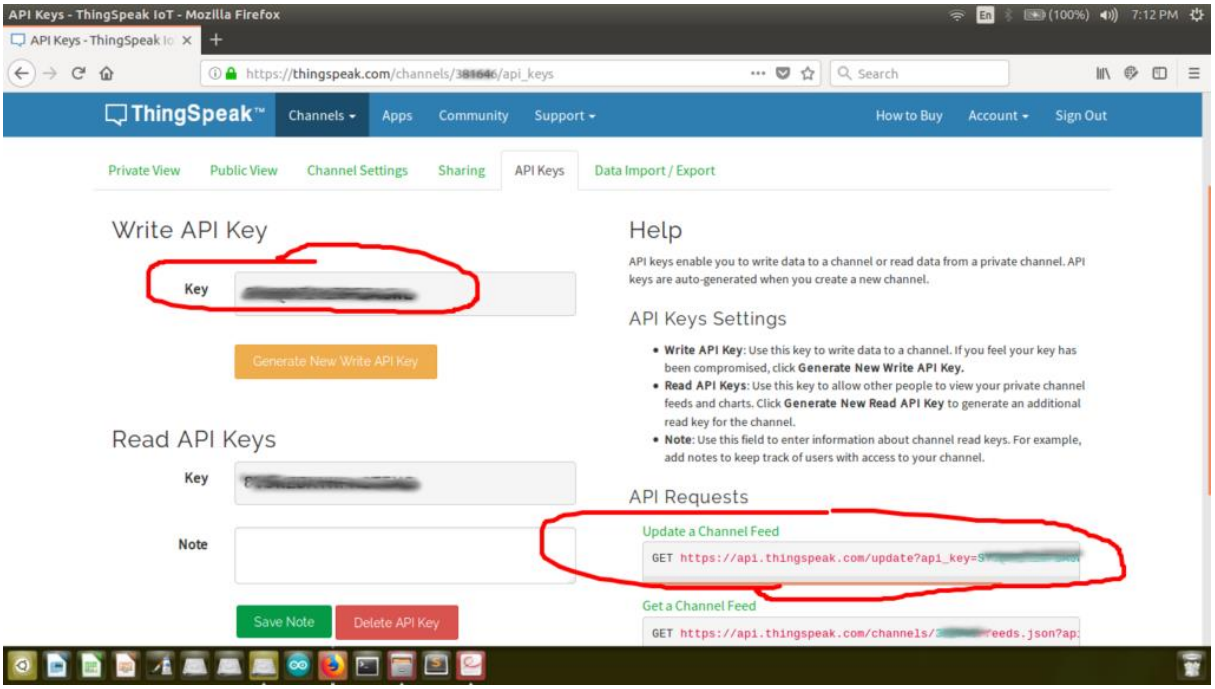


Keep everything else as it is. Blank or default values. and click on Save Channel.



Step 2.2: Getting API Key

Click on API Key Tab and look for these two fields Write Api Key and Update channel feed line.



This line is important for data upload to cloud server

GET https://api.thingspeak.com/update?api_key=akjshfajkhfowei&field1=0

First parameter is Api Key. Do not share API key it makes decision of which user it is. In most cases I create own cloud server as per requirements. Second parameter is field1=0 here you can pass the ADC value ex. field1=1234

value in front of field1=0 is your sensor data for example if your adc value is 1234 then you call this GET request
with **https://api.thingspeak.com/update?api_key=yourapikey&field1=1234**

For multiple sensors you need to add multiple fields as shown in below example

https://api.thingspeak.com/update?api_key=yourapikey&field1=1234&field2=442

Step 3: Programming ESP8266 to upload data to ThingSpeak cloud server

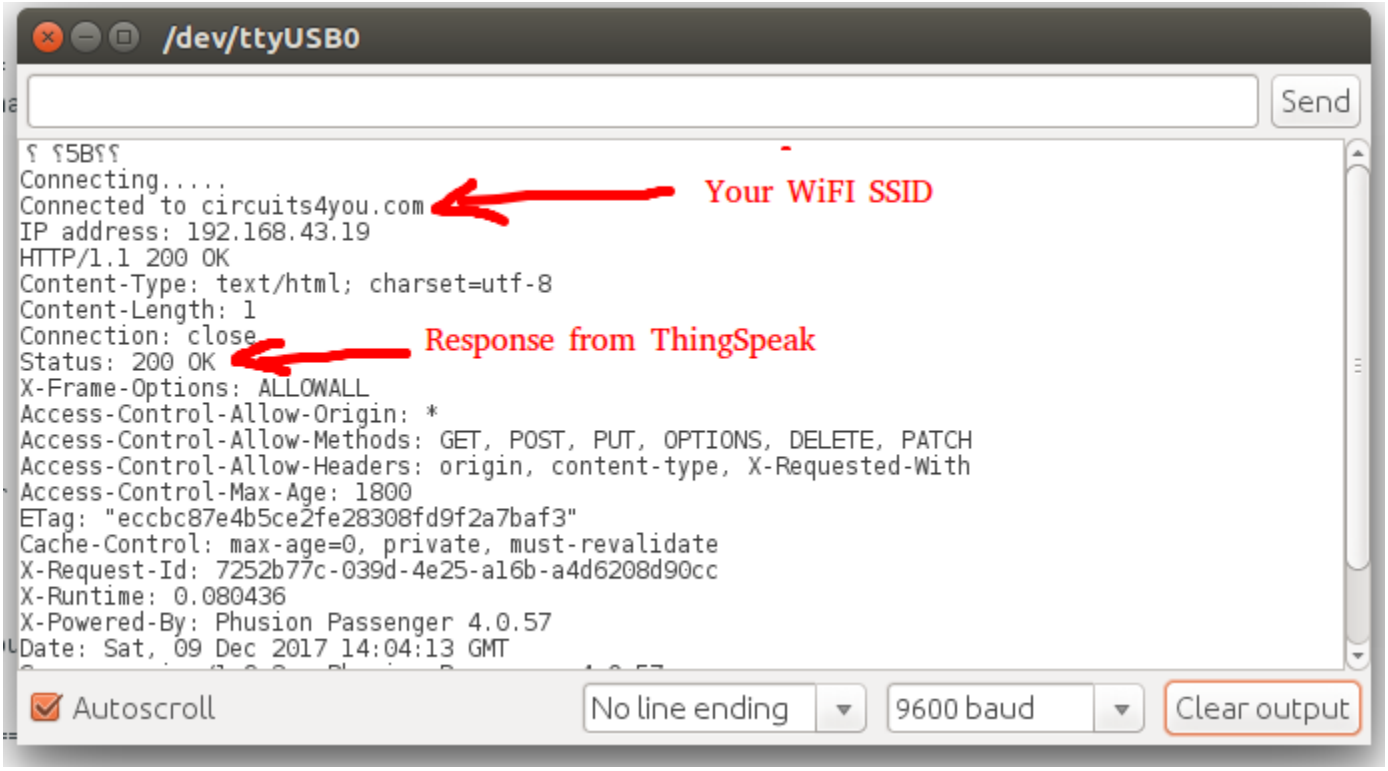
Make Changes in program for API KEY, SSID and PASSWORD

```
/* Connects to WiFi HotSpot. */

#include <ESP8266WiFi.h>

#include <WiFiClient.h>
#include <ESP8266WebServer.h>

/* Set these to your desired credentials. */
const char *ssid = "yournetworkSSID";  //ENTER YOUR WIFI SETTINGS <<<<<<<<<<
const char *password = "yourPassword";

//Web address to read from
const char *host = "api.thingspeak.com";
String apiKey = "Your API KEY";  //ENTER YOUR API KEY <<<<<<<<<<<
//=============================================================================
//                  Power on setup
//=============================================================================

void setup() {
  delay(1000);
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);        //This line hides the viewing of ESP as wifi hotspot
  //WiFi.mode(WIFI_AP_STA);   //Both hotspot and client are enabled
  //WiFi.mode(WIFI_AP);       //Only Access point

  WiFi.begin(ssid, password);     //Connect to your WiFi router
  Serial.println("");

  Serial.print("Connecting");
```

```
  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  //If connection successful show IP address in serial monitor
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  //IP address assigned to your ESP
}

//==============================================================================
//                  Main Program Loop
//==============================================================================
void loop() {
  WiFiClient client;
  const int httpPort = 80; //Port 80 is commonly used for www
  //----------------------------------------------------------------------
  //Connect to host, host(web site) is define at top
  if(!client.connect(host, httpPort)){
    Serial.println("Connection Failed");
    delay(300);
    return; //Keep retrying until we get connected
  }

//----------------------------------------------------------------------
  //Make GET request as pet HTTP GET Protocol format
  String ADCData;
  int adcvalue=analogRead(A0);  //Read Analog value of LDR
  ADCData = String(adcvalue);   //String to interger conversion
  String Link="GET /update?api_key="+apiKey+"&field1=";  //Requeste webpage
  Link = Link + ADCData;
  Link = Link + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection: close\r\n\r\n";
  client.print(Link);
  delay(100);

//----------------------------------------------------------------------
  //Wait for server to respond with timeout of 5 Seconds
  int timeout=0;
  while((!client.available()) && (timeout < 1000))     //Wait 5 seconds for data
  {
    delay(10);  //Use this with time out
    timeout++;
  }

//----------------------------------------------------------------------
  //If data is available before time out read it.
  if(timeout < 500)
  {
     while(client.available()){
        Serial.println(client.readString()); //Response from ThingSpeak
     }
  }
```

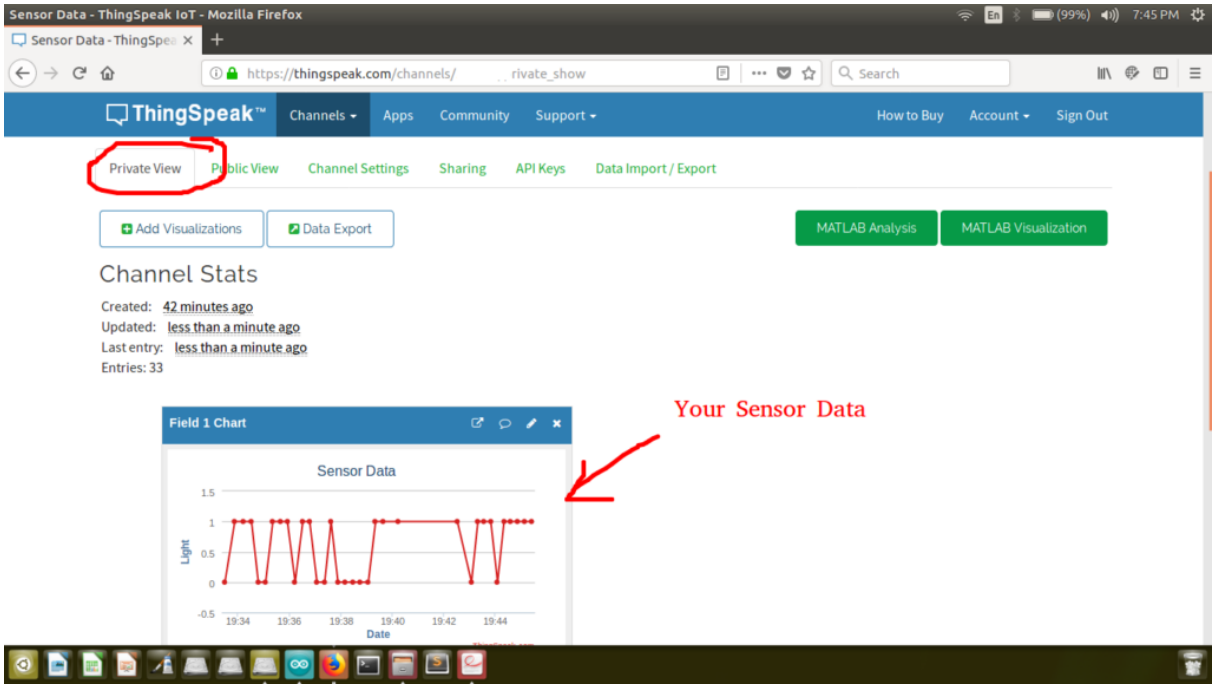
```
 else
 {
    Serial.println("Request timeout..");
 }

 delay(5000);  //Read Web Page every 5 seconds
}
//=
```

Upload Program and Open Serial monitor with baud rate of 115200. First it will show connection status to your wifi router then It uploads data to server and displays response from server. Do not try to upload data at very high rate ThingSpeak server will simply blocks it. 5 Seconds update time is okay.



Step 4: Check Data on ThingSpeak Server

Open Your ThingSpeak Account and Click on Private View of Your Channel



You can Create multiple Fields and Upload multiple sensor data and also you can create multiple channels for multiple Data Nodes.

CHAPTER 5

SOURCE CODE

```
//IOT DATA ACQUISITION SYSTEM USING ARDUINO//
#include <LiquidCrystal.h> //include liquid crystal library
LiquidCrystal lcd(12,11,5,4,3,2); //select rs,en,data pins of lcd
const char* ssid = "IOTDA"; //enter ssid of the hotspot network
const char* password = "IOTDA1234"; //enter password of the hotspot
char res[130];
const int relay1=6; //set pin numbers for all objects
const int relay2=7;
const int relay3=8;
const int relay4=13;
const int fire=19;
const int light=18;
const int buzz=A3;
int val,lightt;
int tempe;
float ttt;
int level;
int gas=0,flame=0;


void serialFlush() //check if any serial data is available
{
  while(Serial.available() > 0)
  {
  char t = Serial.read();
  }
}
char check(char* ex,int timeout)
{
  int i=0;
  int j = 0,k=0;
  while (1)
  {
    sl:
    if(Serial.available() > 0)
    {
      res[i] = Serial.read();
      if(res[i] == 0x0a || res[i]=='>' || i == 100)
      {
        i++;
        res[i] = 0;break;
      }
      i++;
    }
    j++;
    if(j == 30000)
    {
```

```
      k++;
      j = 0;
    }
    if(k > timeout)
    {
      return 1;
    }
  }
  if(!strncmp(ex,res,strlen(ex)))
  {
    return 0;
  }
  else
  {
    i=0;
    goto sl;
   }
}

char buff[200];
void upload1(); //call upload1 function

//program setup//
void setup() //setup the pin modes
{
    int i=0;
  char ret;
    pinMode(10, OUTPUT); //trigger of Level sensor
    pinMode(9, INPUT); //echo of Level sensor
    pinMode(light,INPUT); //LDR
    pinMode(fire,INPUT); //Infrared sensor
    pinMode(buz,OUTPUT); //buzzer
    pinMode(relay1,OUTPUT); //ir relay
    pinMode(relay2,OUTPUT); //light relay
    pinMode(relay3,OUTPUT); //gas relay level relay
    pinMode(relay4,OUTPUT); //temp relay
    digitalWrite(relay1,LOW);
    digitalWrite(relay2,HIGH);
    digitalWrite(relay3,LOW);
    digitalWrite(relay4,LOW);
    Serial.begin(9600);
    //start serial communciation at a baud rate of 9600 bits/sec
  lcd.begin(16,2); //initiate 16*2 lcd
  lcd.setCursor(4,0);
  lcd.print("IOT   DATA");
  lcd.setCursor(3,1);
  lcd.print("ACQUISITION"); //displays IOT DATA ACQUISITON
  delay(3000);
  lcd.clear(); //clear lcd screen
  serialFlush();
```

```
  st:
  Serial.println("ATE0");//echo off
  ret  = check((char*)"OK",50);
  Serial.println("AT"); //check wifi module working status
  ret  = check((char*)"OK",50);
  if(ret != 0)
  {
    delay(100);

    delay(100);
   goto st;
  }

  delay(1000);
  Serial.println("AT+CWMODE = 1"); //set wifi mode to station
  ret  = check((char*)"OK",50);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("CONNECTING");

  cagain:
  delay(1000);
  serialFlush();
  Serial.print("AT+CWJAP=\""); //join access point
  Serial.print(ssid);
  Serial.print("\",\"");
  Serial.print(password);
  Serial.println("\"");
  if(check((char*)"OK",300))
  {
    lcd.clear();lcd.setCursor(0, 0);lcd.print("CONNECTING");
    goto cagain;
  }

  delay(1000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("CONNECTED");
  Serial.println("AT+CIPMUX=1"); //set tcp/udp connection to multiple
connection
  delay(1000);
  delay(1000);
  }
  unsigned long int duration = 0;
  int upload=0,count=0;

//main operation in repeat mode//
void loop()
{
  digitalWrite(10, LOW);//level sensor
  delayMicroseconds(2);
```

```
  digitalWrite(10, HIGH);
  delayMicroseconds(10);
  digitalWrite(10, LOW);
  ttt=pulseIn(9, HIGH);
  level=ttt*0.01732;//level value
  gas = analogRead(3);//gas sensor
  float TT=( gas/1024.0)*5000;gas=TT;//gas value
  delay(50);
  val = analogRead(2);float T=( val/1024.0)*5000; tempe= T/10;
 //temparature     value

    lcd.setCursor(0,0);
    lcd.print("TEM:");lcd.print(tempe);lcd.print("");
    lcd.setCursor(8,0);
    lcd.print("LEV:");lcd.print(level);lcd.print("");
    delay(3000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("GAS:");lcd.print(gas);lcd.print("");
    delay(3000);

upload1();//call upload function

//condition if fire detected
if(digitalRead(fire) == 0)
{
  lcd.setCursor(0, 1);lcd.print("Fire detected");
  digitalWrite(relay1,HIGH);
  tone(buzz,1000);
  delay(1000);
  noTone(buzz)
  flame = 1;
  upload = 1;
}
else
{
  flame = 0;
  digitalWrite(relay1,LOW);
  upload = 1;
}
//condition if no light detected
if(digitalRead(light) == 1)
{
  lcd.setCursor(0, 1);lcd.print("No Light!");
  digitalWrite(relay2,HIGH);
  delay(300);
  lightt = 0;
  upload = 1;
}
else
{
```

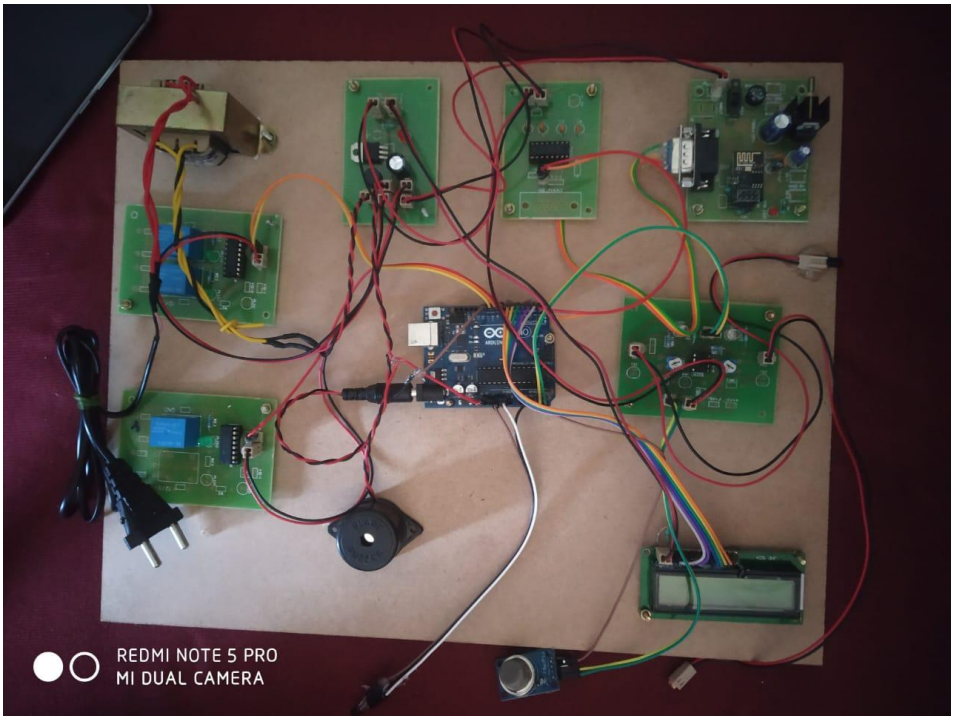
```
  lcd.setCursor(0,1);lcd.print("Light Detected");
  lightt = 1;
  digitalWrite(relay2,LOW);
  upload = 1;
}

//condition if temperature is high
if(tempe >=35)
{
  digitalWrite(relay4,HIGH);
  lcd.setCursor(0, 1);lcd.print("HIGH TEMPERATURE");
  tone(buzz,1000);
  delay(1000);
  noTone(buzz)
  upload = 1;
  delay(30);
}
else
{
  digitalWrite(relay4,LOW);
  upload = 1;
}

//condition if gas is detected
if(gas > 1500)
{
  digitalWrite(relay3,HIGH);
  lcd.setCursor(0, 1);lcd.print("GAS DETECTED  ");
  tone(buzz,1000);
  delay(1000);
  noTone(buzz)
  upload = 1;
}
if(gas < 1500)
{
  digitalWrite(relay3,LOW);
  lcd.setCursor(0, 1);lcd.print("                ");
  upload = 1;
}

//condition if level is high
if(level<19)
{
  digitalWrite(relay3,HIGH);
  lcd.setCursor(0,1);lcd.print("HIGH LEVEL");
  tone(buzz,1000);
  delay(1000);
  noTone(buzz)
  upload = 1;
}
else
```

```
{
  digitalWrite(relay3,LOW);
  lcd.setCursor(0, 1);lcd.print("                ");
  upload = 1;
}


count++;
if(upload == 1||count == 2)
{
  count =0;
  upload = 0;
  upload1();
}

}


void upload1() //uploading sensor data to server
  {
  delay(200);
  serialFlush();
  Serial.println("AT+CIPSTART=5,\"TCP\",\"184.106.153.149\",80");
  //thingspeak IP address and HTTP port selection
       {
       delay(100);
       serialFlush();
       Serial.println("AT+CIPSEND=5,100");
       //send data to server via wifi module at 100 bytes data length
       if(!check((char*)">",50))
       {
       delay(500);


            serialFlush();
            Serial.print("GET /update?api_key=CKWOP5AYN5Q1MKMZ&");
            //API key of our thingspeak server
            sprintf(buff,"field1=%04u",tempe);
            //graph of temperature on thingspeak website
            Serial.print(buff);
            sprintf(buff,"&field2=%04u",gas);
            //graph of gas on thingspeak website
            Serial.print(buff);
            sprintf(buff,"&field3=%04u",flame);
            //graph of flame on thingspeak website
            Serial.print(buff);
            sprintf(buff,"&field4=%04u",lightt);
            //graph of light on thingspeak website
            Serial.print(buff);
            sprintf(buff,"&field5=%04u",level);
            //graph of level on thingspeak website
```

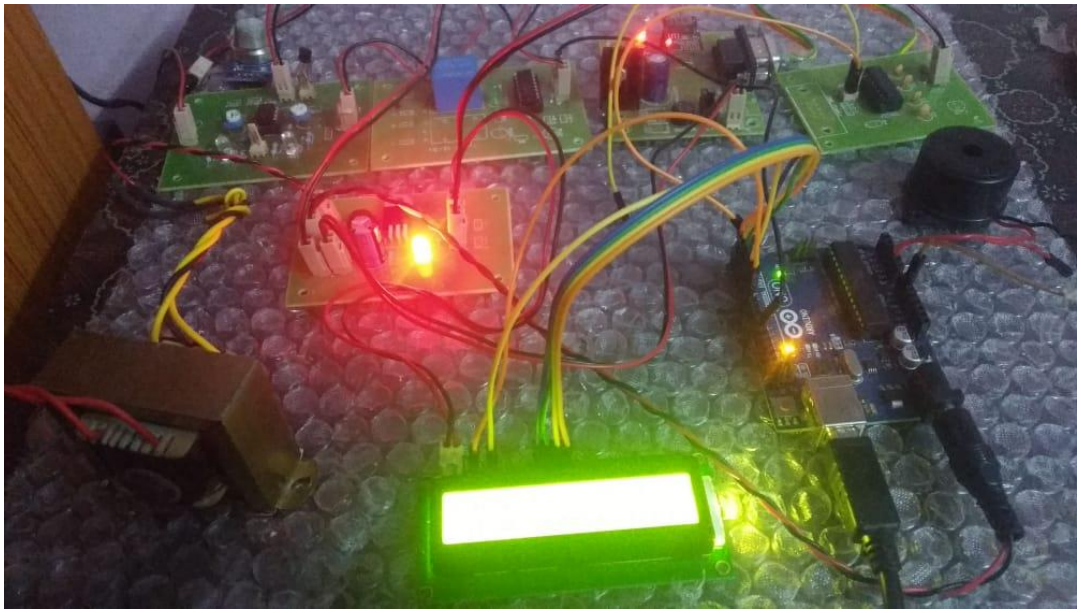
```
          Serial.print(buff);

          Serial.println("");
          if(!check((char*)"OK",200))
          {
              delay(500);

              Serial.println("AT+CIPCLOSE");//close tcp/udp connection
          }

      } //data sending

    } //IP address and port selection

  } //uploading sensor to data
```

CHAPTER 6
RESULT

The connections are made as per the circuit diagram and the program code is uploaded to the Arduino Uno board.

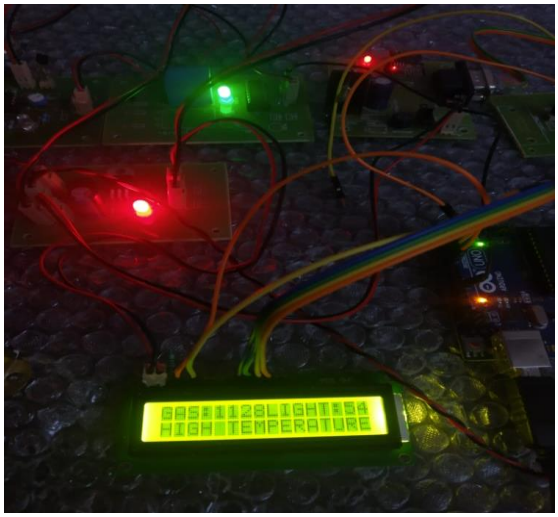


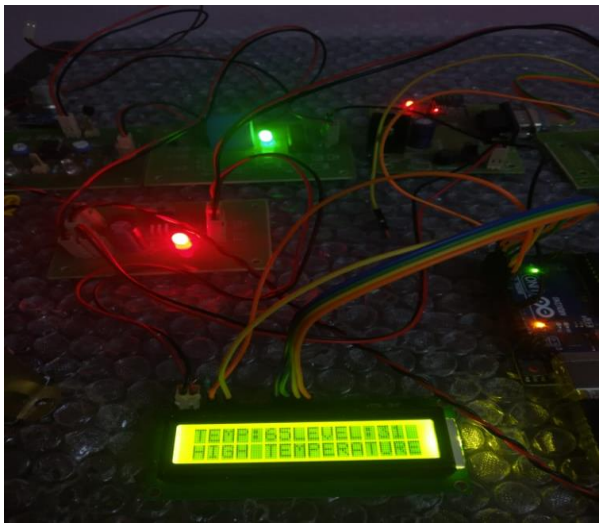
Power supply is given and the LCD shows the parameters based on the given program code.

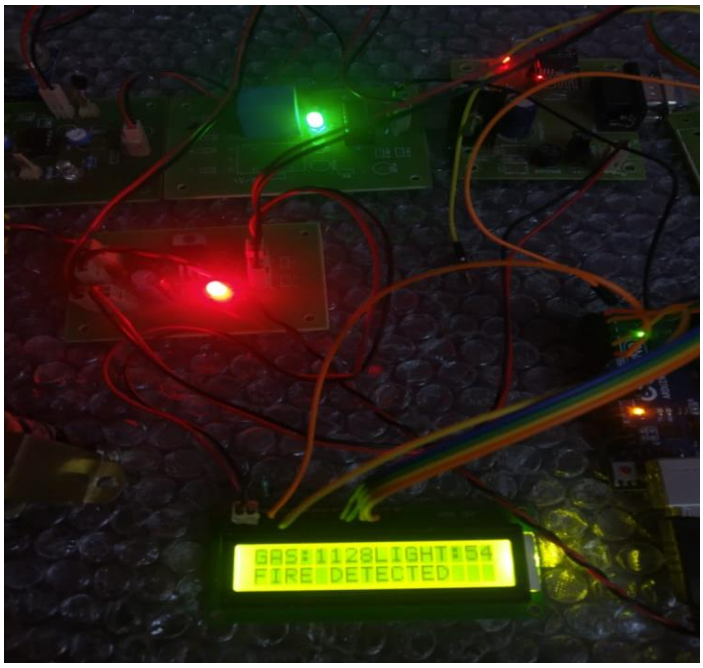


The LCD shows the parameters detected from the sensors i.e *volume of gas in ppm , temperature in Celsius , light in luminous flux per unit area , level in centimetres.*

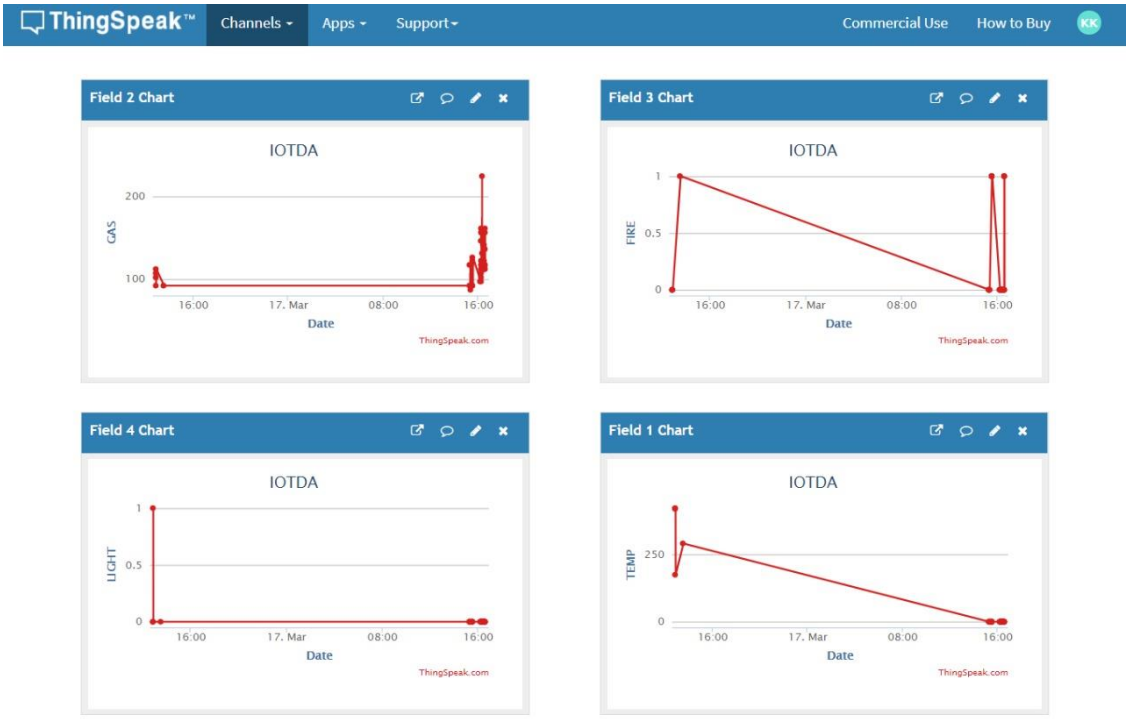
In case any parameter exceeds the threshold level which is set by the user immediately the buzzer makes a beep sound and the corresponding relay is turned on which drives the further circuit to control the parameter.







The sensor data stored in WiFi module automatically uploads it into the web server hosted by Thingspeak in our case with the help of a hotspot or access point and we can see the data in



the form of graphical representations in the website by logging into our account which can be accessed from anywhere in the world.

CHAPTER 7

SUMMARY AND CONCLUSION

A discussion on how the aims and objectives are met is presented. An overall conclusion of IOT based industrial data acquisition has become more efficient, more applicable to today's applications. The work presented in this project was directed towards pushing IOT technology to the next level. The principle of operation of IOT based industrial data acquisition system was shown by operating the Arduino Uno Rev3 attached with embedded system and various sensors to measure industrial parameters. This result in a more efficient in operation because it is connected to a common free IOT based web page specially built to notify or email the responsible authority automatically so reduces the stress of constant monitoring. We can enhance the data acquisition system project to also provide a required control signal by implementing the controllers we need using various softwares like Matlab and Lab view.

Machines are integrated with sensors like temperature, pressure, fuel, rpm etc. These sensors are connected to a machine controller. This controller collects complete data from the sensors and controls the machine with a logic of operation. IoT helps to retrieve this data collected from the controller with a GPRS enabled gateway and store the data on a server. There are different architectures which can use to collect data on the single server platform. One is with a GPRS gateway integrated with each machine controller and other is a wireless RF (Radio frequency) device installed on each machine controller over mesh communication network with a single GPRS enabled DCU (Data concentrator unit). The server can be a locally installed or it can be a cloud-based server. Cloud-based servers (Like Amazon, Azure, etc.) are in demand for real-time monitoring of machines and equipment. This data can be then seen on to your computer screens or mobile phones.

Once the data is collected on to the server the next step is data analytics which plays the important role. Data analytics helps to set the thresholds and parameters of the machines. Real-time monitoring helps monitor production flow in near real-time to eliminate waste, manage equipment's remotely, setting condition monitoring alerts for preventive maintenance, minimizing downtime and increase throughput.

IoT in manufacturing also known as **Industry 4.0** is the current trend of automation and data exchange in manufacturing technologies. Proper use of real-time analytics and condition monitoring will grow organizations by optimizing production requirements and channelizing right feedback mechanism.

CHAPTER 8
BIBLIOGRAPHY

8.1 References

[1] Probabilistic Recovery of Incomplete Sensed Data in IoT
Berihun Fekade ; Taras Maksymyuk ; Maryan Kyryk ; Minho Jo
Publication Year: 2018, Page(s):2282 - 2292
Cited by: Papers (3)
[2] Narrowband Internet of Things: Simulation and Modeling
Yiming Miao ; Wei Li ; Daxin Tian ; M. Shamim Hossain ;Mohammed F. Alhamid
Publication Year: 2018, Page(s):2304 – 2314
[3] The Future Internet of Things: Secure, Efficient, and Model-Based
Joshua E. Siegel ; Sumeet Kumar ; Sanjay E. Sarma
Publication Year: 2018, Page(s):2386 – 2398
[4] Guest Editorial Special Issue on Emerging Social Internet of Things: Recent Advances and Applications
Giancarlo Fortino ; Mohammad Mehedi Hassan ; Mengchu Zhou ;Andrzej M. Goscinski ; Md Zakirul Alam Bhuiyan ; Jianqiang Li ;Sourav Bhattacharya
Publication Year: 2018, Page(s):2478 – 2482
[5] Evaluating Critical Security Issues of the IoT World: Present and Future Challenges
Mario Frustaci ; Pasquale Pace ; Gianluca Aloi ; Giancarlo Fortino
Publication Year: 2018, Page(s):2483 – 2495
[6] Edge Computing and Social Internet of Things for Large-Scale Smart Environments Development
[7] Ferguson, T. (2002) Have Your Objects Call My Object. Harvard Business Review, June.
[8] Nunberg, G. (2012) The Advent of the Internet: 12th April, Courses.
[9] Kosmatos, E.A., Tselikas, N.D. and Boucouvalas, A.C. (2011) Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture. Advances in Internet of Things: Scientific Research, 1,
[10] Aggarwal, R. and Lal Das, M. (2012) RFID Security in the Context of "Internet of Things". First International Conference on Security of Internet of Things, Kerala, 17-19 August 2012.
[11] Biddlecombe, E. (2009) UN Predicts "Internet of Things". Retrieved July 6 [12] Butler, D. (2020) Computing: Everything, Everywhere. Nature, 440, 402-405.
[13] Dodson, S. (2008) The Net Shapes up to Get Physical. Guardian.
[14] Gershenfeld, N., Krikorian, R. and Cohen, D. (2004) The Internet of Things. Scientific American,
[15] Lombreglia, R. (2010) The Internet of Things, Boston Globe. Retrieved October

8.2 Digital Reference

https://www.arduino.cc

https://www.components101.com

https://www.engineersgarage.com

https://www.elprocus.com

https://www.circuitdigest.com

https://www/tinkercad.com

https://www.instructables.com

https://www.tutorialpoints.com

https://www.circuits4you.com

https://www.electonicshub.org

https://www.geeksforgeeks.org

https://www.electronicwings.com

https://www.hackster.io/

https://www.sparkfun.com

https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide