CHAPTER ONE

SMART CITY

1.1. INTRODUCTION

Smart city is a municipality that uses information and communication technologies to increase operational efficiency, share information with the public and improve both the quality of government services and citizen welfare.

While the exact definition varies depending on whom you talk to, the overarching mission of a smart city is to optimize city functions and drive economic growth while improving quality of life for its citizens using smart technology and data analysis.

Fig 1.1 Smart city

1.2. WHAT IS A SMART CITY?

The term “smart city “was firstly introduced in the end of the 20th century. It is rooted in the implementation of user-friendly information and communication technologies developed by major industries for urban spaces. Its meaning has since been expanded to relate to the future of cities and their development. Smart cities are forward-looking, progressive and resource-efficient while providing at the same time a high quality of life. They promote social and tech-no logical innovations and link existing infrastructures.

They incorporate new energy, traffic and transport concepts that go easy on the environment. Their focus is on new forms of governance and public participation. Intelligent decisions need to be taken at the strategic level if cities want to become smart. It takes more than individual projects but careful decisions on long-term implementations. Considering cities as entire systems can help them achieve their ultimate goal of becoming smart. Smart cities forcefully tackle the current global challenges, such as climate change and scarcity of resources. Their claim is also to secure their economic competitiveness and quality of life for urban populations continuously on the rise.

1.3. SMART CITY CONCEPT

Despite there is some kind of consensus that the label Smart City represents innovation in city management, its services and infrastructures, a common definition of the term has not yet been stated. There is a wide variety of definitions of what a Smart City could be. However, two trends can be clearly distinguished in relation with what are the main aspects that Smart Cities must take into consideration.

On the other hand there is a set of definitions that put emphasis just on one urban aspect (technological, ecological, etc.) leaving apart the rest of the circumstances involved in a city. This group of mono-topic descriptions are misunderstanding that the final goal of a Smart City is to provide a new approach to urban management in which all aspects are treated with the interconnection that takes place in the real life of the city. Improving just one part of an urban ecosystem does not imply that the problems of the whole are being solved.

On the other hand there are some authors that emphasize how the main difference of the Smart City concept is the interconnection of all the urban aspects. The tangled problems between urbanization are infrastructural, social and institutional at the same time and this intertwining is reflected in the Smart City concept. From the definitions, it can be noticed that infrastructures are a central piece of the Smart City and that technology is the enabler that makes it possible, but it is the combination, connection and integration of all systems what becomes fundamental for a city being truly smart.

From these definitions it can be concluded that the Smart City concept implies a comprehensive approach to city management and development. These definitions show a balance of the technological, economic and social factors involved in an urban ecosystem. The definitions reflect a holistic approach to the urban problems taking advantage of the new technologies so that the urban model and the relationships among the stakeholders can be redefined.

1.4. CLIMATE CHANGE

Climate change is one of the most pressing issues we are currently faced with. CO2 emissions must be reduced in the decades to come while measures need to be taken to reign in global warming, floods and extended heat waves. Cities are responsible for approximately three quarters of greenhouse gases worldwide. Being major polluters they are also called upon to provide solutions.

1.5. FEATURES OF A SMART CITY; SMART CITY TECHNOLOGIES

Emerging trends such as automation, machine learning and the internet of things (IoT) are driving smart city adoption.

Theoretically, any area of city management can be incorporated into a smart city initiative. A classic example is the smart parking meter that uses an app to help drivers find available parking spaces without prolonged circling of crowded city blocks. The smart meter also enables digital payment, so there's no risk of coming up short of coins for the meter.

Also in the transportation arena, smart traffic management is used to monitor and analyze traffic flows to optimize streetlights to prevent roadways from becoming too congested based on time of day or rush-hour schedules. Smart public transit is another facet of smart cities, used to ensure public transportation meets user demand. Smart transit companies are able to coordinate services and fulfill riders' needs in real time,

improving efficiency and rider satisfaction. Ride-sharing and bike-sharing are also common services in a smart city.

Energy conservation and efficiency are major focuses of smart cities. Using smart sensors, smart streetlights dim when there aren't cars or pedestrians on the roadways. Smart grid technology can be used to improve operations, maintenance and planning, and to supply power on demand and monitor energy outages.

Smart city initiatives also aim to monitor and address environmental concerns such as climate change and air pollution. Sanitation can also be improved with smart technology, be it using internet-connected trash cans and IoT-enabled fleet management systems for waste collection and removal, or using sensors to measure water parameters and guarantee the quality of drinking water at the front end of the system, with proper wastewater removal and drainage at the back end.

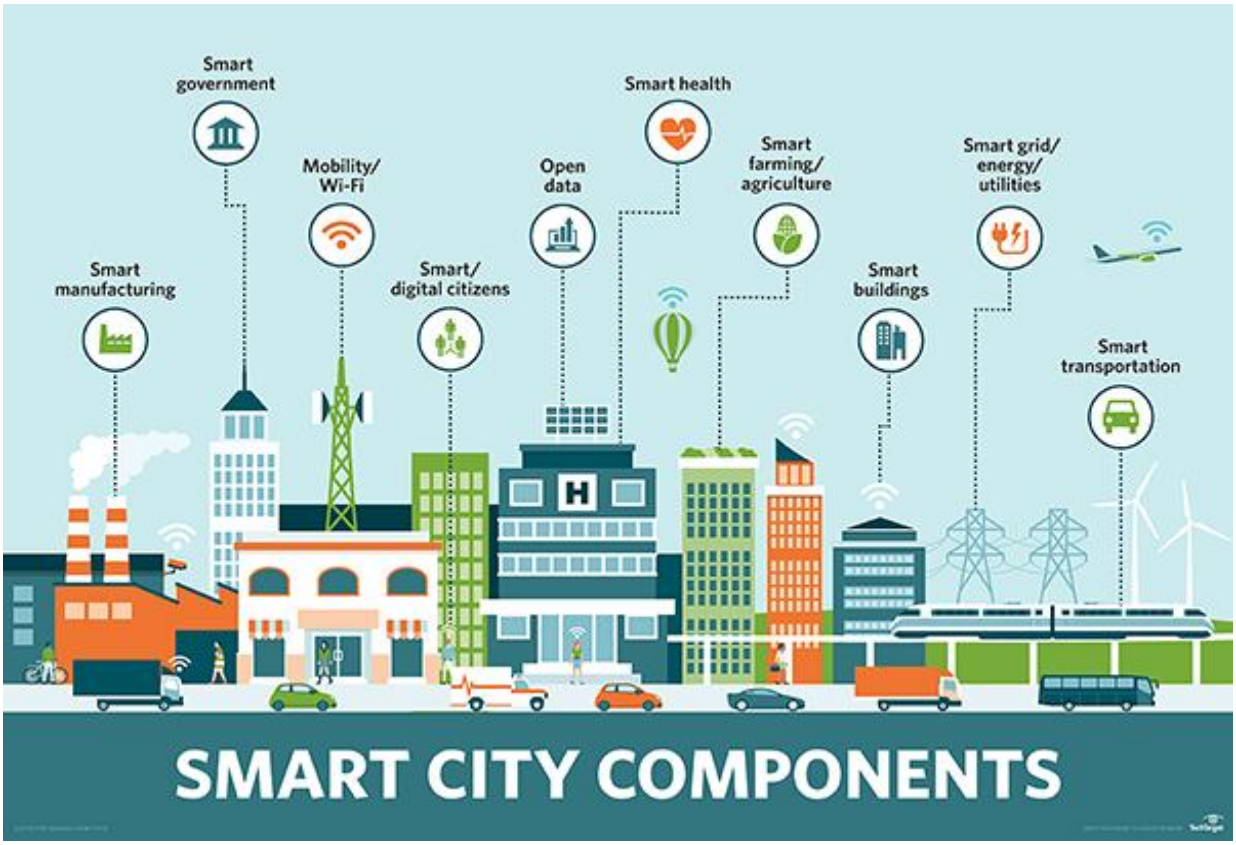


Fig 1.2 Smart city IOT

Smart city technology is increasingly being used to improve public safety, from monitoring areas of high crime to improving emergency preparedness with sensors. For example, smart sensors can be critical components of an early warning system before droughts, floods, landslides or hurricanes.

Smart buildings are also often part of a smart city project. Legacy infrastructure can be retrofitted and new buildings constructed with sensors to not only provide real-time space management and ensure public safety, but also to monitor the structural health of buildings. Attaching sensors to buildings and other structures can detect wear and tear and notify officials when repairs are needed. Citizens can help in this matter, notifying officials through a smart city app when repairs are needed in buildings and public infrastructure, such as potholes. Sensors can also be used to detect leaks in water mains and other pipe systems, helping reduce costs and improve efficiency of public workers.

Smart city technologies also bring efficiencies to urban manufacturing and urban farming, including job creation, energy efficiency, space management and fresher goods for consumers.

1.6. SMART CITIES FOSTER SUSTAINABILITY

Sustainability is another major facet of smart cities. Urbanization is expected to increase even more in the coming years -- today, 80% of the U.S. population lives in metropolitan areas versus 60% just 50 years ago. Smart technology will help cities sustain growth and improve efficiency for citizen welfare and government efficiency in urban areas in the years to come.

Water meters and manhole covers are just a couple of the other city components monitored by smart sensors. Free and/or publically available Wi-Fi is another benefit smart cities often include.

1.7. SMART CITY CHALLENGES AND CONCERNS

Smart city initiatives must include the people it aims to help: its residents, businesspeople and visitors. City leaders must not only raise awareness of the benefits of the smart city technologies being implemented, but also promote the use of open, democratized data to its citizens. If people know what they are participating in and the benefits it can bring, they are more likely to engage.

Fostering collaboration between the public and private sector and city residents is key to creating a smart citizen who will be engaged and empowered and positively contribute to the city and community. New and innovative collaboration methods can improve engagement. Smart city projects should include plans to make the data transparent and available to citizens, often through an open data portal or mobile app. This enables residents to engage with the data and understand what it is used for. Through a smart city app, residents may also be able to complete personal chores, such as viewing their home's energy consumption, paying bills and finding efficient public transportation.

Smart city opponents worry that city managers will not keep data privacy and security top of mind, fearing the exposure of the data that citizens produce on a daily basis to the risk of hacking or misuse. Additionally, the presence of sensors and cameras may be perceived as an invasion of privacy or government surveillance. To address this, smart city data collected should be anonymized and not be personally identifiable information.

CHAPTER TWO

PLAN OF WORK

2.1. OBJECTIVE

Cities are the main poles of human and economic activity. They hold the potential to create synergies allowing great development opportunities to their inhabitants. However, they also generate a wide range of problems that can be difficult to tackle as they grow in size and complexity. Cities are also the places where inequalities are stronger and, if they are not properly managed, their negative effects can surpass the positive ones. Urban areas need to manage their development, supporting economic competitiveness, while enhancing social cohesion, environmental sustainability and an increased quality of life of their citizens.

2.2. GENERAL BLOCK DIAGRAM

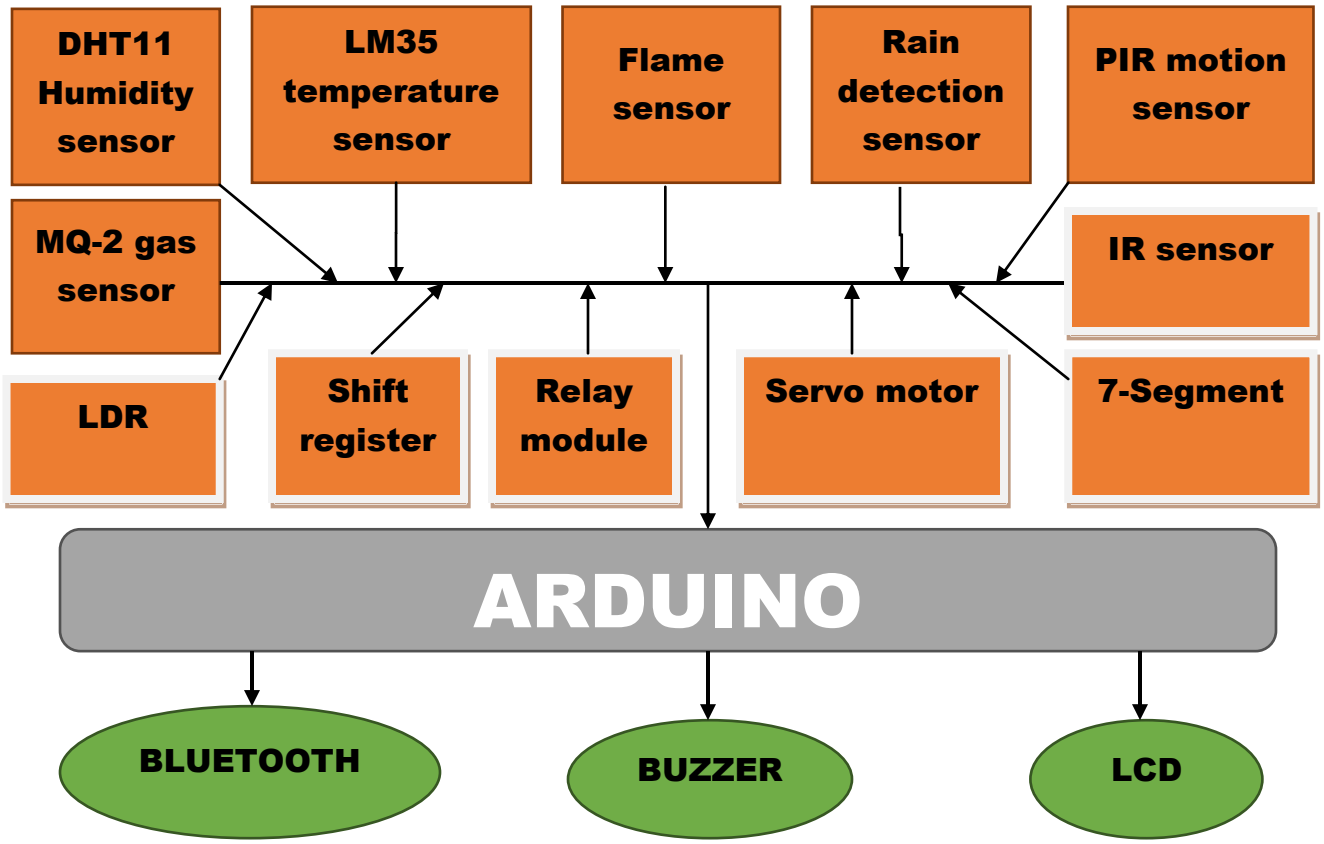


Fig 2.1 Block diagram

2.3. ANALOG TO DIGITAL CONVERT

Connecting digital circuitry to sensor devices is simple if the sensor devices are inherently digital themselves. Switches, relays, and encoders are easily interfaced with gate circuits due to the on/off nature of their signals. However, when analog devices are involved, interfacing becomes much more complex. What is needed is a way to electronically translate analog signals into digital (binary) quantities, and vice versa. An analog-to-digital converter, or ADC, performs the former task while a digital-to-analog converter, or DAC, performs the latter.

An ADC inputs an analog electrical signal such as voltage or current and outputs a binary number. In block diagram form, it can be represented as such:

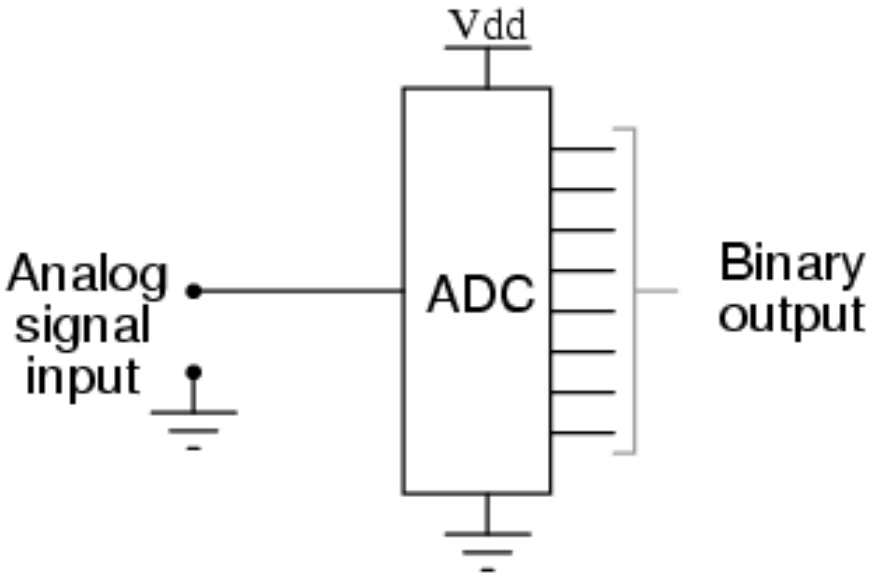


Fig 2.2 ADC

A DAC, on the other hand, inputs a binary number and outputs an analog voltage or current signal. In block diagram form, it looks like this:

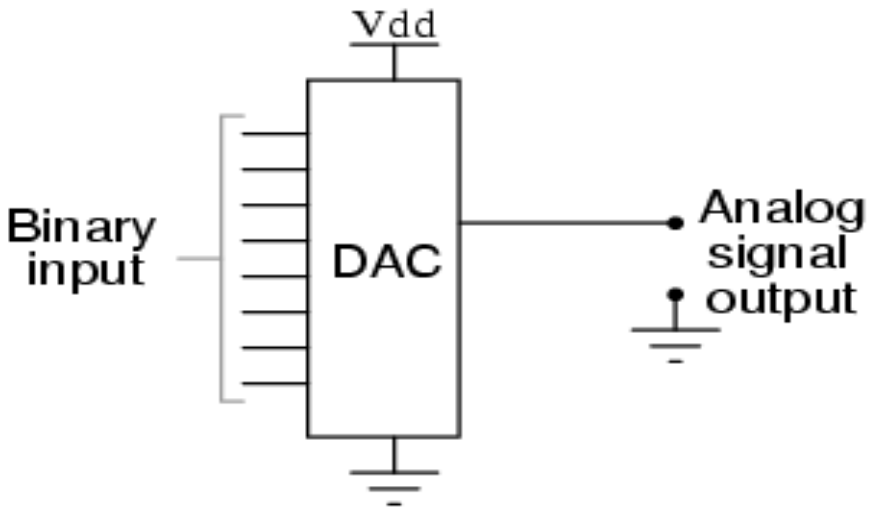


Fig 2.3 DAC

Together, they are often used in digital systems to provide complete interface with analog sensors and output devices for control systems such as those used in automotive engine controls:

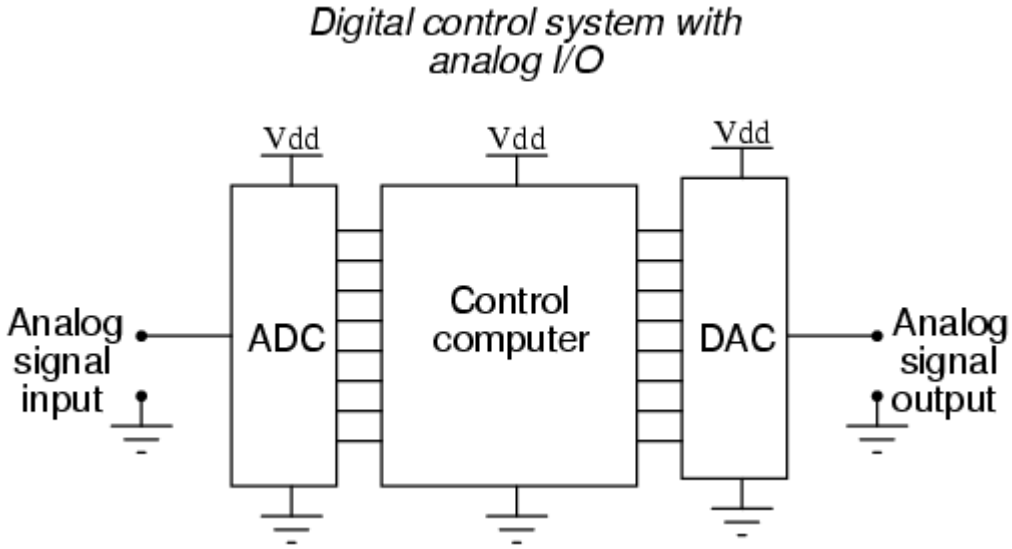


Fig 2.4 vice versa

It is much easier to convert a digital signal into an analog signal than it is to do the reverse. Therefore, we will begin with DAC circuitry and then move to ADC circuitry.

2.3.1. FLASH ADC

Also called the *parallel* A/D converter, this circuit is the simplest to understand. It is formed of a series of comparators, each one comparing the input signal to a unique reference voltage. The comparator outputs connect to the inputs of a priority encoder circuit, which then produces a binary output. The following illustration shows a 3-bit flash ADC circuit:

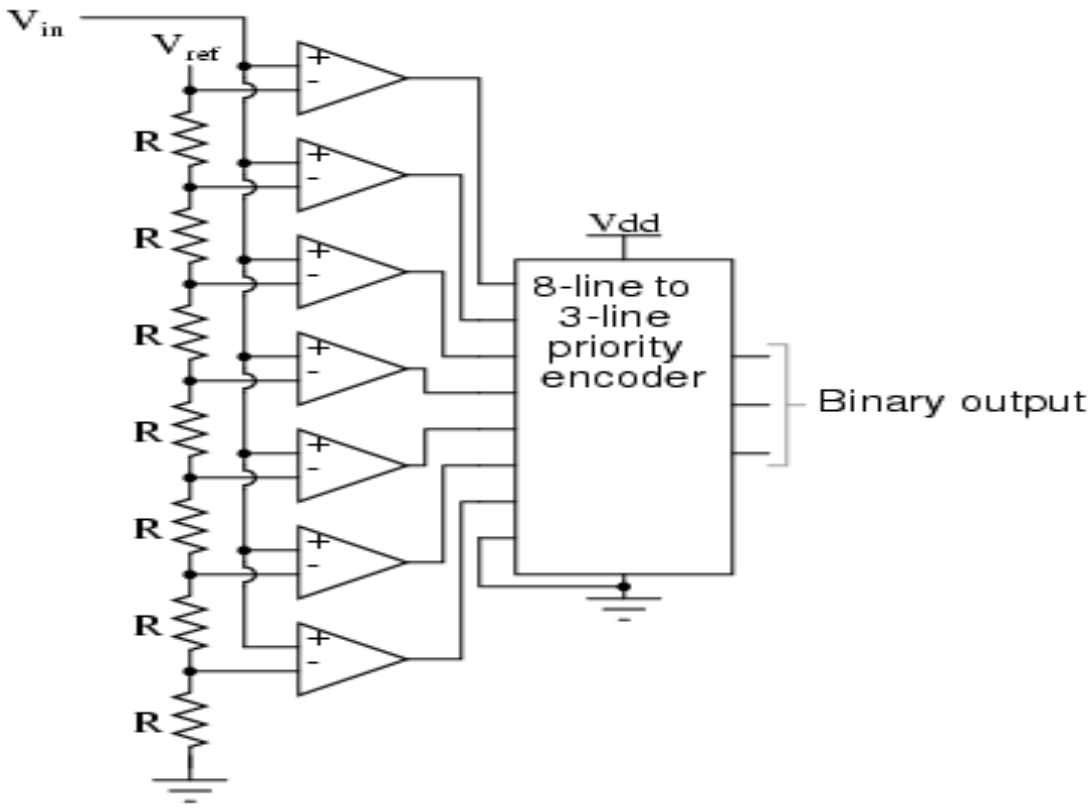


Fig 2.5 Flash ADC

V_{ref} is a stable reference voltage provided by a precision voltage regulator as part of the converter circuit, not shown in the schematic. As the analog input voltage exceeds the reference voltage at each comparator, the comparator outputs will sequentially saturate to a high state. The priority encoder generates a binary number based on the highest-order active input, ignoring all other active inputs.

When operated, the flash ADC produces an output that looks something like this:

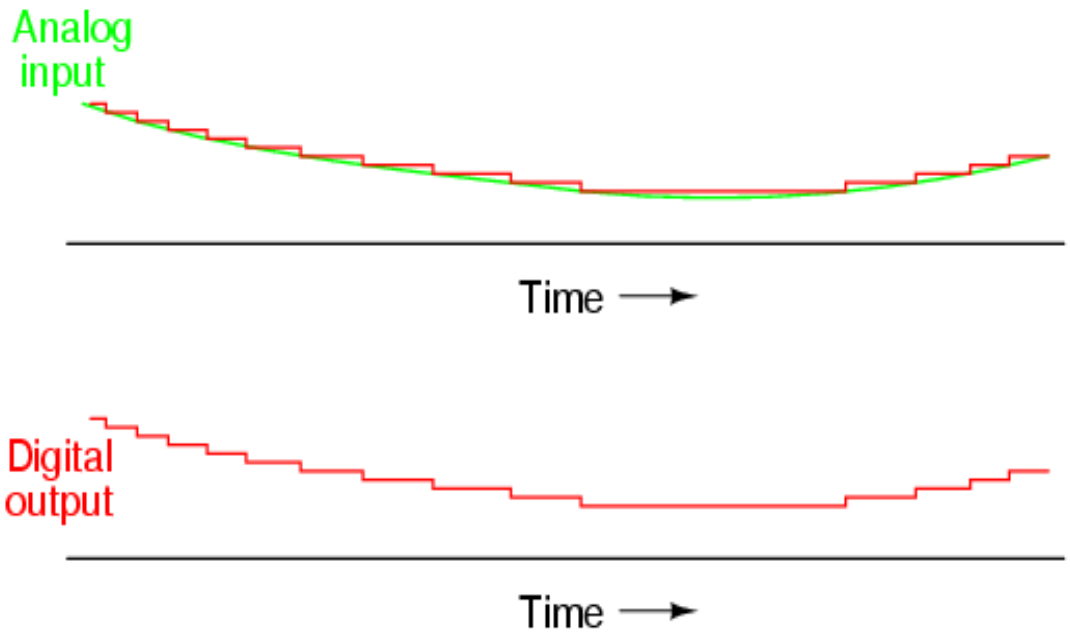


Fig 2.6 Signal

For this particular application, a regular priority encoder with all its inherent complexity isn't necessary. Due to the nature of the sequential comparator output states (each comparator saturating "high" in sequence from lowest to highest), the same "highest-order-input selection" effect may be realized through a set of Exclusive-OR gates, allowing the use of a simpler, non-priority encoder.

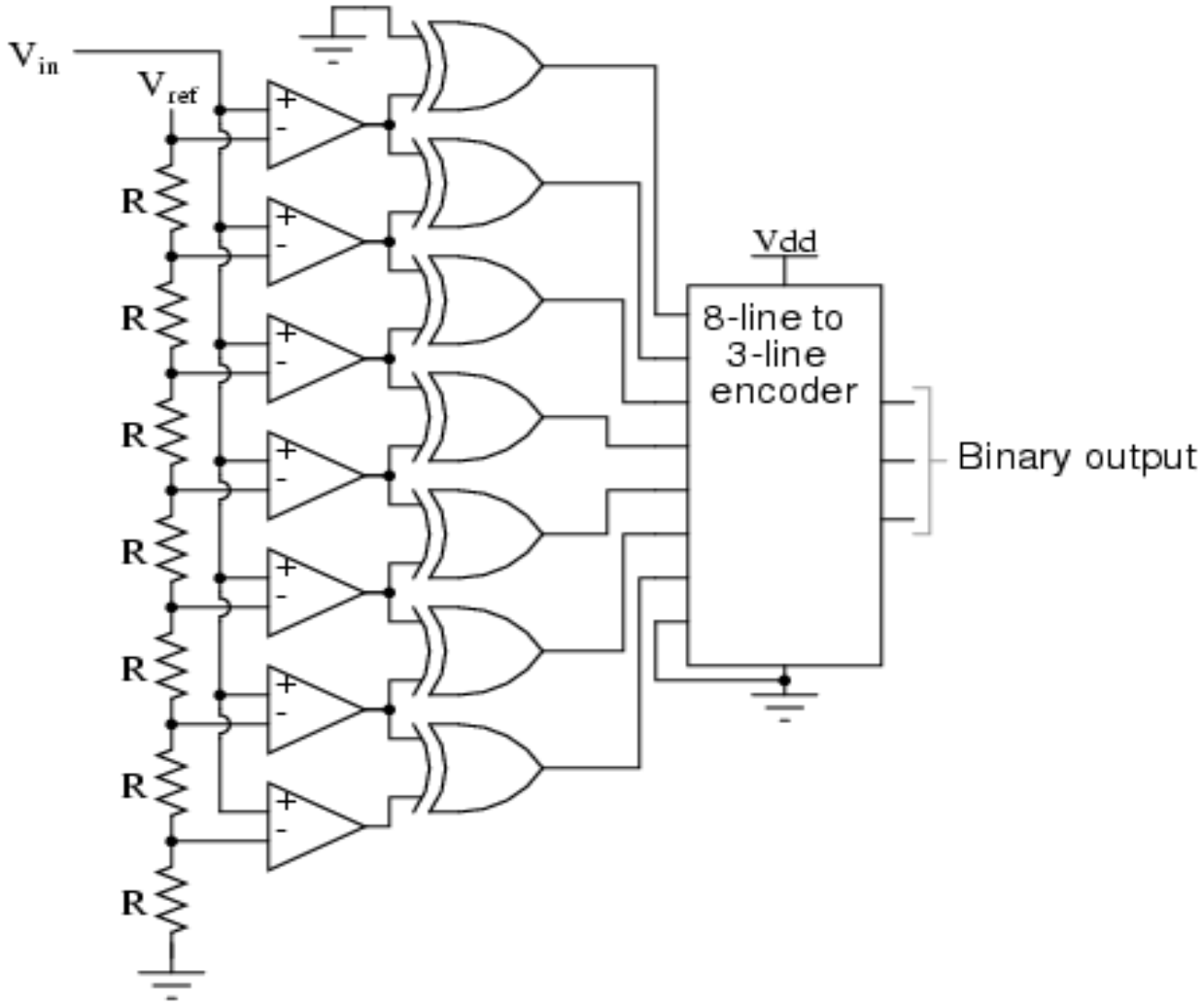


Fig 2.7 Circuit diagram

And, of course, the encoder circuit itself can be made from a matrix of diodes, demonstrating just how simply this converter design may be constructed:

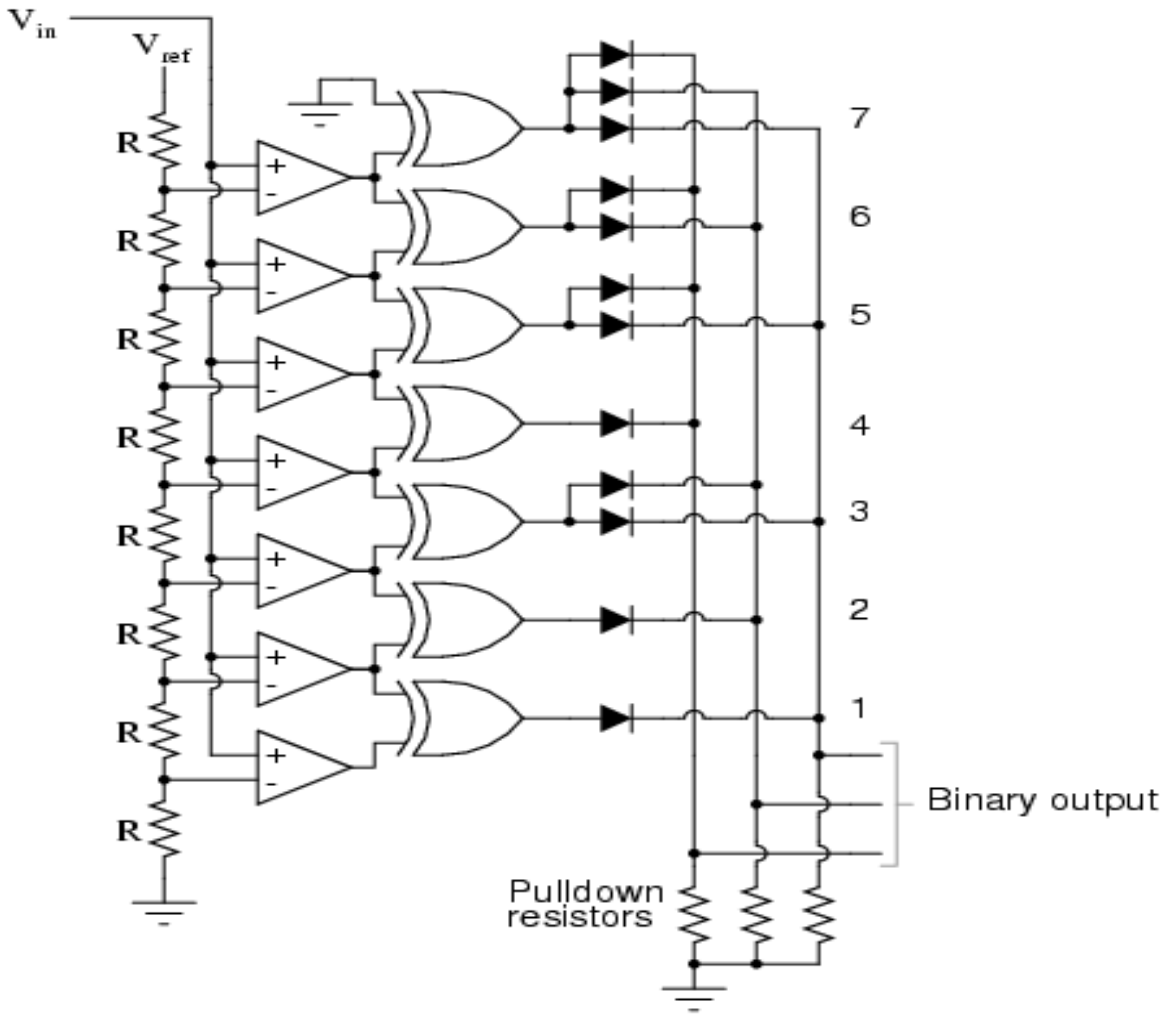


Fig 2.8 Circuit diagram

Not only is the flash converter the simplest in terms of operational theory, but it is the most efficient of the ADC technologies in terms of speed, being limited only in comparator and gate propagation delays. Unfortunately, it is the most component-intensive for any given number of output bits. This three-bit flash ADC requires seven comparators. A four-bit version would require 15 comparators. With each additional output bit, the number of required comparators doubles. Considering that eight bits is generally considered the minimum necessary for any practical ADC (255 comparators needed!), the flash methodology quickly shows its weakness.

An additional advantage of the flash converter, often overlooked, is the ability for it to produce a non-linear output. With equal-value resistors in the reference voltage

divider network, each successive binary count represents the same amount of analog signal increase, providing a proportional response. For special applications, however, the resistor values in the divider network may be made non-equal. This gives the ADC a custom, nonlinear response to the analog input signal. No other ADC design is able to grant this signal-conditioning behavior with just a few component value changes.

2.3.2. RAMP ADC

Also known as the *stair step-ramp*, or simply *counter* A/D converter, this is also fairly easy to understand but unfortunately suffers from several limitations.

The basic idea is to connect the output of a free-running binary counter to the input of a DAC, then compare the analog output of the DAC with the analog input signal to be digitized and use the comparator's output to tell the counter when to stop counting and reset. The following schematic shows the basic idea:

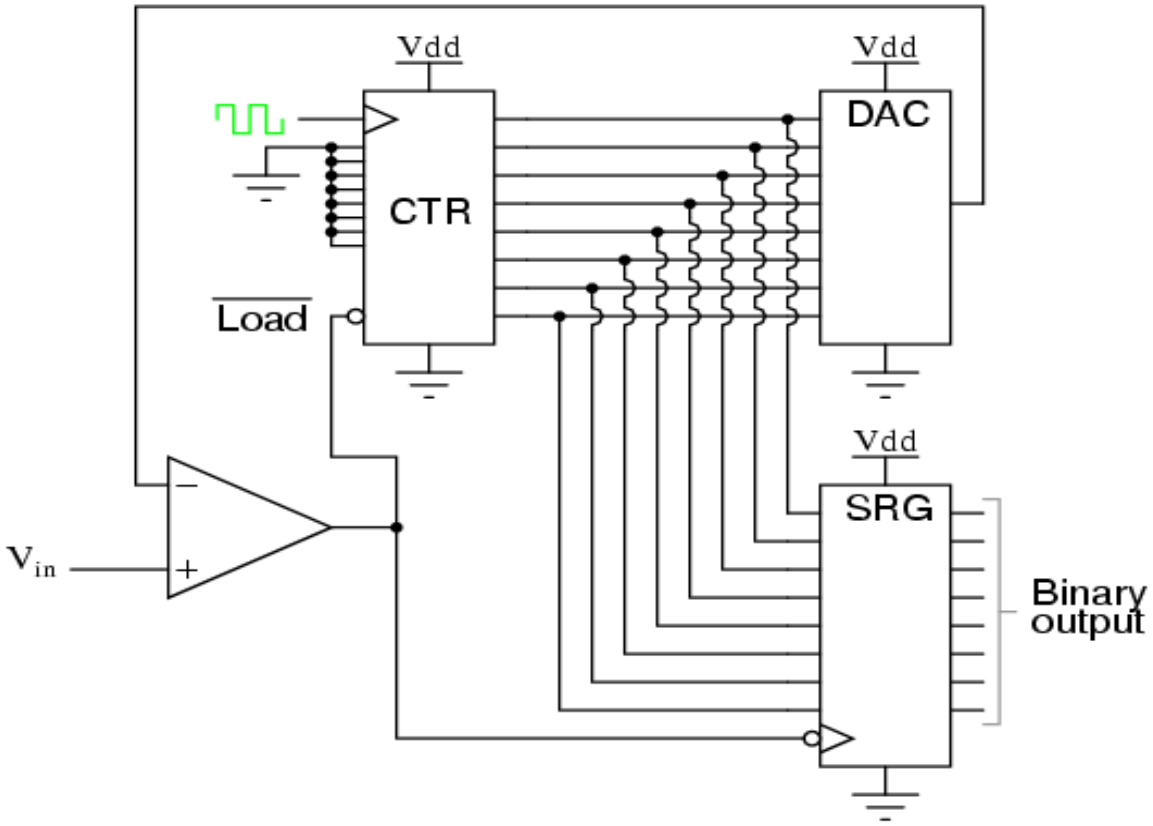


Fig 2.9 Block diagram

As the counter counts up with each clock pulse, the DAC outputs a slightly higher (more positive) voltage. This voltage is compared against the input voltage by the comparator. If the input voltage is greater than the DAC output, the comparator's output will be high and the counter will continue counting normally. Eventually, though, the DAC output will exceed the input voltage, causing the comparator's output to go low. This will cause two things to happen: first, the high-to-low transition of the comparator's output will cause the shift register to "load" whatever binary count is being output by the counter, thus updating the ADC circuit's output; secondly, the counter will receive a low signal on the active-low LOAD input, causing it to reset to 00000000 on the next clock pulse.

The effect of this circuit is to produce a DAC output that ramps up to whatever level the analog input signal is at, output the binary number corresponding to that level, and start over again. Plotted over time, it looks like this:

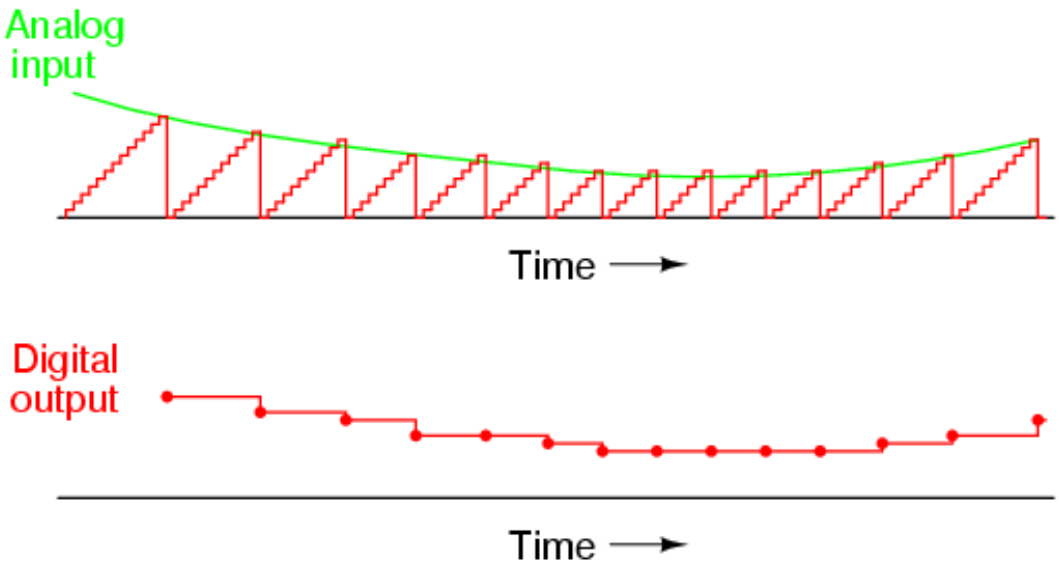


Fig 2.10 signal

Note how the time between updates (new digital output values) changes depending on how high the input voltage is. For low signal levels, the updates are rather close-spaced. For higher signal levels, they are spaced further apart in time:

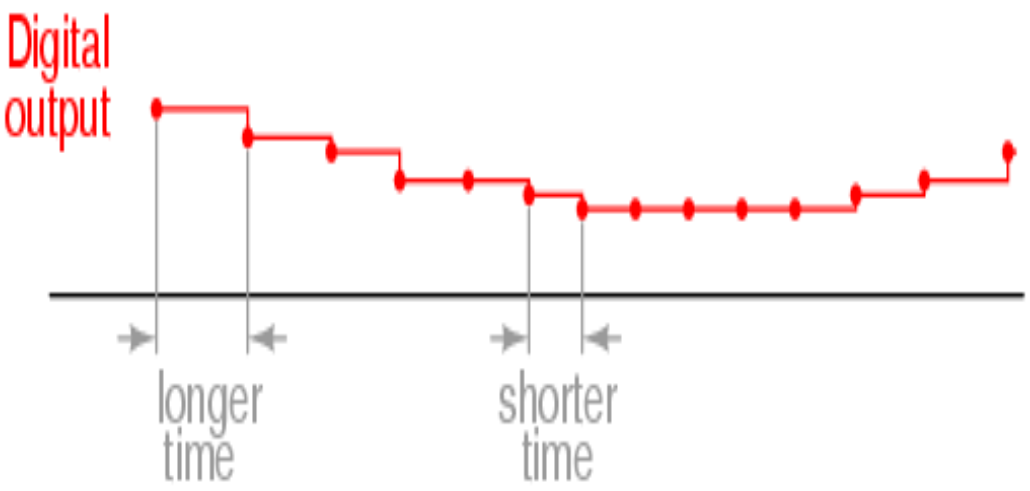


Fig 2.11 Digital signal

For many ADC applications, this variation in update frequency (sample time) would not be acceptable. This, and the fact that the circuit's need to count all the way from 0 at the beginning of each count cycle makes for relatively slow sampling of the analog signal, places the digital-ramp ADC at a disadvantage to other counter strategies.

2.3.3. SAR ADC

The Successive Approximation Register ADC is a must-know.

One of the most common analog-to-digital converters used in applications requiring a sampling rate under 10 MSPS is the Successive Approximation Register ADC. This ADC is ideal for applications requiring a resolution between 8-16 bits. For more information on resolution and sampling rates, please refer to the first in this series of articles: Deciphering Resolution and Sampling Rate. The SAR ADC is one of the most intuitive analog-to-digital converters to understand and once we know how this type of ADC works, it becomes apparent where its strengths and weaknesses lie.

Basic Operation of the SAR ADC:-

The basic successive approximation register analog-to-digital converter is shown in the schematic below:

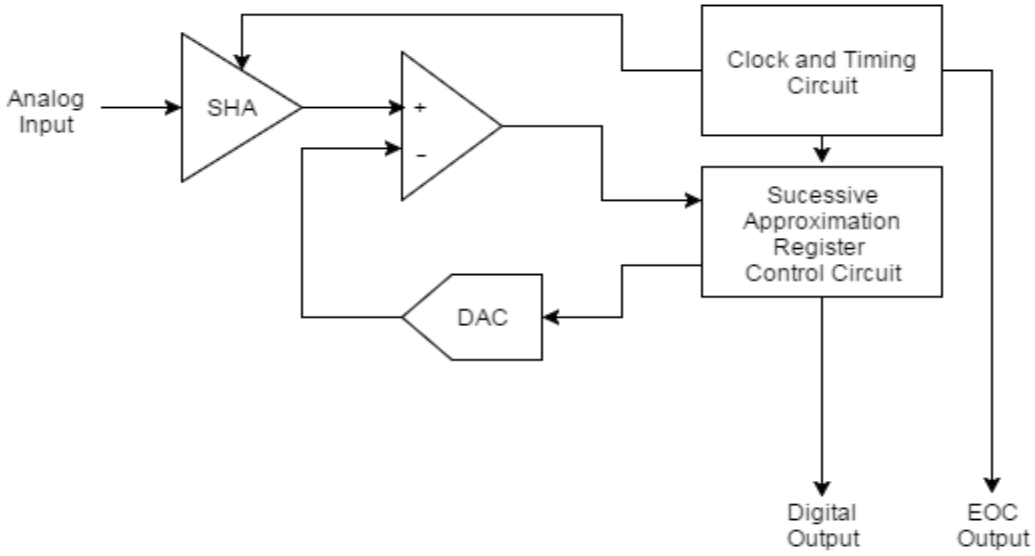


Fig 2.12 Block diagram

The SAR ADC does the following things for each sample:

1. The analog signal is sampled and held.
2. For each bit, the SAR logic outputs a binary code to the DAC that is dependent on the current bit under scrutiny and the previous bits already approximated. The comparator is used to determine the state of the current bit.
3. Once all bits have been approximated, the digital approximation is output at the end of the conversion (EOC).

The SAR operation is best explained as a binary search algorithm. Consider the code shown below. In this code, the current bit under scrutiny is set to 1. The resultant binary code from this is output to the DAC. This is compared to the analog input. If the result of the DAC output subtracted from the analog input is less than 0 the bit under scrutiny is set to 0.

2.4. MICROCONTROLLERS

A microcontroller (or MCU for microcontroller unit) is a small computer on a single integrated circuit. In modern terminology, it is similar to, but less sophisticated than, a system on a chip or SoC, a SoC may include a microcontroller as one of its components.

A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

2.4.1. WHAT IS ARDUINO?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

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

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

Fig 2.13 Arduino logo

2.4.2. WHY ARDUINO?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

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

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

2.4.3. ARDUINO VS MICROCONTROLLER

A microcontroller: on the other hand is a standalone single-chip IC that contains a CPU, read-only memory to store the program, RAM to store variables used in the execution of the program, and various I/O buses to connect to the outside world such as SPI, I2C, UART and others. By itself, it cannot execute any programs without being programmed via an external interface to a PC. A microcontroller may also need an external crystal to provide a clock, however some have an internal clock.

Some microcontrollers (such as Microchip's PIC32) have two sections of flash memory; one to hold initialization (boot) code, and another to store the application. This makes it easier to update the application code in-place.

For your purpose, you would want to use a microcontroller, not a microprocessor. To use a microcontroller, you would either have to design your own board, or buy some sort of development board.

A microcontroller is an IC that contains a CPU as well as some amount or RAM, ROM and other peripherals. Microcontrollers can function without external memory or storage.

Normally, microcontrollers are either programmed before being soldered to a PCB or are programmable using In-System-Programming (ISP or ICSP) connectors via a special "programmer" device attached to a personal computer.

Typical microcontrollers are much simpler and slower than typical microprocessors but I believe the distinction is mostly one of scale and application.

It is found, for example, in simple appliances such as basic washing machines.

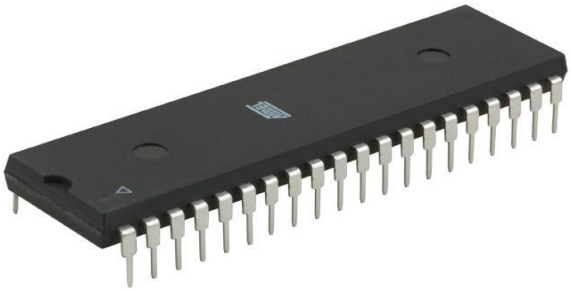


Fig 2.14 Microcontroller

2.4.4. ARDUINO VS MICROPROCESSOR

A microprocessor is typically found in a desktop PC or laptop and contains a CPU and an external memory interface plus various I/O buses to connect to the outside world such as SPI, I2C, UART, USB, LCD and others. A microprocessor will also have an external crystal to provide a clock.

Most microprocessors have no read-only memory on the chip; instead there is an external chip on the motherboard where the initial boot code is located. On Intel-based PC's, this is called the Basic Input/output System (BIOS) and also contains I/O routines in addition to the initial boot code. The boot code starts by doing a Power-On Self-Test (POST) and then looks to see where to load the next stage of the boot code -- from a hard drive, CD (or in olden days) a floppy disk. This second level boot then loads the operating system. (There may even be three levels of boot code in some systems.)
Some microprocessors (usually ones targeted for smart phones and tablets, which have limited boot options) have a small amount of read-only memory that contains the initial boot code.

I refer to the boot code as read-only; actually on some systems, it can be updated. However this is fairly risky; if something goes wrong the system may no longer boot.

Unlike microcontrollers, which execute their programs out of read-only memory, after booting up microprocessors load their programs into external RAM and execute it from there.

A microprocessor is an IC that contains only a central processing unit (CPU). The IC does not contain RAM, ROM or other peripherals. The IC may contain cache memory but it is not designed to be usable without any external memory.

Microprocessors cannot store programs internally and therefore typically load software when powered on, this usually involves a complex multi-stage "boot" process where "firmware" is loaded from external ROM and eventually an operating system is loaded from other storage media (e.g. hard disk).

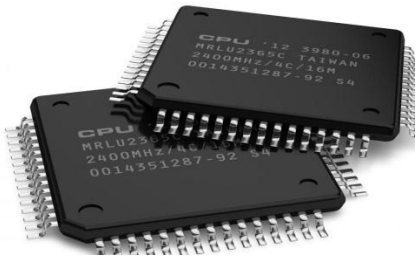


Fig 2.15 Microprocessor

CHAPTER THREE

HARDWARE CONNECTIONS

3.1. ARDUINO MEGA

The Arduino Mega is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

The Mega 2560 R3 also adds SDA and SCL pins next to the AREF. In addition, there are two new pins placed near the RESET pin. One is the IOREF that allow the shields to adapt to the voltage provided from the board. The other is a not connected and is reserved for future purposes. The Mega 2560 R3 works with all existing shields but can adapt to new shields which use these additional pins.

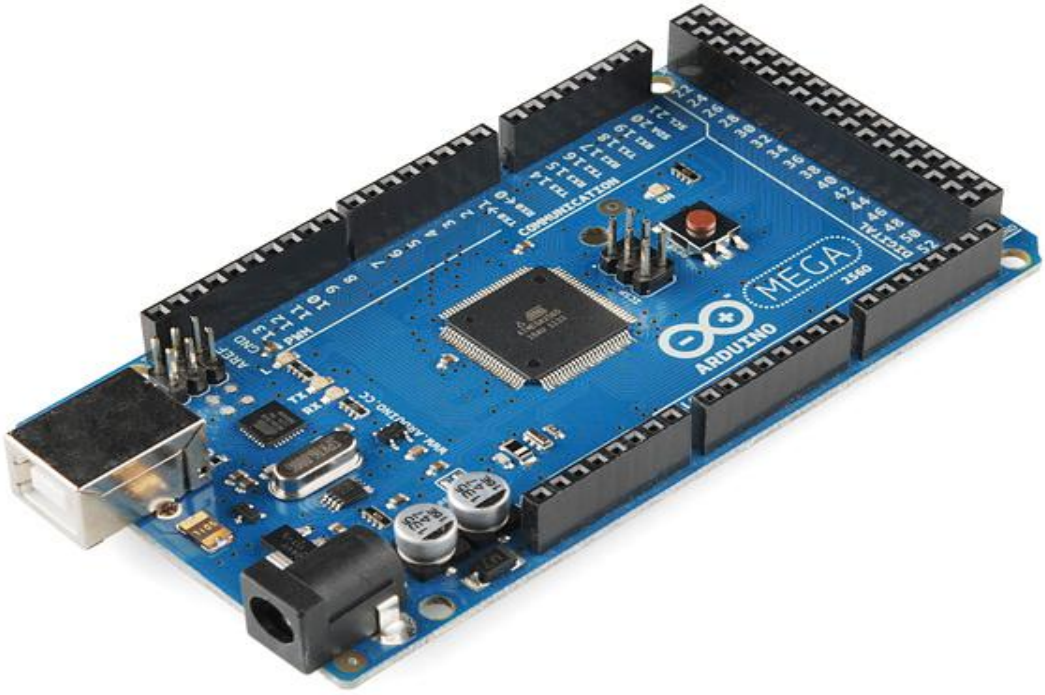


Fig 3.1.Arduino Mega

3.2. ARDUINO UNO

The Arduino UNO is a widely used open-source microcontroller board based on the ATmega328P microcontroller that comes preprogrammed with a bootloader that allows uploading a new code to the MCU without the use of an external hardware programmer and it's developed by Arduino CC. 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 Arduino Uno features are 14 digital input/output pins (of which 6 of them can be used as PWM outputs), 6 Analog pins, a -16MHz- quartz crystal, a USB connection, a power jack, an In-Circuit Serial Programming (ICSP) header and a reset button. External interrupts (typically 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.

The Arduino UNO can be programmed with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9 volt battery, as it accepts voltages between 7 and 20 volts.

"Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, but now evolved to newer releases.

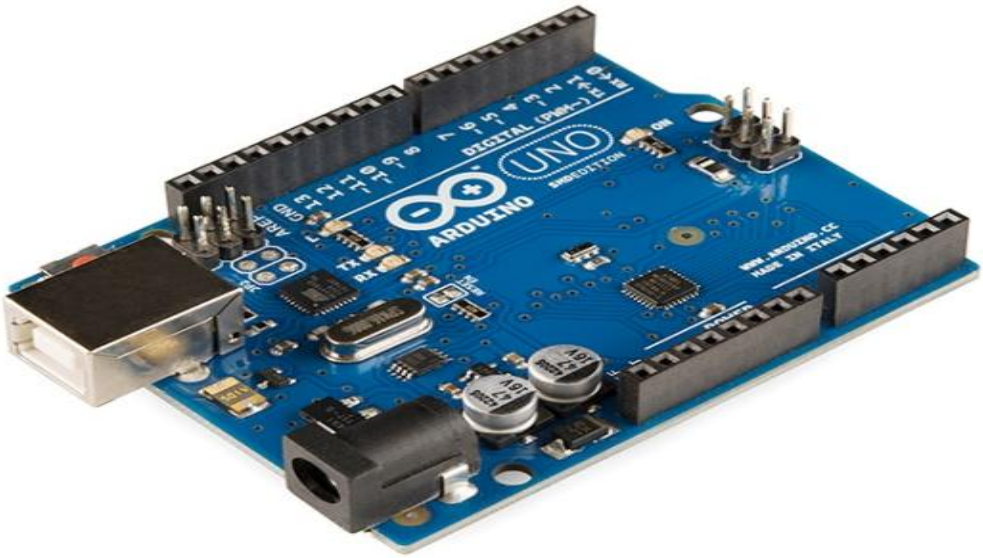


Fig.3.2 Arduino UNO

Table3-1 Arduino mega vs Arduino uno

	Arduino UNO	Arduino MEGA
Microcontroller	**ATmega328p**	**ATmega2560**
I/O's	**14**	**54**
Analog Inputs	**6**	**16**
PWM's	**6**	**14**
Flash memory	**32KB**	**128KB**
SRAM	**2KB**	**8KB**
EEPROM	**1KB**	**4KB**

3.3. PIR MOTION SENSOR

PIR Sensor is short for Passive IR, which basically has a pyroelectric material inside it, such material is made of a crystalline material that generates an electric charge when it receives an infra-red wave radiated from a moving living orgasm (human/animal) that cannot be seen by the human eye.

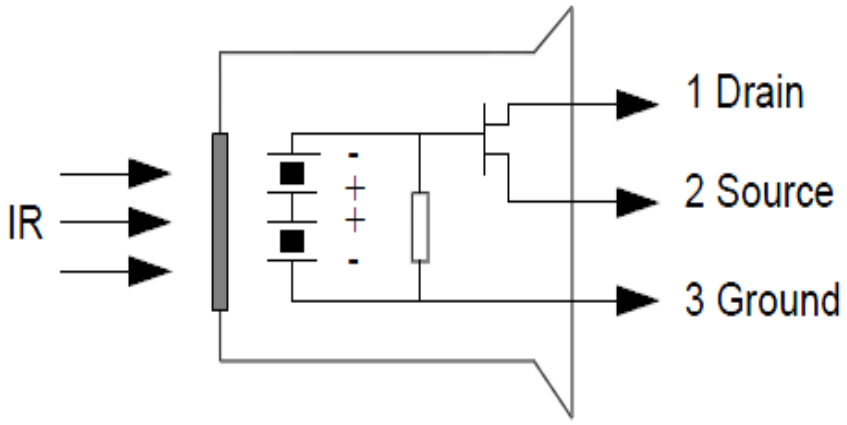


Fig 3.3 PIR Circuit diagram

The module actually consists of a Pyroelectric sensor which generates energy when exposed to heat.

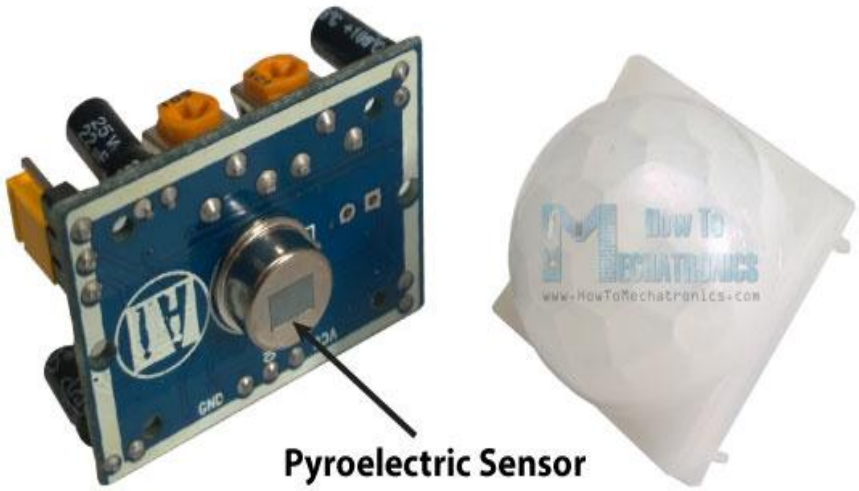


Fig 3.4 PIR Motion sensor

That means when a living orgasm (human body/animal body) gets in the range of the sensor it will detect a movement because the human body or animal body emits heat energy in a form of infrared radiation. That's where the name of the sensor comes from, a Passive Infra-Red sensor. And the term "passive" means that sensor is not using any energy for detecting purposes, it just works by detecting the energy given off by the other objects.

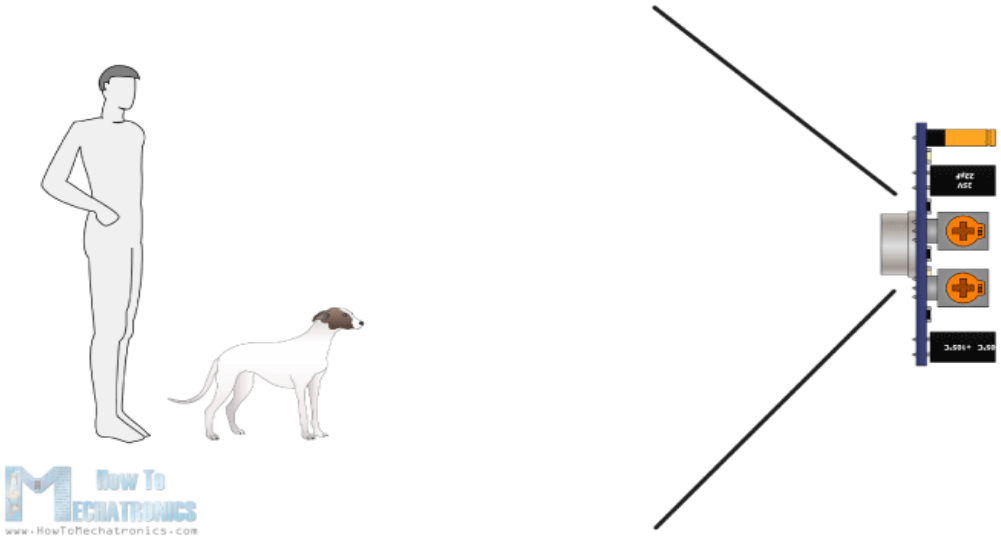


Fig 3.5 Detection range

The module also consists of a specially designed cover named Fresnel lens, which focuses the infrared signals onto the pyroelectric sensor.

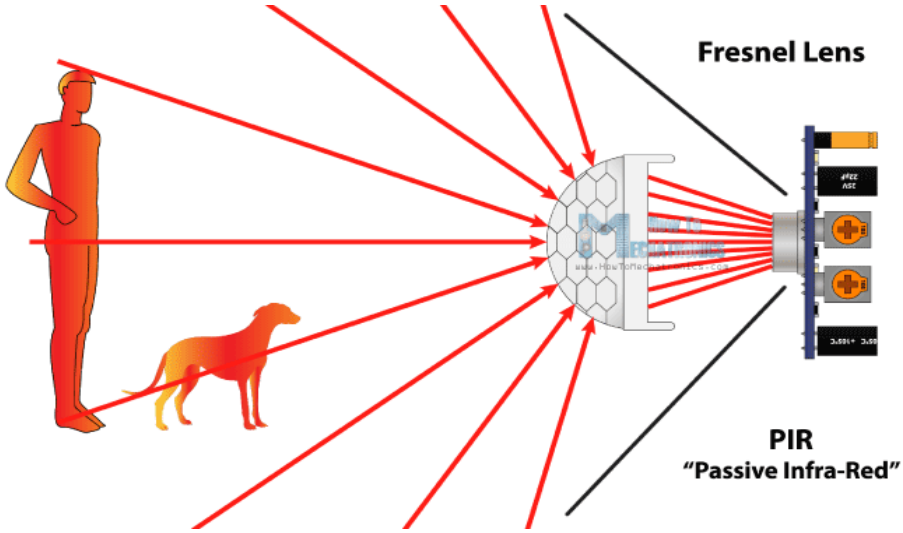


Fig 3.6 Detection principle

Now we look at the Fresnel lens, it's a spherical shaped lens made of high density polythene and what it does is that it enlarges the window of the object so that the detection area becomes larger.

The module has just three pins, a Ground and a VCC for powering the module and an output pin which gives high logic level if an object is detected. Also it has two potentiometers. One for adjusting the sensitivity of the sensor and the other for adjusting the time the output signal stays high when object is detected. This time can be adjusted from 0.3 seconds up to 5 minutes.

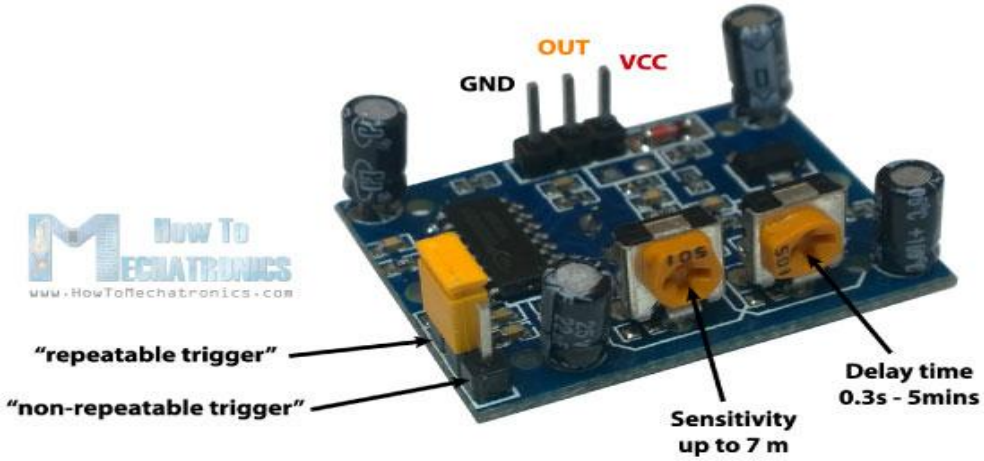


Fig 3.7 Pin out

The module has three more pins with a jumper between two of them. These pins are for selecting the trigger modes. The first one is called “non-repeatable trigger” and works like this: when the sensor output is high and the delay time is over, the output will automatically change from high to low level. The other mode called “repeatable trigger” will keep the output high all the time until the detected object is present in sensor’s range.

Motion sensor specifications:-

- Wide range for the input voltage ranging from +4v and 12v (+5v typically)
- Low power consumption (about 65mA)
- Has two operating modes (repeatable made & non-repeatable mode)
- Covers distance up to 7m with maximum angle = 110°
- Operating temperature is from -20° up to 80° Celsius.

3.4. FLAME DETECTOR

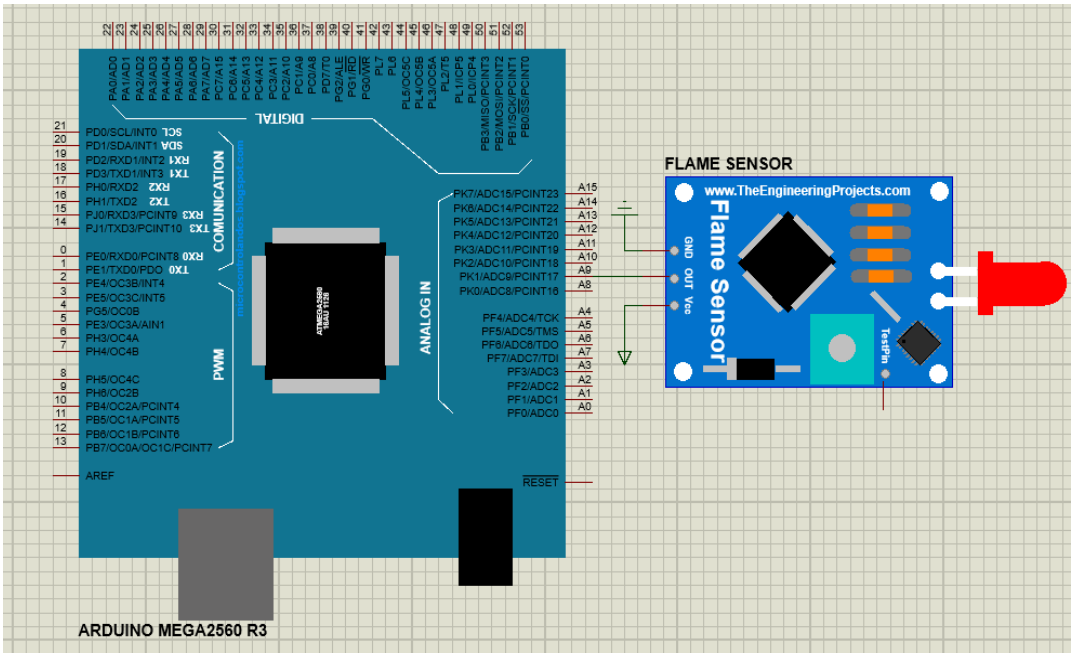


Fig 3.8 Circuit diagram of flame sensor

The Flame Sensor is used to detect the fire or light sources that have wavelengths ranging from 760nm to 1100nm. The module consists of an IR sensor, potentiometer, OP-Amp comparator circuitry and a led indicator. When a flame will be detected, the module will turn on its red led. This module is sensitive to flame but it can also detect ordinary light. The detection angle is 60 degrees. The sensitivity of this sensor is adjustable and it also has a stable performance.

It has both outputs, analog and digital. The analog output gives us a real time voltage output signal on thermal resistance while the digital output allows us to set a threshold via a potentiometer. In our tutorial we are going to use both of these outputs one by one and see how the sensor works. We can use the flame sensor to make an alarm when detecting the fire, for safety purpose in many projects and in many more ways.

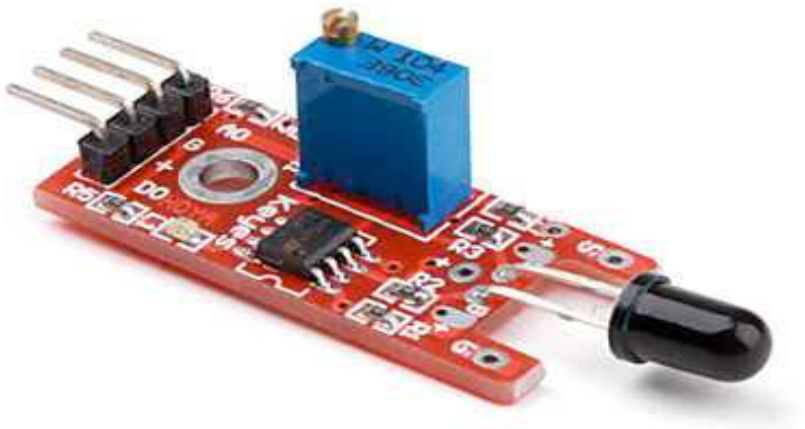


Fig 3.9 Flame sensor

The pins configurations are as follows (from left to right):-

A0: This is the analog pin and this will be connected to the analog pin of the Arduino.

G: This is the ground pin and this will be connected to the ground of the Arduino.

+: This is the input voltage pin of the sensor and this will be connected to the +5V of Arduino.

D0: This is the digital pin and this will be connected to the digital pin of Arduino.

The flame sensor specifications are as follows:-

- The operating voltage is from +3.3v and +5V.
- Detecting distance is up to 100cm and it could be increased if the intensity of flame became higher.
- The maximum detection angle of the flame sensor module is about 60 degrees.
- It gives us both analog output and digital output.
- It has a led indicator, which indicates that whether the flame is detected or not.
- The threshold value can be changes by rotating the top of potentiometer.

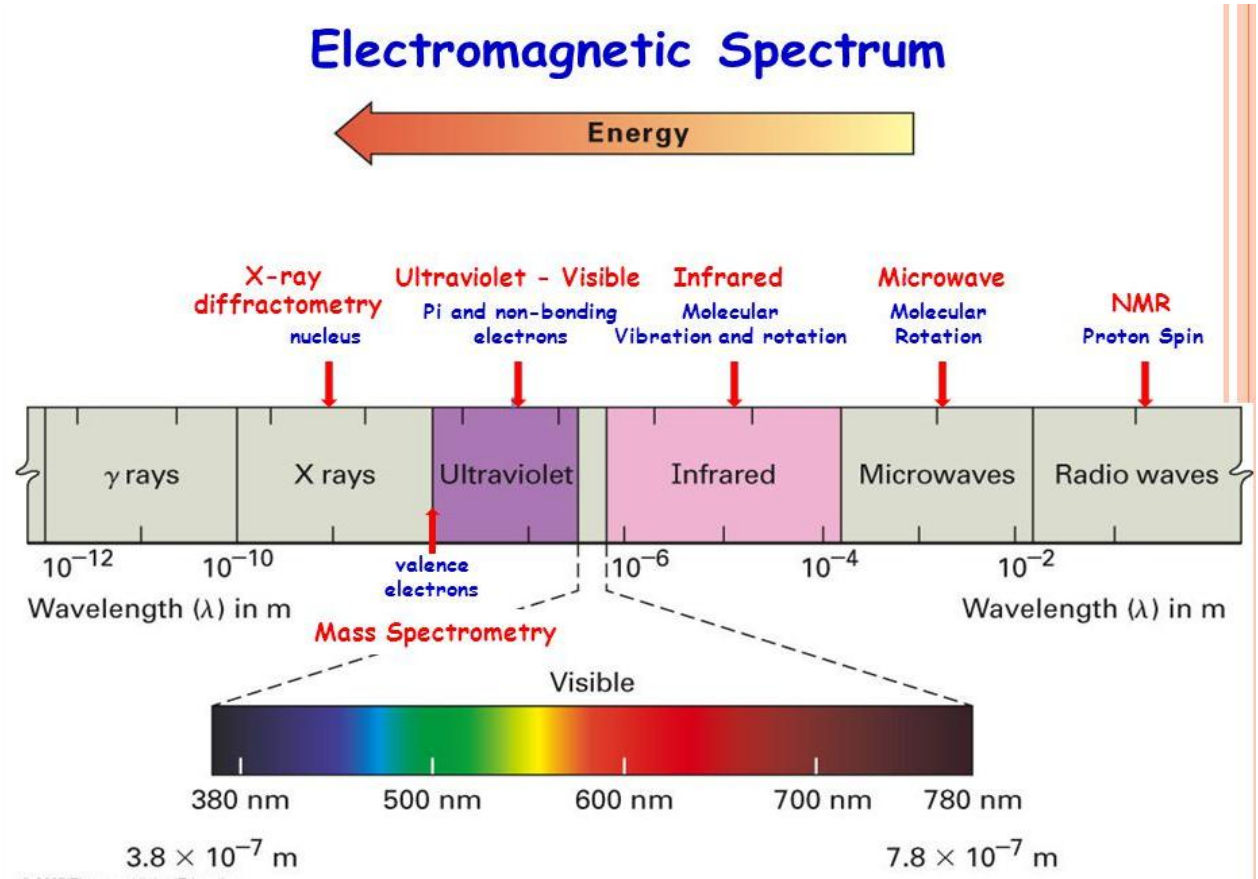


Fig.3.10 Electromagnetic spectrum

3.5. SOUND SENSOR

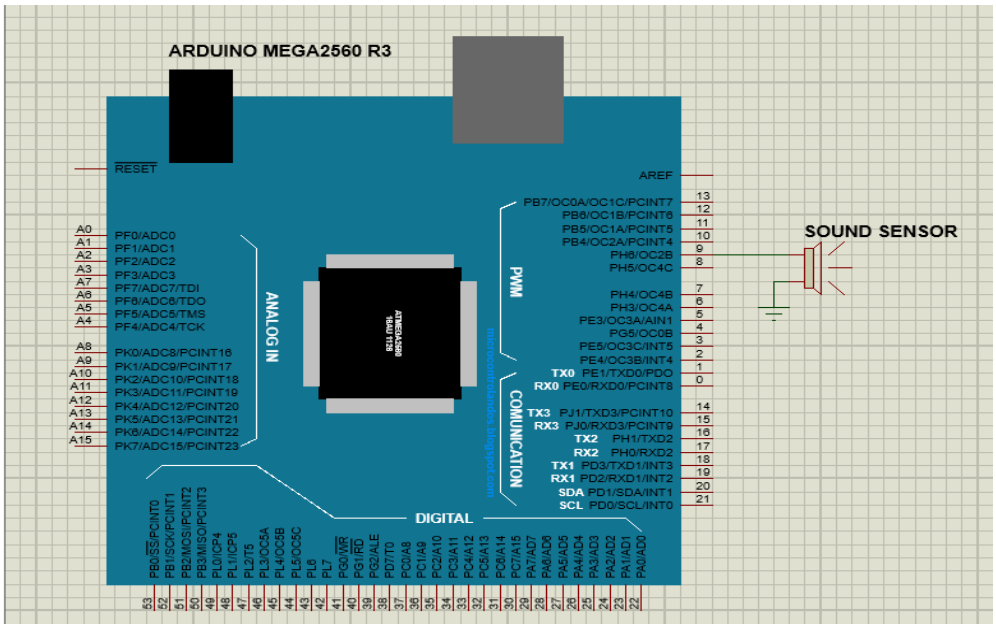


Fig 3.11 Circuit diagram of sound detector

This module allows you to detect when sound has exceeded a set point you select. Sound is detected via a microphone and fed into an LM393 op amp.

The sound level set point is adjusted via an on board potentiometer. When the sound level exceeds the set point, an LED on the module is illuminated and the output is sent low.

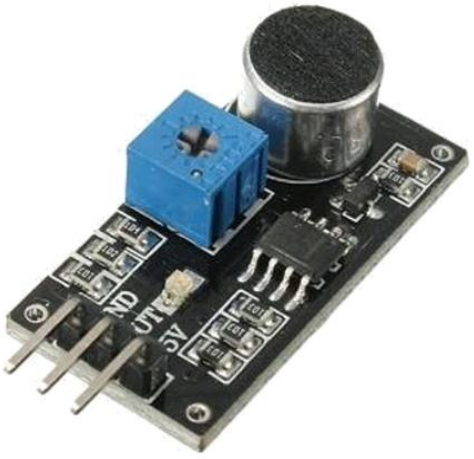


Fig 3.12 Sound detection sensor

Given that this device measures whether or not sound has exceeded a threshold, you're basically left with determining what it is you want to do. What I mean by this is that you can do something when it is quiet and/or you can do something when it is loud. For example:

- You could detect whether a motor is running or not.
- You could set a threshold on pump sound so that you know whether there is cavitation or not.
- In the presence of no sound, you might want to create an ambiance by turning on music.
- In the presence of no sound and no motion, you may go into an energy savings mode and turn off the lights.
    - When less sensitive, it takes more sound to trigger the device
    - When more sensitive, it takes less sound to trigger the device

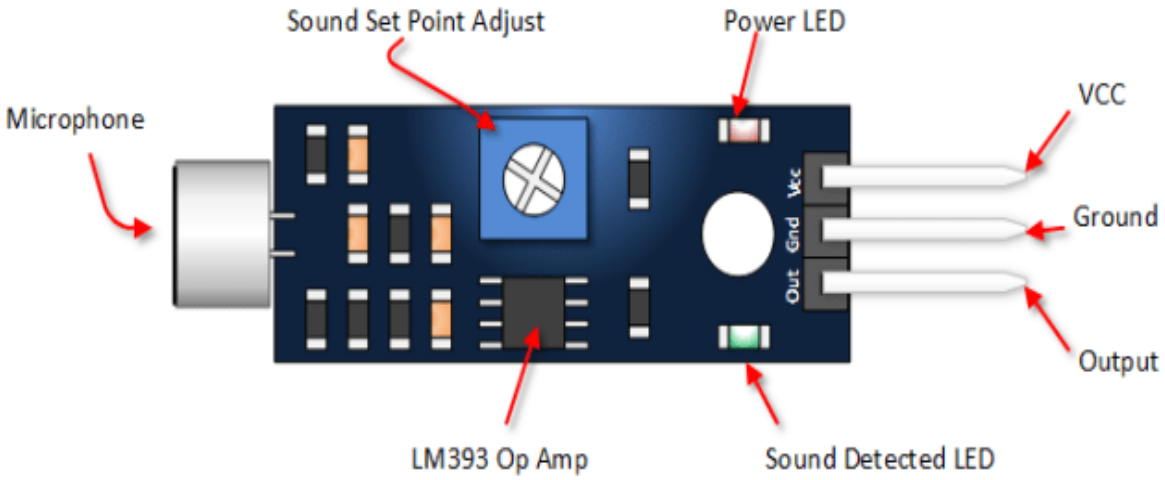


Fig 3.13 Pin out

3.6. RAIN DETECTOR

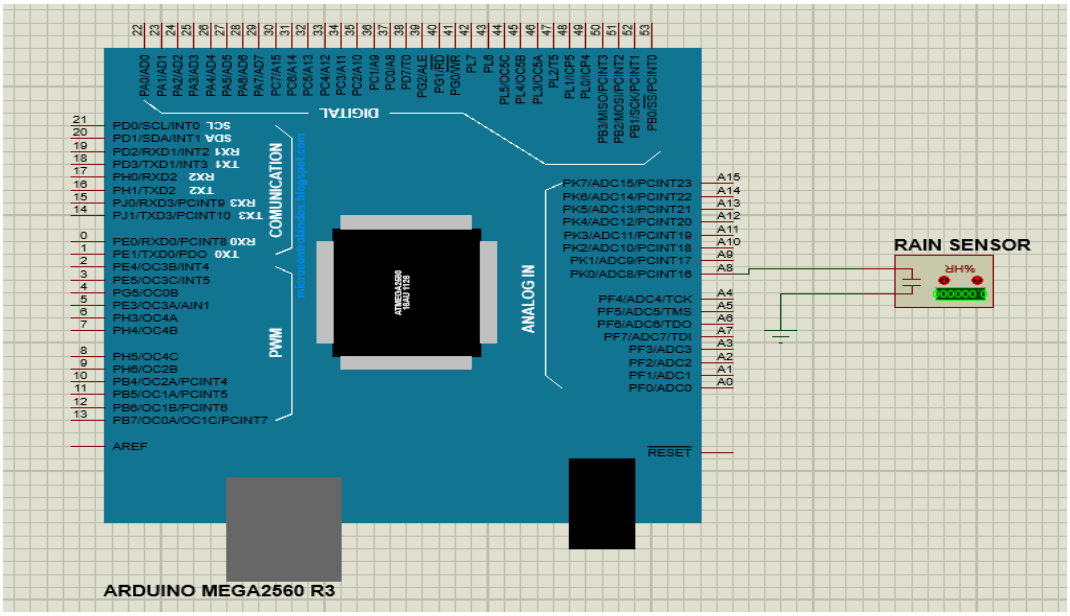


Fig 3.14 Circuit diagram of rain detector

Raindrop sensor is basically a board on which nickel is coated in the form of lines. It works on the principal of resistance. When there is no rain drop on board. The resistance is high so we gets high voltage according to V=IR. When rain drop present it reduces the resistance because water is a good conductor of electricity and the presence of water connects nickel lines in parallel so resistance is reduced and so does the voltage drop across it.

It consists of two parts one is a black or red board with nickel layers on it and other is an integrated chip provided with some output pins. Board has 2 output pin and chip has 6 pins.

Fig 3.15 Rain detection sensor

Pins configurations are as follows:-

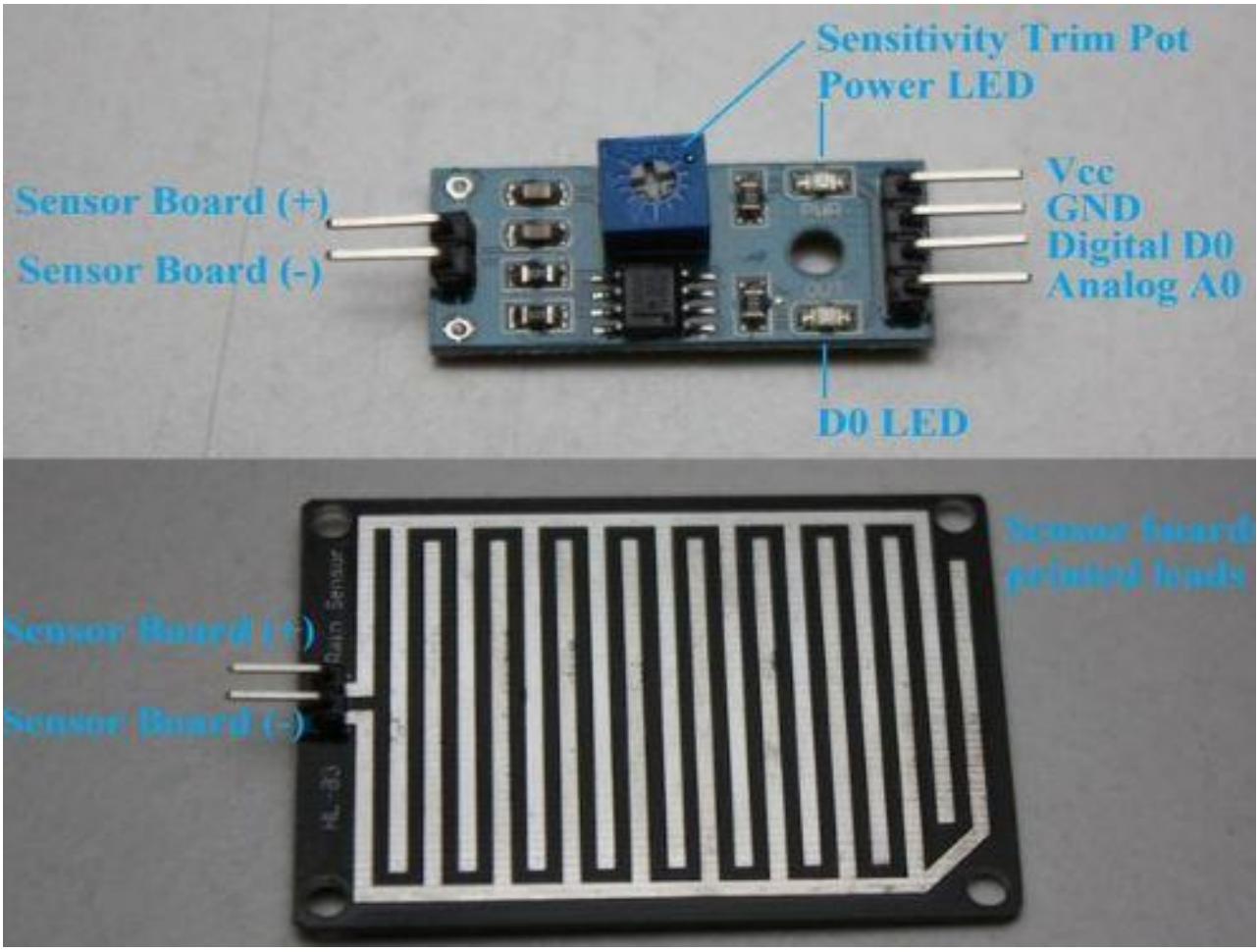


Fig.3.16 Pins configurations

Specifications of Raindrop Sensor are as follows:-

- It's a digital sensor.
- Adopts high quality double sided material.
- Area: 5cm x 4cm nickel plate on side.
- Anti-oxidation, anti-conductivity, with long use time.
- Comparator output signal is a good and clean waveform
- The potentiometer adjusts the sensitivity.
- Operating voltage is +5V.
- Output format: Digital output (0 and 1) or analog output.
- Small board PCB size: 3.2cm x 1.4cm.
- Uses a wide voltage LM393 comparator.

3.7. MQ-2 SMOKE DETECTOR

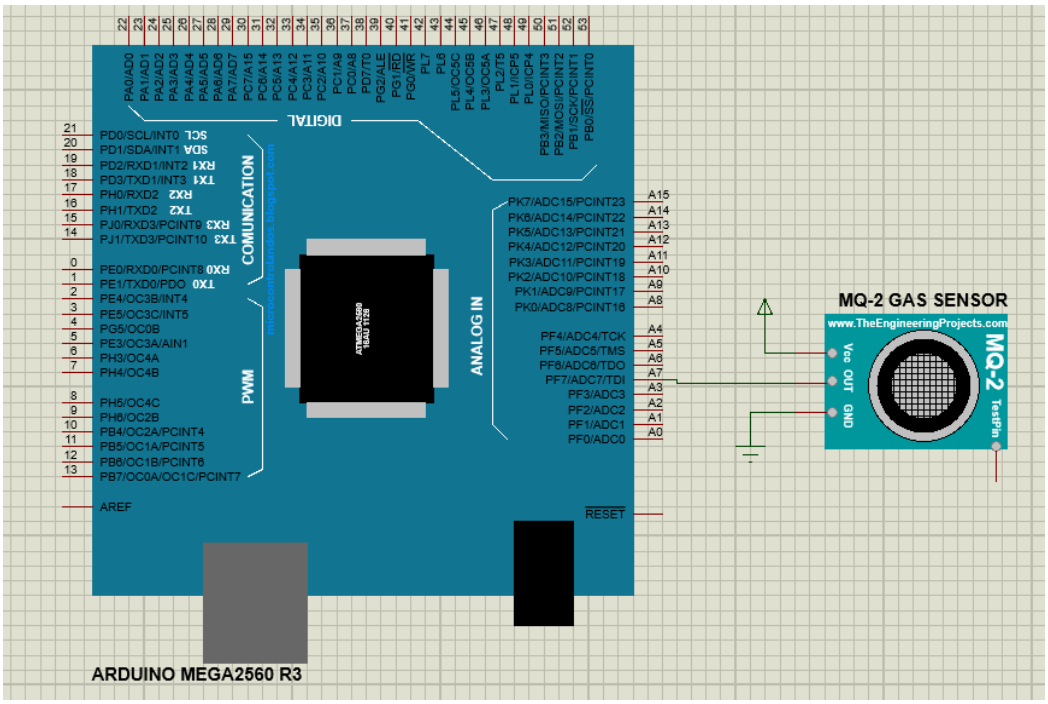


Fig 3.17 Circuit diagram of MQ-2 smoke detector

MQ-2 is basically a general purpose gas sensor (similar to MQ-5) which can sense a broad range of gases like LPG, Butane, Methane (CH4), Hydrogen and in addition to these gases MQ-2 is sensitive to smoke as well.

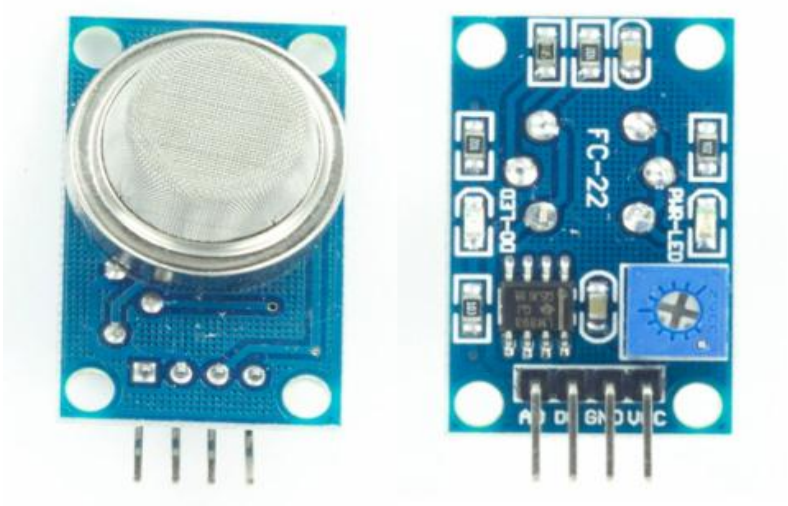


Fig 3.18 MQ-2 gas sensor

Pins configurations are as follows (from left to right):-

1- Analog output (A0)
2- Digital output (D0)
3- Ground (GND)
4- Vcc (+5v)

MQ-2 gas sensor specifications:-

- Operating voltage is +5v.
- Analog output and digital output.
- Wide detecting scope.
- Fast response and high sensitivity.
- Stable and long life.
- Can be used to detect gases like LPG, alcohol, propane, hydrogen, CO-2 and even methane.

Both MQ-5 and MQ-2 are basically gas sensors but their range of sensing different gas levels vary. For example, MQ-5 can sense LPG in a broader range of 200 ppm to 10,000 ppm, whereas the range of MQ-2 for LPG is short and is from 5,000 ppm to 10,000 ppm. Similarly MQ-2 is sensitive to smoke where as MQ-5 is not that sensitive to smoke. So we cannot choose MQ-5 gas sensor to design a smoke alarm or smoke involving applications.

MQ-2 can sense methane (CH4) up to 20,000 ppm where as MQ-5 can sense CH4 only up to 10,000 ppm. The difference between MQ-5 and MQ-2 therefore lies in its range of values. We choose the right sensor based on the application requirement; say for example – We cannot choose MQ-2 for sensing low levels of LPG in the range of 700 ppm because MQ-2 is insensitive low levels of LPG (its range begins at 5,000 ppm and extends to 10,000 ppm). So for applications to sense low levels of LPG, MQ-5 is the ideal choice as it can sense values starting from 200 ppm.

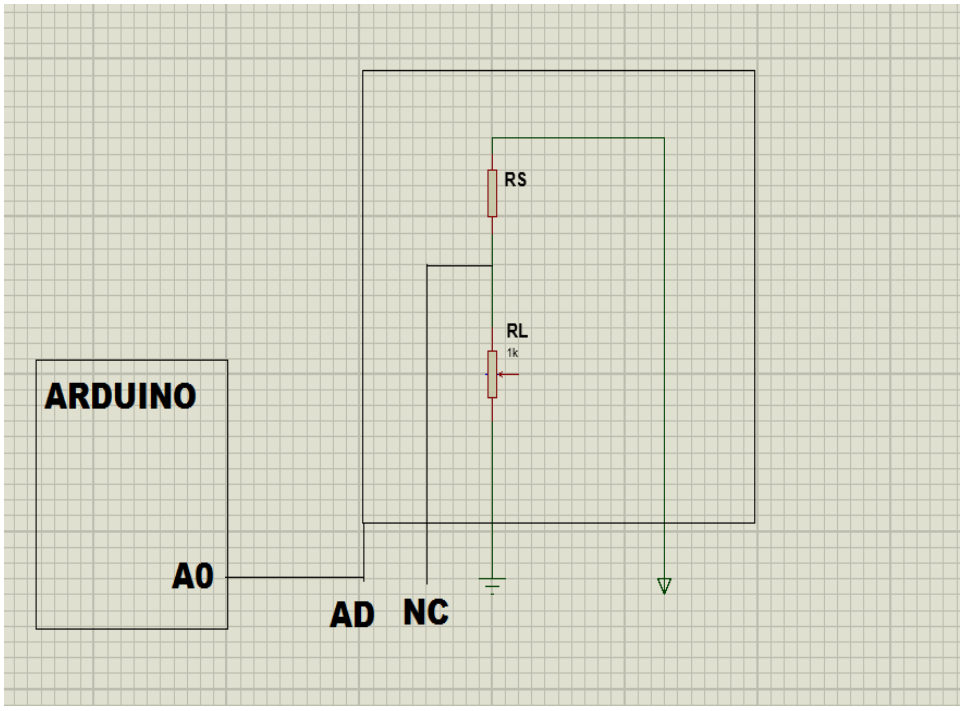


Fig 3.19 Schematic

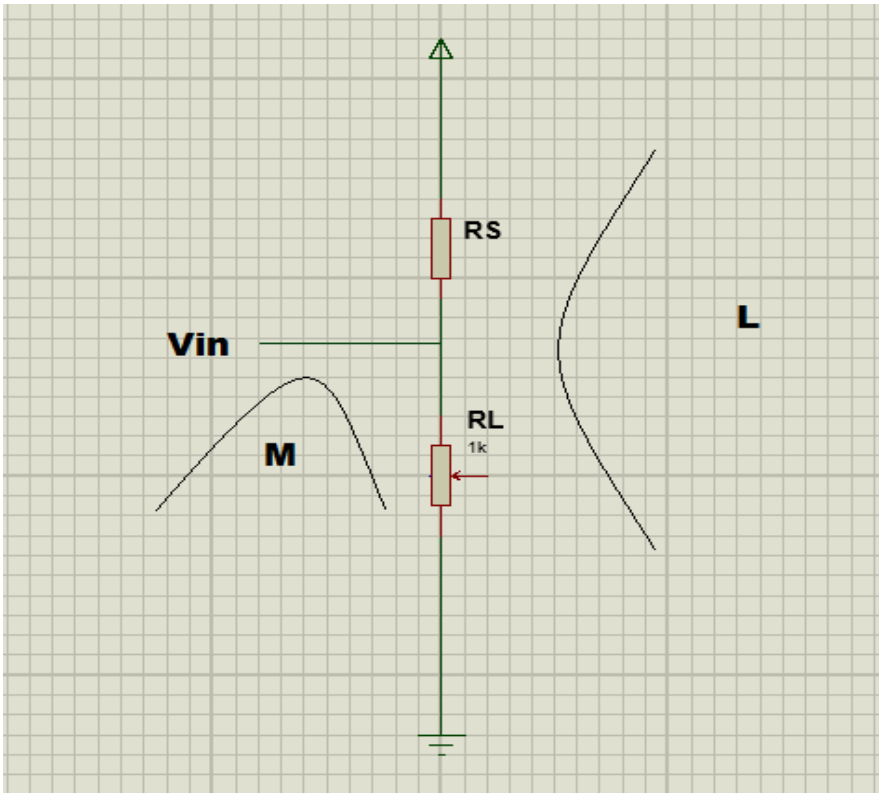


Fig 3.20 Circuit analysis

Equations

L)

*-5+I * Rs + I = 0* *equation (2.1)*

M)

-Vin + I = 0 *equation (2.2)*

I = Vin *equation (2.3)*

From 2.3 into 2.1

*-5 + Vin * Rs + Vin = 0* *equation (2.4)*

Rs = 5 – Vin / Vin *equation (2.5)*

3.8. DHT11 HUMIDITY SENSOR

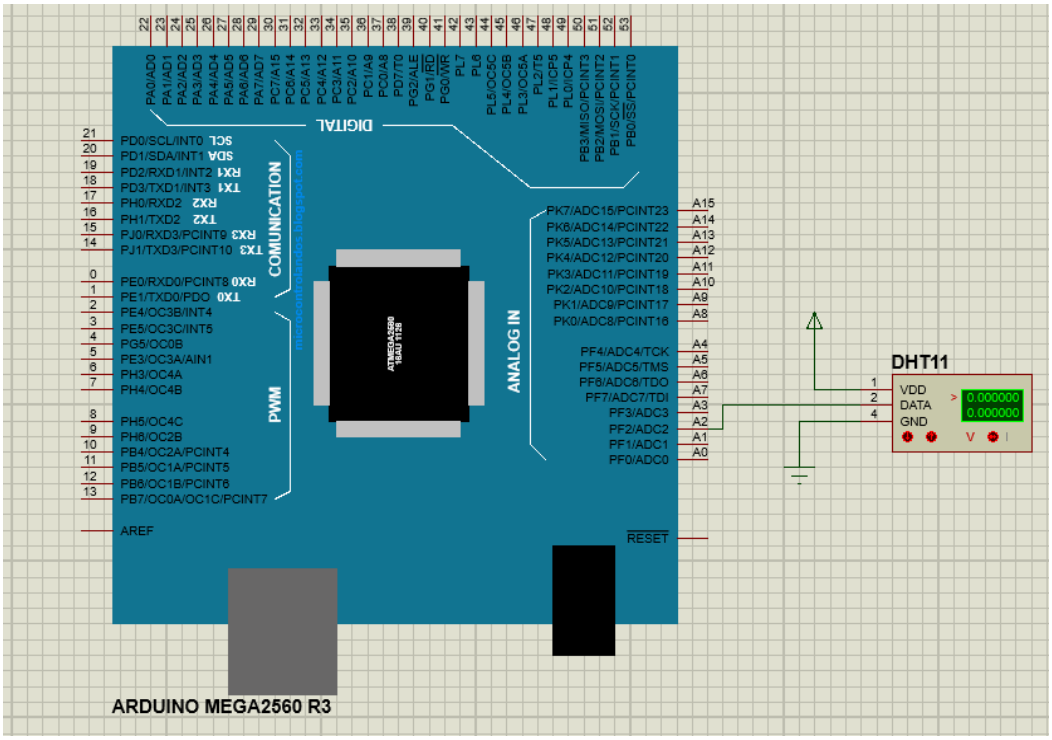


Fig 3.21 Circuit diagram of DHT11 humidity sensor

The DHT11 is a basic, low-cost temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out an analog signal on the data pin.

The DHT11 temperature sensor can be considered as a variable resistance that changes its value as the temperature changes.

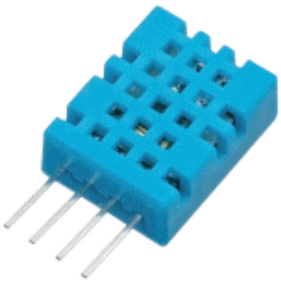
One of the most important specifications of the DHT11 is the sampling rate, in which it's 1Hz or one reading every second, and also the DHT11 has a small body size. The operating voltage of such sensors is from 3 to 5.5volts, while the maximum current used when measuring is 2.5mA.

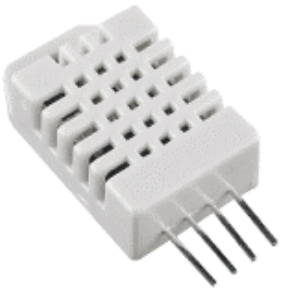
Each DHT11 sensor is strictly calibrated in the laboratory that is extremely accurate on temperature and humidity calibrations. The calibration coefficients are stored as programs in the OTP memory, which are used by the sensor's internal signal detecting process. The single-wire serial interface makes system integration quick and easy. Its small size, low power consumption and up-to-20 meter signal transmission making it the best choice for various applications, including those most demanding ones. The component is 4-pin single row pin package.

The sensors DHT11 and DHT22 are very popular for electronics because they are very cheap and still providing great performance. Here are the main specifications and differences between these two sensors:

The DHT22 is a more expensive version than the DHT11 which obviously has better specifications. Its temperature measuring range is from -40 to +125 degrees Celsius with +-0.5 degrees accuracy, while the DHT11 temperature range is from 0 to 50 degrees Celsius with +-2 degrees accuracy. Also the DHT22 sensor has better humidity measuring range, from 0 to 100% with 2-5% accuracy, while the DHT11 humidity range is from 20 to 80% with 5% accuracy.

Table3-2 DHT11 and DHT22 sensors





DHT11		DHT22
0 - 50°C / ± 2°C	*Temperature Range*	-40 - 125 °C / ± 0.5 °C
20 - 80% / ± 5%	*Humidity Range*	0 - 100 % / ± 2-5%
1Hz (one reading every second)	*Sampling Rate*	0.5 Hz (one reading every two seconds)
15.5mm x 12mm x 5.5mm	*Body Size*	15.1mm x 25mm x 7.7mm
3 - 5V	*Operating Voltage*	3 - 5V
2.5mA	*Max Current During Measuring*	2.5mA

There are two specifications where the DHT11 is better than the DHT22. That's the sampling rate which for the DHT11 is 1Hz or one reading every second, while the DHT22 sampling rate is 0.5Hz or one reading every two seconds and also the DHT11 has smaller body size. The operating voltage of both sensors is from 3 to 5 volts, while the max current used when measuring is 2.5mA.

They consist of a humidity sensing component, a NTC temperature sensor (or thermistor) and an IC on the back side of the sensor.

The specifications of the DHT11 humidity sensor are as follows:-

- Operating voltage is ranging from +3v and +5.5v (+5v typically).
- Sampling rate is 1Hz/sec.
- Low power consumption.
- Long transmission distance & long time stability
- Measuring range for the humidity is 20-90% and for the temperature is 0-50°C, where:
  - The temperature accuracy = 2°C.
  - The humidity accuracy = 5°C.

Pins configurations are as follows:-

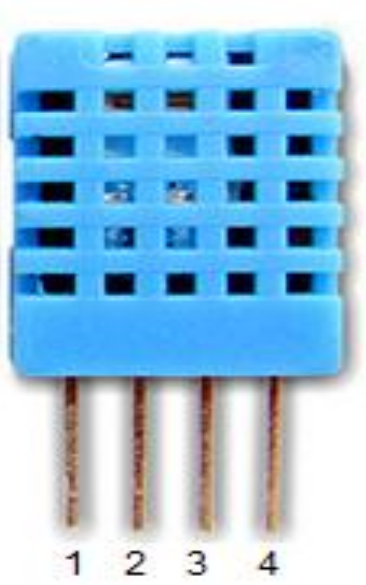


Fig.3.22 DHT11 sensor pins configurations

1- Vcc (+5v typically)
2- Data
3- Not connected (NC)
4- Ground (GND)

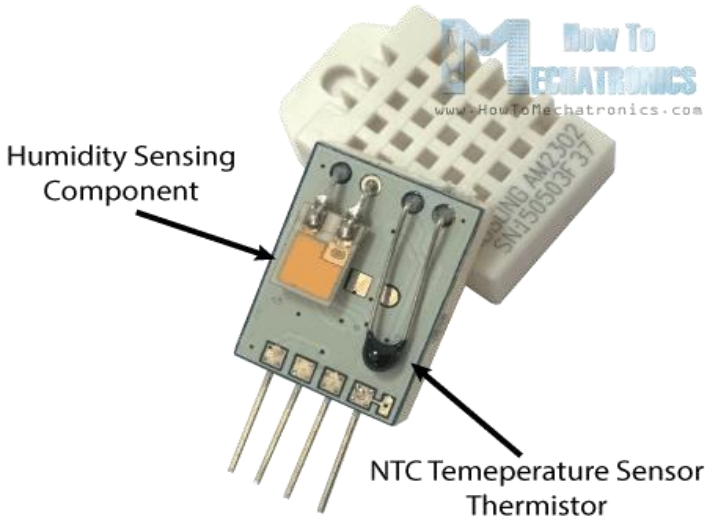


Fig 3.23 Internal components

For measuring humidity they use the humidity sensing component which has two electrodes with moisture holding substrate between them. So as the humidity changes, the conductivity of the substrate changes or the resistance between these electrodes changes. This change in resistance is measured and processed by the IC which makes it ready to be read by a microcontroller.

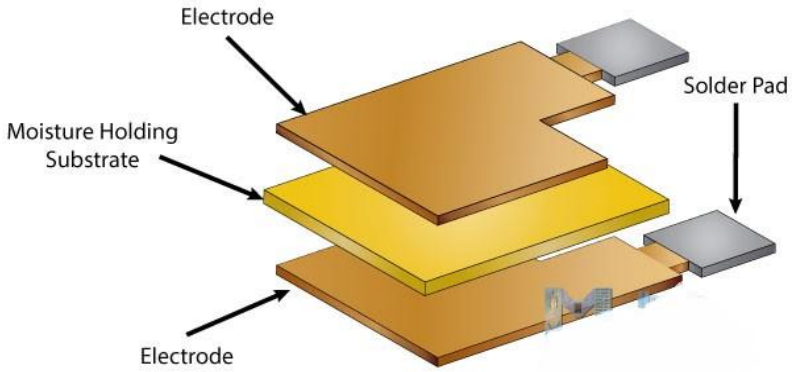


Fig 3.24 Layers

On the other hand, for measuring temperature these sensors use a NTC temperature sensor or a thermistor. A thermistor is actually a variable resistor that changes its resistance with change of the temperature. These sensors are made by sintering of semi conductive materials such as ceramics or polymers in order to provide larger changes in the resistance with just small changes in temperature. The term “NTC” means “Negative Temperature Coefficient”, which means that the resistance decreases with increase of the temperature.

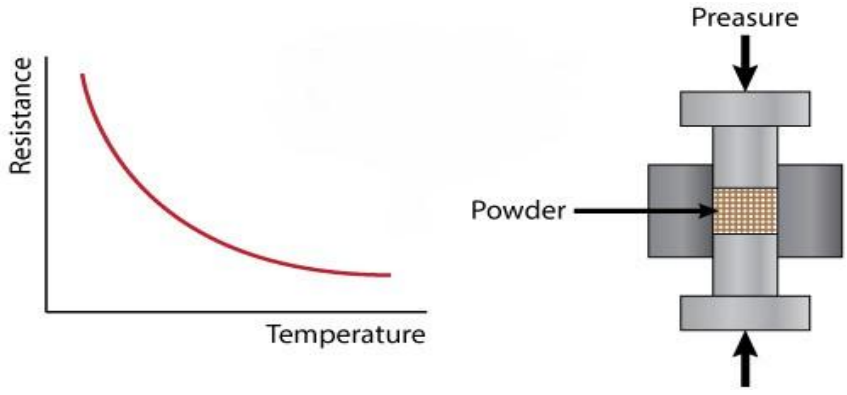


Fig 3.25 Graph

3.9. LM35 TEMPERATURE SENSOR

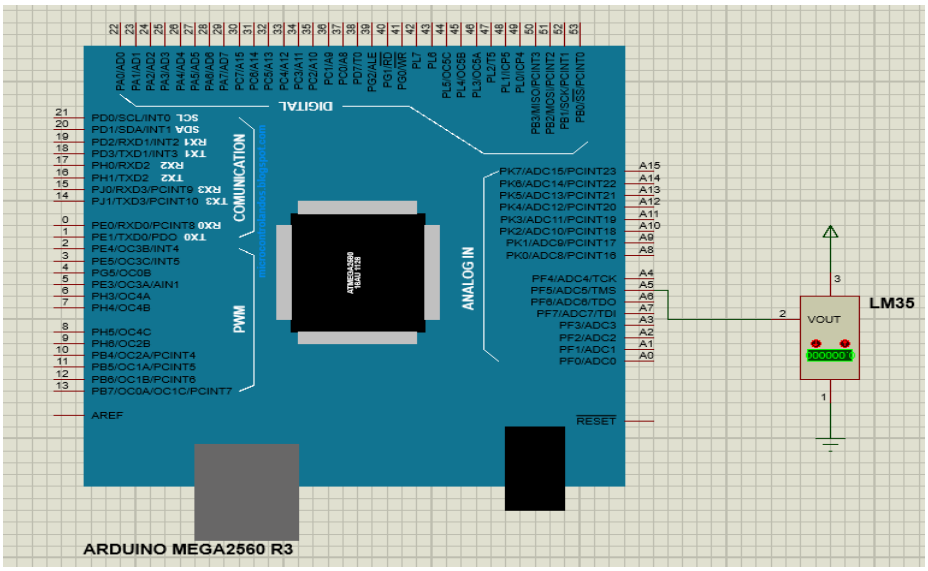


Fig 3.26 Circuit diagram of LM35 temperature sensor

LM35 is an analog, linear temperature sensor whose output voltage varies linearly with change in temperature. LM35 is three terminal linear temperature sensor from National semiconductors. It can measure temperature from **-55 degree Celsius to +150 degree Celsius.** The voltage output of the LM35 increases 10mV per degree Celsius rise in temperature. LM35 can be operated from a 5V supply and the stand by current is less than 60uA.

Fig 3.27 LM35

LM35 is an analog temperature sensor. This means that the output of LM35 is an analog signal. Microcontrollers don't accept analog signals as their input directly. We need to convert this analog output signal to digital before we can feed it to a microcontroller's input. For this purpose, we can use an ADC (Analog to Digital Converter). If we are using a basic microcontroller like 8051, we need to use an external ADC to convert analog output from LM35 to digital. We then feed the output of ADC (converted digital value) to input of 8051. But modern day boards like Arduino and most modern day micro controllers come with inbuilt ADC. Our Arduino Uno has an in built 10 bit ADC (6 channel).

We can make use of this in built ADC of Arduino to convert the analog output of LM35 to digital output. Since Arduino Uno has a 6 channel inbuilt ADC, there are 6 analog input pins numbered from A0 to A5. Connect analog out of LM35 to any of these analog input pins of Arduino.

Connect LM35 to Arduino Uno as shown in circuit diagram. The +5v for LM35 can be taken from the +5v out pin of Arduino Uno. Also the ground pin of LM35 can be connected to GND pin of Arduino Uno. Connect Vout (the analog out of LM35) to any of the analog input pin of Arduino Uno. In this circuit diagram, we have connected Vout of LM35 to A1 of Arduino.

LM35 is available in the market in 3 series variations – LM35A, LM35C and LM35D series. The main differences between these 3 versions of LM35 IC are in their range of temperature measurements. The LM35D series is designed to measure from 0 degree Celsius to 100 degree Celsius, whereas the LM35A series is designed to measure a wider range of -55 degree Celsius to 155 degree Celsius. The LM35C series is designed to measure from -40 degree Celsius to 110 degree Celsius.

3.10. IR SENSOR

The IR Sensor (or Infra-Red sensor) is a general purpose digital sensor. Here we use it for collision detection. The module consists of an IR emitter and IR receiver pair. The high precision IR receiver always detects an IR signal.

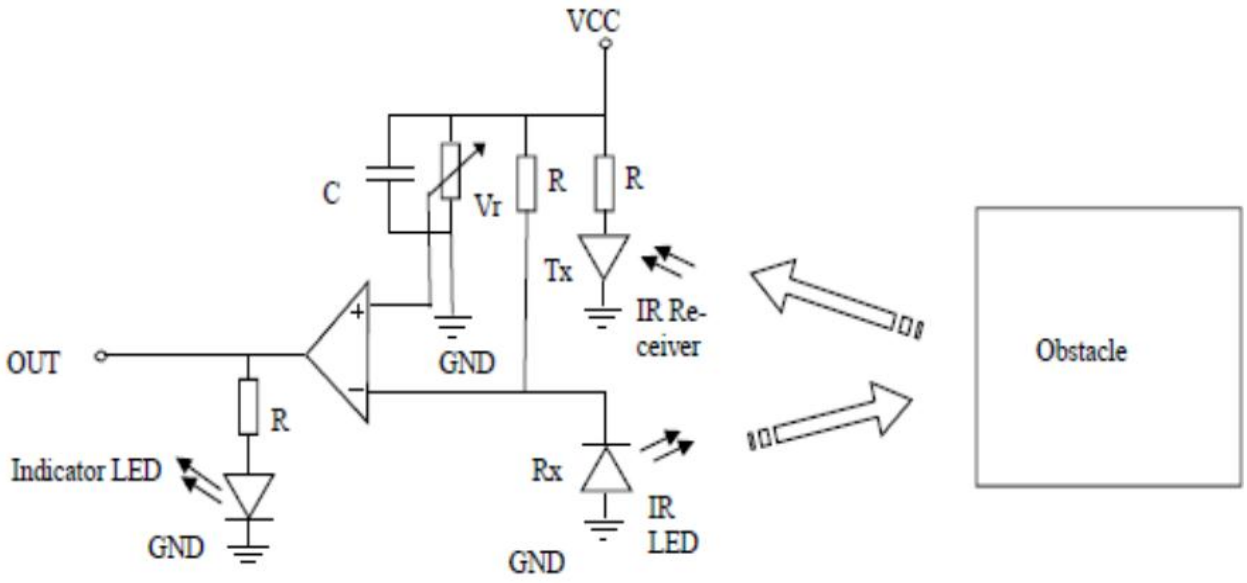


Fig.3.28 IR sensor circuit diagram

The module consists of a 358 comparator IC. The output of sensor is high whenever it receives IR frequency and low otherwise. The on-board LED indicator helps user to check status of the sensor without using any additional hardware. The power consumption of this module is low as it gives a digital output.

The sensitivity of the IR Sensor is tuned using the potentiometer. The potentiometer is tuneable (or changeable) in both directions. Initially tune the potentiometer in clockwise direction such that the Indicator LED starts glowing. Once that is achieved, turn the potentiometer just enough in anti-clockwise direction to turn off the Indicator LED. At this point the sensitivity of the receiver is maximum. Thus, its sensing distance is maximum at this point. If the sensing distance (i.e., Sensitivity) of the receiver is needed to be reduced, then one can tune the potentiometer in the anti-clockwise direction from this point.

Further, if the orientation of both Tx and Rx LED's is parallel to each other, such that both are facing outwards, then their sensitivity is maximum. If they are moved away from each other, such that they are inclined to each other at their soldered end, then their sensitivity reduces.

Tuned sensitivity of the sensors is limited to the surroundings. Once tuned for a particular surrounding, they will work perfectly until the IR illumination conditions of that region nearly constant. For example, if the potentiometer is tuned inside room/building for maximum sensitivity and then taken out in open sunlight, it will require retuning, since sun's rays also contain Infrared (IR) frequencies, thus acting as an IR source (transmitter). This will disturb the receiver's sensing capacity. Hence it needs to be retuned (or changed) to work perfectly in the new surroundings.

The output of IR receiver goes low when it receives IR signal. Hence the output pin is normally low because, though the IR LED is continuously transmitting, due to no obstacle, nothing is reflected back to the IR receiver. The indication LED is off. However, when an obstacle is encountered, the output of IR receiver goes high as the IR signal is reflected from the obstacle surface. So this drives the output of the comparator high, so this turns the indication LED on.

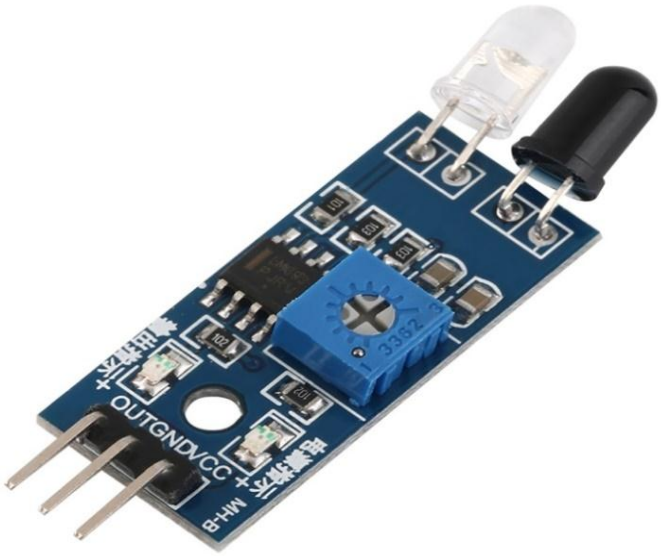


Fig.3.29 IR sensor

Table3-3 IR sensor Pins Configurations (from left to right)

Pin No.	Connection	Description
1	Output	Digital Output (High or Low)
2	Ground	Connected to circuit ground
3	VCC	Connected to the circuit supply

3.11. LDR Photocell

The Light Dependent Resistors (LDRs) or photocells are sensors that allow you to detect light. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they often appear in toys, gadgets and many other appliances. They are often referred to as CdS photoconductive cells (as they are made of Cadmium-Sulfide). Such light dependent resistors are also called photoresistors and they come in many sizes and specifications.

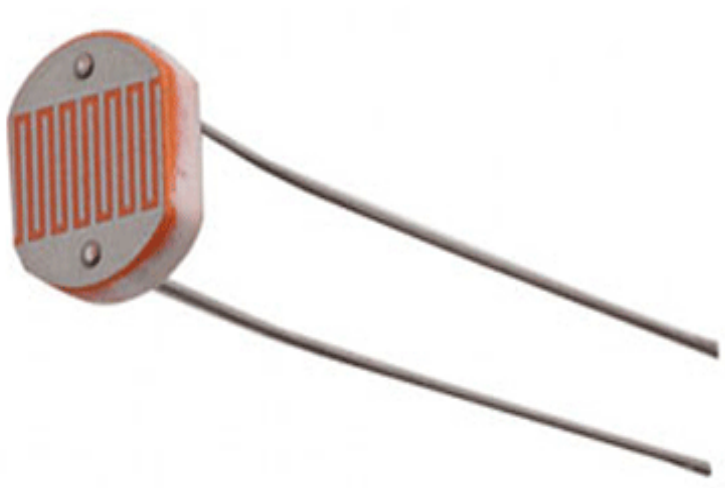


Fig.3.30 LDR

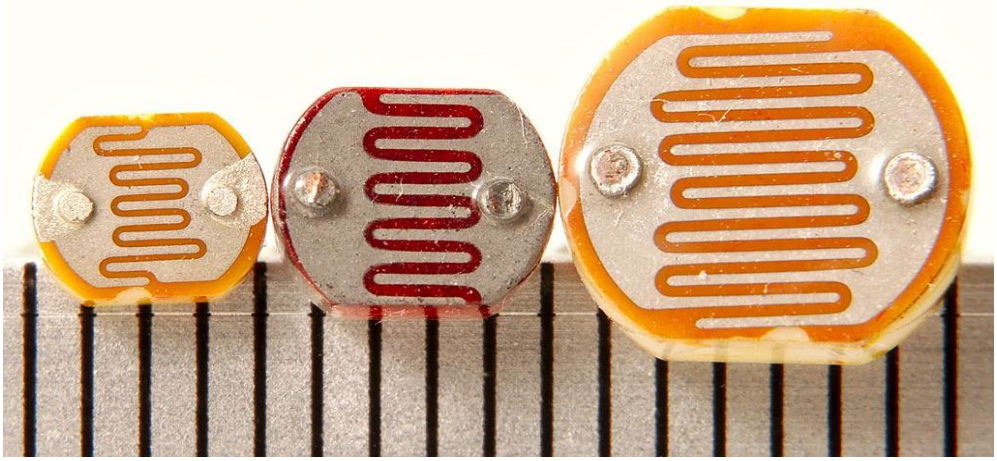


Fig.3.31 Different sizes LDRs

Such light dependent resistor (LDR) or photocell is basically a resistor that changes its resistive value (in ohms) depending on how much light is shining onto the squiggly face.

The easiest way to measure a resistive sensor is to connect one end to Power and the other end to a pull-down resistor to ground. Then the point between the fixed pull down resistor (that could have any value e.g.10kΩ) and the variable photocell resistor (LDR) is connected to the analog input of a microcontroller such as an Arduino.

Because photocells are basically resistors, they are non-polarized. That means you can connect them in any direction and they'll work just fine!.

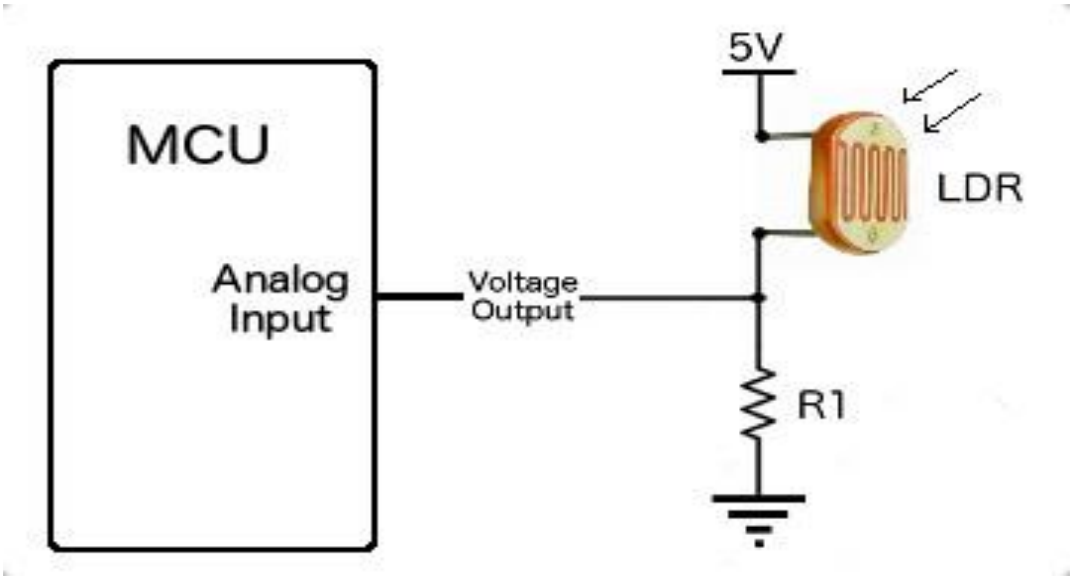


Fig.3.32 LDR circuit diagram

Principle of operation:-

Such LDRs are made of high resistance semiconductor material. When light hits the device, the photons give electrons energy. This makes them jump into the conductive band and thereby conduct electricity.

A photoresistor is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several mega-ohms (MΩ), while in the light, that photoresistor can have a resistance as low as a few ohms. If the incident light on a photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band.

Therefore, the resulting free electrons (and their hole-partners) conduct electricity, thereby lowering resistance.

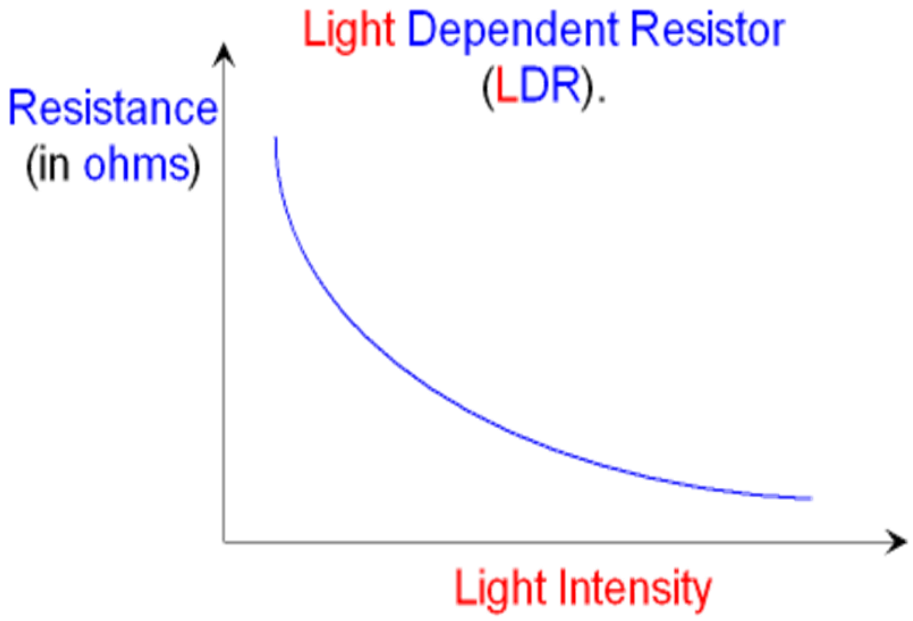


Fig.3.33 Relation between the light intensity and resistance

The sensitivity of a photodetector is the relationship between the light falling on the device and the resulting output signal. Like the human eye, the relative sensitivity of a photoconductive cell is dependent on the wavelength (color) of the incident light. Each photoconductor material type has its own unique spectral response curve or plot.

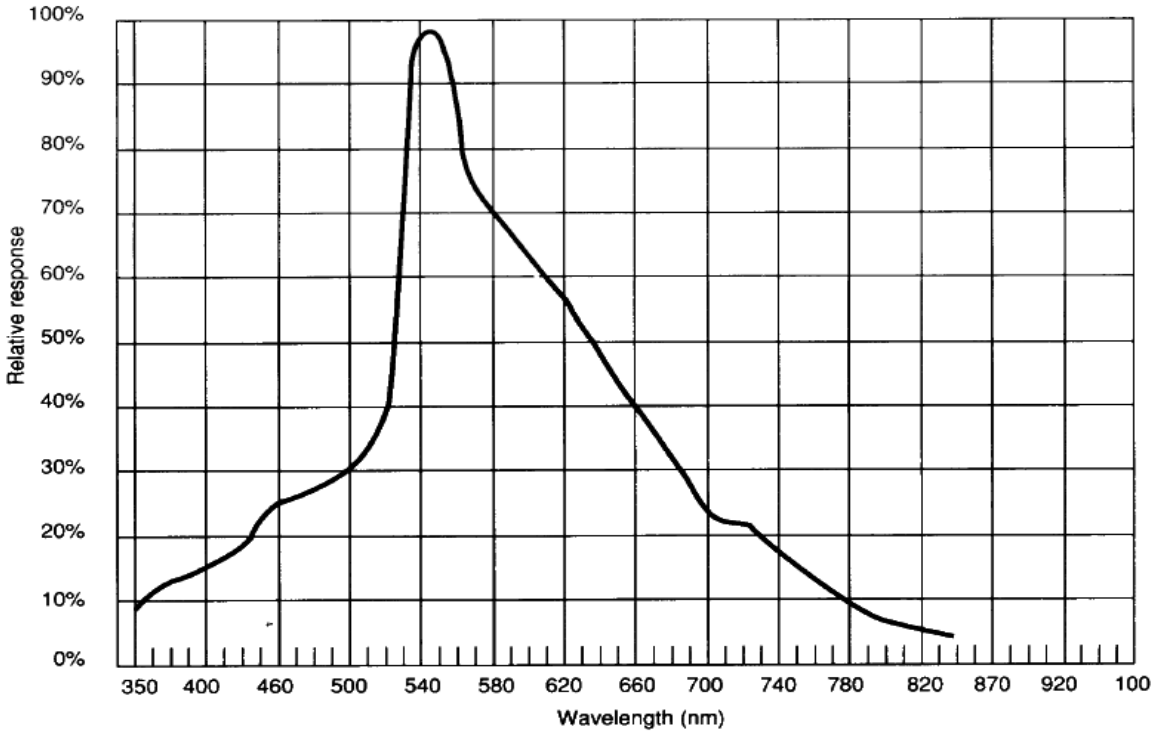


Fig.3.34 Spectral response of LDR

3.12. SERVO MOTOR SG-90

The Servo motor is tiny in size and lightweight with high output power. It can rotate approximately 180 degrees (90° in each direction) and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control such a servo motor.

Servo motors are a combination of motor, gearbox and sensor that are often found in remote-controlled applications in order to control rotation process.

The servo motor comes with a 3 horns (arms) and hardware and it will fit in small places and it works in a closed-loop mechanism that uses position feedback to control its motion and final position and the operation of how it works to turn into different positions will be explained below.

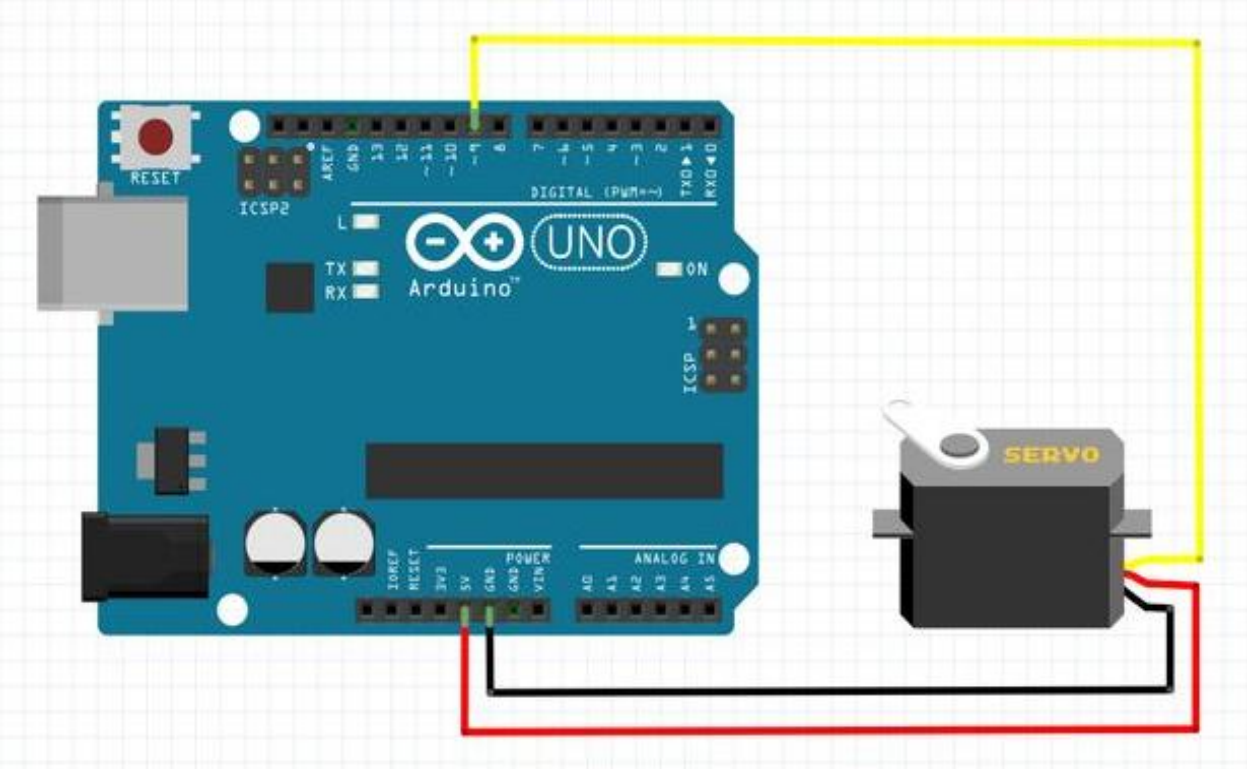


Fig.3.35 Circuit diagram of Servo motor SG-90

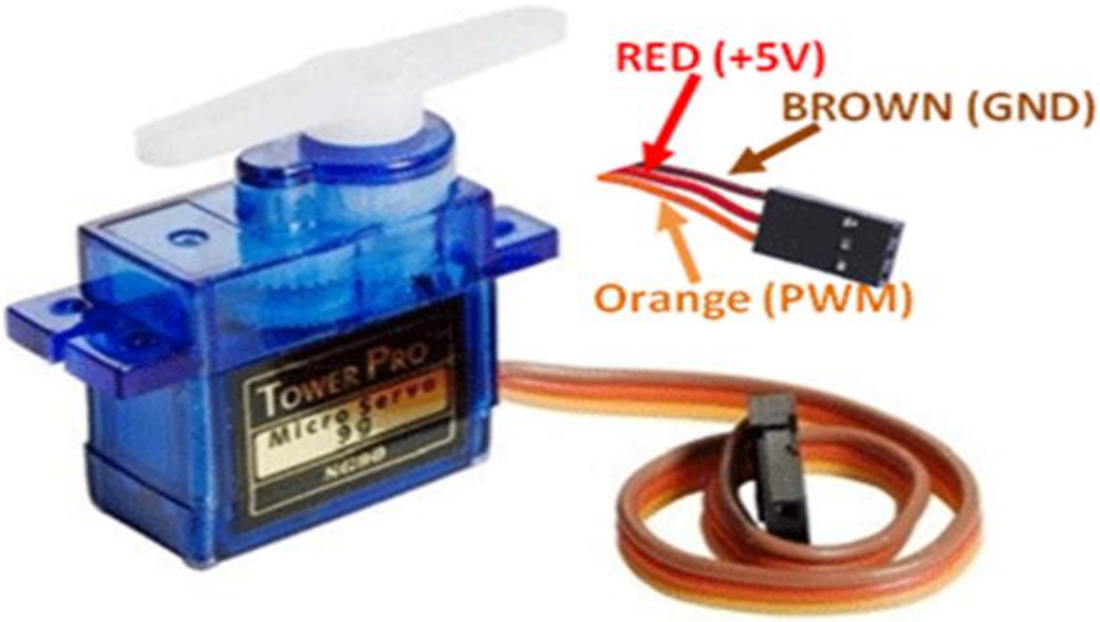


Fig.3.36 Servo motor SG-90

Table3-4 Servo motor SG-90 wires configurations

Wire number	Wire color	Description
1	Brown	Ground wire connected to the ground of system
2	Red	It's connected to the power supply typically (+5v)
3	Orange	A PWM signal is given in through this wire to control the motor

Principle of operation:-

The main idea of how the servo motor works is that it consists of an axe and as this axe rotates the value of resistance changes then, the servo motor changes that resistance value into a specific angle.

As we know there exist three wires coming out of this motor. To make this motor rotate, we have to power the motor with +5V using the Red and Brown wire then send a PWM signal to the Orange color wire. Hence we need something that could generate PWM signals in order to make this motor work, we can use like a 555 Timer or other Microcontroller platforms like Arduino, PIC, ARM.

By moving into the next point, in order to know how to control the direction of motion of the motor we can say that:-

The PWM signal produced should have a frequency of 50Hz as the PWM period should be 20ms. In which the On-Time (time of duty cycle) can vary from 1ms to 2ms. So when the on-time is 1ms the position of motor will be in 0° and when 1.5ms the motor will be 90°, similarly when it is 2ms it will be 180°. So, by varying the on-time from 1ms to 2ms the motor can be controlled from 0° to 180°.

(e.g. in order to make the position of motor is in 45° we should make the time of duty cycle equal to 1.25ms).

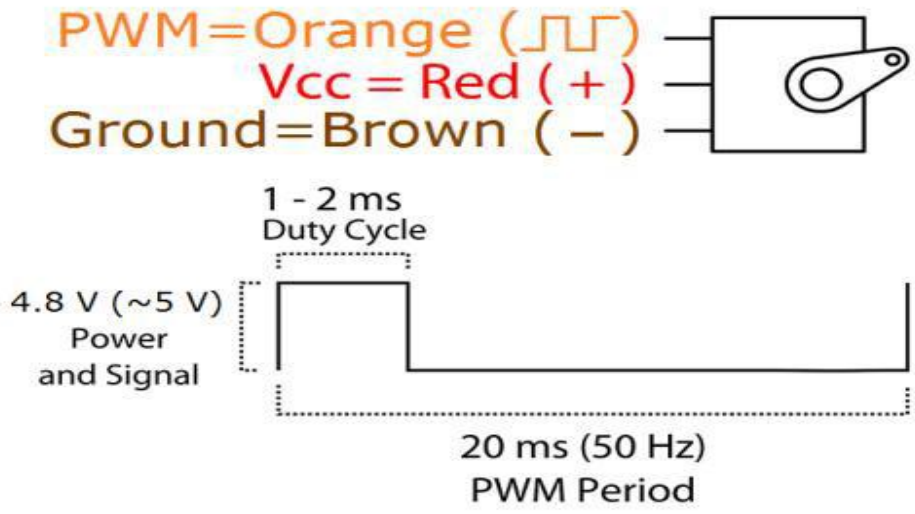


Fig.3.37 PWM period graph

Servo motor SG-90 features:-

- Operating voltage ranges between +4.8v and 6v (typically +5v).
- Rotation between 0° and 180°.
- Operating speed is 0.1s/60°.
- Torque is 2.5kg/cm
- Weight of motor is nearly between 9 and 14.7gms.

Applications of servo motor

- Used as in many robots where position control is required without feedback like robotic arm.
- Commonly used for steering system in RC toys.

3.13. SEVEN SEGMENT DISPLAY (COMMON CATHODE)

A 7-segment display is a form of the efficient electronic LED display devices used in embedded applications for displaying decimal numbers. This display consists of 7 LEDs inside it, in which they are separated into segments, where each can be named as a, b, c, d, e, f and g.

There exist many different sizes and colors to select from. The default and most commonly used one is the 14.20mm with red color display with a black or grey background color.

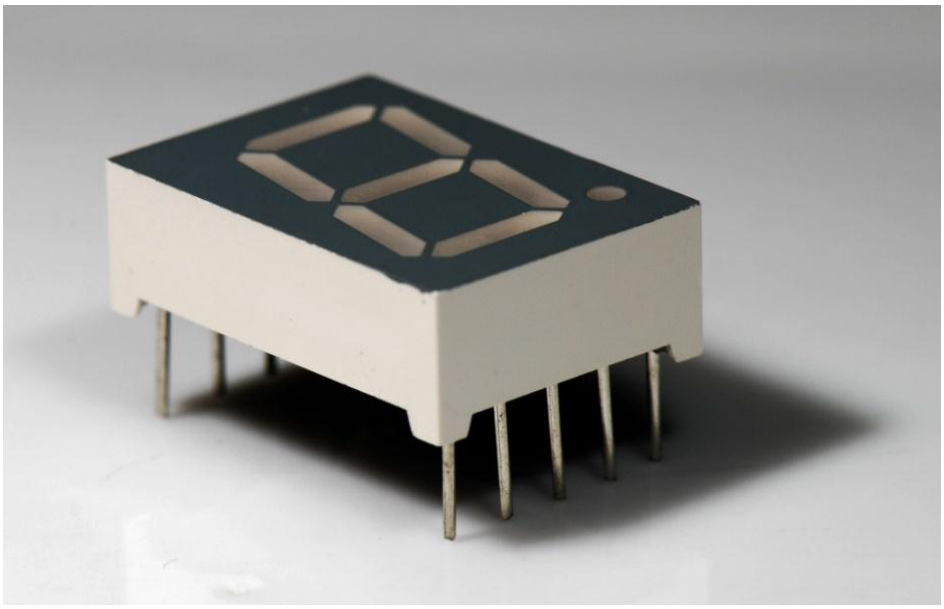


Fig.3.38 Seven segment 14.20mm common cathode

Such 7-segment used consists of 10 pins numbered and named as shown below where: the com pin refers to common cathode or common anode pin (where the Common Cathode is being used) and DP is short for decimal point and it must be made enabled (equal to logic 1) in order to make the numbers be displayed.

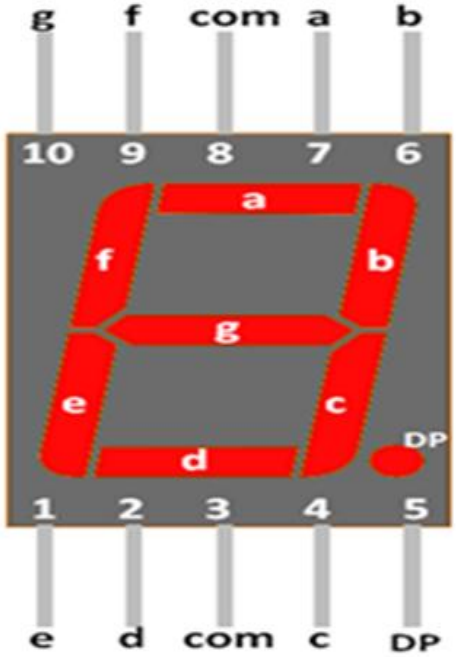


Fig.3.39 Pin diagram

In a simple LED package, typically all of the cathodes (negative terminals) or all of the anodes (positive terminals) of the LEDs segments, are connected and brought out to a common pin; this is referred to as a "common cathode (CC)" or "common anode (CA)" device.

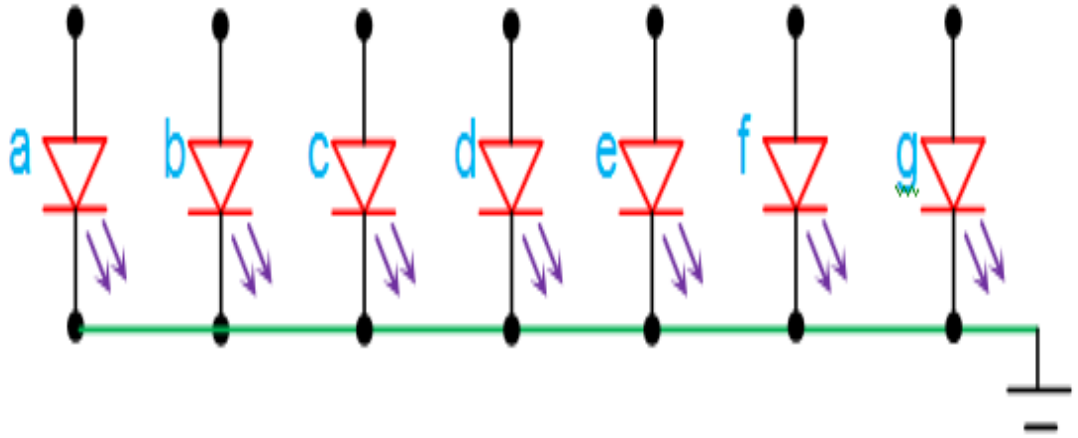


Fig.3.40 Internal circuit of 7-Segment

The 7-segment has the following specifications:-

- Voltage drop per LED segment is about 2.2V (typical)
- Low current operation
- Maximum current consumption: 30mA/segment
- Bright and clear color & font illumination
- Red color display but also there exist: white, blue, yellow and green
- Background color is black or grey
- It could be used in small circuit projects and also it’s compatible with microcontrollers & microprocessors.

The mostly common 7-segment display is used along with a MCU/MPU is the 14.20mm and in that case, the eight segment pins will be connected to the I/O pins of the Microcontroller and the com pin will be connected to the ground or the Vcc depending upon its type (CC/CA). Then these I/O pins can be toggled in a particular sequence to display the desired numbers, in which it could be explained through the table below.

For displaying each number in the seven segment display, its respective sequence is given in the table. For example - if we want to display the number “0” then, we need to glow all the LEDs except LED which belongs to line “g” (see 7 segment pin diagram above), so we need a bit pattern 10111111. Similarly to display “1”we need to glow LEDs associated with b and c, so the bit pattern for this would be 10000110.

3.13.1. DIGIT TO DISPLAY

Table3-5 Seven segment common cathode digit to display

Digit to display	DP	g	f	e	d	c	b	a
0	1	0	1	1	1	1	1	1
1	1	0	0	0	0	1	1	0
2	1	1	0	1	1	0	1	1
3	1	1	0	0	1	1	1	1
4	1	1	1	0	0	1	1	0
5	1	1	1	0	1	1	0	1
6	1	1	1	1	1	1	0	1
7	1	0	0	0	0	1	1	1
8	1	1	1	1	1	1	1	1
9	1	1	1	0	1	1	1	1

Notice that:-

The table is applicable only for Common Cathode type display, if it is a common anode type then simply replace the ‘1’s with ‘0’s and vice versa.

3.13.2. PROBLEM FACING THE OPERATION

In order to make the 7-segment work, we need 8 pins and the Arduino UNO contains 14 pins maximum. While the Arduino mega contains 56 pins maximum, so in both cases we are going to use a large number of pins (8 pins) and actually that’s a kind of problem especially, we are gonna use more applications with the Arduino board.

(Actually 14 pins but the common ground (or positive supply) pins are connected together so the number of pins decreased to 8 pins only).

So in order to solve such problem, we intended to use a 8-bit shift register which in turns, will decrease the number of pins that will make the 7-segment work and it will be explained in detail in the topic below.

3.14. SN74HC595 8-BIT SHIFT REGISTER IC

As explained before, the 7-segment needs a number of 8 pins in order to operate. However, we will use a-8-bit shift register IC with only 3 output pins that will solve such problem.

The SN74HC595 is a simple 8-bit shift register IC. Such device allows additional inputs or outputs to be added to a microcontroller by converting data between parallel and serial formats.

The ’HC595’ devices contain a 8-bit serial-in, parallel-out shift register (which means that we will have a single data input (serial input) and multiple outputs) that feeds a 8-bit D-type storage register.

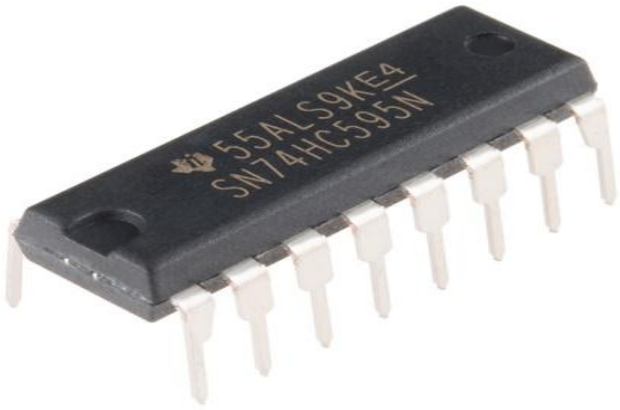


Fig.3.41 SN74HC595 shift register IC

The storage register has parallel 3-state outputs. Separate clocks are provided for both the shift and storage registers. The shift register has a direct overriding clear (SRCLR) input, serial (SER) input, and serial outputs for cascading. When the output-enable (OE) input is high, the outputs are also in the high-impedance state.

Pins configurations are as follows (from the datasheet):-

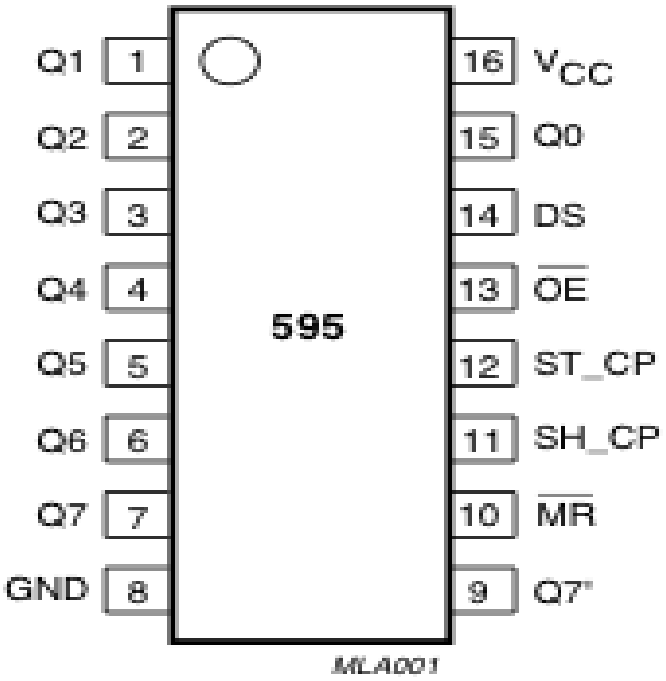


Fig.3.42 Top view of the SN74HC595 shift register IC

Pins 1-7, 15 (Q1-Q7, Q0) ⟶ output pins

Pin 8 (GND) ⟶ ground

Pin 9 (Q7) ⟶ serial out

Pin 10 (MR) ⟶ master reclear, active low

Pin 11 (SH_CP) ⟶ shift register clock pin

Pin 12 (ST_CP) ⟶ storage register clock pin (latch pin)

Pin 13 (OE) ⟶ output enable, active low

Pin 14 (DS) ⟶ serial data input

Pin 16 (Vcc) ⟶ positive supply voltage

However, we will connect only three pins of the shift register to the output pins on the Arduino, and they are:

- Pin 11 (SH_CP - clock pin) ⟶ the shift pin helps to calculate the output configuration.
- Pin 12 (ST_CP - latch pin) ⟶ when the latch pin is low, the current 8-bit information will not change, but when it is high then, the new 8-bit information will be activated

- Pin 14 (DS - data input pin) ⟶ this pin loads the 8-bits into the shift register one by one.

While the pins on the left of the shift register (from Q1 to Q7 and also Q0 - pins 1-7 and 15) will be instead, connected to the output pins of the 7-segment as follows:

Shift-register pin		7-Segment pin
Q0	⟶	Pin a
Q1	⟶	Pin b
Q2	⟶	Pin c
Q3	⟶	Pin d
Q4	⟶	Pin e
Q5	⟶	Pin f
Q6	⟶	Pin g
Q7	⟶	Pin dp

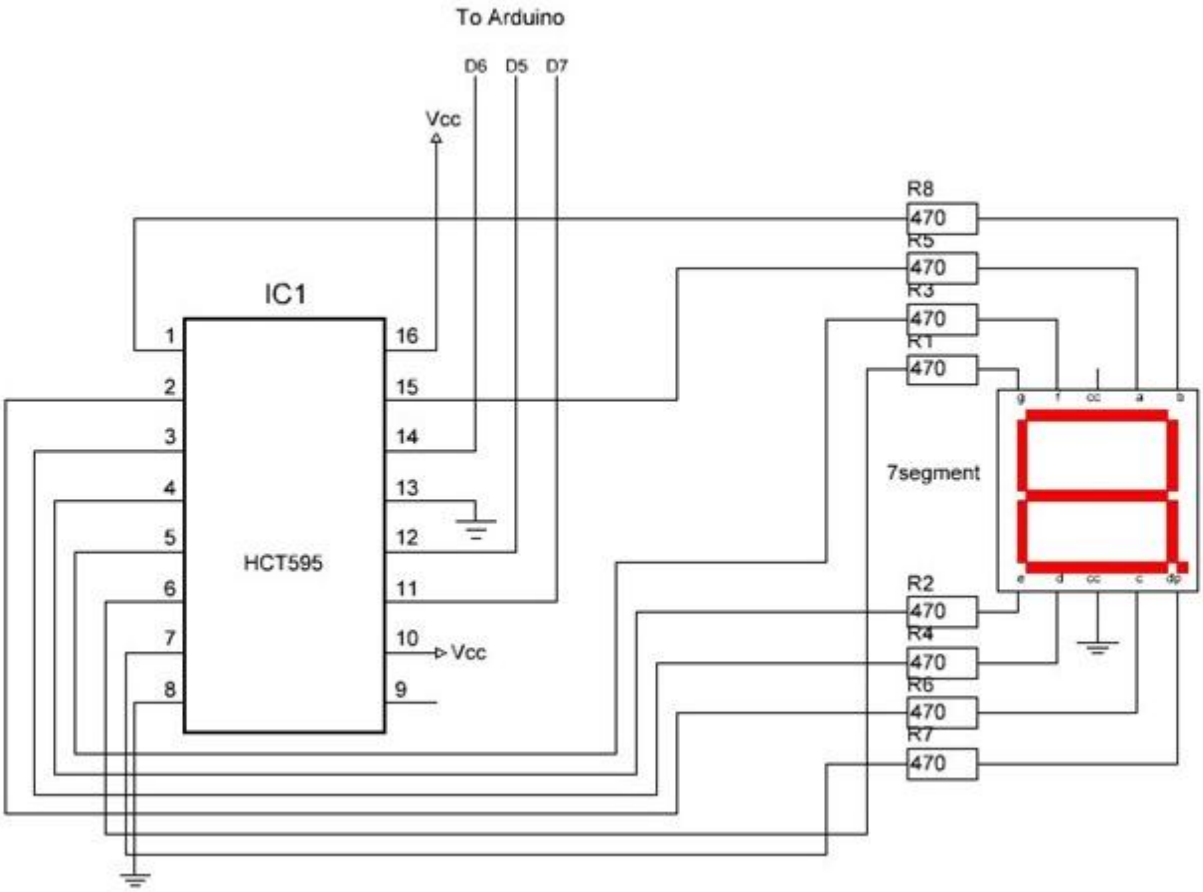


Fig.3.43 Circuit diagram

In our project we intended to use resistors of 220 ohms instead of 470 ohms in order to increase the intensity of the 7-segment LEDs and make them brighter and clearer.

Specifications of SN74HC595 8-bit shift register IC are as follows:-

- It's a 8-bit serial-in, parallel-out shift register
- Wide operating voltage ranging from +2V to +6V
- Clock frequency (for 2v-6v) ranges between 5 and 29 MHz (16 MHz typically)
- High-Current 3-State Outputs
- Low input current of 1 μA max
- Low power consumption (80-μA max)
- Typical period time (tpd) = 62.5 ns
- ±6mA output drive at 5V
- Operating temperature ranging from -40°C to +85°C.

3.15. RELAY MODULE

The 2-channel (+5v) relay module is an electrically operated switch that allows you to turn on or off a circuit using voltage and/or current much higher than a microcontroller could handle. There is no connection between the low voltage circuit operated by the microcontroller and the high power circuit. The relay protects each circuit from each other.

Each channel in the module has three connections named NC (or normally closed), COM, and NO (or normally opened) and each channel also needs a 15-20 mA driver current.

The 2-channel relay module is a general purpose module that used to control various appliances and other equipments with large current. It can be controlled directly with a +3.3v or +5v logic signals from a microcontroller like Arduino, AVR, PIC and so on.

Depending on the input signal trigger mode, the jumper cap can be placed at high level effective mode which 'closes' the normally opened (NO) switch at high level input and at low level effective mode which operates the same but at low level input

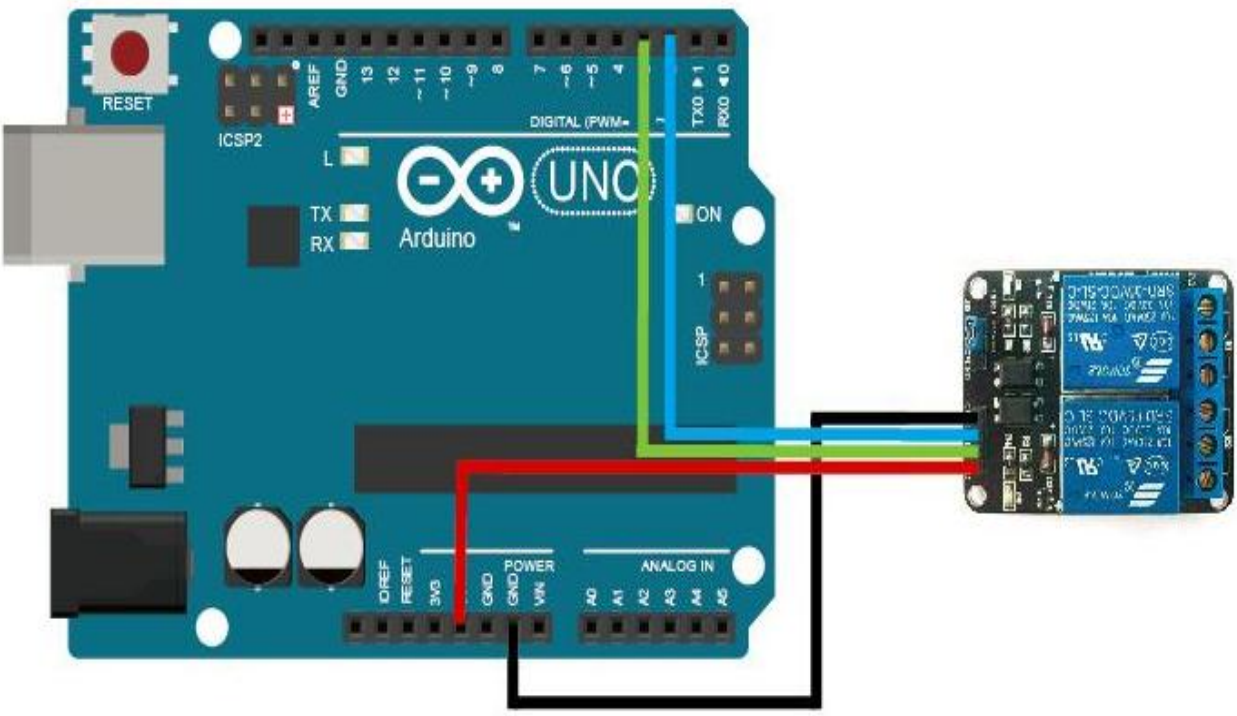


Fig.3.44 Circuit diagram of relay module

Fig.4.45 Relay module

Pins configurations are as follows (from left to right):-

1) VCC: 5V DC
2) COM: 5V DC
3) IN1: high/low output
4) IN2: high/low output
5) GND: ground

Specifications of the 2-channel relay module are as follows:-

- 2-channel relay module with low level trigger expansion board is compatible with Arduino.
- It has a standard interface that can be controlled directly by microcontrollers like Arduino, AVR, PIC, etc…
- Operating voltage works with logic signals from +3.3v or 5V devices, with max. output values: 10A / 30VDC, 10A / 250VAC.
- Module comes with diode current protection, short response time
- PCB Size: 45.8mm x 32.4mm.

3.16. LCD

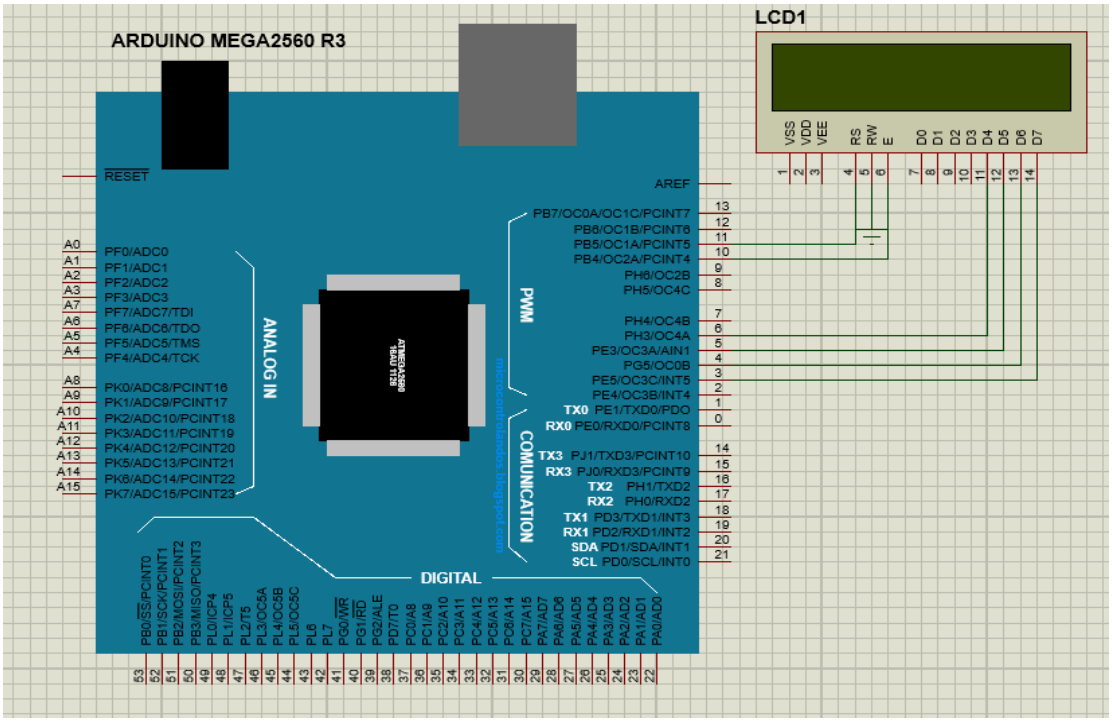


Fig 3.46 Circuit diagram of LCD

A **Liquid Crystal Display** commonly abbreviated as **LCD** is basically a display unit built using Liquid Crystal technology. When we build real life/real world electronics based projects, we need a medium/device to display output values and messages. The most basic form of electronic display available is 7 Segment display – which has its own

limitations. The next best available option is Liquid Crystal Displays which comes in different size specifications. Out of all available LCD modules in market, the most commonly used one is **16×2 LCD Module** which can display 32 ASCII characters in 2 lines (16 characters in 1 line). Other commonly used LCD displays are 20×4 Character LCD, Nokia 5110 LCD module, 128×64 Graphical LCD Display and 2.4 inch TFT Touch screen LCD display.

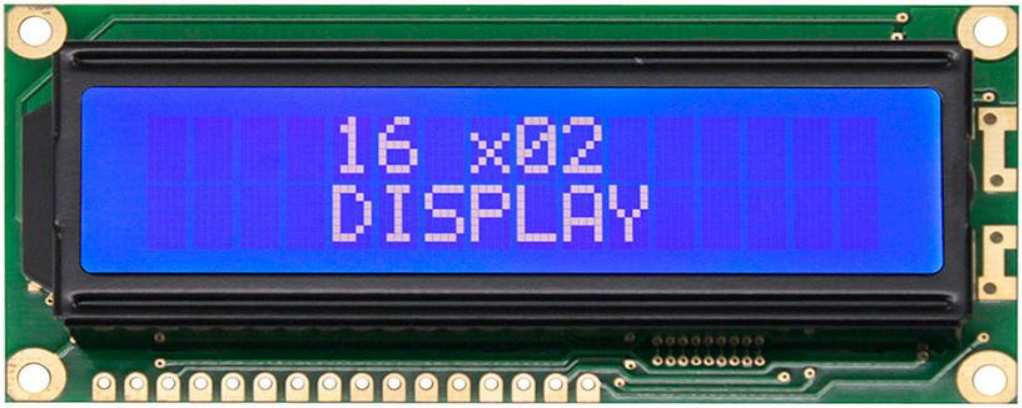


Fig 3.47 LCD

LCD modules form a very important part in many Arduino based embedded system designs. So the knowledge on interfacing LCD module to Arduino is very essential in designing embedded systems. This section of the article is about interfacing an Arduino to 16×2 LCD. JHD162A is the LCD module used here. JHD162A is a 16×2 LCD module based on the **HD44780 driver from Hitachi**. The JHD162A has 16 pins and can be operated in 4-bit mode (using only 4 data lines) or 8-bit mode (using all 8 data lines). Here we are using the LCD module in 4-bit mode.

The JHD162A lcd module has 16 pins and can be operated in 4-bit mode or 8-bit mode. Here we are using the LCD module in 4-bit mode. Before going in to the details of the project, let's have a look at the JHD162A LCD module. The schematic of a JHD162A LCD pin diagram is given below.

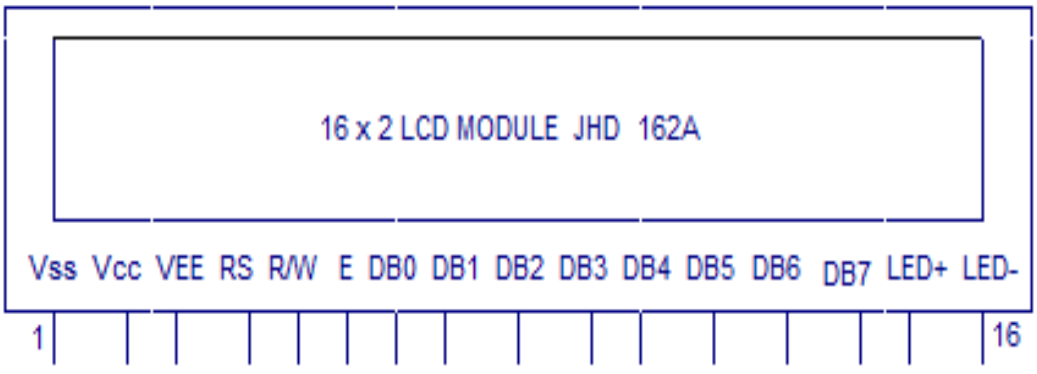


Fig 3.48 Pin diagram

The name and functions of each pin of the 16×2 LCD module is given below.

Pin1 (Vss): Ground pin of the LCD module.

Pin2 (Vcc): Power to LCD module (+5V supply is given to this pin)

Pin3 (VEE): Contrast adjustment pin. This is done by connecting the ends of a 10K potentiometer to +5V and ground and then connecting the slider pin to the VEE pin. The voltage at the VEE pin defines the contrast. The normal setting is between 0.4 and 0.9V.

Pin4 (RS): Register select pin. The JHD162A has two registers namely command register and data register. Logic HIGH at RS pin selects data register and logic LOW at RS pin selects command register. If we make the RS pin HIGH and feed an input to the data lines (DB0 to DB7), this input will be treated as data to display on LCD screen. If we make the RS pin LOW and feed an input to the data lines, then this will be treated as a command (a command to be written to LCD controller – like positioning cursor or clear screen or scroll).

Pin5 (R/W): Read/Write modes. This pin is used for selecting between read and write modes. Logic HIGH at this pin activates read mode and logic LOW at this pin activates write mode.

Pin6 (E): This pin is meant for enabling the LCD module. A HIGH to LOW signal at this pin will enable the module.

Pin7 (DB0) to Pin14 (DB7): These are data pins. The commands and data are fed to the LCD module though these pins.

Pin15 (LED+): Anode of the back light LED. When operated on 5V, a 560 ohm resistor should be connected in series to this pin. In Arduino based projects the back light LED can be powered from the 3.3V source on the Arduino board.

Pin16 (LED-): Cathode of the back light LED.

3.17. BLUETOOTH MODULE

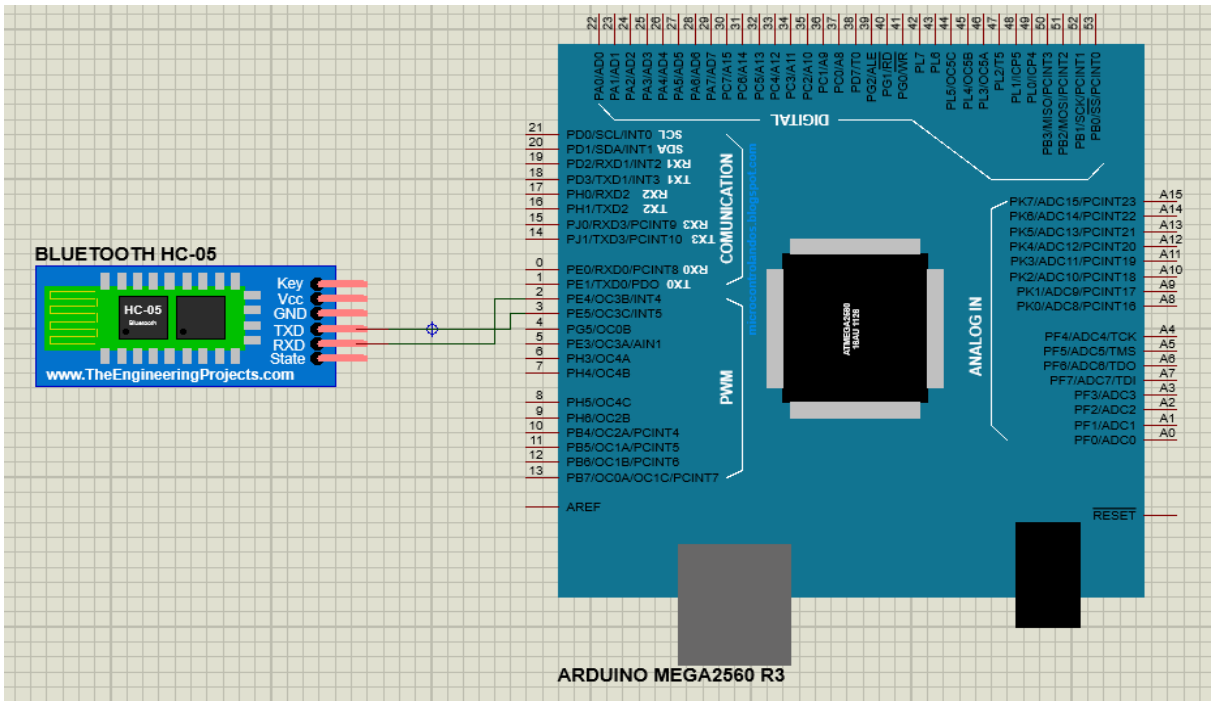


Fig 3.49 Circuit diagram of Bluetooth module

The circuit is so simple and small, there is only few connection to be made

Arduino Pins Bluetooth Pins

RX (Pin 0) ————-> TX

TX (Pin 1) ————-> RX

5V ————-> VCC

GND ————-> GND

HC 05/06 works on serial communication. Here the android app is designed sending serial data to the Bluetooth module when certain button is pressed.

The Bluetooth module at the other end receives the data and send to Arduino through the TX pin of Bluetooth module (RX pin of Arduino). The Code fed to Arduino check the received data and compares.

We connect the Bluetooth module with any android mobile with application named “Bluetooth RC controller” it has all controls that can move the car and it works by sending characters to module and Arduino receives it and take action according to the code.

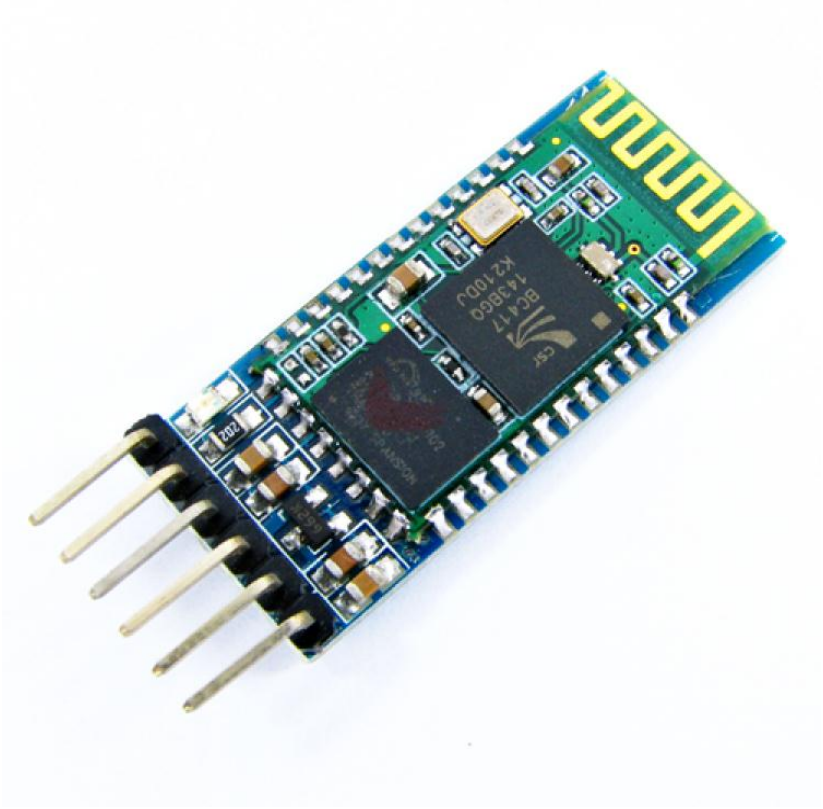


Fig 3.50 Bluetooth module

3.18. BUZZER

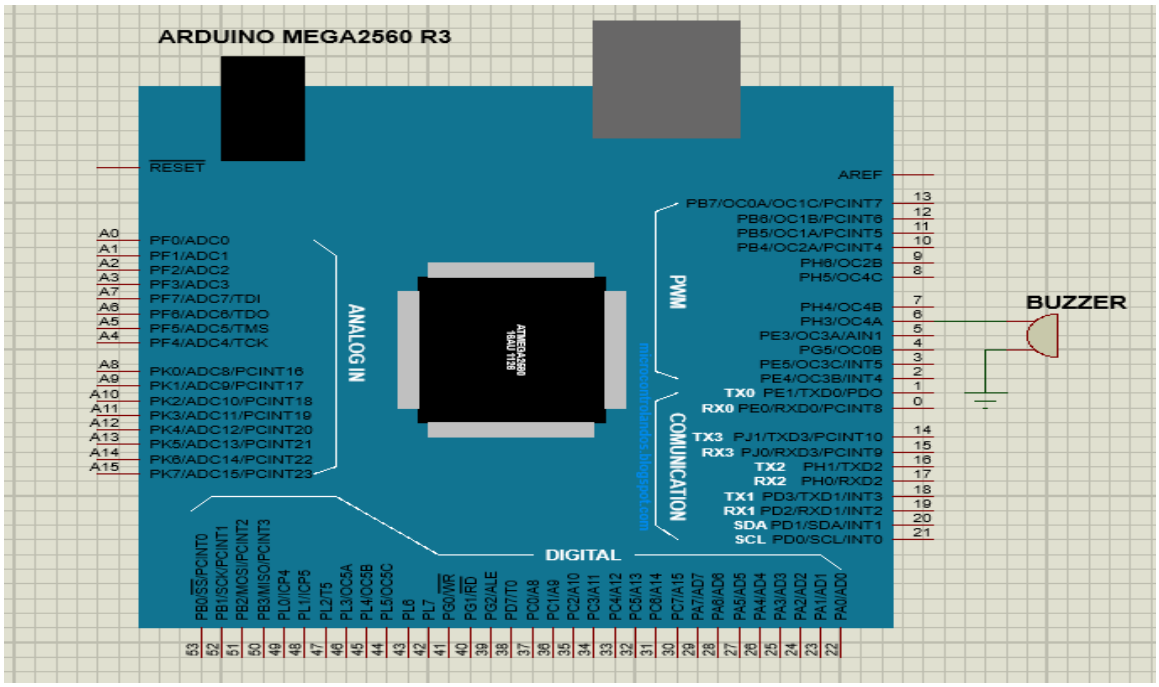


Fig 3.51 Circuit diagram of buzzer

This is a piezo buzzer that has built in circuitry that produces the audible buzzer tone. A plain piezo disk will not work in this circuit as it does not have any circuitry to drive it, although an Arduino sketch could be written to drive a piezo disk that is connected to an Arduino pin.

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Fig 3.52 Buzzer

The project is to take reading from different sensors (flame, rain, smoke, motion, sound, temperature and humidity) all connected to Arduino mega and show this readings on LCD and gives alerts on a buzzer and send it by Bluetooth module.

CHAPTER FOUR

SOFTWARE

4.1. ARDUINO IDE

We use Arduino IDE to write the code and upload it to the Arduino board and in this section we will show how to write a code to make the car moves using Arduino.

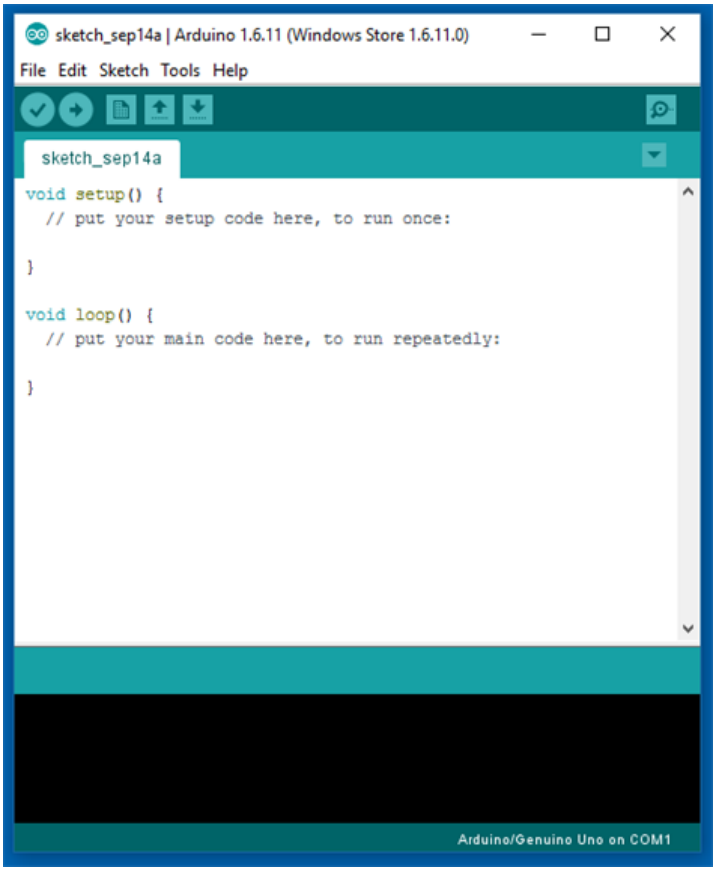
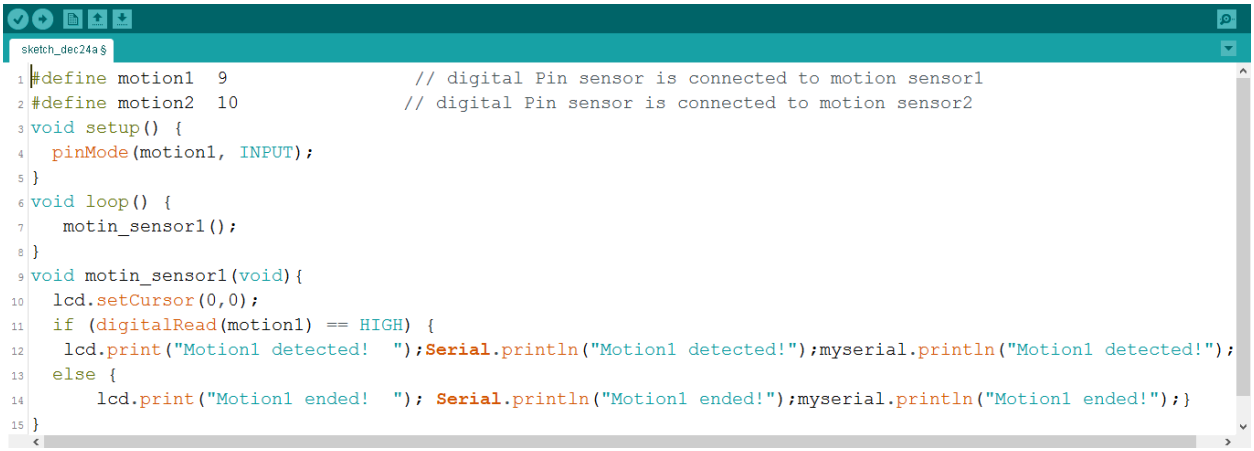


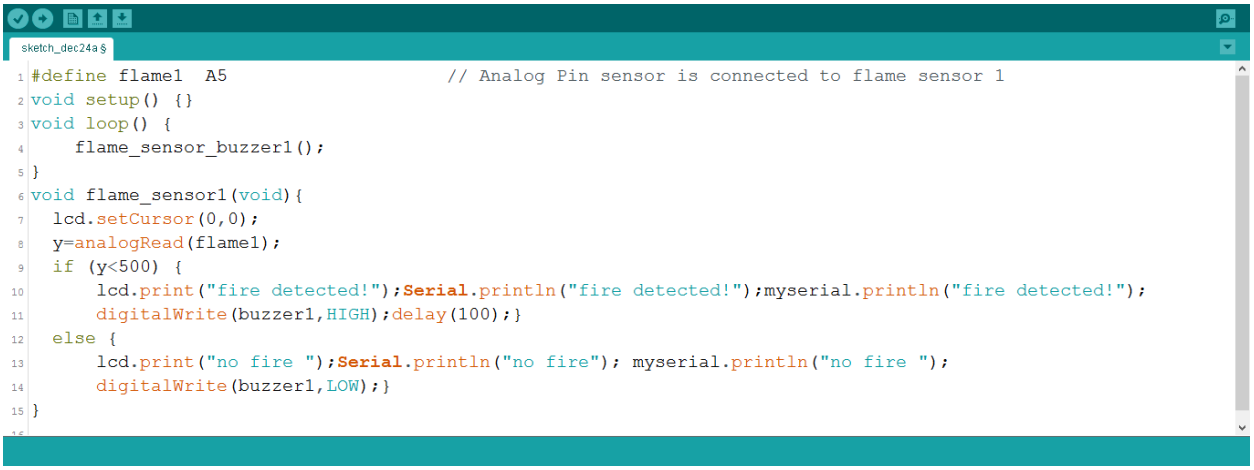
Fig 4.1.Arduino IDE

4.2. MOTION SENSOR



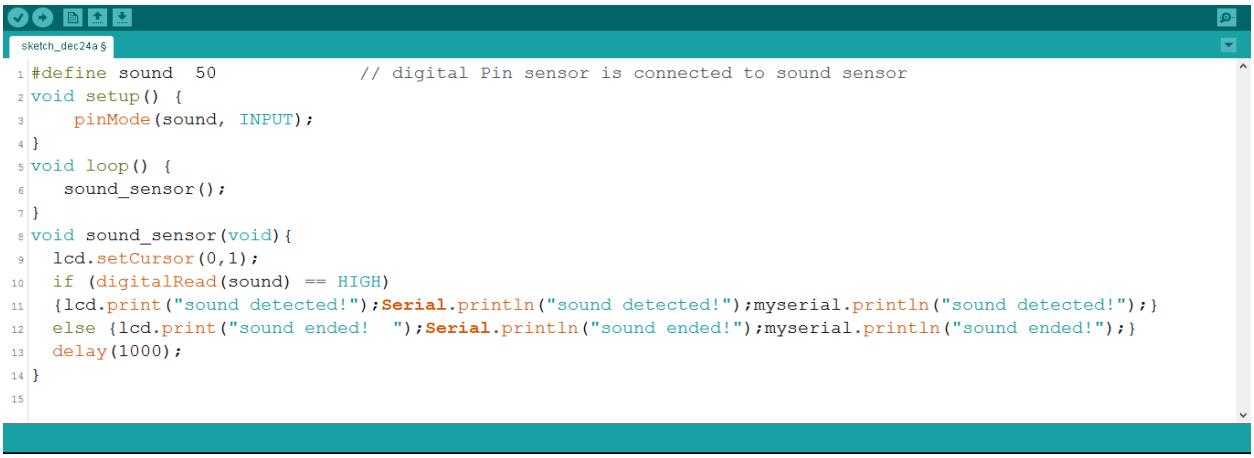
```
#define motion1  9                 // digital Pin sensor is connected to motion sensor1
#define motion2  10                // digital Pin sensor is connected to motion sensor2
void setup() {
  pinMode(motion1, INPUT);
}
void loop() {
   motin_sensor1();
}
void motin_sensor1(void){
  lcd.setCursor(0,0);
  if (digitalRead(motion1) == HIGH) {
   lcd.print("Motion1 detected!  ");Serial.println("Motion1 detected!");myserial.println("Motion1 detected!");
  else {
      lcd.print("Motion1 ended!  "); Serial.println("Motion1 ended!");myserial.println("Motion1 ended!");}
}
```

4.3. FLAME DETECTOR



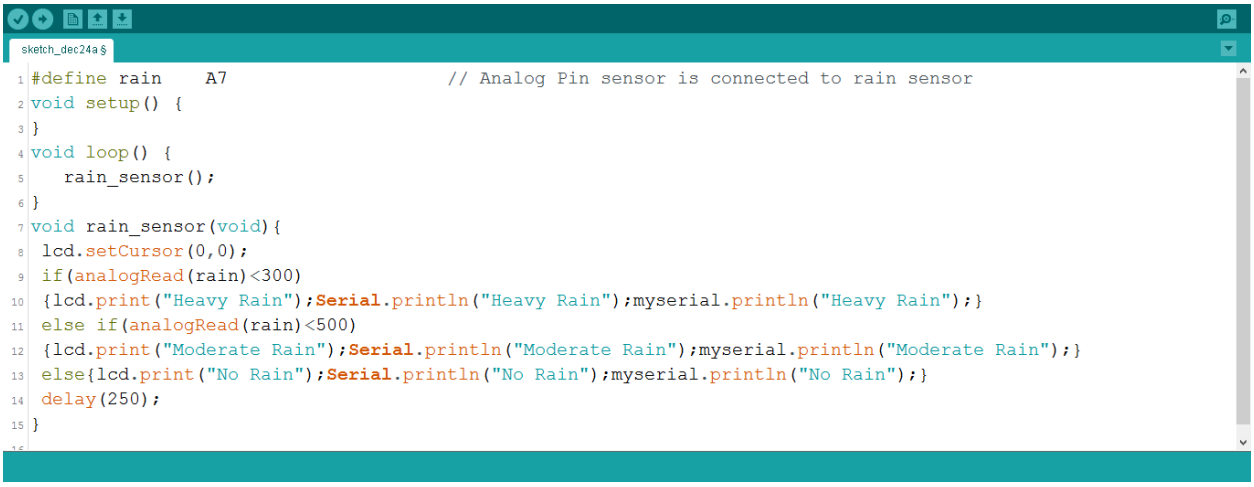
```
#define flame1  A5                    // Analog Pin sensor is connected to flame sensor 1
void setup() {}
void loop() {
    flame_sensor_buzzer1();
}
void flame_sensor1(void){
  lcd.setCursor(0,0);
  y=analogRead(flame1);
  if (y<500) {
      lcd.print("fire detected!");Serial.println("fire detected!");myserial.println("fire detected!");
      digitalWrite(buzzer1,HIGH);delay(100);}
  else {
      lcd.print("no fire ");Serial.println("no fire"); myserial.println("no fire ");
      digitalWrite(buzzer1,LOW);}
}
```

4.4. SOUND SENSOR



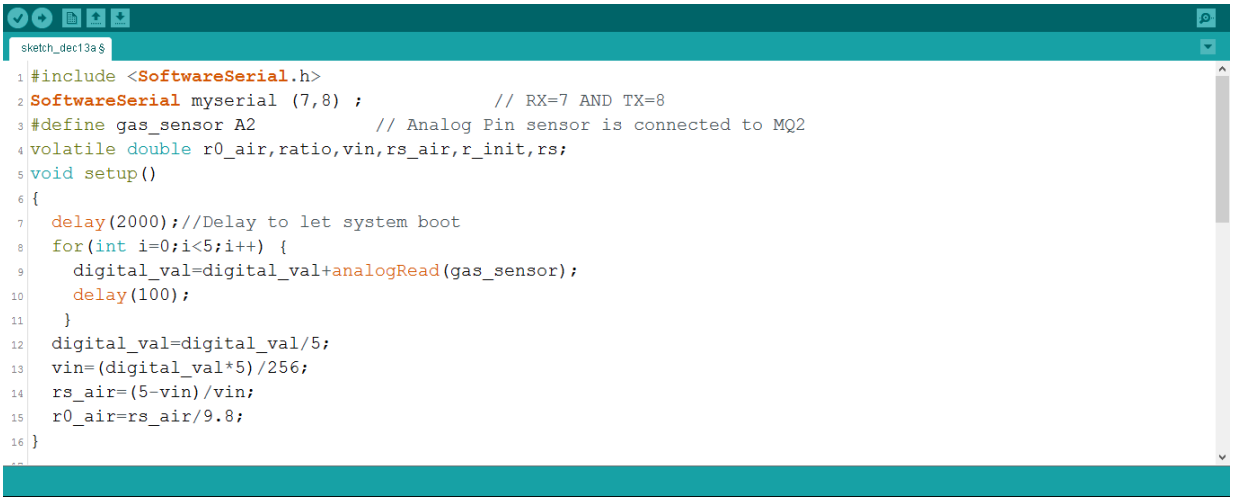
```
#define sound  50             // digital Pin sensor is connected to sound sensor
void setup() {
    pinMode(sound, INPUT);
}
void loop() {
   sound_sensor();
}
void sound_sensor(void){
  lcd.setCursor(0,1);
  if (digitalRead(sound) == HIGH)
  {lcd.print("sound detected!");Serial.println("sound detected!");myserial.println("sound detected!");}
  else {lcd.print("sound ended!  ");Serial.println("sound ended!");myserial.println("sound ended!");}
  delay(1000);
}
```

4.5. RAIN DETECTOR

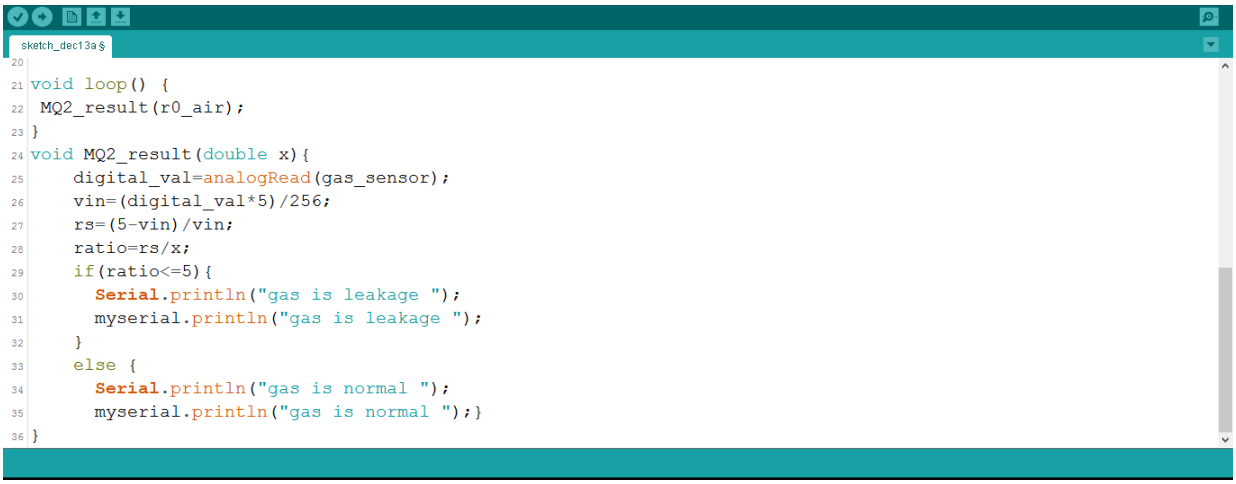


```
sketch_dec24a §
#define rain    A7                    // Analog Pin sensor is connected to rain sensor
void setup() {
}
void loop() {
   rain_sensor();
}
void rain_sensor(void){
 lcd.setCursor(0,0);
 if(analogRead(rain)<300)
 {lcd.print("Heavy Rain");Serial.println("Heavy Rain");myserial.println("Heavy Rain");}
 else if(analogRead(rain)<500)
 {lcd.print("Moderate Rain");Serial.println("Moderate Rain");myserial.println("Moderate Rain");}
 else{lcd.print("No Rain");Serial.println("No Rain");myserial.println("No Rain");}
 delay(250);
}
```

4.6. MQ-2 SMOKE DETECTOR



```
sketch_dec13a §
#include <SoftwareSerial.h>
SoftwareSerial myserial (7,8) ;            // RX=7 AND TX=8
#define gas_sensor A2           // Analog Pin sensor is connected to MQ2
volatile double r0_air,ratio,vin,rs_air,r_init,rs;
void setup()
{
  delay(2000);//Delay to let system boot
  for(int i=0;i<5;i++) {
    digital_val=digital_val+analogRead(gas_sensor);
    delay(100);
   }
  digital_val=digital_val/5;
  vin=(digital_val*5)/256;
  rs_air=(5-vin)/vin;
  r0_air=rs_air/9.8;
}
```



```
sketch_dec13a §

void loop() {
 MQ2_result(r0_air);
}
void MQ2_result(double x){
    digital_val=analogRead(gas_sensor);
    vin=(digital_val*5)/256;
    rs=(5-vin)/vin;
    ratio=rs/x;
    if(ratio<=5){
      Serial.println("gas is leakage ");
      myserial.println("gas is leakage ");
    }
    else {
      Serial.println("gas is normal ");
      myserial.println("gas is normal ");}
}
```

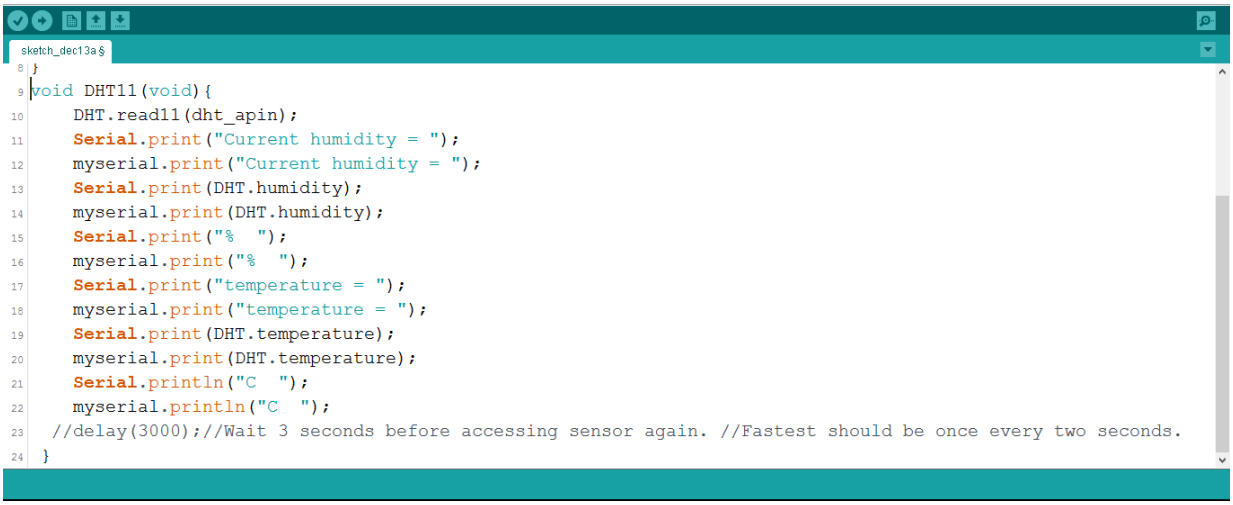
4.7. DHT11 HUMIDITY SENSOR

```
#include <dht.h>
#define dht_apin A0                 // Analog Pin sensor is connected to DHT11
dht DHT;

void setup()
{
  Serial.begin (9600) ;

}

void loop()
{
    DHT11();
    delay(500);
}
```



```
}
void DHT11(void){
    DHT.read11(dht_apin);
    Serial.print("Current humidity = ");
    myserial.print("Current humidity = ");
    Serial.print(DHT.humidity);
    myserial.print(DHT.humidity);
    Serial.print("%  ");
    myserial.print("%  ");
    Serial.print("temperature = ");
    myserial.print("temperature = ");
    Serial.print(DHT.temperature);
    myserial.print(DHT.temperature);
    Serial.println("C  ");
    myserial.println("C  ");
  //delay(3000);//Wait 3 seconds before accessing sensor again. //Fastest should be once every two seconds.
 }
```

4.8. LM35 TEMPERATURE SENSOR

```
#include <SoftwareSerial.h>
SoftwareSerial myserial (7,8) ;            // RX=7 AND TX=8
volatile int adc_count;
volatile float T;
void setup()
{
  myserial.begin(9600);
  Serial.begin (9600) ;

}
void loop()
{
  temperature_sensor();
  delay(500);
}
```

```
sketch_dec13a §

void temperature_sensor(void)
{
  adc_count=analogRead(lm);
  T =(adc_count*0.48828125);
  Serial.print("TEMPERATURE = ");
  myserial.print("TEMPERATURE = ");
  Serial.print(T);
  myserial.print(T);
  Serial.println(" *C");
  myserial.println(" *C");
}
```

4.9. LCD

```
sketch_dec21a §

#include<LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // sets the interfacing pins
void setup()
{
 lcd.begin(16, 2);  // initializes the 16x2 LCD
}
void loop()
{
  lcd.setCursor(0,0);           //sets the cursor at row 0 column 0
  lcd.print("16x2 LCD MODULE"); // prints 16x2 LCD MODULE
  lcd.setCursor(2,1);           //sets the cursor at row 1 column 2
  lcd.print("HELLO WORLD");     // prints HELLO WORLD
}
```

4.10. BLUETOOTH

```
sketch_jan05a §

#include <SoftwareSerial.h>
SoftwareSerial myserial (2,3) ;// bluetooth : Tx = 2 , Rx = 3
void setup()
{
  myserial.begin(9600);
  Serial.begin (9600) ;
}
void loop()
{
  if(myserial.available() )
  { Serial.write( myserial.read() ); }
  if (Serial.available () )
  { myserial.write(Serial.read() );}
  delay(500);
}
```

4.11. BUZZER

sketch_dec21a §

```
const int buzzer = 9; //buzzer to arduino pin 9
void setup()
{
  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output
}
void loop()
{
  tone(buzzer, 1000); // Send 1KHz sound signal...
  delay(1000);        // ...for 1 sec
  noTone(buzzer);     // Stop sound...
  delay(1000);        // ...for 1sec
}
```

4.12. WHOLE PROJECT

final_code_smart_city.ino §

```
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
#include <dht.h>
#define dht_apin A0                // Analog Pin sensor is connected to DHT11
#define dht_apin1 A1                // Analog Pin sensor is connected to DHT11
#define gas_sensor A2           // Analog Pin sensor is connected to MQ2
#define lm1  A3                    // Analog Pin sensor is connected to lm35
#define lm2  A4                    // Analog Pin sensor is connected to lm35
#define flame1  A5                    // Analog Pin sensor is connected to flame sensor 1
#define flame2  A6                    // Analog Pin sensor is connected to flame sensor 2
#define rain    A7                    // Analog Pin sensor is connected to rain sensor
#define motion1  9                 // digital Pin sensor is connected to motion sensor1
#define motion2  10               // digital Pin sensor is connected to motion sensor2
#define sound  50             // digital Pin sensor is connected to sound sensor
#define buzzer1  51             // digital Pin sensor is connected to buzzer
```

final_code_smart_city.ino §

```

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
SoftwareSerial myserial (7,8) ;            // RX=7 AND TX=8




volatile double r0_air,ratio,vin,rs_air,r_init,rs;
volatile int digital_val=0;
volatile int adc_count,x,y;
volatile float T;

dht DHT;

```

final_code_smart_city.ino §

```
void setup(){
  myserial.begin(9600); Serial.begin (9600) ;
  lcd.begin(16, 2);
  pinMode(motion1, INPUT);
  pinMode(motion2, INPUT);
  pinMode(sound, INPUT);
  pinMode(buzzer1,OUTPUT);
  delay(2000);//Delay to let system boot
  for(int i=0;i<5;i++){
    digital_val=digital_val+analogRead(gas_sensor);
    delay(100);}
  digital_val=digital_val/5;vin=(digital_val*5)/256;
  rs_air=(5-vin)/vin;
  r0_air=rs_air/9.8;
}
```

final_code_smart_city.ino §

```
void loop(){
  delay(1000);lcd.clear();
    DHT11_1();
    flame_sensor_buzzer1();flame_sensor_buzzer2();delay(500);lcd.clear();
    DHT11_2();
    flame_sensor_buzzer1();flame_sensor_buzzer2();delay(1000);lcd.clear();
    MQ2_result(r0_air);
    flame_sensor_buzzer1();flame_sensor_buzzer2();
    sound_sensor();delay(500);lcd.clear();
    temperature_sensor_1();temperature_sensor_2();
    flame_sensor_buzzer1();flame_sensor_buzzer2();delay(3000);
    motin_sensor1();motin_sensor2();
    flame_sensor_buzzer1();flame_sensor_buzzer2();delay(1000);lcd.clear();
    flame_sensor1();flame_sensor2();delay(1000); lcd.clear();
   rain_sensor();delay(2000);
}// end loop()
```

final_code_smart_city.ino §

```


void DHT11_1(void)
{
    lcd.setCursor(0, 0);
    DHT.read11(dht_apin);
    lcd.print("humidity1=");Serial.print("humidity1=");myserial.print("humidity1=");
    lcd.print(DHT.humidity);Serial.print(DHT.humidity);myserial.print(DHT.humidity);
    lcd.print("% ");Serial.print("% ");myserial.print("% ");
    lcd.setCursor(0, 1);
    lcd.print("temperatur1=");Serial.print("temperatur1=");myserial.print("temperatur1=");
    lcd.print(DHT.temperature); Serial.print(DHT.temperature); myserial.print(DHT.temperature);
    lcd.print("C  ");Serial.println("C  ");myserial.println("C  ");
   delay(3000);//Wait 3 seconds before accessing sensor again.
}
```

final_code_smart_city.ino §

```
89
90 void DHT11_2(void)
91 {
92     lcd.setCursor(0, 0);
93     DHT.read11(dht_apin1);
94     lcd.print("humidity2=");Serial.print("humidity2=");myserial.print("humidity2=");
95     lcd.print(DHT.humidity);Serial.print(DHT.humidity);myserial.print(DHT.humidity);
96     lcd.print("% ");Serial.print("% ");myserial.print("% ");
97     lcd.setCursor(0, 1);
98     lcd.print("temperatur2=");Serial.print("temperatur2=");myserial.print("temperatur2=");
99     lcd.print(DHT.temperature); Serial.print(DHT.temperature); myserial.print(DHT.temperature);
100    lcd.print("C  ");Serial.println("C  ");myserial.println("C  ");
101    delay(3000);//Wait 3 seconds before accessing sensor again.
102 }
103
104
```

final_code_smart_city.ino §

```
104
105  void temperature_sensor_1(void)
106 {
107   lcd.setCursor(0,0);
108   adc_count=analogRead(lm1);
109   T =(adc_count*0.48828125);
110   lcd.print("TEMPERATUR3="); Serial.print("TEMPERATUR3="); myserial.print("TEMPERATUR3=");
111   lcd.print(T); Serial.print(T);myserial.print(T);
112   lcd.print("*c");
113   Serial.println(" *C");
114    myserial.println(" *C");
115 }
116
117
118
```

final_code_smart_city.ino §

```
120  void temperature_sensor_2(void)
121 {
122   lcd.setCursor(0,1);
123   adc_count=analogRead(lm2);
124   T =(adc_count*0.48828125);
125   lcd.print("TEMPERATUR4=");
126   Serial.print("TEMPERATUR4=");
127   myserial.print("TEMPERATUR4=");
128   lcd.print(T);
129   Serial.print(T);
130   myserial.print(T);
131   lcd.print("*c");
132   Serial.println(" *C");
133    myserial.println(" *C");
134 }
```

final_code_smart_city.ino §

```
void MQ2_result(double x)
{
  digital_val=analogRead(gas_sensor);
  vin=(digital_val*5)/256;
  rs=(5-vin)/vin;
  ratio=rs/x;
  if(ratio<=5){
   lcd.setCursor(0,0);
   Serial.println("gas is leakage ");myserial.println("gas is leakage ");lcd.print("gas is leakage ");}
  else{
    lcd.setCursor(0,0);
   Serial.println("gas is normal ");myserial.println("gas is normal ");lcd.print("gas is normal ");}
delay(3000);
}
```

final_code_smart_city.ino §

```
void motin_sensor1(void)
{
  lcd.setCursor(0,0);
  if (digitalRead(motion1) == HIGH) {
      lcd.print("Motion1 detected!  ");
      Serial.println("Motion1 detected!");
      myserial.println("Motion1 detected!");
  }
  else {
      lcd.print("Motion1 ended!  ");
      Serial.println("Motion1 ended!");
      myserial.println("Motion1 ended!");
  }
}
```

final_code_smart_city.ino §

```
void motin_sensor2(void)
{
  lcd.setCursor(0,1);
  if (digitalRead(motion2) == HIGH) {
      lcd.print("Motion2 detected!  ");
      Serial.println("Motion2 detected!");
      myserial.println("Motion2 detected!");
  }
  else
  {
      lcd.print("Motion2 ended!  ");
      Serial.println("Motion2 ended!");
      myserial.println("Motion2 ended!");
  }
}
```

final_code_smart_city.ino §

```
184 void sound_sensor(void)
185 {
186   lcd.setCursor(0,1);
187   if (digitalRead(sound) == HIGH) {
188       lcd.print("sound detected!");
189       Serial.println("sound detected!");
190       myserial.println("sound detected!");
191   }
192   else {
193       lcd.print("sound ended!  ");
194       Serial.println("sound ended!");
195       myserial.println("sound ended!");
196   }
197   delay(1000);
198 }
```

final_code_smart_city.ino §

```
200
201 void flame_sensor1(void)
202 {
203   lcd.setCursor(0,0);
204   y=analogRead(flame1);
205   if (y<500) {
206       lcd.print("fire detected!");Serial.println("fire detected!"); myserial.println("fire detected!");
207       digitalWrite(buzzer1,HIGH);
208       delay(100);
209   }
210   else {
211       lcd.print("no fire ");Serial.println("no fire"); myserial.println("no fire ");
212       digitalWrite(buzzer1,LOW);
213   }
214 }
215
```

final_code_smart_city.ino §

```
218 void flame_sensor2(void)
219 {
220   lcd.setCursor(0,1);
221   x=analogRead(flame2);
222   if (x<500) {
223       lcd.print("fire detected!");Serial.println("fire detected!"); myserial.println("fire detected!");
224       digitalWrite(buzzer1,HIGH);delay(100);
225   }
226   else {
227       lcd.print("no fire ");
228       Serial.println("no fire");
229       myserial.println("no fire ");
230       digitalWrite(buzzer1,LOW);
231   }
232 }
```

final_code_smart_city.ino §

```
235
236
237
238 void rain_sensor(void)
239 {
240  lcd.setCursor(0,0);
241  if(analogRead(rain)<300){
242  lcd.print("Heavy Rain");Serial.println("Heavy Rain");myserial.println("Heavy Rain");}
243  else if(analogRead(rain)<500){
244  lcd.print("Moderate Rain");Serial.println("Moderate Rain");myserial.println("Moderate Rain");}
245  else{lcd.print("No Rain");Serial.println("No Rain");myserial.println("No Rain");}
246  delay(250);
247 }
248
249
250
```

final_code_smart_city.ino §

```
249
250
251 void flame_sensor_buzzer1(void)
252 {
253   y=analogRead(flame1);
254   if (y<500) { digitalWrite(buzzer1,HIGH);}
255   else {digitalWrite(buzzer1,LOW);}
256
257 }
258 void flame_sensor_buzzer2(void)
259 {
260   x=analogRead(flame2);
261   if (x<500){digitalWrite(buzzer1,HIGH);delay(100);}
262   else { digitalWrite(buzzer1,LOW);}
263  }
264
```

CONCLUSION

Our projects aims to the implementation of technology in all our living ways starting with our homes the place we live in so it took the main things that can affect any home and we try to make a solution for it as the rain falling detect it and alert the user to take proper action that can save more lives and make new technologies that can help in the development in the smart technology that can help citizen that lives in this city

REFERENCES

[1] https://www.quora.com/Which-is-better-arduino-or-8051-microcontroller

[2] http://www.varesano.net/blog/fabio/what Arduino why we choose it what can we do it

[3] http://www.raviyp.com/embedded/158-difference-between-arduino-and-microcontroller

[4] https://www.arduino.cc/en/Guide/Introduction

[5] https://learn.sparkfun.com/tutorials/arduino-comparison-guide

[6] https://www.allaboutcircuits.com/textbook/digital/#chpt-13

[7] https://circuitdigest.com/microcontroller-projects/arduino-smoke-detector-on-pcb-using-mq2-gas-sensor

[8] http://howtomechatronics.com/tutorials/arduino/how-pir-sensor-works-and-how-to-use-it-with-arduino/

[9] http://microcontrollerslab.com/flame-sensor-arduino-fire-detection/

[10] http://microcontrollerslab.com/raindrop-sensor-arduino-detector/

[11] http://henrysbench.capnfatz.com/henrys-bench/arduino- sensors-and-input/arduino-sound-detection-sensor-tutorial-and-user-manual/

[12] http://internetofthingsagenda.techtarget.com/definition/smart-city

[13] https://www.wien.gv.at/stadtentwicklung/studien/pdf/b008403j.pdf

[14] https://www.springer.com/cda/content/document/cda.../9783319277523-c2.pdf

[15] https://e-radionica.com/productdata/LD3361BS.pdf

[16] http://www.ladyada.net/media/sensors/RE200B.pdf

[17] http://rubus.net.nz/?page_id=613

[18] https://arduinodiy.wordpress.com/2012/05/06/7-segment-display-on-arduino-with-595/

[19] http://www.energiazero.org/arduino_sensori/2_channel_5v_10a_relay_module.pdf

[20] http://wiki.sunfounder.cc/index.php?title=2_Channel_5V_Relay_Module