Student Minor Research Project

LIGHT ANIMATIONS USING ARDUINO & MATLAB

Under RUSA 2.0 Scheme

(Through Ch.S.D.St.Theresa's College for Women (Autonomous), Eluru, AP)

Submitted by

Mr G Bharat Sai , III B.Sc. MPE (Reg.No.11704022)

Mr S Raja Shekhar, III B.Sc. MPE (Reg.No.11704038)

Ms Ch Sindhu, III B.Sc. MECs (Reg.No.11705049)

Under the guidance of

Dr K Venkateswarlu

HOD of Electronics & Project Advisor

Department Of Electronics

SRI Y N COLLEGE

(AUTONOMOUS)

Thrice Accredited by NAAC at 'A' Grade

Recognized by UGC as "College with Potential for Excellence"

Narsapur-534275, AP, India

December-2019

Department Of Electronics

SRI Y N COLLEGE

(AUTONOMOUS)

Thrice Accredited by NAAC at 'A' Grade

Recognized by UGC as "College with Potential for Excellence"

Narsapur-534275, AP, India

CERTIFICATE

*This is to certify that the project work entitled "***Light Animations using Arduino and MATLAB***" is bonafied work carried out by Mr G Bharat Sai (Reg.No: 11704022), Mr S Raja Shekhar (Reg.No: 11704038), Ms Ch Sindhu (Reg.No: 11705049), submitted in Third Year of the degree B.Sc. in Electronics during the year 2019-20 is an authentic work under my supervision and guidance.*

To the best of my knowledge, the matter embodied in the project work has not been submitted to any other College/Institute.

Date: 29-12-2019

Dr K Venkateswarlu
Project Advisor
Department of Electronics

ACKNOWLEDGEMENT

We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our Project advisor, ***Dr K Venkateswarlu,*** *Head, Department of Electronics,* ***Sri Y N College (Autonomous), Narsapur*** *in bringing this report to a successful completion.*

We are grateful to ***Mr K Vinaya Phaneendhra****, Lecturer, Department of Electronics for permitting us to make use of the facilities available in the department to carry out the project successfully. Last but not the least we express our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.*

Finally we extend our gratefulness to one and all who are directly or indirectly involved in the successful completion of this project work.

Mr. G Bharat Sai	**Mr. S Raja Shekhar**	**Ms. Ch Sindhu**
III.B.Sc.MPE	III.B.Sc.MPE	III.B.Sc.MECs
Reg. No 11704022	Reg. No.11704038	Reg.No.11705049

DECLARATION

We, the undersigned, declare that the project entitled "**Light Animations using Arduino and MATLAB**", being submitted in Third Year of Bachelor of Science in Electronics, Sri Y N College (Autonomous), is the work carried out by us.

Mr. G Bharat Sai	**Mr. S Raja Shekhar**	**Ms. Ch Sindhu**
III.B.Sc.MPE	III.B.Sc.MPE	III.B.Sc.MECs
Reg. No 11704022	Reg. No.11704038	Reg.No.11705049

Contents **Page No.**

List of Figures & Tables Page No.

1. ABSTRACT

The main objective of this project to develop, a MATLAB-based graphical user interface (GUI) approach to control the glowing pattern of a number of light-emitting diodes (LEDs). Use of GUI is advantageous since the user can control illumination patterns while performing other tasks in the PC. Light animations are visually appealing and hence widely used for advertising purposes.

In this project Arduino IDE used to program the Arduino Uno. GUI application program has been developed in R2014a version of MATLAB. After installing this MATLAB version in PC. The project file, MATLAB will try to communicate with the board. After successful communication is established, controlled the LEDs by pressing the corresponding pushbuttons in the GUI.

2. INTRODUCTION TO ARDUINO

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.

Why Arduino?

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 - 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, Netmedia's BX-24, Phidgets, MIT's Handyboard, 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
- 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.

2.1 INTRODUCTION TO ARDUINO UNO

- **Arduino Uno** is a microcontroller board developed by Arduino.cc which is an open-source electronics platform mainly based on AVR microcontroller Atmega328.
- First Arduino project was started in Interaction Design Institute Ivrea in 2003 by David Cuartielles and Massimo Banzi with the intention of providing a cheap and flexible way to students and professional for controlling a number of devices in the real world.
- The current version of Arduino Uno comes with USB interface, 6 analog input pins, 14 I/O digital ports that are used to connect with external electronic circuits. Out of 14 I/O ports, 6 pins can be used for PWM output.
- It allows the designers to control and sense the external electronic devices in the real world.
- This board comes with all the features required to run the controller and can be directly connected to the computer through USB cable that is used to transfer the code to the controller using IDE (Integrated Development Environment) software, mainly developed to program Arduino. IDE is equally compatible with Windows, MAC or Linux Systems, however, Windows is preferable to use. Programming languages like C and C++ are used in IDE.
- Apart from USB, battery or AC to DC adopter can also be used to power the board.
- Arduino Uno boards are quite similar to other boards in Arduino family in terms of use and functionality, however, Uno boards don't come with FTDI USB to Serial driver chip.
- There are many versions of Uno boards available, however, Arduino Nano V3 and Arduino Uno are the most official versions that come with Atmega328 8-bit AVR Atmel microcontroller where RAM memory is 32KB.
- When nature and functionality of the task go complex, Mirco SD card can be added in the boards to make them store more information.

Features of Arduino Uno:

- Arduino Uno comes with USB interface i.e. USB port is added on the board to develop serial communication with the computer.

- Atmega328 microcontroller is placed on the board that comes with a number of features like timers, counters, interrupts, PWM, CPU, I/O pins and based on a 16MHz clock that helps in producing more frequency and number of instructions per cycle.

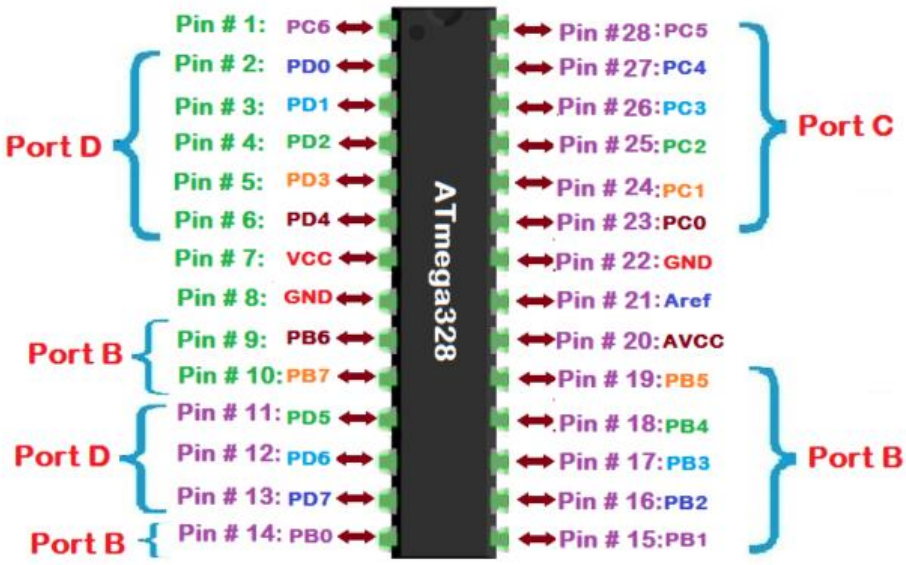


Fig 2.1 (a) Pin Diagram of ATmega328

- It is an open source platform where anyone can modify and optimize the board based on the number of instructions and task they want to achieve.
- This board comes with a built-in regulation feature which keeps the voltage under control when the device is connected to the external device.
- Reset pin is added in the board that reset the whole board and takes the running program in the initial stage. This pin is useful when board hangs up in the middle of the running program; pushing this pin will clear everything up in the program and starts the program right from the beginning.
- There are 14 I/O digital and 6 analog pins incorporated in the board that allows the external connection with any circuit with the board. These pins provide the flexibility and ease of use to the external devices that can be connected through these pins. There is no hard and fast interface required to connect the devices to the board. Simply plug the external device into the pins of the board that are laid out on the board in the form of the header.

- The 6 analog pins are marked as A0 to A5 and come with a resolution of 10bits. These pins measure from 0 to 5V, however, they can be configured to the high range using analogReference() function and AREF pin.
- 13KB of flash memory is used to store the number of instructions in the form of code.
- Only 5 V is required to turn the board on, which can be achieved directly using USB port or external adopter, however, it can support external power source up to 12 V which can be regulated and limit to 5 V or 3.3 V based on the requirement of the project.

Arduino Uno Pinout:

Arduino Uno is based on AVR microcontroller called Atmega328. This controller comes with 2KB SRAM, 32KB of flash memory, 1KB of EEPROM. Arduino Board comes with 14 digital pins and 6 analog pins. ON-chip ADC is used to sample these pins. A 16 MHz frequency crystal oscillator is equipped on the board. Following figure shows the pinout of the Arduino Uno Board.

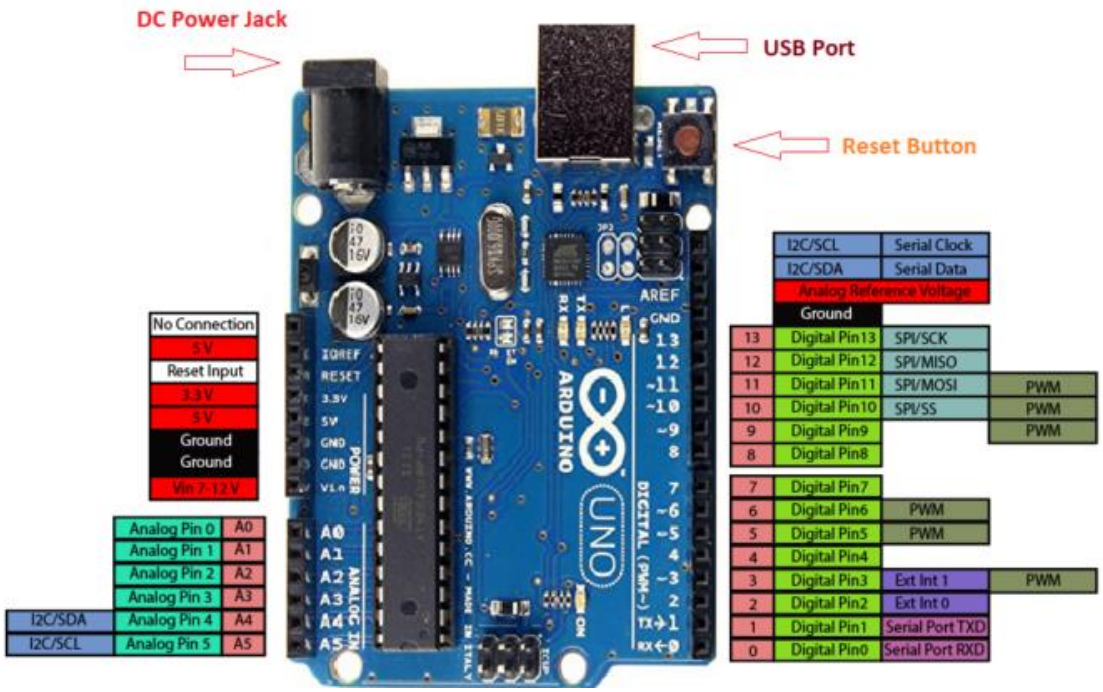


Fig 2.1 (b) Pinout of Arduino Uno

Pin Description:

There are several I/O digital and analog pins placed on the board which operates at 5V. These pins come with standard operating ratings ranging between 20mA to 40mA. Internal pull-up resistors are used in the board that limits the current exceeding from the given operating conditions. However, too much increase in current makes these resisters useless and damages the device.

LED: Arduino Uno comes with built-in LED which is connected through pin 13. Providing HIGH value to the pin will turn it ON and LOW will turn it OFF.

Vin: It is the input voltage provided to the Arduino Board. It is different than 5 V supplied through a USB port. This pin is used to supply voltage. If a voltage is provided through power jack, it can be accessed through this pin.

5V: This board comes with the ability to provide voltage regulation. 5V pin is used to provide output regulated voltage. The board is powered up using three ways i.e. USB, Vin pin of the board or DC power jack.

USB supports voltage around 5V while Vin and Power Jack support a voltage ranges between 7V to 20V. It is recommended to operate the board on 5V. It is important to note that, if a voltage is supplied through 5V or 3.3V pins, they result in bypassing the voltage regulation that can damage the board if voltage surpasses from its limit.

GND: These are ground pins. More than one ground pins are provided on the board which can be used as per requirement.

Reset: This pin is incorporated on the board which resets the program running on the board. Instead of physical reset on the board, IDE comes with a feature of resetting the board through programming.

IOREF: This pin is very useful for providing voltage reference to the board. A shield is used to read the voltage across this pin which then select the proper power source.

PWM: PWM is provided by 3,5,6,9,10, 11pins. These pins are configured to provided 8-bit output PWM.

SPI: It is known as Serial Peripheral Interface. Four pins 10(SS), 11(MOSI), 12(MISO), 13(SCK) provide SPI communication with the help of SPI library.

AREF: It is called Analog Reference. This pin is used for providing a reference voltage to the analog inputs.

TWI: It is called Two-wire Interface. TWI communication is accessed through Wire Library. A4 and A5 pins are used for this purpose.

Serial Communication: Serial communication is carried out through two pins called Pin 0 (Rx) and Pin 1 (Tx).

Rx pin is used to receive data while Tx pin is used to transmit data.

External Interrupts: Pin 2 and 3 are used for providing external interrupts. An interrupt is called by providing LOW or changing value.

Communication and Programming:

Arduino Uno comes with an ability of interfacing with other other Arduino boards, microcontrollers and computer. The Atmega328 placed on the board provides serial communication using pins like Rx and Tx.

The Atmega16U2 incorporated on the board provides a pathway for serial communication using USB com drivers. Serial monitor is provided on the IDE software which is used to send or receive text data from the board. If LEDs placed on the Rx and Tx pins will flash, they indicate the transmission of data.

Arduino Uno is programmed using Arduino Software which is a cross-platform application called IDE written in Java. The AVR microcontroller Atmega328 laid out on the base comes with builtin bootloader that sets you free from using a separate burner to upload the program on the board.

3. INTRODUCTION TO MATLAB

What Is MATLAB?

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which

toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The MATLAB System

The MATLAB system consists of five main parts:

The MATLAB language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

The MATLAB working environment.

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

Handle Graphics.

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

The MATLAB mathematical function library.

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

The MATLAB Application Program Interface (API).

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

A minimum MATLABsession:

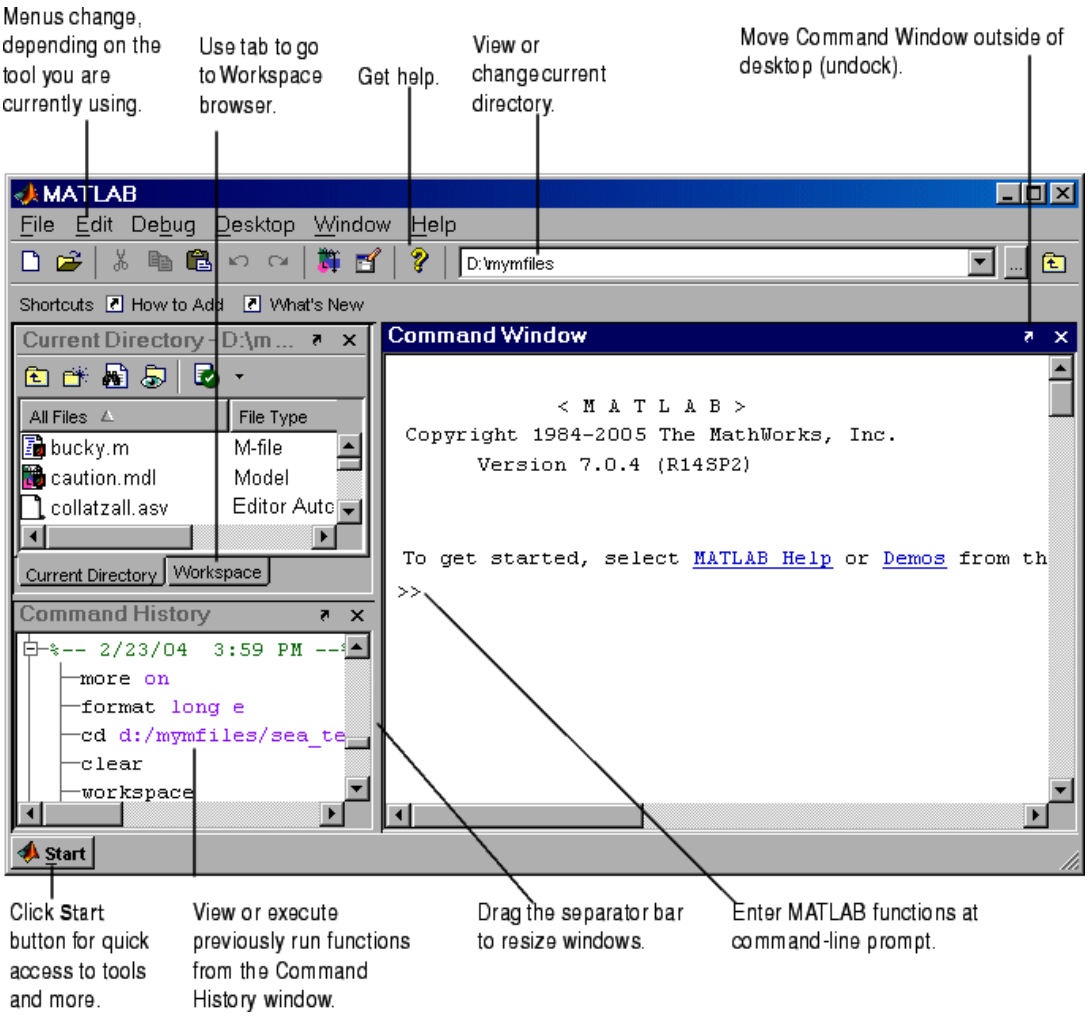
The goal of this *minimum* session (also called *starting* and *exiting* sessions) is to learn the first steps:

- How to log on
- Invoke MATLAB
- Do a few simple calculations
- How to quit MATLAB

Starting MATLAB:

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut *icon* (MATLAB 7.0.4) on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains *other* windows. The major tools within or accessible from the desktop are:

- The COMMAND WINDOW
- The COMMAND HISTORY
- The WORKSPACE
- The CURRENT DIRECTORY
- The HELP BROWSER
- The START button



The graphical interface to the MATLAB workspace

When MATLAB is started for the first time, the screen looks like the one that shown in the Figure 1.1. This illustration also shows the default configuration of the MATLAB desktop. You can customize the arrangement of tools and documents to suit your needs.

Now, we are interested in doing some simple calculations. We will assume that you have sufficient understanding of your computer under which MATLAB is being run.

You are now faced with the MATLAB desktop on your computer, which contains the prompt (>>) in the Command Window. Usually, there are 2 types of prompt:

>> for full version

EDU> for educational version

NOTE: To simplify the notation, we will use this prompt, >>, as a standard prompt sign, though our MATLAB version is for educational purpose.

Quitting MATLAB

To end your MATLAB session, type **quit** in the Command Window, or select **File Exit** $\longrightarrow$ **MATLAB** in the desktop main menu.

4. IN4007

Diodes are used to convert AC into DC these are used as half wave rectifier or full wave rectifier. Three points must he kept in mind while using any type of diode.

1.Maximum forward current capacity

2.Maximum reverse voltage capacity

3.Maximum forward voltage capacity

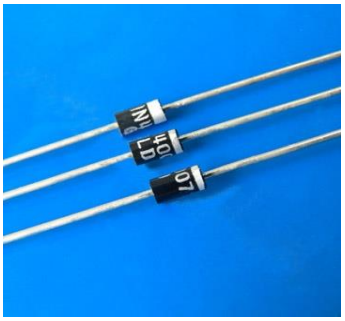


Fig 4(a): 1N4007 diodes

The number and voltage capacity of some of the important diodes available in the market are as follows:

- Diodes of number IN4001, IN4002, IN4003, IN4004, IN4005, IN4006 and IN4007 have maximum reverse bias voltage capacity of 50V and maximum forward current capacity of 1 Amp.
- Diode of same capacities can be used in place of one another. Besides this diode of more capacity can be used in place of diode of low capacity but diode of low capacity cannot be used in place of diode of high capacity. For example, in place of IN4002; IN4001 or IN4007 can be used but IN4001 or IN4002 cannot be used in place of IN4007.The diode BY125made by company BEL is equivalent of diode from IN4001 to IN4003. BY 126 is equivalent to diodes IN4004 to 4006 and BY 127 is equivalent to diode IN4007.

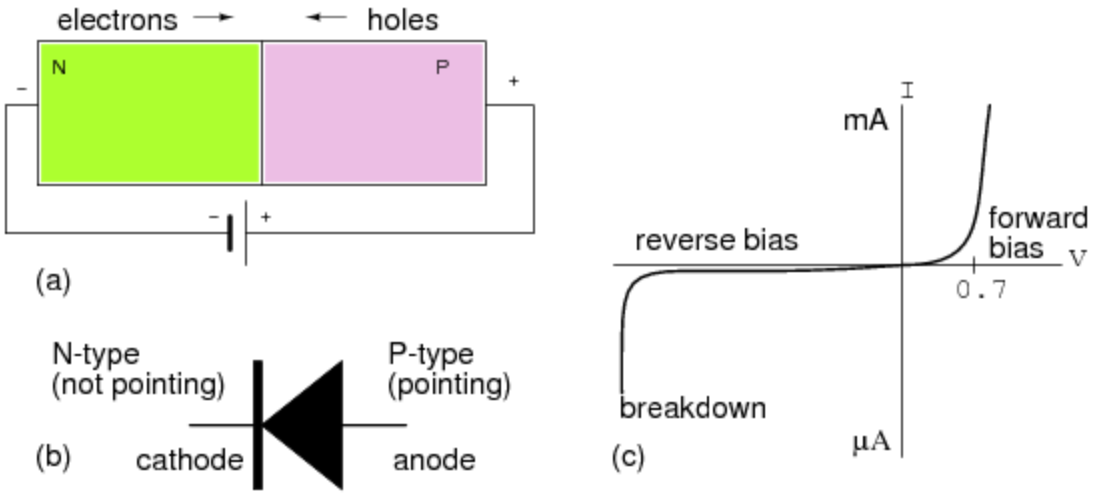


Fig 4 (b):PN Junction diode

PN JUNCTION OPERATION

Now that you are familiar with P- and N-type materials, how these materials are joined together toform a diode, and the function of the diode, let us continue our discussion with the operation of the PNjunction. But before we can understand how the PN junction works, we must first consider current flow inthe materials that make up the junction and what happens initially within the junction when these twomaterials are joined together.

Current Flow in the N-Type Material

Conduction in the N-type semiconductor, or crystal, is similar to conduction in a copper wire. Thatis, with voltage applied across the material, electrons will move through the crystal just as current wouldflow in a copper wire. This is shown in figure 1-15. The positive potential of the battery will attract thefree electrons in the crystal. These electrons will leave the crystal and flow into the positive terminal ofthe battery. As an electron leaves the crystal, an electron from the negative terminal of the battery willenter the crystal, thus completing the current path. Therefore, the majority current carriers in the N-typematerial (electrons) are repelled by the negative side of the battery and move through the crystal towardthe positive side of the battery.

Current Flow in the P-Type Material

Current flow through the P-type material is illustrated. Conduction in the P material isby positive holes, instead of negative electrons. A hole moves from the positive terminal of the P materialto the negative terminal. Electrons from the external circuit enter the negative terminal of the material andfill holes in the vicinity of this terminal. At the positive terminal, electrons are removed from the covalentbonds, thus creating new holes. This process continues as the steady stream of holes (hole current) movestoward the negative terminal

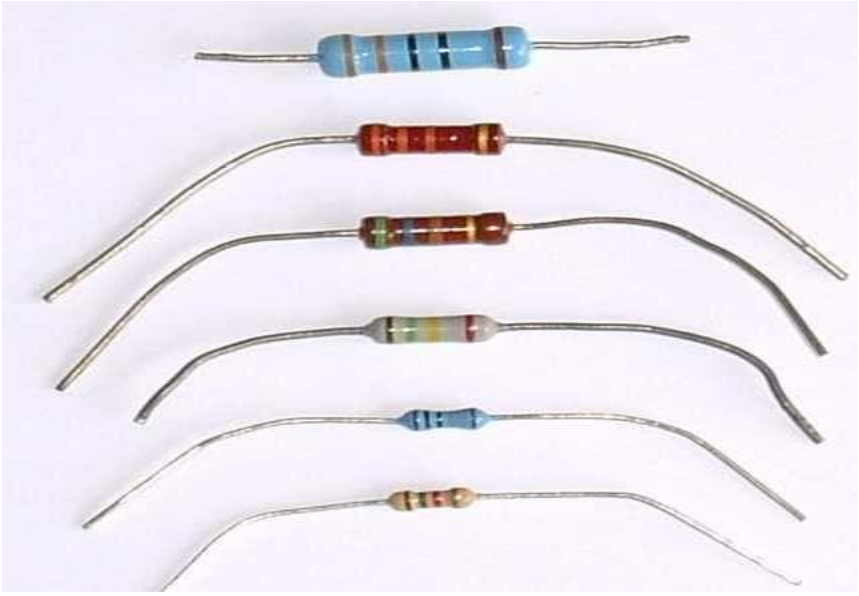
5. RESISTORS

A resistor is a two-terminal electronic component designed to oppose an electric current by producing a voltage drop between its terminals in proportion to the current, that is, in accordance with Ohm's law:

$$V = IR$$

Resistors are used as part of electrical networks and electronic circuits. They are extremely commonplace in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel/chrome).

The primary characteristics of resistors are their resistance and the power they can dissipate. Other characteristics include temperature coefficient, noise, and inductance. Less well-known is critical resistance, the value below which power dissipation limits the maximum permitted current flow, and above which the limit is applied voltage. Critical resistance depends upon the materials constituting the resistor as well as its physical dimensions; it's determined by design. Resistors can be integrated into hybrid and printed circuits, as well as integrated circuits. Size, and position of leads (or terminals) are relevant to equipment designers; resistors must be physically large enough not to overheat when dissipating their power.



A resistor is a two-terminalpassiveelectronic component which implements electrical resistance as a circuit element. When a voltage V is applied across the terminals of a resistor, a

current I will flow through the resistor in direct proportion to that voltage. The reciprocal of the constant of proportionality is known as the resistance R, since, with a given voltage V, a larger value of R further "resists" the flow of current I as given by Ohm's law:

$$I = \frac{V}{R}$$

Resistors are common elements of electrical networks and electronic circuits and are ubiquitous in most electronic equipment. Practical resistors can be made of various compounds and films, as well as resistance wire (wire made of a high-resistivity alloy, such as nickel-chrome). Resistors are also implemented within integrated circuits, particularly analog devices, and can also be integrated into hybrid and printed circuits.

The electrical functionality of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than 9 orders of magnitude. When specifying that resistance in an electronic design, the required precision of the resistance may require attention to the manufacturing tolerance of the chosen resistor, according to its specific application. The temperature coefficient of the resistance may also be of concern in some precision applications. Practical resistors are also specified as having a maximum power rating which must exceed the anticipated power dissipation of that resistor in a particular circuit: this is mainly of concern in power electronics applications. Resistors with higher power ratings are physically larger and may require heat sinking. In a high voltage circuit, attention must sometimes be paid to the rated maximum working voltage of the resistor.

The series inductance of a practical resistor causes its behavior to depart from ohms law; this specification can be important in some high-frequency applications for smaller values of resistance. In a low-noise amplifier or pre-amp the noise characteristics of a resistor may be an issue. The unwanted inductance, excess noise, and temperature coefficient are mainly dependent on the technology used in manufacturing the resistor. They are not normally specified individually for a particular family of resistors manufactured using a particular technology. A family of discrete resistors is also characterized according to its form factor, that is, the size of the device and position of its leads (or terminals) which is relevant in the practical manufacturing of circuits using them.

Units

The ohm (symbol: Ω) is the SI unit of electrical resistance, named after Georg Simon Ohm. An ohm is equivalent to a volt per ampere. Since resistors are specified and manufactured over a very large range of values, the derived units of milliohm (1 mΩ = 10^{-3} Ω), kilohm (1 kΩ = 10^{3} Ω), and megohm (1 MΩ = 10^{6} Ω) are also in common usage.

The reciprocal of resistance R is called conductance G = 1/R and is measured in Siemens (SI unit), sometimes referred to as a mho. Thus a Siemens is the reciprocal of an ohm: $S = \Omega^{-1}$. Although the concept of conductance is often used in circuit analysis, practical resistors are always specified in terms of their resistance (ohms) rather than conductance.

Theory of operation

Ohm's law

The behaviour of an ideal resistor is dictated by the relationship specified in Ohm's law:

$$V = I \cdot R$$

Ohm's law states that the voltage (V) across a resistor is proportional to the current (I) passing through it, where the constant of proportionality is the resistance (R).

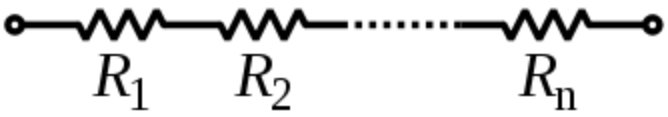
Equivalently, Ohm's law can be stated:

$$I = \frac{V}{R}$$

This formulation of Ohm's law states that, when a voltage (V) is present across a resistance (R), a current (I) will flow through the resistance. This is directly used in practical computations. For example, if a 300 ohm resistor is attached across the terminals of a 12 volt battery, then a current of 12 / 300 = 0.04 amperes (or 40 mill amperes) will flow through that resistor.

Series and parallel resistors

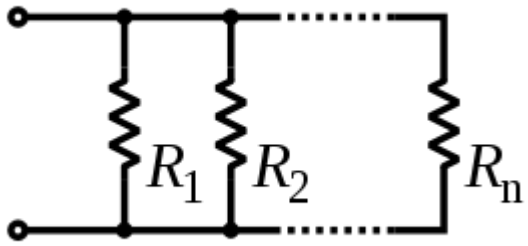
In a series configuration, the current through all of the resistors is the same, but the voltage across each resistor will be in proportion to its resistance. The potential difference (voltage) seen across the network is the sum of those voltages, thus the total resistance can be found as the sum of those resistances:



$$R_{\text{eq}} = R_1 + R_2 + \cdots + R_n$$

As a special case, the resistance of N resistors connected in series, each of the same resistance R, is given by NR.

Resistors in a parallel configuration are each subject to the same potential difference (voltage), however the currents through them add. The conductance of the resistors then add to determine the conductance of the network. Thus the equivalent resistance (R_{eq}) of the network can be computed:



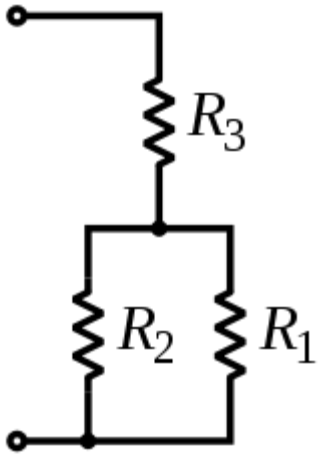
$$\frac{1}{R_{\text{eq}}} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}$$

The parallel equivalent resistance can be represented in equations by two vertical lines "||" (as in geometry) as a simplified notation. For the case of two resistors in parallel, this can be calculated using:

$$R_{\text{eq}} = R_1 \| R_2 = \frac{R_1 R_2}{R_1 + R_2}$$

As a special case, the resistance of N resistors connected in parallel, each of the same resistance R, is given by R/N.

A resistor network that is a combination of parallel and series connections can be broken up into smaller parts that are either one or the other. For instance,

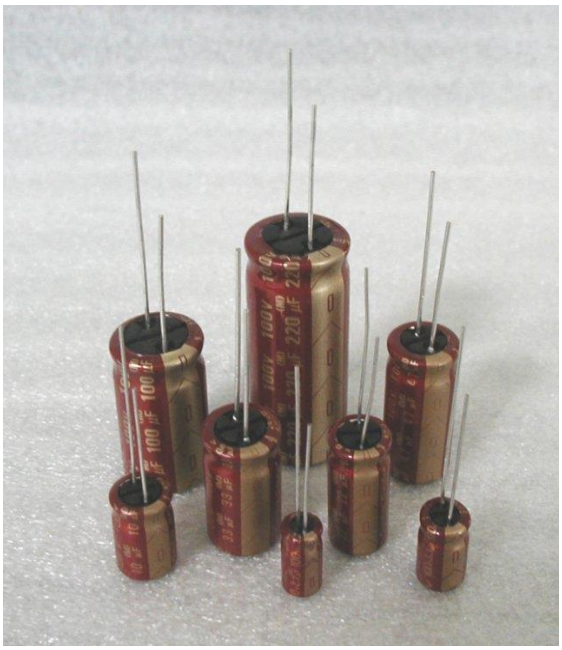


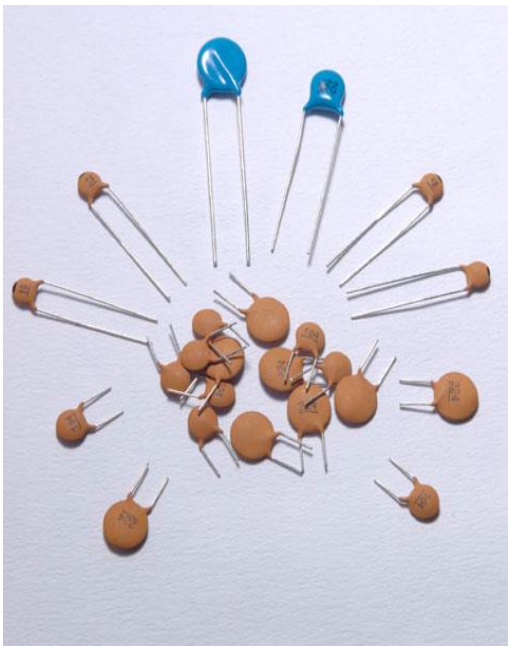
$$R_{\text{eq}} = (R_1 \| R_2) + R_3 = \frac{R_1 R_2}{R_1 + R_2} + R_3$$

However, some complex networks of resistors cannot be resolved in this manner, requiring more sophisticated circuit analysis. For instance, consider a cube, each edge of which has been replaced by a resistor. What then is the resistance that would be measured between two opposite vertices? In the case of 12 equivalent resistors, it can be shown that the corner-to-corner resistance is $5/6$ of the individual resistance. More generally, the Y-Δ transform, or matrix methods can be used to solve such a problem. One practical application of these relationships is that a non-standard value of resistance can generally be synthesized by connecting a number of standard values in series and/or parallel. This can also be used to obtain a resistance with a higher power rating than that of the individual resistors used. In the special case of N identical resistors all connected in series or all connected in parallel, the power rating of the individual resistors is thereby multiplied by N.

6. CAPACITORS

A capacitor or condenser is a passive electronic component consisting of a pair of conductors separated by a dielectric. When a voltage potential difference exists between the conductors, an electric field is present in the dielectric. This field stores energy and produces a mechanical force between the plates. The effect is greatest between wide, flat, parallel, narrowly separated conductors.





An ideal capacitor is characterized by a single constant value, capacitance, which is measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them. In practice, the dielectric between the plates passes a small amount of leakage current. The conductors and leads introduce an equivalent series resistance and the dielectric has an electric field strength limit resulting in a breakdown voltage.

The properties of capacitors in a circuit may determine the resonant frequency and quality factor of a resonant circuit, power dissipation and operating frequency in a digital logic circuit, energy capacity in a high-power system, and many other important aspects.

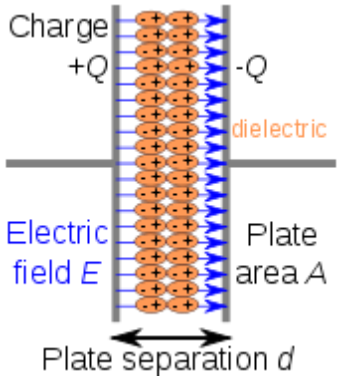
A capacitor (formerly known as condenser) is a device for storing electric charge. The forms of practical capacitors vary widely, but all contain at least two conductors separated by a

non-conductor. Capacitors used as parts of electrical systems, for example, consist of metal foils separated by a layer of insulating film.

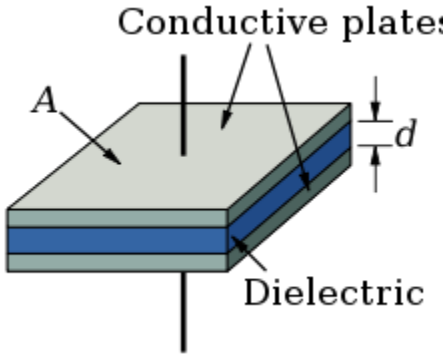
Capacitors are widely used in electronic circuits for blocking direct current while allowing alternating current to pass, in filter networks, for smoothing the output of power supplies, in the resonant circuits that tune radios to particular frequencies and for many other purposes.

A capacitor is a passiveelectronic component consisting of a pair of conductors separated by a dielectric (insulator). When there is a potential difference (voltage) across the conductors, a static electric field develops in the dielectric that stores energy and produces a mechanical force between the conductors. An ideal capacitor is characterized by a single constant value, capacitance, measured in farads. This is the ratio of the electric charge on each conductor to the potential difference between them.

The capacitance is greatest when there is a narrow separation between large areas of conductor, hence capacitor conductors are often called "plates", referring to an early means of construction. In practice the dielectric between the plates passes a small amount of leakage current and also has an electric field strength limit, resulting in a breakdown voltage, while the conductors and leads introduce an undesired inductance and resistance.



Parallel plate model



Dielectric is placed between two conducting plates, each of area A and with a separation of d.

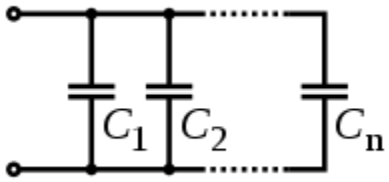
The simplest capacitor consists of two parallel conductive plates separated by a dielectric with permittivityε (such as air). The model may also be used to make qualitative predictions for other device geometries. The plates are considered to extend uniformly over an area A and a charge density ±ρ = ±Q/A exists on their surface. Assuming that the width of the plates is much greater than their separation d, the electric field near the centre of the device will be uniform with the magnitude E = ρ/ε. The voltage is defined as the line integral of the electric field between the plates

$$V = \int_0^d E\mathrm{d}z = \int_0^d \frac{\rho}{\varepsilon}\mathrm{d}z = \frac{\rho d}{\varepsilon} = \frac{Qd}{\varepsilon A}.$$

Solving this for C = Q/V reveals that capacitance increases with area and decreases with separation

$$C = \frac{\varepsilon A}{d}.$$

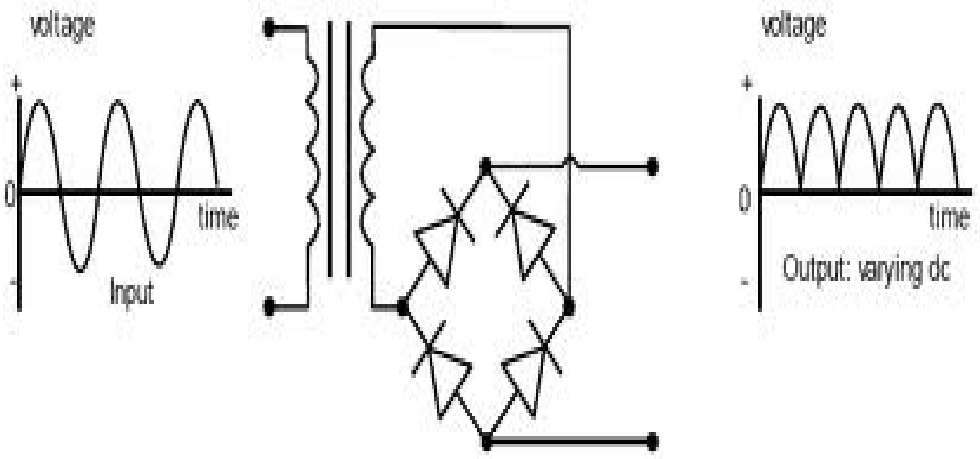
The capacitance is therefore greatest in devices made from materials with a high permittivity.



Several capacitors in parallel

7. RECTIFIERS

A rectifier is an electrical device that converts alternating current (AC), which periodically reverses direction, to direct current (DC), current that flows in only one direction, a process known as rectification. Rectifiers have many uses including as components of power supplies and as detectors of radio signals. Rectifiers may be made of solid statediodes, vacuum tube diodes, mercury arc valves, and other components. The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification. In positive half cycleonly two diodes (1 set of parallel diodes) will conduct, in negative half cycle remaining two diodes will conduct and they will conduct only in forward bias only.



8. LED

A Light-Emitting Diode (LED) is a semiconductor light source. LEDs are used as indicator lamps in many devices, and are increasingly used for lighting. When a light-emitting diode is forward biased (switched on), electrons are able to recombine with holes within the device, releasing energy in the form of photons.

This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy gap of the semiconductor. An LED is often small in area (less than 1 mm^2), and integrated optical components may be used to shape its radiation pattern. LEDs present many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved robustness, smaller size, faster switching, and greater durability and reliability.

Types of LED'S

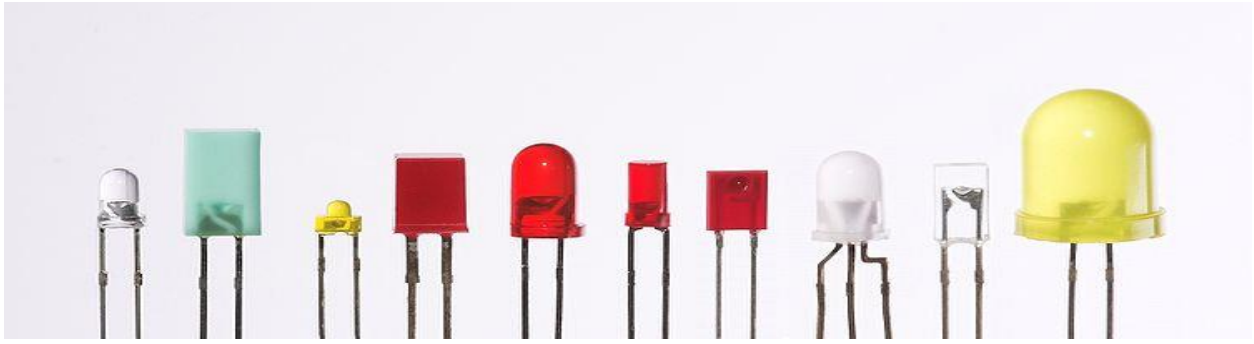


Fig 8(a): Types of LEDs

Light-emitting diodes are used in applications as diverse as replacements for aviation lighting, automotive lighting as well as in traffic signals. The compact size, the possibility of narrow bandwidth, switching speed, and extreme reliability of LEDs has allowed new text and video displays and sensors to be developed, while their high switching rates are also useful in advanced communications technology.

Electronic Symbol:

Fig 8.1(b): Symbol of LED

White LED'S

Light Emitting Diodes (LED) have recently become available that are both white and bright, so bright that they seriously compete with incandescent lamps in lighting applications. They are still pretty expensive as compared to a GOW lamp but draw much less current and project a fairly well focused beam.

When run within their ratings, they are more reliable than lamps as well. Red LEDs are now being used in automotive and truck tail lights and in red traffic signal lights. You will be able to detect them because they look like an array of point sources and they go on and off instantly as compared to conventional incandescent lamps.

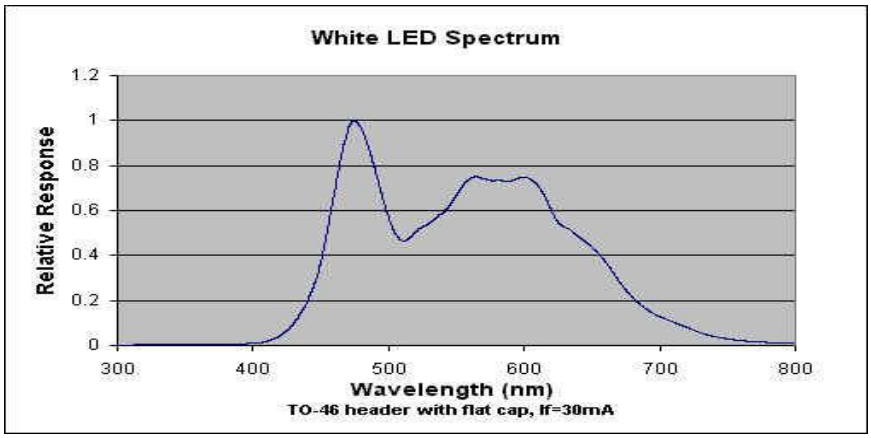


Fig 8.1(c): White LED spectrum

9. PROJECT DESCRIPTION

Light animations are visually appealing and hence widely used for advertising purposes. In this project, we present a MATLAB-based graphical user interface (GUI) approach to control the glowing pattern of a number of light-emitting diodes (LEDs). Use of GUI is advantageous since the user can control illumination patterns while performing other tasks in the PC.

This project creates five different lighting patterns including ring counter and Johnson counter by clicking appropriate pushbuttons in the GUI. The blinking speed of LEDs can also be controlled using fast, normal and slow pushbuttons in the GUI. The author's prototype is shown in Fig. 1 and the MATLAB-based GUI in Fig. 2.

Circuit and working of Project

The circuit for controlling light animations using Arduino is shown in Fig. 3. It consists of an Arduino Uno board, eight LEDs and eight 1-kilo-ohm resistors.

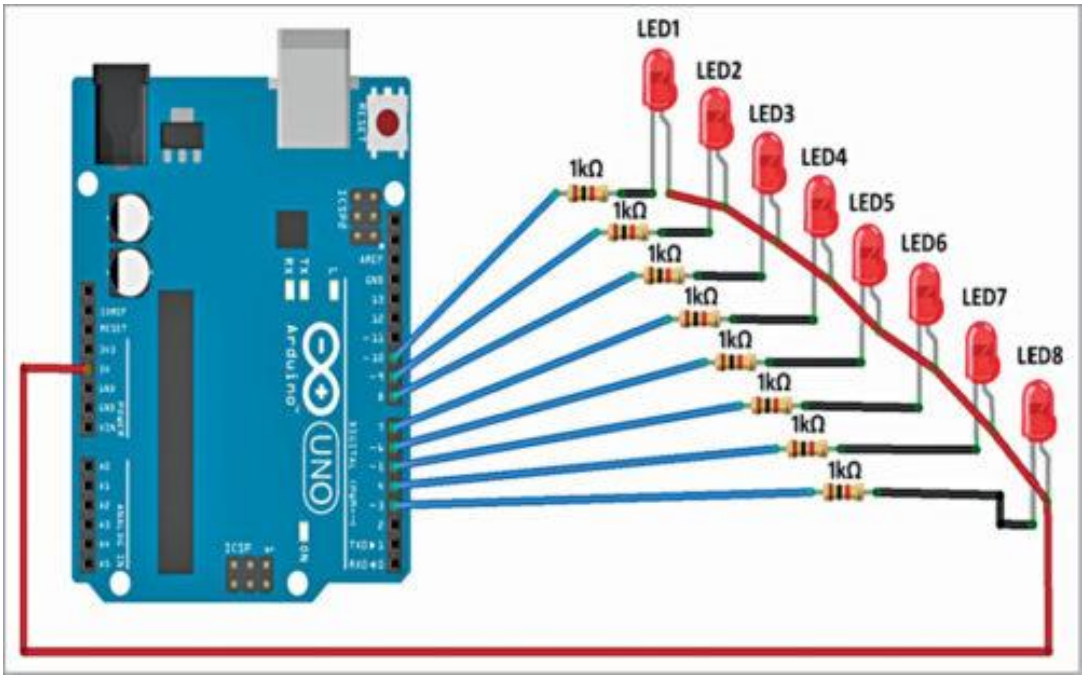


Fig (9) Project Circuit Diagram

Arduino Uno is an AVR ATmega328P microcontroller based development board with six analogue input pins and 14 digital input/output (I/O) pins. The microcontroller has 32kB of ISP flash memory, 2kB RAM and 1kB EEPROM. The board provides serial communication via UART, SPI and I2C. The microcontroller can operate at a clock frequency of 16MHz. In this project, digital I/O pins 3 through 10 of the Arduino are configured as output pins and used to control the illumination of eight LEDs.

When a pushbutton corresponding to a particular pattern is pressed in the GUI, it executes the corresponding callback function in the source program (Illumination001.m.) The callback function executes program statements corresponding to that pattern and sends High or Low control signals to appropriate pins of the Arduino in order to create the desired glowing pattern. Eight series-connected 1-kilo-ohm resistors limit current flow through LEDs.

10. CODING

```
function varargout = Illumination001(varargin)
% ILLUMINATION001 MATLAB code for Illumination001.fig
%      ILLUMINATION001, by itself, creates a new ILLUMINATION001 or raises
the existing
%      singleton*.
%
%      H = ILLUMINATION001 returns the handle to a new ILLUMINATION001 or
the handle to
%      the existing singleton*.
%
%      ILLUMINATION001('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in ILLUMINATION001.M with the given input
arguments.
%
%      ILLUMINATION001('Property','Value',...) creates a new
ILLUMINATION001 or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Illumination001_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Illumination001_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Illumination001

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Illumination001_OpeningFcn, ...
                   'gui_OutputFcn',  @Illumination001_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT


% --- Executes just before Illumination001 is made visible.
function Illumination001_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Illumination001 (see VARARGIN)

% Choose default command line output for Illumination001
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Illumination001 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Illumination001_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
% Get default command line output from handles structure
varargout{1} = handles.output;
clear a;
global delay;
delay=0.5;
global a;
a = arduino('COM8');

for i=3:10
    a.pinMode(i,'output');
end
for i=3:10
    a.digitalWrite(i,1);
end

% --- Executes on button press in Pattern1.
function Pattern1_Callback(hObject, eventdata, handles)
% hObject    handle to Pattern1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data)global a;
global delay;

while 1
    for l=3:10
```

```
        a.digitalWrite(l,1);
    end

    for i=1:5
        for n=3:10
            a.digitalWrite(n,0);
            pause(delay);
            a.digitalWrite(n,1);
            pause(delay);
        end
    end
end

% --- Executes on button press in Pattern2.
function Pattern2_Callback(hObject, eventdata, handles)
% hObject    handle to Pattern2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global a;
global delay;
while 1
    for m=3:10
        a.digitalWrite(m,1);
    end

    for i=3:10
        for j=3:i
            a.digitalWrite(j,0);
            pause(delay);
        end

        for m=3:10
            a.digitalWrite(m,1);
        end
        pause(delay);
    end
end


% --- Executes on button press in Pattern3.
function Pattern3_Callback(hObject, eventdata, handles)
% hObject    handle to Pattern3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global a;
global delay;
while 1
    for j=3:10
        a.digitalWrite(j,1);
    end
    pause(delay);
    for k=3:10
        a.digitalWrite(k,0);
```

```
        pause(delay);
    end
    for k=3:10
        a.digitalWrite(k,1);
        pause(delay);
    end
end


% --- Executes on button press in FAST.
function FAST_Callback(hObject, eventdata, handles)
% hObject    handle to FAST (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global delay;
delay=0.2;


% --- Executes on button press in SLOW.
function SLOW_Callback(hObject, eventdata, handles)
% hObject    handle to SLOW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global delay;
delay=0.8;


% --- Executes on button press in Pattern4.
function Pattern4_Callback(hObject, eventdata, handles)
% hObject    handle to Pattern4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global a;
global delay;

while 1
    for i=3:10
        a.digitalWrite(i,1);
    end

    for n=3:10
        a.digitalWrite(n,0);
        pause(delay);
        a.digitalWrite(n,1);
        pause(delay);
    end
    for n=10:-1:3
        a.digitalWrite(n,0);
        pause(delay);
        a.digitalWrite(n,1);
        pause(delay);
    end
end
```

```
% --- Executes on button press in NORMAL.
function NORMAL_Callback(hObject, eventdata, handles)
% hObject    handle to NORMAL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global delay;
delay=0.5;


% --- Executes on button press in Pattern5.
function Pattern5_Callback(hObject, eventdata, handles)
% hObject    handle to Pattern5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global a;
global delay;
while 1
    for m=3:10
        a.digitalWrite(m,1);
    end

    for i=3:10
        for j=3:i
            a.digitalWrite(j,0);
            pause(delay);
        end
    end

    for m=10:-1:3
        for j=10:-1:m
            a.digitalWrite(j,1);
            pause(delay);
        end
    end
        pause(delay);
end


% --- Executes on button press in STOP.
function STOP_Callback(hObject, eventdata, handles)
% hObject    handle to STOP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data
global a;
clear all
```

11. SOFTWARE

We used Arduino IDE to program the Arduino Uno., install the Arduino IDE and note the installation directory.

We developed 'Legacy MATLAB and Simulink Support for Arduino' package. We Connect the Arduino Uno board to PC. From Device Manager, note the COM port at which the Arduino Uno board is connected. From Tools menu in the Arduino IDE, select the board as Arduino Uno, and the COM Port number noted earlier. Upload code to the Arduino Uno board by pressing Upload button in the IDE.

This GUI application program has been developed in R2014a version of MATLAB. After installing this MATLAB version in PC, open the file install_arduino.m present in the directory where you copied contents of Arduino IO folder. This code will correctly install and save the path of the Arduino support package.

Edit the line a=arduino('COM9') with the COM port number in your PC where the Arduino Uno board has been connected. When you run the project file, MATLAB will try to communicate with the board. After successful communication is established, we can control LEDs by pressing the corresponding pushbuttons in the GUI.

12. RESULT

We have implemented project with 100% accuracy. During the testing process of project we find Five different lighting patterns including ring counter and Johnson counter by clicking appropriate pushbuttons in the GUI. The blinking speed of LEDs can also be controlled using fast, normal and slow pushbuttons in the GUI.

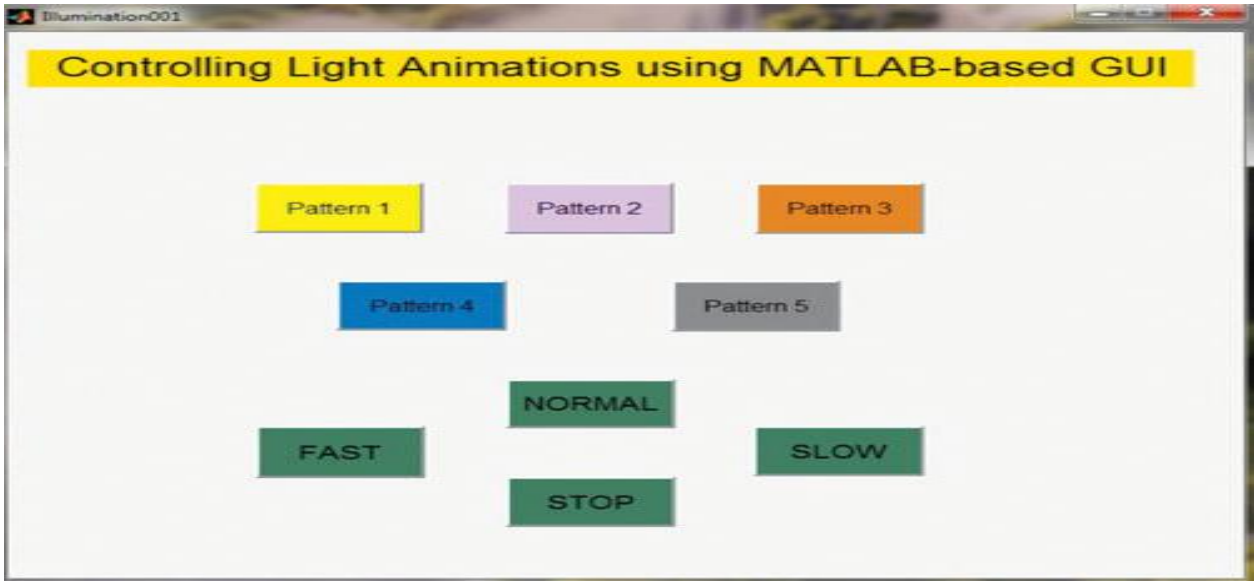


Fig 12(a): MATLAB-based GUI

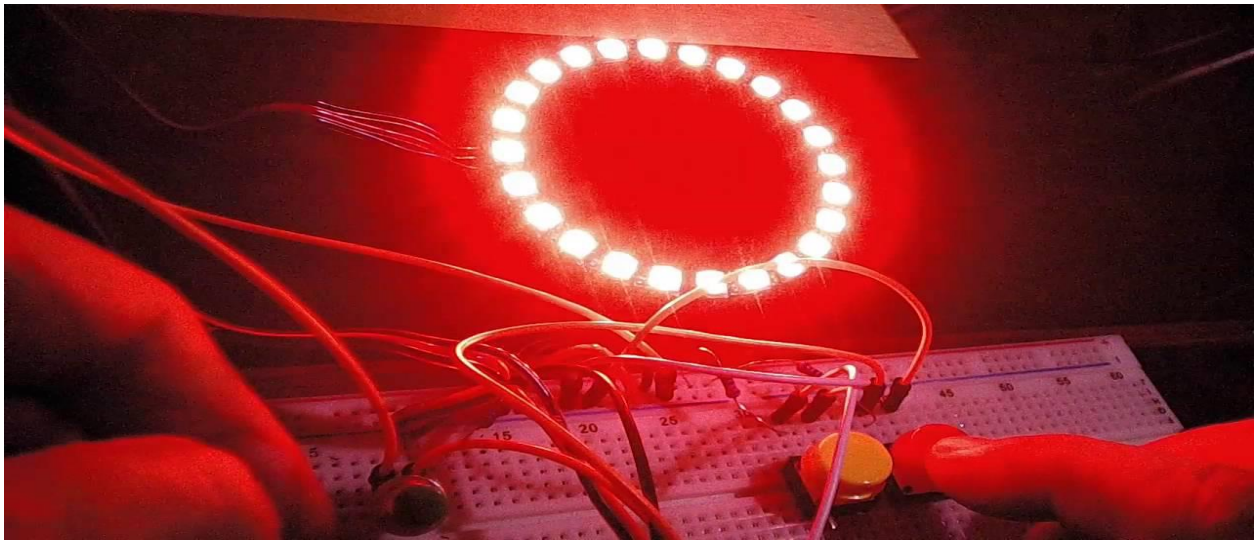


Fig 12(b) Prototype of the project

13. CONCLUSION

The Project which we designed has its focal point on controlling different lighting patterns including ring counter and Johnson counter by clicking appropriate pushbuttons in the GUI. Use of GUI is advantageous since the user can control illumination patterns while performing other tasks in the PC. Light animations are visually appealing and hence widely used for advertising purposes.

14. BIBLIOGRAPHY

TEXT BOOKS REFERED:

1. Arduino Programming: The Ultimate Beginner's Guide to Learn Arduino Programming by Ryan Turner.
2. Understanding Matlab: A Textbook for Beginners by S.S. Alam S.N. Alam
3. MATLAB for Machine Learning by Giuseppe Ciaburro
4. "The ATMEGA Microcontroller and Embedded systems" by Mazidi.
5. MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence by Phil Kim
6. ATMEGA 328 Data Sheets.

WEBSITES

- www.mathworks.com
- www.atmel.com
- www.arduino.cc
- www.wikipedia.org
- www.alldatasheets.com