

“ARDUINO BASED XY-PLOTTER”

By

Aarushi Gupta – 101063

Shradha Pandey – 101067

Anamika Sarkar – 101141

Under the Supervision of

Mr. Munish Sood



May, 2014

Submitted in partial fulfillment of the Degree of
Bachelor of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT

TABLE OF CONTENTS

	List of figures	iv
	List of Tables	vi
	Certificate from Supervisor	vii
	Acknowledgement	viii
	Summary	ix
Chapter-1	Introduction	1
1.1	History of Plotter	1
1.2	Block diagram	6
Chapter- 2	Technical Details	7
2.1	How is Digital image stored in Pc?	7
2.2	GLCD Library	8
2.3	Bitmap2LCD	8
2.4	Voltage Regulator	8
2.5	Arduino Module	10
2.6	Push Button Switches	15
2.7	Slotted opto isolators	15
2.8	Potentiometer	18
2.9	Comparator	19
2.10	Pull up resistors	22
2.11	Optocoupler	23
2.12	H-Bridge	26
2.13	DC geared motor	29
2.14	Wooden Work	37
Chapter -3	Implementation in the Project	38
3.1	Hardware	38
3.1.1	Voltage regulator in the project	38
3.1.2	Arduino in the project	39
3.1.3	Switches in the project	39
3.1.4	Slotted coupler in the project	39
3.1.5	Potentiometer in the project	40

3.1.6	Comparator in the project	40
3.1.7	Pull up resistors in our project	41
3.1.8	Optocoupler in the project	42
3.1.9	H-bridge in project	43
3.2	Software	45
3.2.1	BITMAP2LCD v 1.5a	45
3.2.2	ARDUINO v1.0.5 windows	47
3.2.3	Plotting Process	51
3.2.4	Algorithm	52
Chapter-4	Result	54
Chapter-5	Future Scope	55
APPENDIX		57
REFERENCES		71

LIST OF FIGURES

FIGURE NO.	PAGE NO.
Figure 1: Calcomp565	2
Figure 2: Flatbed plotter	2
Figure 3: HP 7470	3
Figure 4: Evolution of printer	3
Figure 5: Vinyl cutter	4
Figure 6: Static cutting table	5
Figure 7: Block Diagram	6
Figure 8: Gray code of letter 'a'	7
Figure 9: Positive and negative voltage regulator	9
Figure 10: Voltage regulator 7805	9
Figure 11: Arduino Uno Module	14
Figure 12: Push Button Switches	15
Figure 13: Slotted opto isolator	16
Figure 14: Internal connection of IC MOC7811	17
Figure 15: Trimmer resistance	18
Figure 16: Voltage divider	18
Figure 17: Three lugs of potentiometer	18
Figure 18: Potentiometer 103	19
Figure 19: Symbolic representation of comparator	19
Figure 20: Pin diagram of LM358	21
Figure 21: General connection of pull up resistance with button switch	22
Figure 22: Optocoupler	24
Figure 23: Internal circuitry of FL817F343	25
Figure 24: LED driving a phototransistor	26
Figure 25: H-bridge	27
Figure 26: Current Flow in H-bridge	27
Figure 27: Different combinations of switches	28
Figure 28: Gear motor	29
Figure 29: Dc motor internal structure	31


Figure 30: Left hand rule	31
Figure 31: Vector diagram of field	32
Figure 32: Principle of DC Motor	34
Figure 33: Armature at $\alpha=0$	35
Figure 34: Armature at $0<\alpha<90$	35
Figure 35: Armature at $\alpha=90$	36
Figure 36: Voltage regulator circuitry	38
Figure 37: Switch circuitry	39
Figure 38: Comparator circuitry	41
Figure 39: A103J	41
Figure 40: A103J internal circuitry	42
Figure 41: Circuitry of motor driver	43
Figure 42: H-Bridge circuitry	45
Figure 43: Screen-shot of bitmap2lcd	46
Figure 44: Screenshot of encoded table of constants	46

LIST OF TABLES

TABLE NO.	PAGE NO.
Table 1: Features of Arduino Uno	11
Table 2: Comparator output operation	20
Table 3: Quadrants of H-bridge	28
Table 4: Actions of H-Bridge	44

CERTIFICATE

This is to certify that the work titled “**Arduino based XY Plotter**” is submitted by **Aarushi Gupta, Shradha Pandey, Anamika Sarkar** in the partial fulfillment for the award of degree of Bachelor of Technology in Jaypee University of Information Technology, Wagnaghat, Solan has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for award of this or any other degree or diploma.

Signature of supervisor --- 
Name of Supervisor --- MUNISH SOOD
Designation --- Assistant Professor Grade II
Date --- 27th May 2014

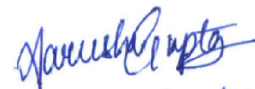
ACKNOWLEDGEMENT

We would like to express our sincere thanks to Mr. Munish Sood, Department of Electronics and Communication Engineering, for his constant encouragement and guidance during this project. His continuous involvement in the progress of our project was of great help to us. We would like to thank him for his valuable suggestions in the project which improved the usefulness of the system as a whole.

We would also like to thank Mr. Mohan Sharma and Mr. Pandey, for their consistent support during the project.

The zeal to accomplish the task of formulating the project could not have been realized without the support and cooperation of faculty members of ECE Department. We are sincerely thankful to Prof. (Dr.) T. S. Lamba, Dean (A & R) and Prof. (Dr.) Sunil V. Bhooshan, HOD(ECE) for their consistent support throughout the project work. We would also like to show our gratitude to the Project Coordinator, Mrs. Meenakshi Sood for her help.

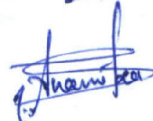
Name of the students



Aarushi Gupta



Shradha Pandey



Anamika Sarkar

SUMMARY

In this project, we have developed a two axes plotter i.e. x-axis and y-axis. Although we are using a third axis i.e. z-axis but that is used for the purpose of inscribing a mark on the sheet. All these axes are controlled using the Arduino Module. We decided to use the Arduino Module over the commonly used 8051 microcontroller because it is a new emerging powerful module. Basically, the image to be plotted is fed into the arduino after it is converted into the table of constant which consists of data regarding the positioning of black and white pixels. This table of constant is formed using the Bitmap2lcd software. After that, using the arduino software's sketch, this table of constant is plotted as the image on the sheet.

CHAPTER 1 INTRODUCTION

An X-Y plotter as by its name, is a plotter that operates in three axes of motion—x-axis, y-axis and z-axis, in order to draw raster graphics or the vector graphics. Raster graphics are the graphics, which are stored as the dot matrix data structure consisting the rectangular grid of the pixels whereas, the vector graphics are the ones which can be represented using the geometric primitives having the mathematical equations i.e. lines, polygons, points, shape or a curve. The term XY plotter was used to differentiate it from standard plotters. These standard plotters had control only over the "y" axis, and the "x" axis was being continuously fed to provide a plot of some variable with time. Plotters are different from Inkjet and Laser printers which are very common these days. Plotter inscribe the dots which are commonly known as pixels in the digital image world in case of raster image and for the vector graphics it is more like a pen on paper. On the other hand, inkjet and laser printers use a very fine matrix of dots to form images, such that while a line may appear continuous to the naked eye, it in fact is a discrete set of points. But with the X-Y plotter for raster graphic, even though they plot bitwise, we don't have the very fine points and thus they appear distinctly on the sheet.

The x-axis and y-axis motors are being used for plotting according to the coordinate of pixels of the image, on the x-y plane of the canvas. X-Y plotter is using z-axis to raise and to lower the pen for inscribing the dot. Each axis is powered using the dc motor. All the electronics to move the motors accordingly, are connected to the Arduino, directly or indirectly and powered by a 12 volt wall adaptor.

1.1) HISTORY OF PLOTTER^[1]

The plotter is a printer like device used for printing vector graphics. In past, plotters were used in computer –aided design like applications, though they have been replaced with wide-format conventional printers. A plotter draws pictures on paper using pen or marker. Plotters are mainly used to design ships and machines, plotting plans for buildings and so on.

1.1.1) 1959- *CALCOMP 565*

Calcomp565, drum plotter was one of the first computer graphics output devices. The computer could control the rotation of an 11 inch wide drum and the horizontal movement of pen holder over the drum. Pen was pressed against the paper scrolling across the drum. The pen is connected to

solenoids that lift the pen off the paper. It needs the perforation along the edges. This arrangement allows the pen to draw lines on the paper under computer control.



Figure 1: Calcomp 565

1.1.2) LATE 1960'S AND 1970'S- FLATBED PLOTTER

It is a small desktop-sized plotter. It has a travelling bar over which pens were mounted. The X-axis represented by the motion of the bar back and forth across the plotting table whereas the Y-axis represented by the motion of the pen along the length of the bar. This plotter is relatively slow because of the mass of the bar.



Figure 2: Flatbed plotter^[2]

1.1.3) 1980- HP 7470

It is a small and light weight plotter. It is based on the 'grit-wheel' mechanism. It does not require the perforation along the edges. It has urethane-coated rollers, the grit wheels at the opposite edges of the sheet press against the rollers and form tiny indentation in the sheet. The grit wheels keeps the sheet in proper registration as the sheet move back and forth, much like teeth of two gears

meshing. The carriage with pen moves back and forth in a line between the grit wheels, it is just acts as the orthogonal axis.

This small ‘home-use’ plotter became popular for desktop business graphics and engineering laboratories, but as they were very slow they were not suitable for general purposes



Figure 3: HP 7470^[3]

Conventional printers would be required for general purposes. Hewlett Packard introduces ‘word chart’ for HP 2647, which used the plotter to plot large letters on a transparency. This was the fore runner of modern PowerPoint chart. With the availability of high resolution inkjet and laser printers, inexpensive memory and fast computing overpowered the pen plotters. However, the grit wheel mechanism is still used in inkjet-based printers and large format engineering plotters.



Figure 4: Evolution of printer

1.1.4) VINYL CUTTER

Vinyl cutter is also known as a cutting plotter. It is basically used by the businesses which makes professional posters and bill board signs. It is popularly used to produce sign boards which are weather resistant, posters, and billboards can be created which has adhesive on its back side and

uses self-coloured vinyl film. For large, bright posters used for advertisements the vinyl is applied to car bodies and windows, and to sailboat transoms. It is used to cut tinted vinyl for automotive windows. The hardware of vinyl cutter is almost similar to the traditional plotter except that the pen is replaced by a knife or cutter that is use to cut out shape and in addition the plotter has a system to control pressure that decide how hard the knife to be pressed to cut out the shape. The vinyl knife is shaped just like the traditional plotter pen and to make sure the knife edge should rotate correctly, the knife is mounted over the ball bearings.



Figure 5: Vinyl Cutter^[4]

1.1.5) STATIC CUTTING TABLE

It is a type of cutting plotter used large vacuum table for operations. It is used for cutting porous material basically those materials which are not rigid such as clothes, foam, or leather that too difficult or say impossible to cut with plotters uses rollers. Static cutters also have the ability to cut much thicker and heavier materials compared to a typical roll-fed plotter. A series of small pinholes drilled on the surface of table. The material to be cut is placed on the table and a coversheet of plastic or paper is overlaid onto the material. Air pressure created by the vacuum pump pushes down the coversheet to hold the material in place. The table then operates just like a normal vector plotter.



Figure 6: Static cutting table^[5]

1.1.6) MID-TO-LATE 2000

The artists and hackers began to rediscover the other attractive use of pen plotter as customizable output devices. The lines that are drawn by the pens of plotter on paper are quite different from other outputs produced by other digital techniques. Quite old pen plotters are still function reliably, and many are available for a minimal amount on auctions or other resale websites. While support for driving pen plotters has disappeared from most applications that uses commercial graphics. Nowadays many different types of printers become popular in the market and because they are more efficient, fast and easy to use, plotters are very rare to see in business.

1.1) BLOCK DIAGRAM

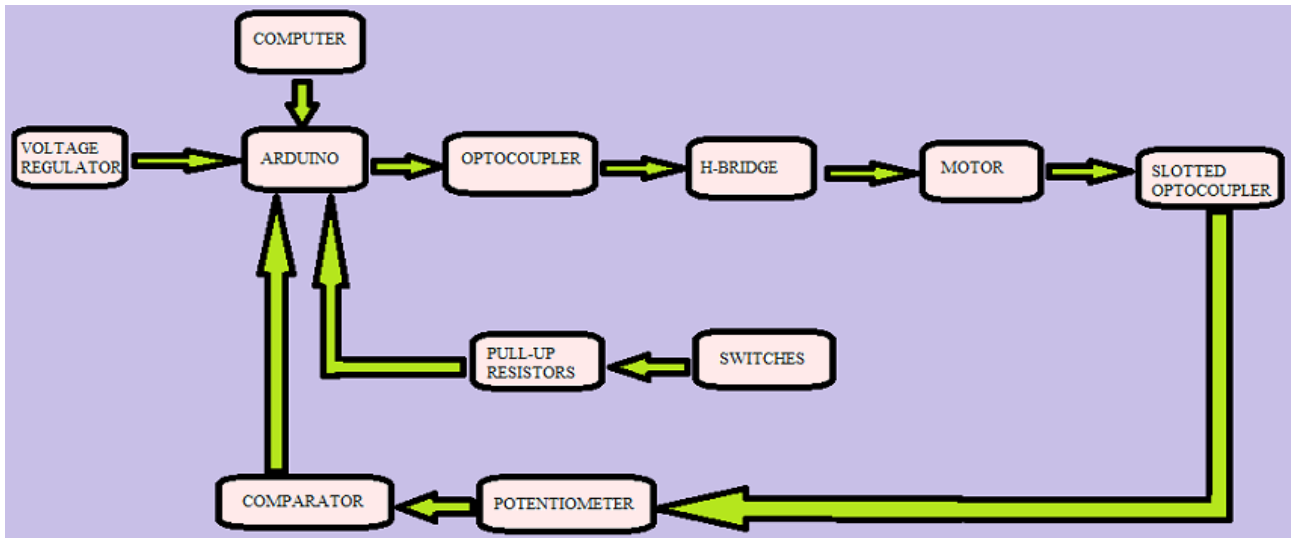


Figure 7: Block Diagram

The images which are to be plotted are converted into the bit form and stored in the memory of Arduino. There are switches which are used to perform specific tasks which depend upon the user. The pull up resistors are used to prevent floating state of the Arduino pins. When we press any switch, Arduino works accordingly and commands the motor to move in a specific direction with the help of motor driver circuitry which consists of optocoupler and H-Bridge. Three motors are used for the motion of x, y and z axes. An interrupt pin is connected to the axes along which each x and y directions' motor rotates. The slotted optocouplers are related to the x and y motors. When the motor rotates, slotted optocoupler detects the interruption. The potentiometer and comparator tells the Arduino that the interrupt had occurred in return Arduino reads the value of the next pixel and commands the motors to rotate according to the Arduino sketch. The Arduino sketch finds out the positioning of the different black pixels and these pixels are plotted on the paper sheet with the help of the z-axis motor, which makes the pen to put a mark on the required coordinates of the xy plane.

CHAPTER 2: TECHNICAL DETAILS

2.1) HOW IS DIGITAL IMAGE STORED IN PC? [6,7,8,9]

A digital image stored in memory is the digitized form of every image. It means that a rectangular picture is divided into large number of squares. It is stored as the numeric representation (normally binary) of a two-dimensional image. It can be vector type or raster type, depending on whether the image resolution. If in a zoomed in picture, the curves of the picture are smooth then it is a vector image while, if the curves are not smooth and we are able to see the pixels present, then it is a raster or bitmapped image. It is difficult to store and transmit vector image. Thus, the vector images are hardly used and in general, the term "digital image" usually refers to raster image or bitmapped images.

A Raster image is the one which have finite set of digital values, called *picture elements* or *pixels*. The digital image contains a fixed number of rows and columns of pixels. Pixels are the smallest individual element in an image, holding quantized values that represent the brightness of a given colour at any specific point.

The letter 'a' might be represented in a 12x14 matrix as depicted in Figure given below.

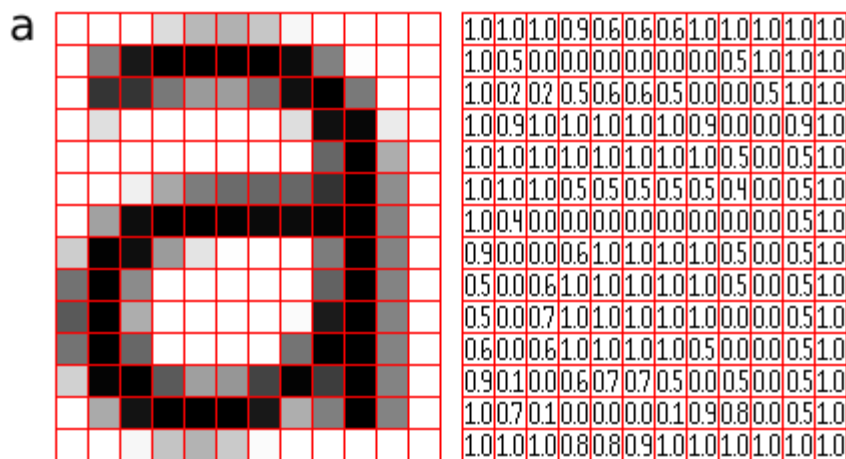


Figure 8: Gray code of letter 'a'^[9]

Typically, the pixels are stored in computer memory as a raster image or raster map, which is a two-dimensional array of small integers.

As we know that, **byte** is a unit of digital information stored in computing and telecommunications. It is also the unit of image stored. An image stored is generally mentioned as M X N byte matrix. Also, 1 byte is equal to eight bits.

2.2) GLCD Library^[11]

GLCD is the software with Tools that enable us to convert windows bitmap files (bmp) to a byte array suitable for graphic LCD displays. Additionally it can be used to create the font sets of our own, out of the system fonts for the displays. Its output is a C Code which is easily usable in a microcontroller project. Although we are not using this software for the encoded output, we are only using it for the library purpose because it is easily imported in the arduino.

2.3) BITMAP2LCD^[10]

It is a design and programming tool for the monographic small LCDs up to 320X240. It is useful in importing bitmaps, paint pictograms with creative inspiration and it can also present our creation into table of constants for its use in different compilers, in our case it is arduino. This tool runs on recent computers with Microsoft XP, Vista, Window 7 and 8. Screen resolutions required is 1024 x 768 or higher. Its target is to offer the most universal tool, compatible with most LCD controllers of the market and the most programming languages and development tools. It allows us to configure this tool according to your own hardware, software and our requests.

2.4) VOLTAGE REGULATOR

Voltage regulator is a device that regulates the input voltage within an acceptable limit. The voltage regulator is needed where it is necessary to keep voltages within the specific range so that the equipment using this voltage can tolerate it without any damage.

The main purpose of a voltage regulator is to provide voltage in a circuit which is merely close to the desired value. Since a power supply produces raw current frequently that would cause damage to the components of the circuit which may cause the failure of the circuit. Electronic voltage regulators are solid-state semiconductor devices which can smoothen out the variations in the flow of current. In many of the cases, they operate as the variable resistances; that is, resistance decreases when the electrical load is heavy and increases when the load is lighter. The 78xx is a series of integrated circuits which can regulate variable voltage to a fixed linear voltage. These ICs used in the circuits in which a regulated power supply needs to operate. They are small, cheap and easy-to- use. The 78xx series is positive voltage regulators means they produce a voltage that is positive relative to a common ground. The series of 79xx devices produces complementary negative voltage regulators^[12].

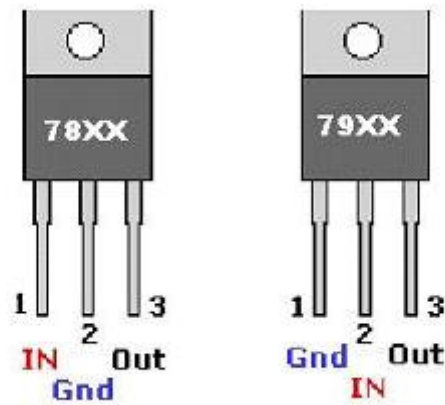


Figure 9: positive and negative voltage regulators^[1]

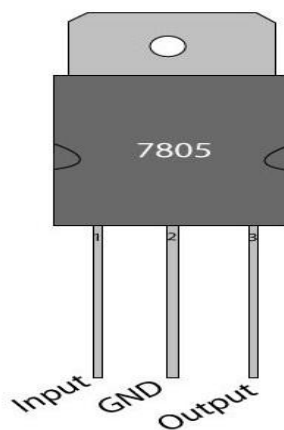


Figure 10: voltage regulator 7805^[13]

2.4.1) ADVANTAGE OF 78XX^[1]

- ICs of this family does not require any other components to provide constant and regulated voltage makes them easy to use, as well as economical and efficient uses of space.
- 78xx series ICs have built-in protection against a circuit drawing too much power. They have protection against overheating and short-circuit; these features make them quite robust in most applications. The current-limiting feature of the 78xx devices not only provides protection to itself, but also to the other components of the circuit.

2.4.2) DISADVANTAGE OF 78XX^[1]

- The input voltage always be higher than the output voltage by some minimum amount (typically 2.5 volts)
- This is the requirement of 78xx that the input voltage should always be higher than the voltage desired as the output; this means that the power that is supplied to 78xx as input will

be more than the output power. The extra input power is dissipated as heat. This means that for some applications an adequate heat sink must be provided, and it is also clearly noticeable that a portion of the input power is wasted during this process, which makes them less efficient.

2.5) ARDUINO MODULE^[14]

Arduino is an open-source electronics prototyping platform which is flexible and has easy-to-use hardware and software. It can be used to develop creative objects and that's why it mainly intended for artists, designers, hobbyists and anyone who is creative.

It takes input from variety of switches and sensors, and controlling of variety of motors, lights and other physical outputs. Projects based on Arduino can be stand-alone; they can communicate with software running on the computer. The Arduino board can be manually assembled or purchased preassembled; the open source IDE can be downloaded for free. The Arduino programming language is nothing but the implementation of wiring. It is similar to the physical computing platform, which is based on the programming needed in multimedia environment.

2.5.1) ARDUINO Uno

Uno is an Italian count equivalent to one and it named to mark the release of Arduino 1.0.

The Arduino Uno is a microcontroller board which uses ATmega328. It consists of 14 digital i/o pins. In those 14 pins, 6 pins are used for PMW outputs, another 6 pins are analog inputs. It has 16 MHz ceramic resonator, a USB connection that is use to connect the Arduino to system, a power jack for power supply. It also has an ICSP header, and a reset button which is used to reset the codes stored in the Arduino microcontroller. The board has everything which is needed to support a microcontroller to function. Arduino can be powered up either by connecting it to the computer through USB cable or by using AC-to-DC adapter or battery which is connected to power jack.

2.5.2) FEATURES OF ARDUINO Uno

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Table 1: Features of Arduino module^[14]

2.5.3) POWER

We can power up Arduino Uno board by two ways firstly we can use USB cable; secondly, power supply can be external one.

External power supply that is non-USB can either be an AC-to-DC adapter or a battery. The adapter has a centre-positive plug of 2.1mm that can be connected into the on board power jack. The leads of the battery should be inserted in the pins of power connector one of which is Gnd and another one is the V_{in} pin.

The board needs 6 to 20 volts provided externally to operate. If the voltage supplied to the board is less than 7 volts the operations of the board may be unstable as 5V pin may supply voltage that will be less than 5 volt. If the supplied voltage is greater than 12 volt the voltage regulator may overheat and it may cause damage to the board. So the voltage should be in the range of 7 to 12 volts.

The power pins of the Arduino Uno board are as follows:

- V_{in} - it is used to provide external voltage to the board (via power jack).
- 5V- this pin gives a regulated 5V as its output from the regulator present on the Arduino board. If we are providing power to the board through the DC power jack the input voltage

to the board will be in range of 7V to 12V, if supply is through the USB connector the input voltage will be 5V, else if we are using the VIN pin of the board then the range will be 7V to 12V. Voltage supplied via the 5V or 3.3V pins bypasses the on board regulator, and can cause damage to board.

- 3V3- this pin outputs regulated 3.3V from the on board regulator.
- GND- it is the ground pin
- IORF- It provides the reference voltage with which the ATmega328 operates.

2.5.4) MEMORY

The ATmega328 has memory of 32 KB. 0.5 KB of this memory is used for the bootloader. It has SRAM (static random access memory) of 2 KB and also have 1 KB of EEPROM which is used to read and write with the EEPROM library.

2.5.5) INPUT AND OUTPUT

It has the 14 digital pins that can be used as an input or output pins. Each needs 5volts to operate. Each pin has an internal pull-up resistor of 20-50 kOhms which is disconnected by default and able to provide or receive 40 mA.

Some pins have specialised functions:

- *Serial (0 and 1 pin)* –It is used to receive (pin 0) and transmit (pin 1) TTL serial data. These pins are connected to USB-to-TTL Serial chip pins of the ATmega8U2 .
- *External Interrupts (2 and 3 pin)*- These pins can be programmed to generate an interrupt on a that either be a low value, a rising or falling edge, or there is a change in value
- *PWM: (3, 5, 6, 9, 10, and 11 pin)* - these pins used to give 8-bit PWM output.
- *SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) pin* - These pins uses SPI library for SPI communication.
- *LED (13 pin)* - At digital pin 13 a built-in LED is connected. The LED glows when the pin is high, else it's off.
- There are 6 analog input pins; labelled as A0, A1, A2, A3, A4 and A5, each of which has resolution of 10 bits that is total 1024 different values can be provided. They measure 5V by default from ground, but the upper end of their range can be changed by using the pin labelled as AREF.
- *TWI: (A4 and A5 pin)* - A4 is also known as SDA pin and A5 is also known as SCL pin. This two pin uses wire library to supports TWI communication.

There are some other pins on the board:

- *AREF*- It is used to provide a reference voltage for the analog inputs. It is Used with `analogReference()`.
- *Reset*- It is used to reset the microcontroller. Basically used to add a reset button to shields which is used to block the one present on the board.

2.5.6) COMMUNICATION

The Arduino Uno board has the ability to communicate with the computer, other Arduino, or the other microcontrollers. The ATmega328 has the ability to communicate serially with the help of digital pin 0 which is used to receive and digital pin 1 is used to transmit. The '16U2 microcontroller needs no external driver; it only uses the standard USB COM drivers. The software used to program Arduino module needs a serial monitor, through which simple textual data are sent to and from the Arduino board. When data is transmitted through USB-to serial chip and USB connected to the computer the on board LEDs of RX and TX will glow.

Any of the digital pins of the Uno board can be enabled for serial communication by using Software Serial library.

2.5.7) USB OVERCURRENT PROTECTION

The Arduino Uno has a polyfuse which is resettable by nature and helps to protect the USB ports of the computer that is used from being short or overcurrent. Generally most of computers have their own subsystems that provide internal protection; the fuse purposely needed for an extra layer of protection. The fuse will break automatically whenever the current applied to the USB port is more than 500mA until the short or overload is rectified.

2.5.8) PHYSICAL CHARACTERISTICS OF ARDUINO UNO

The Uno board has the standard dimensions; the maximum length of the module is 2.7 inches and it is 2.1 inches wide. It also has the USB connector for computer interfacing and power jack to provide power; they are extending beyond the mentioned dimension. To fix the Arduino module to the surface four screw holes on the appropriate position are provided.

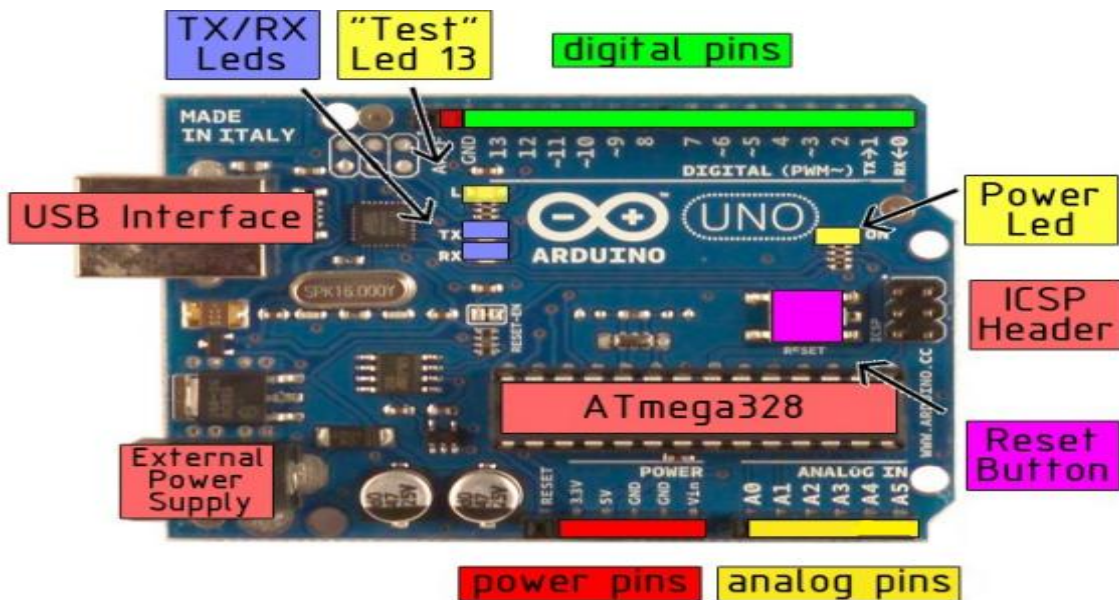


Figure 11: Arduino Uno module^[15]

2.5.9) ARDUINO vs MICROCONTROLLER

There are many other microcontroller platforms that can be used for physical computing. Some of them are Phidgets, Netmedia's BX-24, MIT's Handy board, Parallax Basic Stamp and there are many more which offer similar functionalities. All of these need the knowledge of microcontroller programming to write the code so that the microcontroller is able to perform specific function but these all are easy-to-use package. Though Arduino also simplifies the process of working and to code the microcontroller but has some additional advantages over other platforms especially it is easy to understand for teachers, students and even the interested amateurs can easily use the board.

- **Inexpensive-** Arduino Uno boards does not cost much compared to other microcontroller platforms. It is possible to assemble it manually.
- **Cross platform-** Arduino software has the compatibility with different operating systems such as Windows, Macintosh OSX, and Linux whereas most of microcontroller systems are only compatible to the windows.
- **Simple clear programming environment-**programming is quite simple and easy to use for beginners; even it is flexible enough that work for advanced user.
- **Open source and extensible software-**the software used to program Arduino is an open source tool. The language is simple, easy to understand and can be easily expanded through C++ libraries; it is based on C programming which is one of the great advantage of Arduino over AVR. We can add AVR-C code directly into your Arduino programs.

- **Open source and extensible hardware** - The circuit designers can make their own version of the module by extending or improving its functionalities, as the published plans for the modules are under a Creative Commons license.

2.6) PUSH BUTTON SWITCHES

The machine process can be control with the help of switches. There are many types of switches available in the market. Push button switches are one of the types. It works on simple switch mechanism. Push buttons usually made up of plastic or metal. The surface of the button is designed to be flat so that it can be pressed easily by human finger. Button switches are mostly biased switches means they are momentarily switches which need the user to push for on or off the process. These types of switches are used in many electronic devices like calculators, push button telephone even in many kitchen appliances. They are also use in industrial and commercial appliances.

There are two types of switches “push-to-make” and “push-to-break”. The push-to-make switches are those which when pressed makes contact with electronic system, the contact breaks when it is released. Whereas push-to-make switches breaks contact when pressed.



Figure 12: Push button switches^[16]

2.7) SLOTTED OPTO ISOLATOR^[17]

In a slotted optocoupler there is a slot molded into the package in the space between the LED light source and the phototransistor light sensor. This slot houses transparent windows, so that the LED light can easily reach to the transistor, but can be interrupted by an opaque object (if placed within the slot). These slotted optocoupler can be used in a variety of applications, including end-of-tape detection, limit switching, and liquid level detection. It is also known as **slotted optical switch**

2.7.1) WORKING OF SLOTTED OPTOCOUPLER

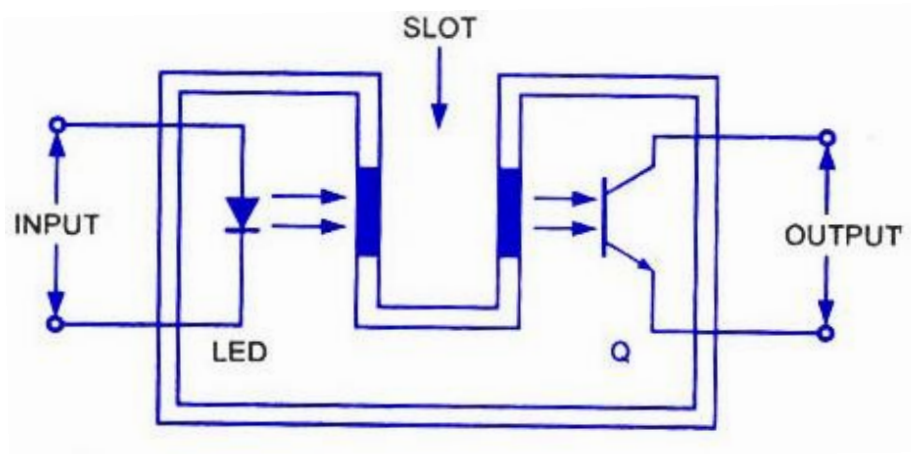


Figure 13: Slotted optocoupler^[20]

It works on following basic principle:

When the light emitted by the IR LED is blocked because of alternating slots of the encoder disc (or by a simple interrupt pin; which we have used in our project), then the logic level of the photo diode changes. And this change in the logic level can be sensed by the microcontroller. This sensor is generally used to give position feedback to the hardware.

2.7.2) IC MOC 7811^[17]

MOC7811 is an IC for the slotted Opto isolators. It has an IR transmitter and a photodiode mounted on it. This slotted opto isolator performs the task of Non-Contact Object Sensing. This is normally used as positional sensor switch (limit switch) or as Position Encoder sensors used to find position of the wheel. The IR LED and Photodiode are mounted and face each other enclosed in plastic body.

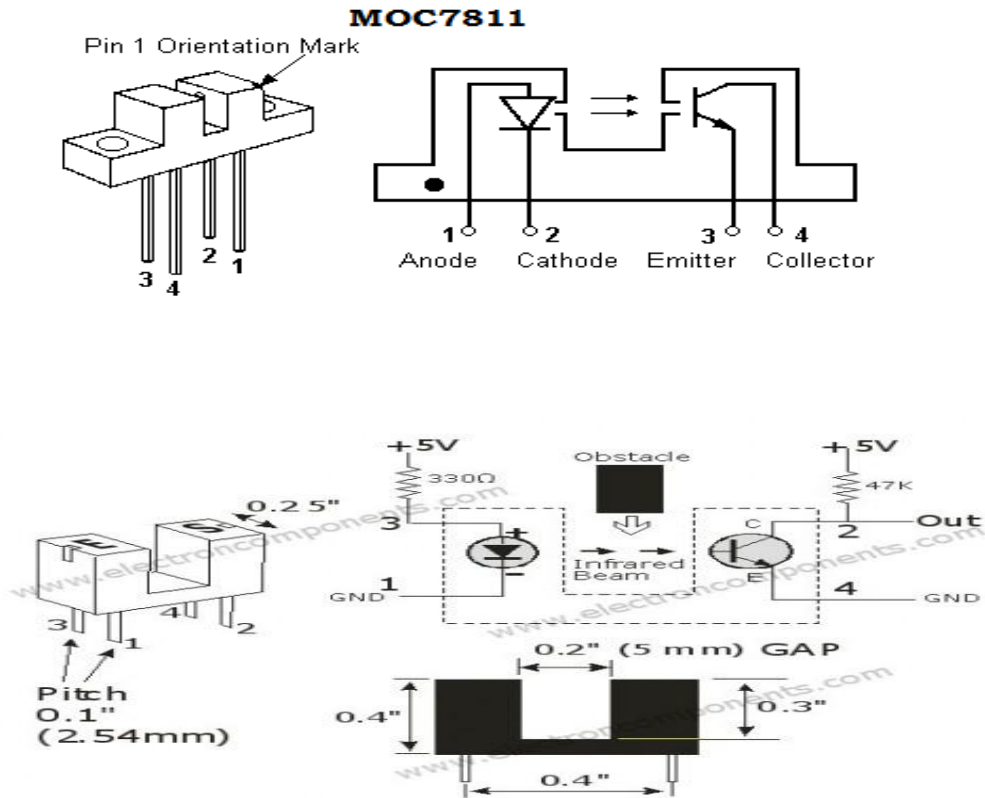


Figure 14: Internal connection of IC MOC7811^[18]

2.7.3) SPECIFICATIONS^[18]

- Mounting hole diameter: 3mm
- Mounting hole spacing: 19mm
- Slot width: 3mm
- Slot depth: 7mm

2.7.4) APPLICATIONS

- DC motor position / velocity control
- Position and velocity servomechanisms
- Factory automation robots
- Numerically controlled machinery
- Computer printers and plotters

2.8) POTENTIOMETER

Potentiometer is a passive electronic component which has an electro-mechanism that helps the user to vary its resistance manually. By varying the value of resistance the current in the circuit can be control.

It is a three terminal resistive component with a sliding contact. It can be implemented as a variable voltage divider if all its terminals are used. If only two terminals are used i.e., one end and the wiper then it will act as the rheostat (a variable resistance).

It is basically used to control the electric devices such as fan regulator, volume control on audio systems and many more. They are also used as position transducer in joystick.

Ways to connect the potentiometer:

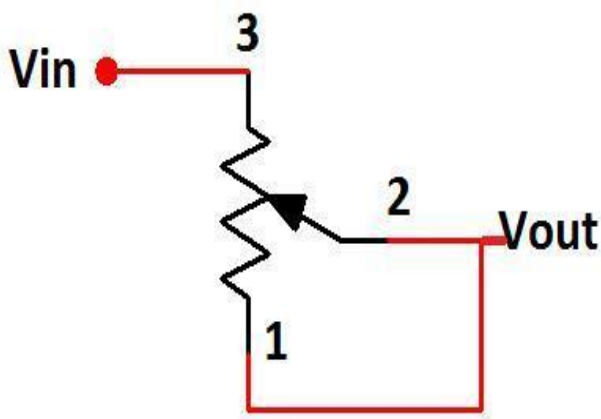


Figure 15: Trimmer resistance^[22]

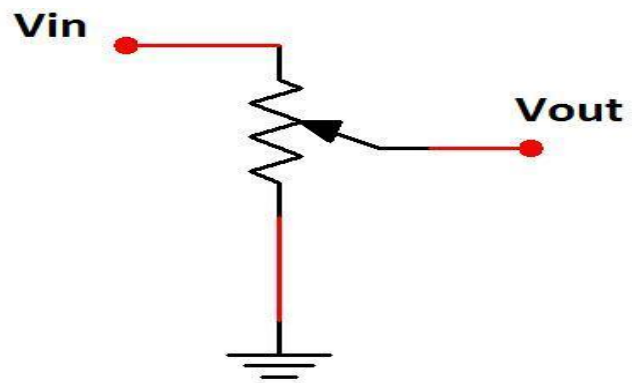


Figure 16: Voltage divider^[22]

2.8.1) THEORY OF POTENTIOMETER

Potentiometer has 3 important parts which are lugs, resistive strip and metal wiper.

Lugs: Any potentiometer conventionally has three lugs. Lugs are shown in the figure as 1,2 and 3.

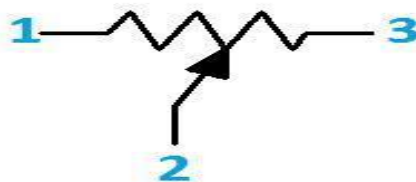


Figure 17: Three lugs of potentiometer

Resistive strip: It is a carbon strip printed on a phenol strip. They are connected to the lugs by metallic contacts. This is the main part of the whole potentiometer structure.

Metal wiper: It is rotated either with the help of a shaft or any other mechanical system. It makes the connection between the lug 1 and lug 2.

Potentiometer has two strips one is resistive and another one is conductive. Resistive strip is responsible for resistance variance feature and the conductive strip helps to carry current to the circuit with respect to the resistance. The metallic wiper is the conducting path between the lug 1 and lug 2. When we rotate the wiper, the resistance between the lug 1 and lug 2 is the part of the resistive strip over which the wiper traversed. Hence resistance varies when we move rotate the wiper.



Figure 18: Potentiometer 103^[21]

2.9) COMPARATOR

A comparator is a circuit that compares two voltages and outputs either a 1 (the voltage at the plus side i.e. +V) or a 0 (the voltage at the negative side i.e. -V) to indicate which one of the two voltages is larger. Comparators are generally used to check whether an input has reached some predetermined value or not. In most cases a comparator is implemented using a dedicated comparator IC, but op-amps may be used as an alternative.

2.9.1) WORKING OF COMPARATOR^[25]

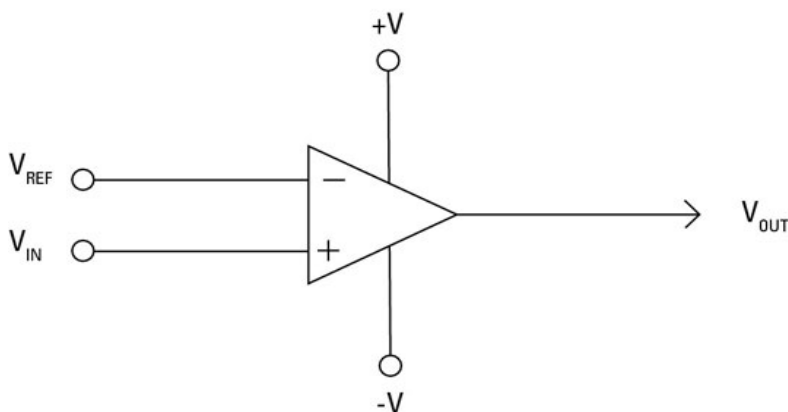


Figure 19: symbolic representation of comparator

In a voltage-comparator circuit, the reference voltage i.e. V_{ref} is applied to the inverting input (V_-) which is the pin 2 of the common op-amp; and then the input voltage V_{in} which is to be compared with the reference voltage is applied to the pin 3 of the op-amp which is the non-inverting input. Since there is no feedback, the circuit amplifies the voltage difference between V_{in} and V_{REF} , and gives the output at V_{out} . If V_{in} is greater than V_{ref} , then voltage at V_{out} will rise to its positive saturation level, i.e. to the voltage at the positive side. If V_{in} is lower than V_{ref} , then V_{out} , will fall to its negative saturation level, equal to the voltage at the negative side.

The value of the output voltage depends on the value of the input voltage relative to the reference voltage; the following table shows different output voltages for different combinations of V_{in} and V_{ref} .

INPUT VOLTAGE V_{in}	OUTPUT VOLTAGE V_o
Less than reference voltage	Negative
Equal to reference voltage	Zero
Greater than reference voltage	Positive

Table 2: comparator output operation

One thing to be kept in mind is that the voltage level for the positive as well as the negative output will be about 1 V less than the power supply. Thus, if the power supply to the op-amp is 5V, the output voltage will be +4V if V_{in} greater than the V_{ref} , 0 V if the V_{in} is equal to the reference voltage, and -4V if the V_{in} is less than the reference voltage.

To eliminate the negative voltage from the output, we can use a diode and send the output through that diode if the input is less than the reference. In such a circuit, a positive voltage appears at the output pin1 if the input voltage is greater than the reference voltage; otherwise, no output voltage exists.

2.9.2) IC LM358^{[23],[24]}

The LM358 op-amps are used in all the conventional op-amp circuits which can now be implemented with much more ease in single power supply systems. For example, the LM358 op-amp can be directly operated off of the standard +5V power supply voltage which is used as a part

of digital systems and it very easily provides the required interface electronics without any need of the extra $\pm 15\text{V}$ power supplies. The LM385 IC comes in an 8-pin DIP (dual in-line package) package as shown in the following figure:

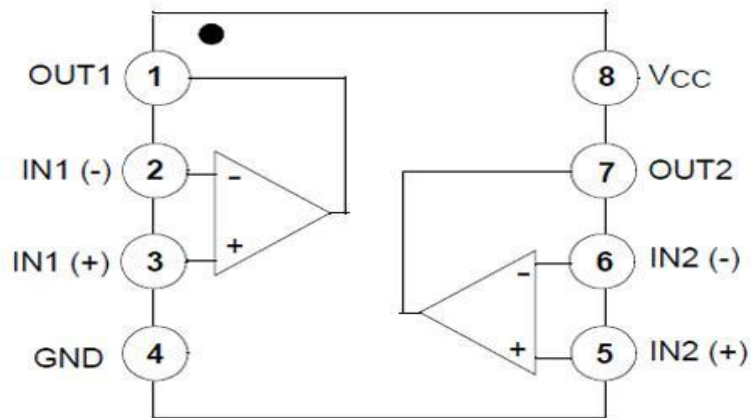


Figure 20: Pin diagram of LM358^[23]

Pin Description:

- Pin 1 and 7 are outputs of comparator
- Pin 2 and 6 are inverting inputs
- Pin 3 and 5 are non-inverting inputs
- Pin 4 is ground (GND)
- Pin 8 is VCC+

2.9.3) FEATURES

- Internally frequency compensated for unity gain
- Large dc voltage gain: 100 dB
- Wide bandwidth
- Wide power supply range: 3V to 32V
- Low input offset voltage: 2 mV
- Input common-mode voltage range includes ground
- Differential input voltage range equal to the power supply voltage
- Very low supply current drain(around 500mA) essentially independent of supply voltage

2.9.4) ADVANTAGES OF LM358:

- Both the two op amps are internally compensated.
- No need for dual supplies.
- Compatibility with all the forms of logic.
- Power drain suitable for battery operation.

2.10) PULL UP RESISTORS

Microcontroller pins has three possible states ON, OFF, and FLOATING. Out of these three only ON and OFF state are useful. The floating state is not useful as this state can't be converted to the Boolean value that is 1 or 0. So if we are working with the 0's and 1's this state must be avoided. Hence to eliminate this state we use the

pull up or pull down resistors to ensure that the pin is in either high state or low state

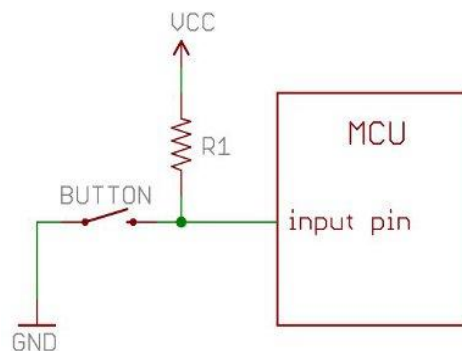


Figure 21: general connection of pull up resistance with button switch^[26]

2.10.1) THEORY OF PULL UP RESISTORS

The pull up resistances is usually used with switches. If we are using pull up resistance the input to the pin will be high when the switch is opened else it will read the low value.

When the switch (say button switch) is not pressed, a small amount of current will flows between the Vcc and the input pin of the microcontroller, and the pin reads a high state. If the switch is pressed the current will flow to ground and the pin reads the low state.

The figure shows the common pull up resistance, if we do not use the resistance R1, Vcc will be shorted to the ground on closing the switch.

For a strong pull up, a low resistor should be use whereas for the weak pull up, the resistor should be high as it is known that the less current flows with high resistance.

The following two conditions must be considered while selecting the value of pull up resistance:

1. The input pin reads low value, *whenever the switch is closed*. The value of resistor R1 will decide how much current will flow from VCC, through the switch, and then to the ground.
2. The input pin reads high value, *whenever the switch is opened*. The value of the pull-up resistor (R1) controls the voltage on the input pin.

2.11) OPTOCOUPLER

An **optocoupler** also known as **optoisolater**, **photocoupler**, or **optical isolator**, is an electronic component that transfers electrical signals between two isolated circuits by using light. An optocoupler prevents from high voltages from affecting the system receiving the signal. Optoisolators which are available in market can withstand input-to-output voltages up to 10 kV and the voltage transients at the speed up to 10 kV/ μ s.

The most common type of optocoupler consists of a led and a phototransistor in the same opaque package. Other types of source-sensor combinations include LED-photodiode, LED-LASCR, and lamp-photoresistor pairs. Generally opto-isolators transfer digital (on-off) signals only, but some techniques now make it possible for them to use analog signals.

2.11.1) WORKING OF OPTOCOUPLER^[41]

An optocoupler contains three main parts:

1. A source of light(which is called as emitter), which is generally a near infrared light-emitting diode (LED), that has capability to convert an electrical input signal into light,
2. A closed optical channel (called as dielectric channel), and
3. A photo-sensor, which can detect the incoming light and can either generates electric energy directly, or modulates electric current flowing from an external power supply.

While understanding the working of optocoupler, One must keep in mind that an optocoupler can transfer only the light signal not the electrical signal .

The sensor can be a photoresistor, a photodiode, a phototransistor, a silicon-controlled rectifier (SCR) or a triac. Because LEDs can sense light in addition to emitting it, construction of symmetrical, bidirectional opto-isolators is possible. An optocoupled solid state relay contains a photodiode opto-isolator which drives a power switch, usually a complementary pair of MOSFETs. A slotted optical switch contains a source of light and a sensor, but its optical channel is open, allowing the modulation of light by external objects obstructing the path of light or reflecting light into the sensor.

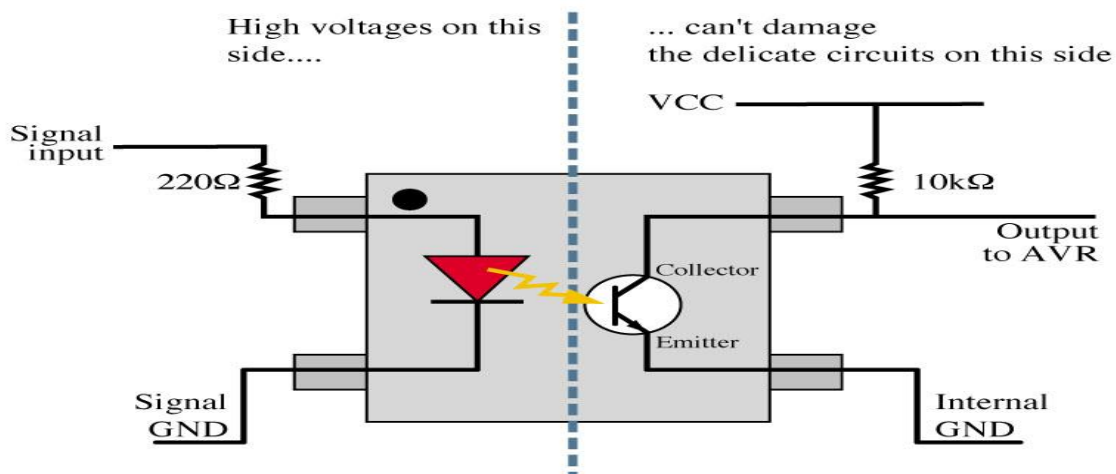


Figure 22: optocoupler

2.11.2) IC FL817F343^[29]

The optocoupler IC which we are using is **FL817F343**. The common optoisolator found in LCD TVs is also 817type optoisolator. Following is an schematic diagram of the optocoupler IC showing the pin numbers.(NPN transistor Optocoupler) in which LED is acting as the source of light, phototransistor as the sensor and showing dielectric barrier in the centre.

To test the optocoupler before using it, we can simple follow these steps. First put the red test lead of the multimeter on the anode of the optocoupler (means the pin number 1) and put the black test lead on the cathode i.e. the pin number 2 of the IC. By doing so, the resistance reading of the analog meter must be infinite. Now place the black test lead on the anode and the red test lead on the cathode, now the analog meter should show a low value resistance reading which is around 20ohms. If we get the low resistance value on both the above two cases, then it would mean that the LED inside the IC is shorted and we need to change this optocoupler with a new and correct IC.

We also need to check the phototransistor, for this first we need to put our analog meter into *10k ohm range. Put the black lead on collector pin and the red on the emitter pin of the optocoupler, we should get a resistance reading of infinite. Now reverse the leads, and by doing so now we should get a large resistance reading which is around 500k ohms or so. If we get low resistance or 0 resistance value in both these cases, then it would mean that the phototransistor is shorted and thus we'll have to replace the optoisolator.

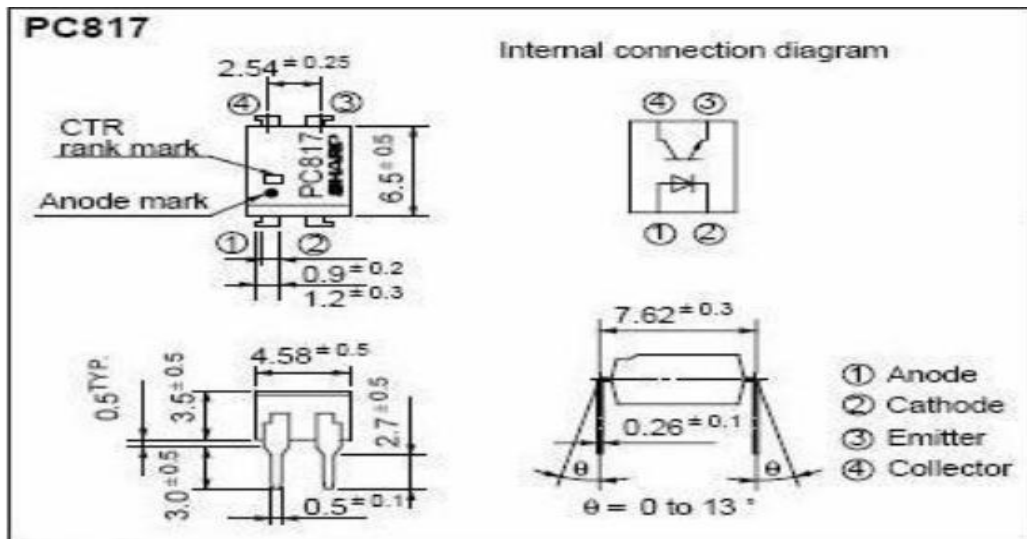
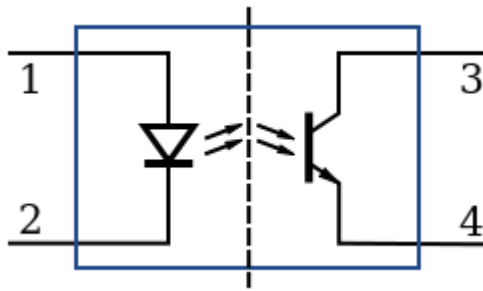


Figure 23: Internal circuitry of FL817F343^[42]

2.11.3) CHARACTERISTICS OF AN OPTOCOUPLER

CTR: The Current Transfer Ratio. The most important parameters out of all in an optoisolator is its optocoupling efficiency. The optocoupling efficiency is maximized by spectrally matching the LED and the phototransistor (which generally rate in the IR range). The optocoupling efficiency of an optocoupler can be easily specified by the output-to-input *current transfer ratio* (CTR) . The current transfer ratio is the ratio of the output current I_c , which is measured at the collector of the phototransistor, to the input current I_F flowing into the LED.

$$CTR = I_c / I_F$$

If used in normal mode, the base of the phototransistor is left open circuit, and in such a condition the optocoupler device has a minimum CTR value of 20 %.

V_{iso} : Input-to-Output Isolation Voltage. It is the value of maximum potential difference that is allowed to exist between the input and output terminals. Typically its values range from 500 V to 4 kV.

$V_{CE(max)}$: Maximum Collector-Emitter Voltage. This is the maximum dc voltage that can be allowed to be applied across the output transistor. Its typical values range from 20 to 80 volts.

Bandwidth: It is the maximum signal frequency (in kHz) that can be usefully passed through the optoisolator when the device is operated in the normal mode. Its typical values range from 20 to 500 kHz, and these values vary according to the different the type of device construction. If used in normal mode, the optoisolator has a useful bandwidth of 300 kHz.

Response Time: The response time is divided into two parts, the **rise time t_r** and the **falltime t^*** . The t_r and t^* are usually around 2 to 5 us in case of a phototransistor output stages.

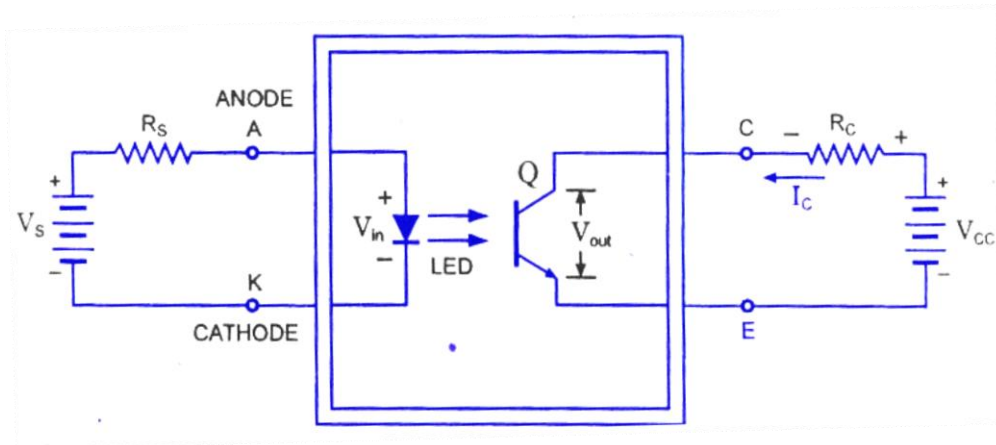


Figure 24: LED driving a phototransistor

2.12) H-BRIDGE

The H-Bridge is the link between digital circuitry and mechanical action. The computer sends the commands which are in binary form, and the high powered actuators follows the command performs the task as ordered by the system. Generally H-bridges are used to control the direction of DC motors in which it rotates. H-Bridge is not so expensive to build and it also pretty efficient to drive and control the motors accordingly.

The H-Bridge is mainly designed to drive a motor either in clockwise or anticlockwise. It is required to reverse the supply, to reverse the rotation of the motor, and this is what exactly H-Bridge do with the motors connected to them. An H-Bridge can be built with various electronics components such as switches, relays, BJTs or MOSFETS. It is nothing but a specific arrangement of transistors that allows a small circuit to control a standard electric DC motor. That is, with the

help of H-bridge, a microcontroller or a logic chip could command the motor when to go forward or reverse, when to brake, and when to coast.

2.12.1) THEORY OF H-BRIDGE

It is also known as "full bridge" this arrangement called as H-bridge because it consists of 4 switching elements and these switches are arranged such that it appears as they are fitted at the corners of the letter 'H' and the motor which is needed to drive is connected to the cross bar. When we see these switches in clockwise order then these switches named as high side left, high side right, low side right, and low side left as mentioned in the image. The switches work in pairs that is switches turned on and off in pairs. Pairs are consisting of two oppositely placed switches. High side left and low side right switches form one pair whereas other one consists of low side left and high side right. If both switches that are positioned on the same side of a bridge turned on, the battery terminals will get short circuited. This phenomenon is known as shoot.

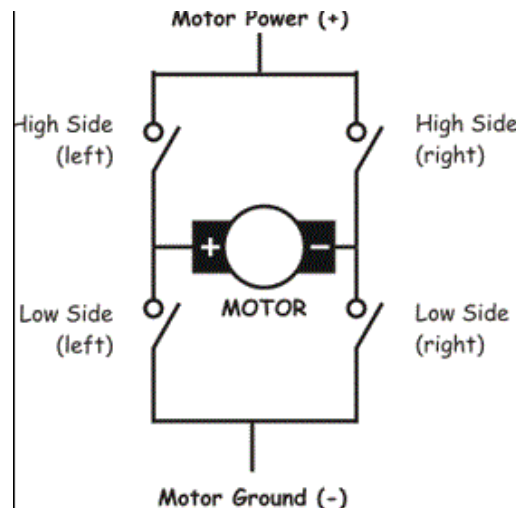


Figure 25: H-bridge^[30]

To power up the motor, any one diagonally opposite pair of switches should be on. Let say the diagonally opposite pair, high side left switch and low side right switch are turned on. The current will flow as it is shown by green mark. As the current flows through the motor and the motor begins

to turn in a direction which is positive.

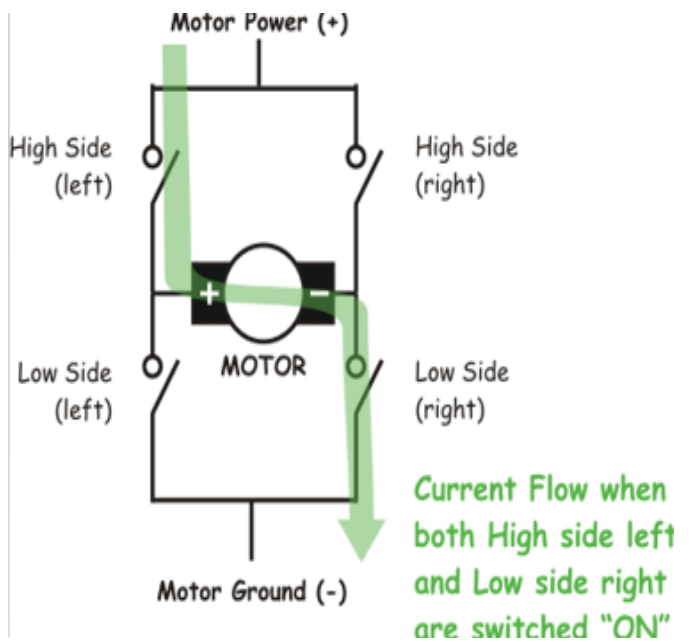


Figure 26: Current Flow in H-bridge^[30]

If another pair of switches are turned on i.e., high side right switch and low side left switch. Then the current will flow in another direction through the motor and the motor starts rotating in the opposite direction. This type of circuits also known as 'four quadrant device'. As each switch may have one of the two states that are on or off, and there are total four switches, there can be 16 states which are possible.

But the states in which switches that are positioned in one side turned on is called the "bad" state, therefore there are only four useful states say only four quadrants are possible where the motor can work.

High Side Left	High Side Right	Lower Left	Lower Right	Quadrant Description
On	Off	Off	On	Motor goes Clockwise
Off	On	On	Off	Motor goes Counter-clockwise
On	On	Off	Off	Motor "brakes" and decelerates
Off	Off	On	On	Motor "brakes" and decelerates

Table 3: quadrants of H-bridge^[31]

The last 2 combinations of switches describe the "short circuit" of the motor due to which the motors generator works against itself. There is also a state where all the transistors are turned off. In that case the motor coasts if it was spinning and happens nothing if it was doing nothing.

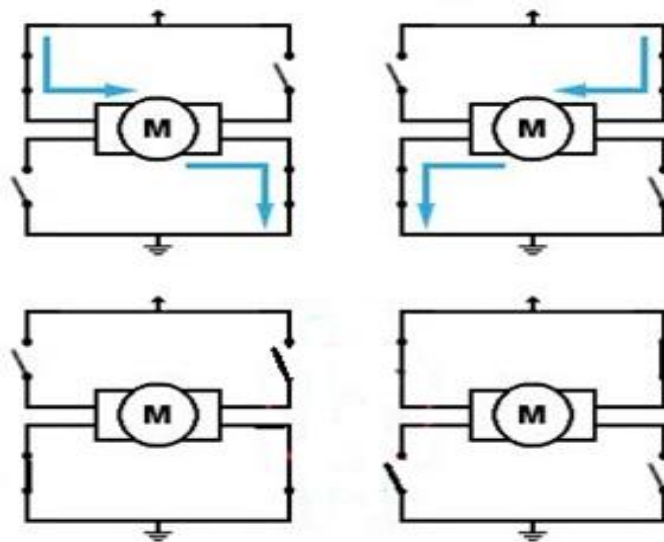


Figure 27: different combinations of switches^[32]

2.13) DC GEARED MOTOR^[33]

Electric motors are everywhere; almost every mechanical movement that we see around us is either caused by an AC or a DC motor. A geared dc motor is a DC motor whose rotor shaft is connected to the load shaft by a gear system. A gear motor, by the virtue of the additional GEAR BOX, reduces the speed of the motor output shaft from its normally high speed (1000-20,000 rpm) to a slower output speed (say 50-200 rpm). We are using three 12V, 50 rpm geared dc motors^[37].

The power of a motor at a given energy can be arranged to have either high speed or high torque. As in the case of an 18-speed mountain bike. If we put the bike in the 1st gear, our pedalling will provide a high torque that can enable us to ride up a steep hill with ease at the expense of going slowly. And if we put the bike on 18th gear, it will be nearly impossible to pedal up the steep hill but will yield excellent speed on the flat ground or going downhill.

The power produced by the electric motor can be manipulated in the same way. The gear assembly helps in increasing the torque and reducing the speed, this concept known as **gear reduction**. And for this reason we go for the GEAR MOTOR. Each gear motor should have a **gear ratio** that specifies the ratio of input speed to output speed of motor output shaft. For example a gear motor with 100:1 gear ratio implies that the actual DC motor output shaft must spin 100 times to make geared output shaft complete 1 revolution.

The following figure shows a DC gear motor (left) and again a DC motor removed from the plastic gear box (right)



Figure 28: gear motor^[34]

The speed of motor (which is in RPM) is counted in terms of number of rotations of the shaft per minute. In any application that requires shaft speeds slower than that of a “straight” motor, gear motors are a highly desirable alternative to conventional belts, gears and chains.

2.13.1) **WORKING PRINCIPLE:**

The gears increase the torque of the motor while reducing the speed.

As we know,

Power=Torque*angular velocity.

Increased torque means it can lift heavier loads.

The DC motor works over a good range of voltage. The higher the input voltage more is the RPM of the motor. Thus the speed of motor depends directly on the input voltage. Per say if the motor works in the range of 6-12V, it will have the least RPM at 6V and maximum at 12 V.

In terms of voltage, we can put the equation as:

$$\text{RPM} = K1 * V$$

K1= induced voltage constant

V=voltage applied

The working of geared motor can be explained by the principle of **conservation of angular momentum**. The gear with smaller radius will cover more RPM than the one with larger radius. However, the larger gear will give more torque to the smaller gear than vice versa. The comparison of angular velocity between input gear (the one that transfers energy) to output gear gives the **gear ratio**. When multiple gears are connected together, conservation of energy is also followed. The direction in which the other gear rotates is always the opposite of the gear adjacent to it.

The equations detailing the working and torque transfer of gears are as follows ^[35] :

$$\zeta_{in} \omega_{in} = \zeta_{out} \omega_{out} \quad , \text{ where}$$

ζ_{in} = input torque by the driver gear.

ω_{in} = angular speed of the driver gear.

ζ_{out} = output torque of the driver gear.

ω_{out} = angular speed of the driven gear.

2.13.2) **DC MOTOR WORKING**

DC motor is the type of electric motor that works on the direct current (DC) electricity. It simply converts direct current (electrical energy) into the mechanical energy.

OPERATION OF DC MOTOR ^[36]

Let's first look to the constructional features of the DC motor. The most basic construction of a dc motor contains a current carrying armature connected to the supply end through a commutator segment and brushes, and placed in between the two opposite poles (north south poles) of a permanent or an electro-magnet. The following figure shows the basic construction of a DC motor.

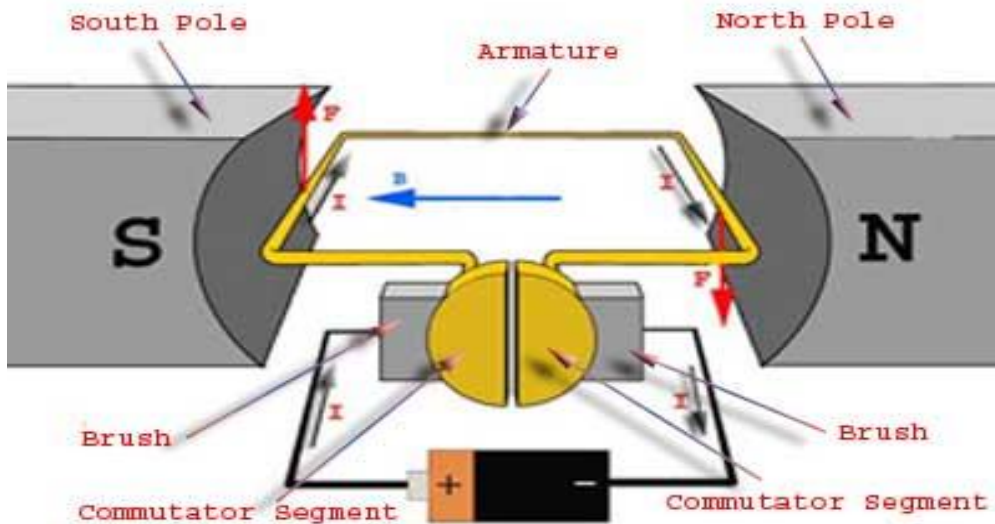


Figure 29: DC motor internal structure

The **operating principle of DC motor** can be understood only with the correct understanding of Fleming's left hand rule, which is used to determine the direction of the force acting on the armature conductors of dc motor. Because the motion of the armature (clockwise or anticlockwise) is the crux of the motors.

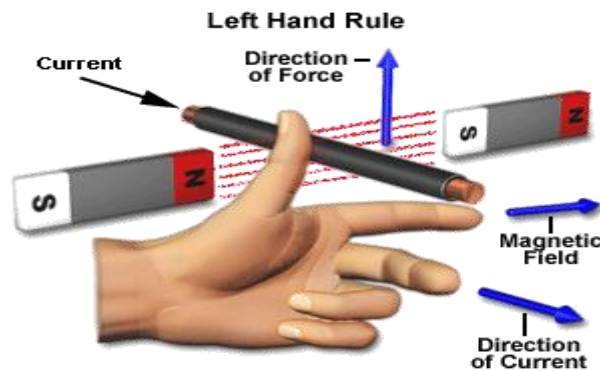


Figure 30: left hand rule

The Fleming's left hand rule says that if we extend the index finger, middle finger and thumb of our left hand in such a way that the electric current carrying conductor is placed in a magnetic field (represented by the index finger) is perpendicular to the direction of current (represented by the middle finger), then the conductor experiences a force in the direction (represented by the thumb) mutually perpendicular to both the direction of field and the current in the conductor.

Determining the magnitude of the force:

Let's consider the following diagram. We know that when an infinitely very small charge dq is made to flow at a velocity 'v' under the influence of an electric field E, and a magnetic field B, then the Lorentz Force dF experienced by the charge is given by the relation :-

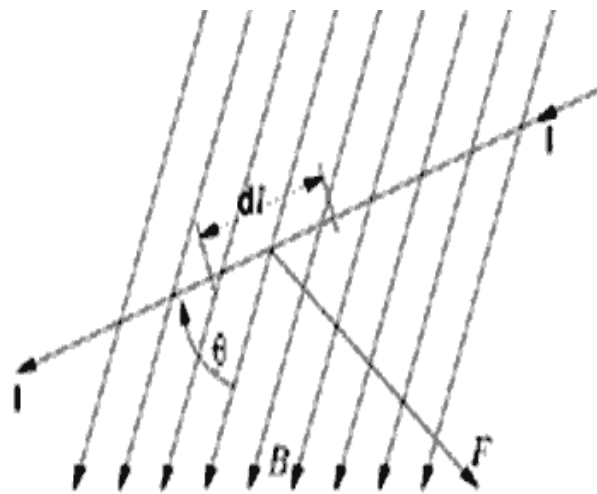


Figure 31: Vector diagram showing field

$$dF = dq(E + v \times B)$$

now let's take $E = 0$ for the **operation of a dc motor**,

$$\therefore dF = dq (0 + v \times B)$$

$$\Rightarrow dF = dqv \times B$$

This means the force is the cross product of dqv and magnetic field B .

$$\text{Or } dF = dq \left(\frac{dL}{dt}\right) \times B \quad [\text{since velocity } v = \frac{dL}{dt}]$$

Where dL is the length of the conductor carrying charge q .

$$\text{Or } dF = (dq/dt) dL \times B$$

$$\text{Or } dF = I dL \times B \quad [\text{Since, current } I = dq/dt]$$

$$\text{or } F = IL \times B = ILB \sin\theta$$

$$\text{or } \mathbf{F} = \mathbf{BIL} \sin\theta$$

From the above diagram showing construction of a DC motor, we can see that the direction of electric current through the armature conductor at all the instances is perpendicular to the field. Hence the force acting on the armature conductor is in the direction which is perpendicular to the both uniform field as well as and current.

$$\text{i.e. } \theta = 90^\circ$$

So if the current in the left hand side (LHS) of the armature conductor to be I , and current at RHS of the armature conductor to be $-I$, as the two current will always flow in the opposite direction with respect to each other.

$$\text{Then the force on the LHS armature conductor, } F_1 = BIL \sin 90^\circ = \mathbf{BIL}$$

$$\text{And similarly force on the RHS conductor, } F_r = B(-I)L \sin 90^\circ = -\mathbf{BIL}$$

Thus we can conclude that the position of the force on either side is equal in magnitude but opposite in direction. And since the two conductors are separated by some distance w , the two opposite forces produce a rotational force or a torque that results in the rotation of the armature conductor. Note that w is the width of the armature turn.

Expression of torque when the armature turn creates an angle of α with its initial position:

$$\text{Torque} = (\text{Tangential force to the direction of armature rotation}) \times (\text{distance})$$

$$\Rightarrow \tau = F \cos\alpha \cdot w$$

$$\Rightarrow \tau = BIL w \cos\alpha$$

α = the angle between the plane of the armature turn and the plane of reference or the initial position of the armature which is along the direction of magnetic field here.

The presence of the term $\cos\alpha$ in the formula for torque shows that unlike force, the torque is not the same for all the positions. It varies with the variation of the angle α .

Principle behind rotation of the motor:

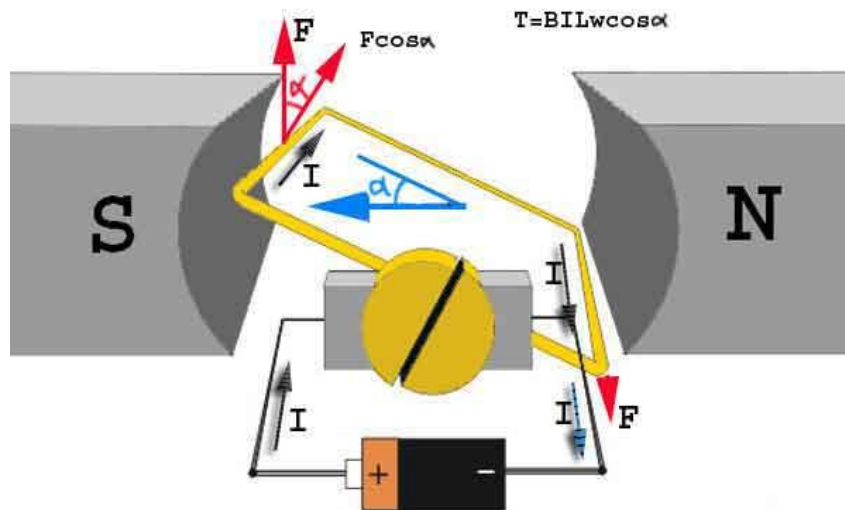


Figure 32: Principle of DC motor

Step1: When initially the armature is in its starting point or reference position.

Then angle $\alpha = 0$

$$\therefore \tau = BIL w \cos 0 = BILw$$

Since $\cos 0 = 1$ i.e. the maximum value of the cosine function, hence torque at this position is maximum and is given by $\tau = \mathbf{BILw}$. This high value of the starting torque helps in overcoming the initial inertia of rest of the armature and sets it into rotation.

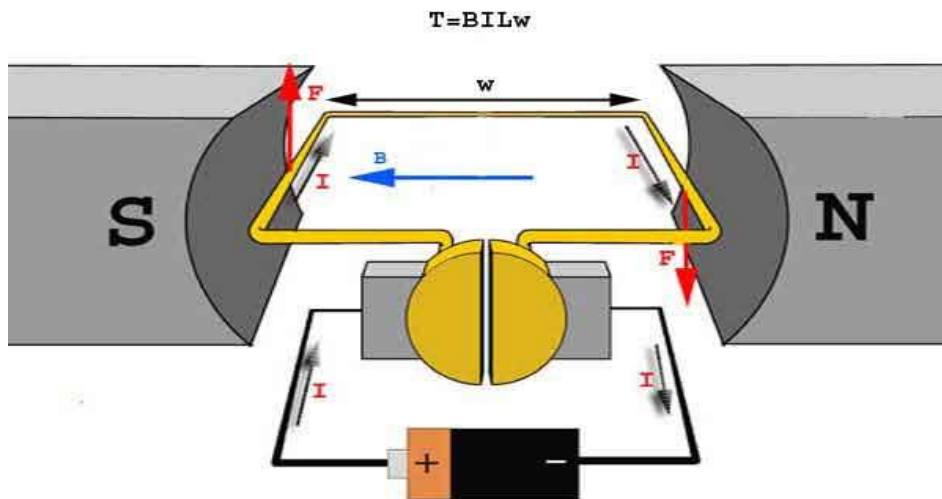


Figure 33: Armature at $\alpha=0$

Step 2: Now once the armature is set in motion, the angle α between the actual position of the armature and its reference initial position keeps on increasing during the path of its rotation until it becomes 90° from its initial position. And thus the term $\cos\alpha$ keeps decreasing and thus the value of torque also keeps decreasing. The torque in this case is given by:

$\tau = BILw\cos\alpha$, which is less than $BILw$ when α is greater than 0° .

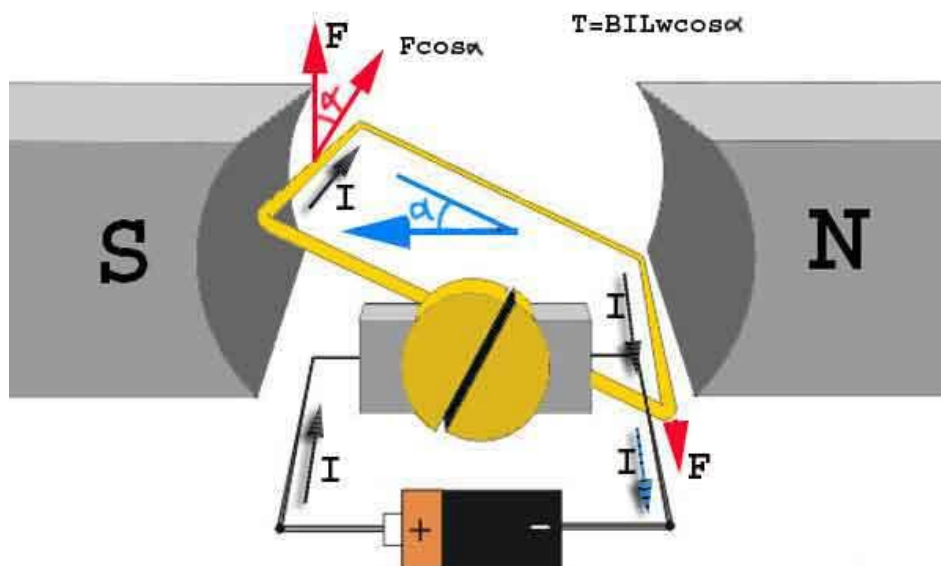


Figure 34: Armature at $0 < \alpha < 90$

Step 3:In the path of the rotation of the armature a point is reached when the actual position of the rotor is completely perpendicular to its initial position. i.e. $\alpha = 90$ and thus $\cos\alpha = 0$.The torque acting on the conductor at this position is given by,

$$\tau = BILw\cos90^\circ = 0$$

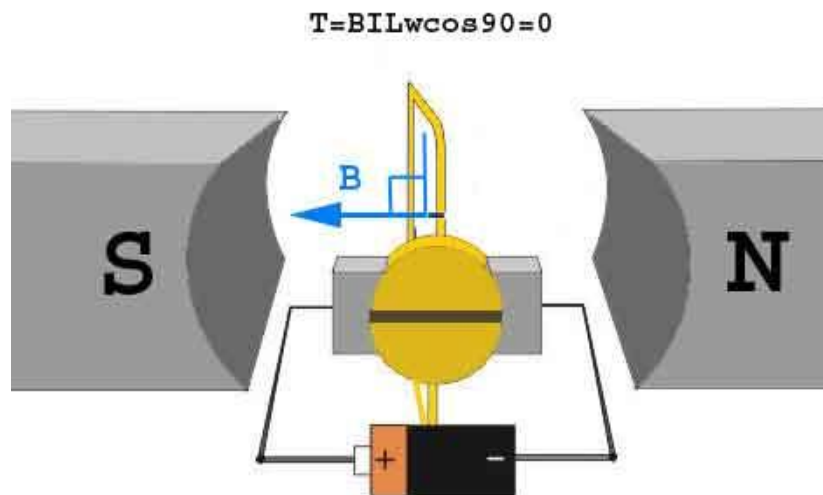


Figure 35: Armature at $\alpha=90$

Though virtually no rotating torque is acting on the armature at this moment, but still the armature does not come to a standstill condition, this is so because of the operation of a dc motor has been engineered in such a way that the inertia of motion at this point is just enough to overcome this point of zero torque. And once the rotor crosses over this position the angle between the actual position of the armature and the initial plane decreases again and torque starts acting on it again. And the same process keeps on repeating and thus the motor keeps rotating.

2.13.3) ELECTRICAL LOSSES IN DC MOTORS:

The electrical loss present in DC motors is of mainly 3 kinds:

1. Losses due to joule's effect in the copper wire of the winding and the brushes.

These losses are integral to the dc motors and gear motors and cannot be eliminated.

2. Losses due to parasitic currents which generate in the lamination pack of the rotor.

These types of losses depend upon the thickness and material used in the rotor. These can also be minimized but can't be eliminated completely.

3. Losses due to attrition.

These losses are tightly connected to the building quality of DC motors and gear motors with permanent magnets.

2.13.4) ADVANTAGES OF USING DC MOTOR^[38]

- Excellent speed control. As the power supply to the motor is connected directly to the motor field, it helps in precised volatage control which, we know is very improtant for speed and torque control applications.
- Design is simpler and cheaper.
- Almost ideal Speed vs Torque motor characteristics.
- The possibility of supplying high specific power.
- Possibility of obtaining variable and continuous dc voltage.
- Simplicity for control (control paradigm)
- Large range of speed controllabilty

2.13.5) THE LIMITAIONS OF USING DC MOTOR^[38]

- Requires high maintance . The maintance of commuatator is always a problem as it conatins copper and mica segments attached to the rotor winding.
- Very vulnerable to dust, the dust deposition results in decreased performance.
- Request for often maintaining.
- Sensitivity according current overloading , sparking (in commutation)

2.14) WOODEN WORK

For the proper alignment of the motors, we need a wooden support. This wooden support should be able to meet the neccessities of the structure of the plotter. The whole wooden structure is inspired from the CNC machine. Thus to get whole wooden work done along with the placement of motors and the base sheet, we approached a skilled carpenter. To achieve this we showed few videos of CNC machine, consequently the carpenter was able to provide us with this wooden structure.

CHAPTER 3: IMPLEMENTATION IN THE PROJECT

3.1) HARDWARE

3.1.1) VOLTAGE REGULATOR IN PROJECT

In order to provide a steady voltage to the entire electronics component used, we need to convert the voltage source into a regulated 5V source. We are using an adaptor which converts the mains voltage to 9V-12V source which is further converted to a ripple free constant 5V source with the help of IC7805.

IC7805 is used as voltage regulator with two 1000 μ F capacitors in IN and OUT pins. 7805 give the regulated 5V as output voltage; the minimum input voltage should be 7.3V.

USE OF CAPACITANCES IN THE CIRCUIT: These capacitors act as filters to remove the ripples from input supply voltage, as we know that in the power supply along with DC some unwanted AC component also exists.

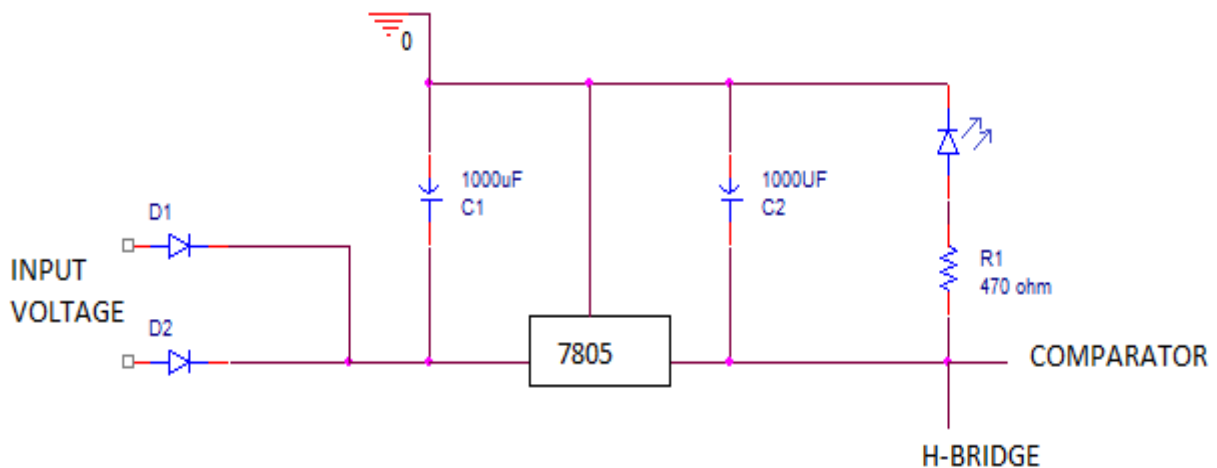


Figure 36: Voltage regulator circuitry

3.1.2) ARDUINO IN THE PROJECT

In this project Arduino is used as an interface between the computer and the plotter. The picture to be plotted by the plotter is stored in the Arduino in its pixel form. Here we are only using the digital pins of the on board microcontroller since the inputs and outputs are in the 1 and 0's form. All the 14 digital input and output pins are used.

- Pin 0 and pin 1 are used to detect the interruptions in the slotted optocoupler.
- Pin 2, 3, 4, 5, 6 and 7 are used as the input pins. These pins get the inputs from the 6 switches that are assigned to perform specific tasks.
- Pin 8, 9, 10, 11, 12 and 13 are the output pins. The output signals from these pins are fed to 6 optocouplers.

3.1.3) SWITCHES IN THE PROJECT

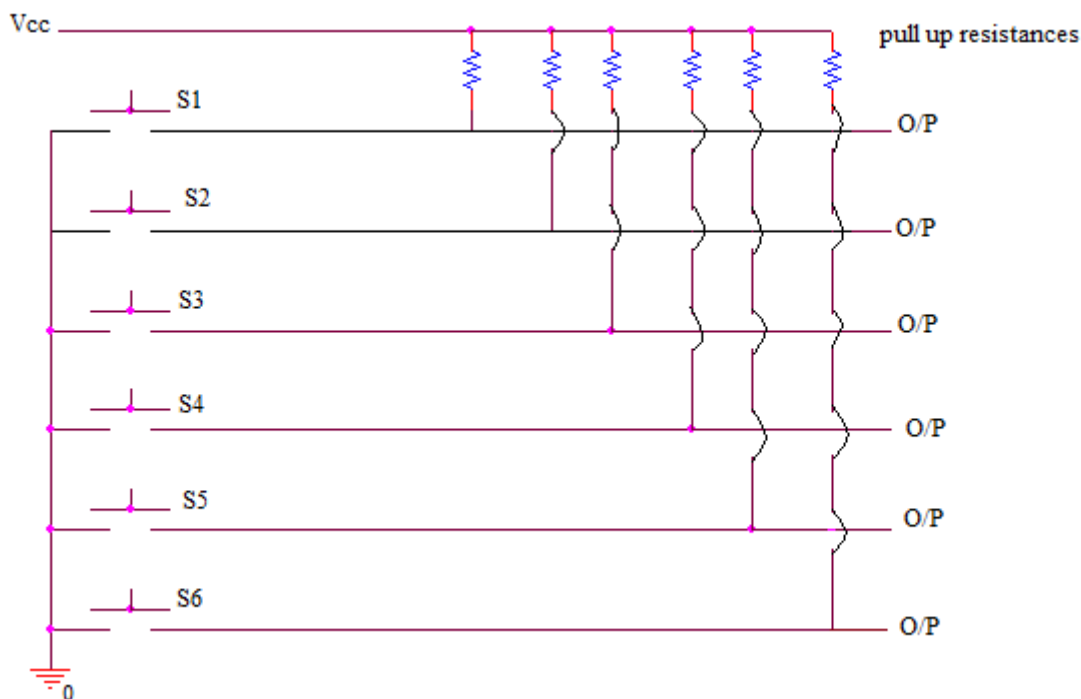


Figure 37: switch circuitry

3.1.4) SLOTTED COUPLER IN THE PROJECT

In the project hardware we are using two slotted couplers. . For the interrupt in the slotted optocoupler we are using a pin made of tin and its tip is covered by a black tape. Whenever this interrupt passes the slot in between the LED and phototransistor, the output of the optocoupler becomes LOW, otherwise it remains HIGH. Their outputs are fed to the two distinct potentiometers on the PCB and the potentiometers after regulating these inputs, further give the information to the

two non-inverting input terminals of the comparator. These slotted optocouplers are used to find out the position of the z-axis along the x-plane and y-plane to provide the Arduino sketch the information about the number of bits travelled by the motors in x and y direction with the help of xcut and ycut variables. These two variables play a very important role in the complete plotting process of the project. The Arduino sketch, which makes the motors run in order to plot the picture, has two functions checkxcut() and checkycut() and these user defined functions continuously keep checking for the xcut and ycut values respectively. According to the value of these variables, motor movement in the specific direction takes place.

3.1.5) POTENTIOMETER IN THE PROJECT

Here we are using trimmer potentiometer 103. It has three pins and a turning knob. These are “set and forget” devices, which are used for a fine tune circuit. It is used by mounting it directly on PCB or bread board. It is used in circuits which need an exact potential that can be ensured with the help of resistance. Trim pot gives the exact voltage at the output (by varying the resistance) and then it forgets about it. There is no shaft attached to the wiper it is only the knob. The resistance of the resistive element of pot 103 is 10 k Ω means the maximum resistance it can provide is 10 k Ω .

3.1.6) COMPARATOR IN THE PROJECT

In our project we are using the IC LM358 to fulfil the function of comparator. As we have seen this IC has 2 comparators built inside it and that's the reason we are using it. We get two outputs from two different potentiometers, these two outputs are named as XCUT and YCUT. These inputs are to be compared with the V_{ref} which is coming from the slotted optocoupler. We need to use the comparator because we have to compare and find out the larger value between the XCUT and this V_{ref} . One end of this XCUT is connected to the potentiometer (which is acting as variator).

If XCUT is greater than V_{ref} then we get HIGH value as output of the comparator and when it is lower, we get LOW value as an output of the comparator. Finally this output is fed to the arduino module. The input coming to the pin 2 of IC LM358 is acting as V_{ref} and is HIGH if there is no interrupt while it is LOW, when no interruption is there. Pin 3 is used for the XCUT and pin 5 for the YCUT, whose output is coming from pin 1 and pin 7 of IC LM358, respectively. The outputs of these comparators are fed to arduino pin 0 and pin 1.

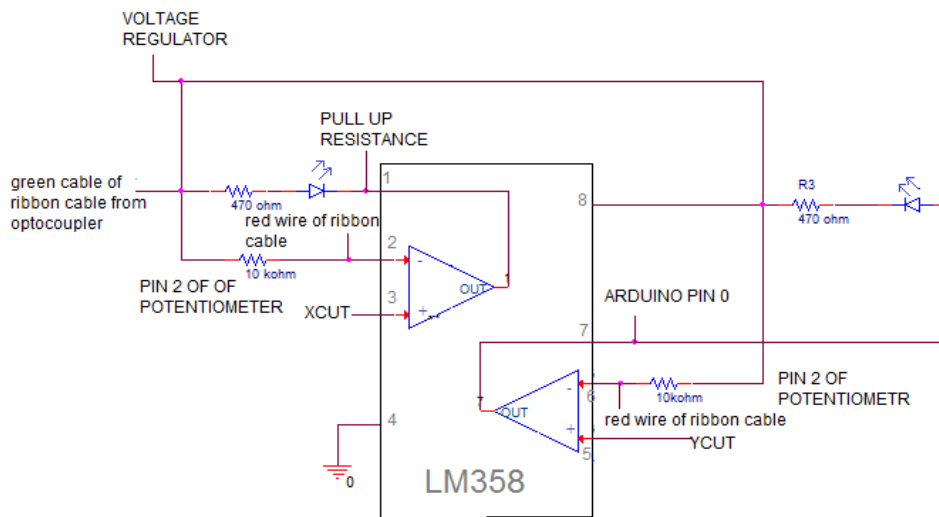


Figure 38: comparator circuitry

3.1.7) PULL UP RESISTORS IN OUR PROJECT

In our project we are using total 6 button switches which are connected to the Arduino. To ensure that the microcontroller are pins not to be in floating state we should either use the pull up or the pull down resistance. Since we are working in active low therefore we uses the pull up resistors. IC A103J- pull up resistor

A103J is a 9 pin network resistor. Network resistors are passive elements consists of multiple resistance. In our project 6 resistances are needed to connect 6 switches to the Arduino. This type of network resistors helps us to condense our circuit with the benefit of low cost. A103J provides resistance of 10 kohms with $\pm 5\%$ tolerance

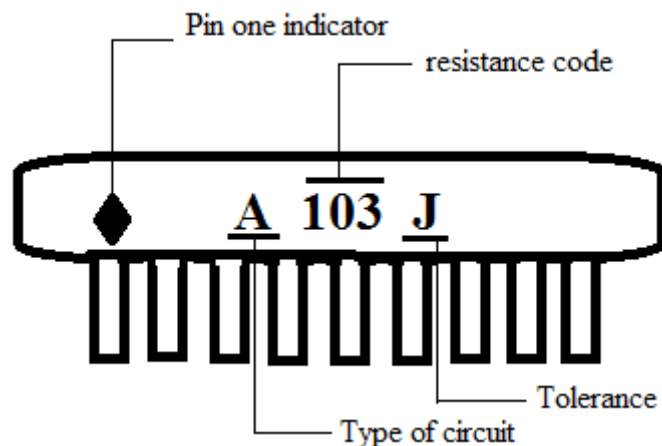


Figure 39: A103J

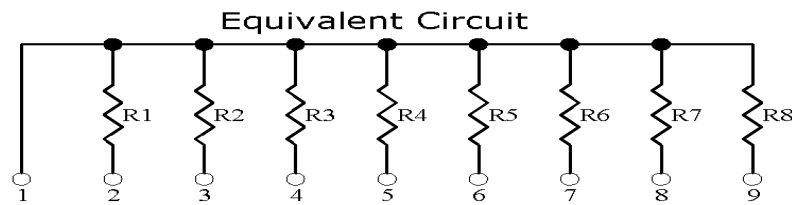


Figure 40: A103J internal circuitry^[26]

3.1.8) OPTOCOUPLER IN THE PROJECT

The optocoupler's function in the circuit:

- In our project, it detects the interrupt. Whenever there is an interrupt, the path of light is obstructed and thus the output pin to the motor driver becomes low.
- This device also prevents high voltages from components in one side of a circuit from damaging or interfering with components on the other side of the circuit. Optoisolators are elegant devices with very little about them to "go wrong". And they are "**fail-safe**". This means that if something goes wrong, your system will simply won't work, it will stop working but nothing moves in a bad direction. Let's take the example of a car: The engine is "fail safe". If it dies, we just drift to the side of the road and stop. Our brakes are not failing safe. They fail, and we get in trouble.

CONNECTIONS

The following figure is the circuit diagram to show how the connections are made on the PCB. We can see that our optocoupler IC is connected to the H-bridge to drive the motor. Thus the output at the pin number 4 of each optoisolator IC governs the direction in which the motor will rotate.

In our circuit we are using six optoisolators and three H-bridges to drive our three geared DC motors. Three motors are used for motion of the plotter's pen in the X, Y and Z direction respectively. Each motor can move clock wise or anticlockwise, thus giving the two possible motions of the plotter. Thus each H-bridge is connected with two optoisolator ICs. The pin number 8 to 13 of the arduino is the six output values which are fed to the pin number 1 of the six different optocouplers. The pin number 8 to 13 are for yup, ydn, zdn, zup, xdn and xup respectively. Let's explain it with the example of one motor, let's take the motor for Y-axis movement (we can call it as motor-x). The pin number 8 of the arduino possess the value of the variable yup, and this value is fed to the pin number 1 of the optocoupler (this is the optocoupler which is connected to the H-bridge to run motor-y) and at the same time the output of the pin 9 (the ydn) of the arduino is fed to the pin 1 of the optocoupler which is connected to the opposite side of the same H-bridge.

Now if the $yup=0$ and $ydn=1$, then the pin4 of first optocoupler (i.e. the one whose input is yup) gives the high output but the pin4 of the other optocoupler gives the low output. And these outputs are directly connected to the H-bridge. This makes the H-ridge to rotate the motor-y in the clockwise direction thus letting the plotter to move in the yup direction, and thus fulfilling the task of motor movement. In the same way rest of the four optocouplers help in the x and z axes movement of the motors. Our circuit is working in the active-low all the time and thus $yup=0$ at pin8 of the Arduino will result into the yup motion of the plotter.

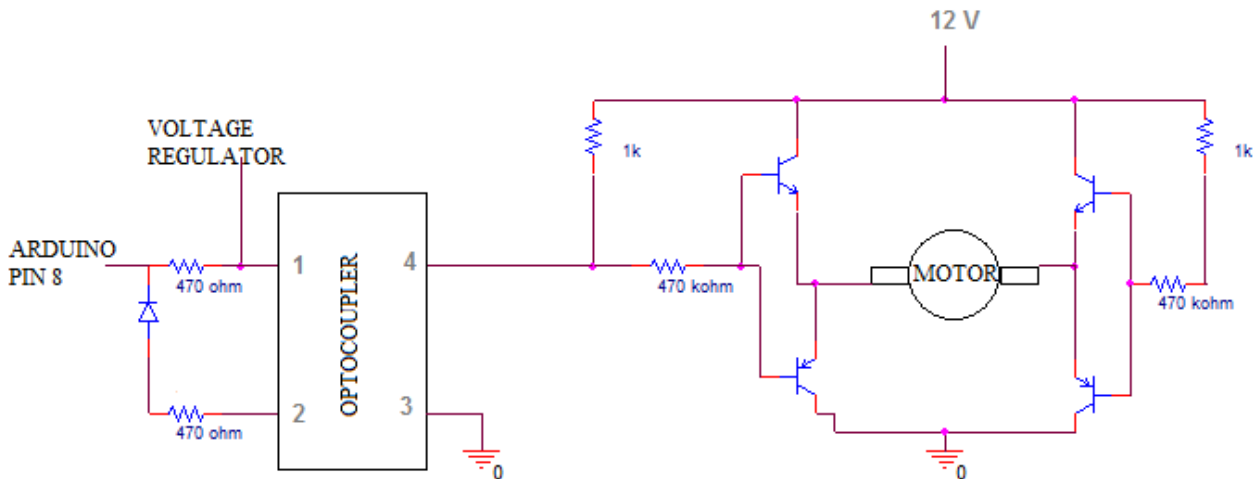


Figure 41: circuitry of motor driver

3.1.8) H-BRIDGE IN THE PROJECT

Here, we have built 3 H-Bridges using 4 BJTs (bipolar junction transistors) each to drive motors (X-axis, Y-axis, Z-axis). Each H-Bridge is connected to the two optocouplers. The optocouplers receives the output signals from the Arduino and accordingly the H-bridge controls the motors to plot. In H-bridge we are making use of the concept of BJTs as switches. Physical switches are very impractical since no one is going to sit there and flipping the switches in pairs so that their motor could move forward or in reverse. That's where the transistors come in the picture. A transistor is as a solid state switch which acts as a close switch when its base terminal is provided with a small current. Because only a small current is required to activate a transistor and it is sufficient to complete one half of the circuit. When a small current is applied to the base, comparatively larger current is allowed to flow from collector to emitter. Basically a transistor controls a larger current by allowing a small current. In this case the emitter should always be grounded.

We are using 2 NPN and 2 PNP BJTs to build an H-Bridge in order to prevent short circuits and optimize current flow. We are using NPN transistor for top two and PNP for bottom two. The

collector of both NPNs is connected to 12 volts, whereas collector of 2 PNPs connected to ground. The emitter of NPNs and PNPs are connected to the motor. To operate the BJT a voltage is applied to the base, and in result a sudden current of electrons passes through the emitter and collector. Let say A and B are marked as two control lines on which logic voltage is to apply. And these control lines A and B are the outputs from the two optocouplers connected to the same motor. Since we have only two pins, which can only have a binary control, there are four possible activities that can take place.

Let's take the example of the working of the H-bridge which drives the Y axis motor for the yup and ydn motion. The pin 8 and pin 9 of the arduino possess the yup and ydn variables respectively. Let's take the case when pin8=0 and pin9=1, output of these pins are fed to the pin1 of two different optocouplers. Output of these two optocouplers is providing the inputs A and B to the H-bridge. When pin8=0 that means the LED connected to the optocoupler will glow and this LED shows the condition of the LED which is inside the optocoupler. Now we know that with the given low output at pin8 of arduino, the LED inside the optocoupler glows and that's why the output will be high. Thus now A=1. And at the same time the opposite working will be there with the other optocoupler connected to the H-bridge. Pin9=1, means LED connected to the second optocoupler as well as the LED inside will not glow and thus the output of the optocoupler will be low, that means B=0.

Now when A=1 and B=0, the direction of rotation of the motor will be decided by using the concept of BJT as switches. As A=1 that means NPN will give 0 and PNP will give 1 and at the same time because B=0, the right hand side NPN and PNP will give 1 and 0 respectively and thus causing the motor rotate in the clockwise direction. Similarly different combination of xup, xdn, zup and zdn will lead to motion of motors and consecutively the plotter. The following table shows the possible outcomes of different combinations of A and B inputs. Note that we can never have the condition in which the A and B inputs both are simultaneously high as we know xup and xdn can never be equal to 0 at the same time cause this would mean that the motor should be running in xup as well as in xdn direction at the same time. It's because we are using same motor for up and down motion of 1 axis. And also it must be taken care that A and B never become 1 simultaneously because this will damage the circuit completely.

A	B	Action
0	0	Nothing happens, the motor is turned off
0	1	Motor rotates counter clockwise
1	0	Motor rotates clockwise
1	1	Not possible in the case of this project

Table 4: Actions of H-Bridge

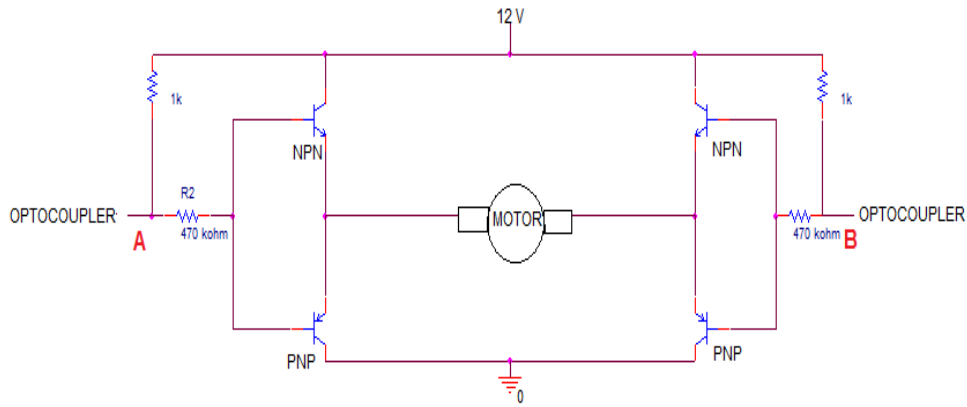


Figure 42: H-Bridge circuitry

3.2) SOFTWARE

This part includes the description of the way we used the different softwares. There are two softwares which we have used namely Bitmap2lcd v1.5a and Arduino v1.0.5 windows. Also we have used GLCD software, but to a limited extent. We have only used its library, that too for including the header files in our arduino code.

3.2.1) BITMAP2LCD v 1.5a

STEPS:

- 1) For a new canvass, we first select the canvas size as 96X40, from the LCD matrix option. (This size of 96X40 is chosen because of speed limitation of our project.)
- 2) Now we import a picture in our canvas. This picture is automatically resized according to our canvas. Also when the picture is imported onto the canvas, it is automatically filtered into a black and white pixel picture i.e. into a raster image. While using this software we can also edit the raster image according to our need.
- 3) With the help of toolbar, we are able to get the encoded file of table of constants for the image. This file is saved with “.h” extension in the GLCD library. This file contains the arrayed data about the image.

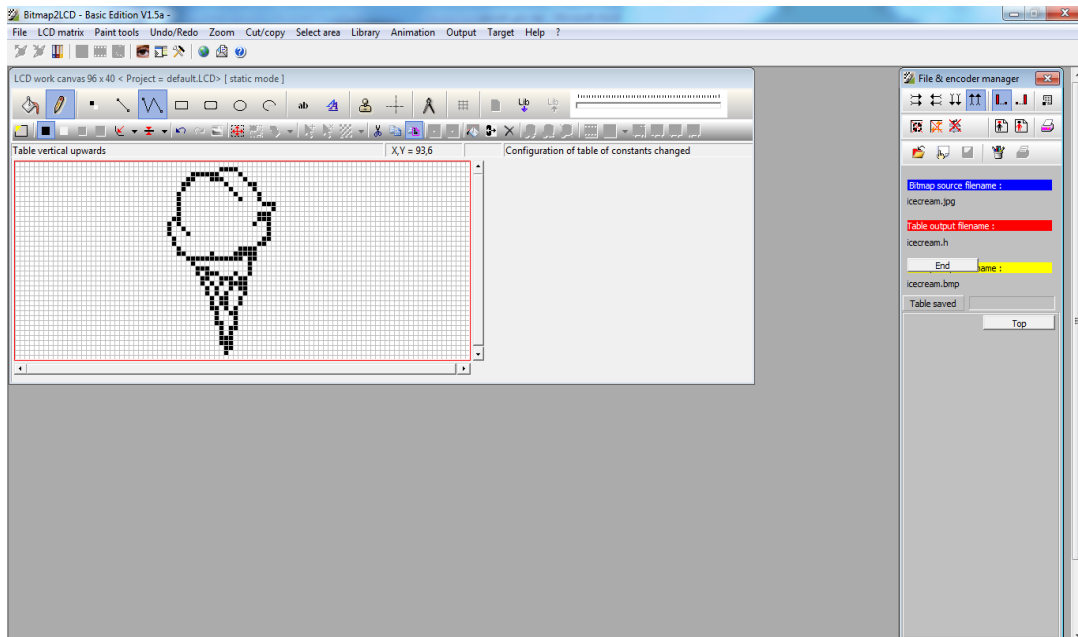


Figure 43: screen-shot of bitmap2lcd

Above figure shows the outlook of the software window, in which the imported image of ice-cream on the canvas is converted into the black and white raster image.

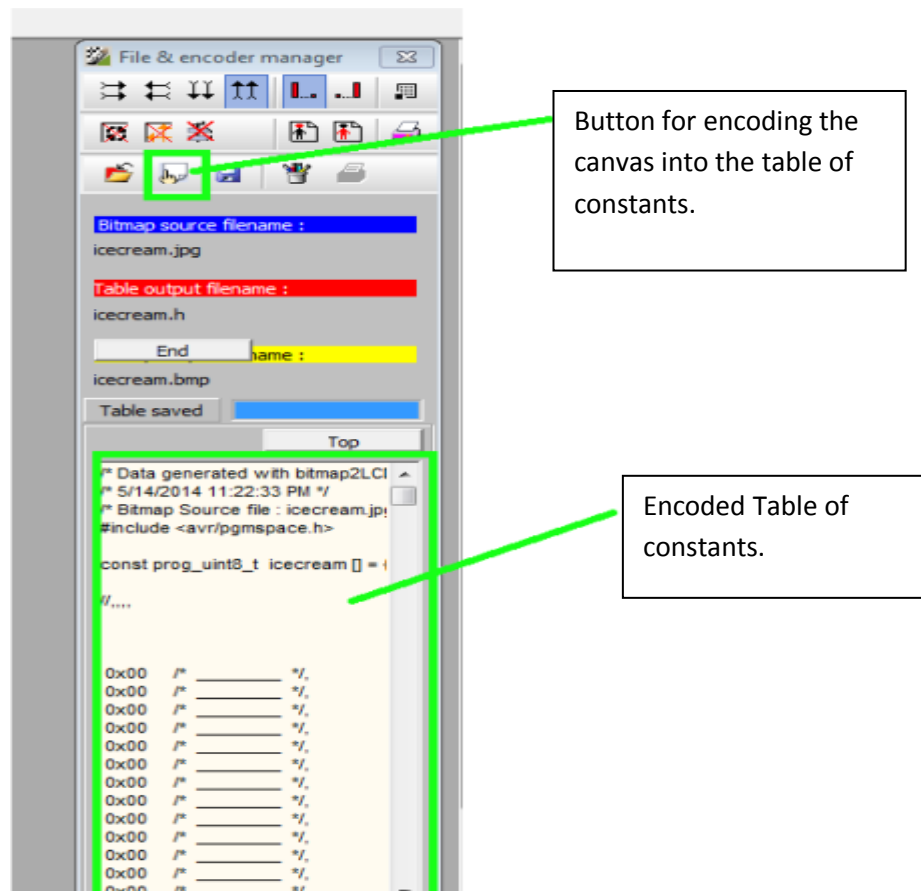


Figure 44: Screenshot of encoded table of constants

3.2.2) ARDUINO v1.0.5 windows

Arduino 1.0.5 is open-source environment software. Thus, this property makes it easy to write code and upload it to the i/o board. It can run on Windows, Mac OS X, and Linux. The code is written in C/C++, Java and based on Processing, avr-gcc, and other open source software. Because it is open source software thus it is very user friendly. A new file in the Arduino is called as the sketch.

Here, in our project it is basically used to load the stored image from the computer and also for the control of the motors. Arduino consists of various keywords and inbuilt functions. Out of all those, the one which are used in our project are discussed below.

a) INBUILT FUNCTIONS^[39]:

□ setup():

This function is called when a sketch starts. It is used to initialize variables, pin modes, start using libraries, etc. After every power up or reset of the Arduino board, this function will run only for one time.

□ Loop()

After creating a setup() function, the loop() function as its name suggests, loops consecutively, allows the program to change and respond. It is used to actively control the Arduino board.

KEYWORDS:

As we are only using the digital i/o pins, following given are the keywords:

□ pinMode(pin, mode)

Configures the specified pin to behave either as an input or an output.

Parameters:

Pin: the number of the pin whose mode we wish to set

Mode: INPUT, OUTPUT, or INPUT_PULLUP.

Returns:

None

□ digitalWrite(pin, value)

Used to make the value of a digital pin to be HIGH or LOW.

If the pin is acting as an OUTPUT , its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

And if the pin is acting as an INPUT, this function will enable (HIGH) or disable (LOW) the internal pull up on the input pin.

Parameters:

Pin: the pin number whose voltage we wish to set

Value: HIGH or LOW.

Returns:

None

□ digitalRead(pin)

This function reads the value from a specified digital pin, and tells whether it is HIGH or LOW.

Parameters:

Pin: the number of the digital pin we want to read. It only takes *integer* value.

Returns:

HIGH or LOW

□ Delay (ms)

It is used to pause by delaying the program for the amount of time (in mili-seconds) specified as parameter.

Parameters:

ms: the number of milliseconds to pause (*unsigned long*).

Returns:

None

b) USER DEFINED VARIABLES USED IN SKETCH

All the user defined variables of this project, are of integer type. The variables used are pixel, x-axis, ycut, xcut, z, i, j and k. In addition to these, there are variables used for defining the state of switches in terms of voltage as 1 or 0 i.e. HIGH or LOW. These are keystate1, keystate2, keystate3 and keystate5.

c) USER – DEFINED FUNCTIONS USED IN SKETCH

- mark()

DESCRIPTION: z-axis pins are namely zdn (pin number 10) and zup (pin number 11). For making pen to go down, voltage of pin 10 is made low. Z-axis motor move downwards when the pin 10 value is low because the whole system is working on active low. Further it

is stopped by making voltage high after a fixed delay. A dot is inscribed. After that zup pin is made low resulting in pen to go up.

- checkycut() and checkxcut()

DESCRIPTION: Here in these functions we keep on checking the value of ycut and xcut, respectively for checkycut() and checkxcut(), until their value is changed from 1 to 0. Here “0” implies the interruption have taken place i.e. a black pixel is present.

- yuprun(), ydnrun(), xuprun() and xdnrun()

DESCRIPTION: These functions are made for the movement of the motors in y-axis and x-axis. Each of these functions, have their own variables which corresponds according to their respective names i.e. yup for yuprun(), ydn for ydnrun(), xup for xuprun() and xdn for xdnrun(). In these, first we made the voltage of anyone of the pins representing yup, ydn, xup or xdn low. This results in the movement of the motor in particular direction. Then in this function we check for the ycut and xcut, using checkycut() and checkxcut(), respectively. Until the value of ycut and xcut is “1”, the motors keep moving in the direction it is going and if their value becomes “0”, then it comes out of checkycut() or checkxcut() function and the voltage of the pin yup (or ydn or xup or xdn) becomes high, resulting in stopping of the motor.

d) PUSH BUTTON SWITCHES

Their function is controlled by the coding on the sketch. Whenever the value of voltage on the Arduino pins goes LOW, then the switch connected to that low voltage pin is pressed. With this knowledge we are able to design the sketch that assign particular work to the particular switch or the combination of switches. These switches enabled us to manually control the motors movement.

Result of pressing any specific key or the combination of keys on the PCB is given as follows:

- KEY 4 → for resetting the whole system and stopping right away.
- KEY1+KEY5 → yup
- KEY1+KEY6 → ydn
- KEY2+KEY5 → xup
- KEY2+KEY6 → xdn
- KEY3+KEY5 → zup

- KEY3+KEY6 → zdn
- KEY 1 → for printing image 1 stored in arduino.
- KEY 2 → for printing image 2 stored in arduino.
- KEY 3 → for printing image 3 stored in arduino.
- KEY 5 → for printing image 4 stored in arduino.

e) LIBRARY FILES INCLUDED ARE:

For including the GLCD library and the encoded files of image, following header files are included:

- 1) glcd.h
- 2) fonts/allFonts.h
- 3) home.h
- 4) icecream.h
- 5) smiley.h
- 6) star.h

Here, the last four mentioned files are bitmap2lcd encoded files of image.

f) PINS ON ARDUINO BOARD ARE INTIALIZED AS:

- xint is assigned to PIN 0;
- yint is assigned to PIN 1;
- key1 is assigned to PIN 2;
- key2is assigned to PIN 3;
- key3is assigned to PIN 4;
- key4is assigned to PIN 5;
- key5 is assigned to PIN 6;
- key6 is assigned to PIN 7;

- yup is assigned to PIN 8;
- ydn is assigned to PIN 9;
- zdn is assigned to PIN 10;
- zup is assigned to PIN 11;
- xdn is assigned to PIN 12;
- xup is assigned to PIN 13;

g) INITIALIZATION OF CONTEXT SWITCH THAT DEFINES INITIAL STAGE

- int keystate1 = 0;
- int keystate2 = 0;
- int keystate3 = 0;
- int keystate5 = 0;

3.2.3) PLOTTING PROCESS:

In this section we will discuss the whole printing process. In this we will tell about the movement of motors on each axis.

1. It works according to the encoded file from bitmap2lcd. This encoded file consists of the data about the image in the form of byte array. This file tells about the black pixel and the white pixel. Thus, as the canvass is of 96X40 bit image then for the byte array it is converted into 96X5 byte array.
2. A particular image is assigned to a particular switch. Thus, we keep noticing which key is being pressed.
3. Regardless of which image is selected, we go with two loops -- 'i' and 'j', which varies from 0 to 4 and 0 to 95 respectively. This 'i' loop varying from 0 to 4, it represents the 40 bit or the 5 bytes. While 0 to 95 of the 'j' loop represents the 96 bit value.

4. Next step is to find the black pixel interruption in the 'i'th loop, for the 'j'th value. For this we keep on checking the array value. If it is greater than zero, then it means there is an black pixel interruption, otherwise not.
5. Next step is to find that out of which one among the 8 bits(equivalent to 1 byte) on the y axis is the black bit.

For finding which of the pixel among jth X ith value is black, we keep on checking the resultant value of the bitwise AND binary operation, of the array data with octal numbers 0x80, 0x40, 0x20,0x01, 0x08, 0x04, 0x02 and 0x01. If the resultant value of any of this binary operated data is greater than zero, then mark() function is called, followed by the xuprun().

When the motor reaches the 8th bit on the y-axis then it have to come back to the 1st bit from where it had gone up, for this xdnrun() is called using another 'z' loop .

After running every time for the 'i'th loop, the motor reaches the last bit i.e. 96th bit on the x-axis, then ydnrun() is called ensuring that the pen have reaches the zero coordinate of the x-axis. And after that, for the next 8 bits plotting along y-axis, we shifts the x-axis on the 9th bit using xuprun().

By using this strategy of dividing the canvass into the five blocks of 96X8bits, it becomes easy to directly use the encoded file from the Bitmap2lcd software.

3.2.4) ALGORITHM:

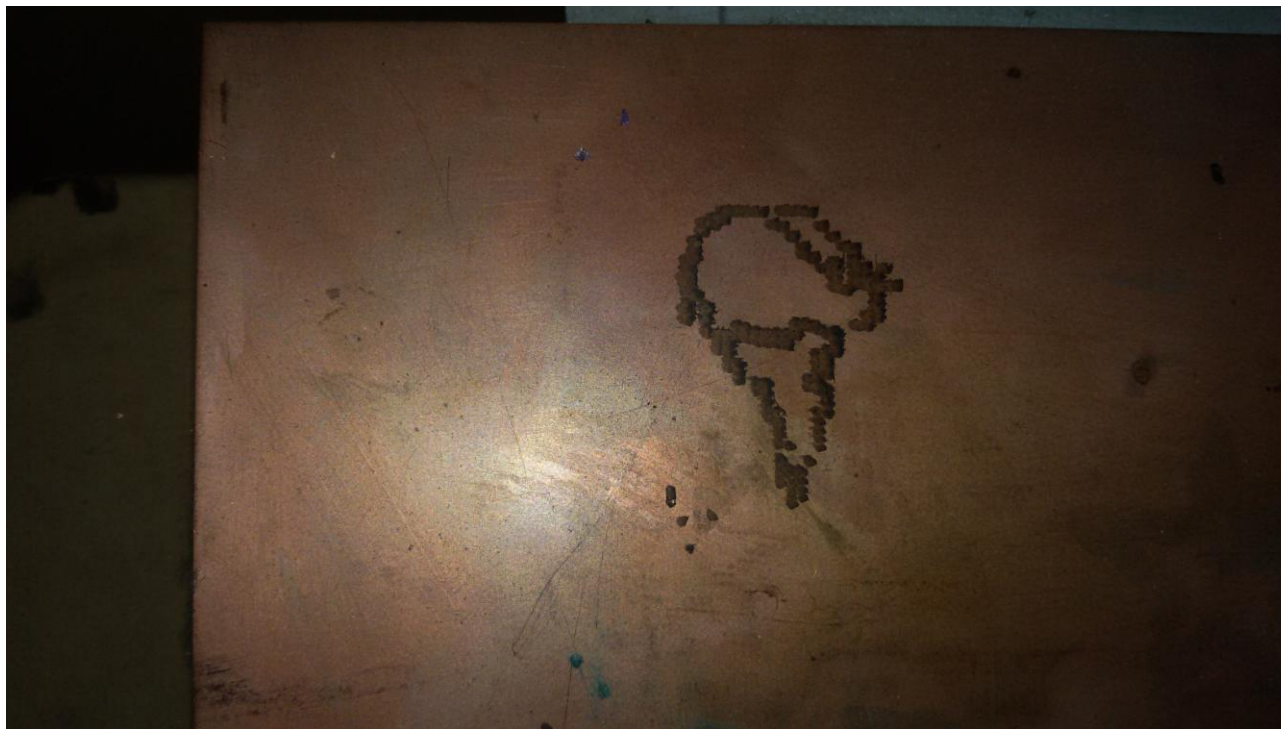
1. Firstly, we include the header files from GLCD library, which enable us to use the inbuilt GLCD files and also the header files of the images to be plotted, which were created by using Bitmap2lcd software.
2. Then we assign the names to the arduino board pins, according to the way they are connected in the hardware setup.
3. Initializations:
 - The variables defining the state of switches (keystate1, keystate2, keystate3, keystate5) are set as 0.
 - The variables used in user defined functions such as xcut and ycut are initialized as 1.
 - Lastly, the variables for the image array such as pixel, x-axis, z, i, j and k are set as 0.
4. Next step is to define the user-defined functions namely, mark(), checkxcut(), checkycut(), xuprun(), xdnrun(), yuprun() and ydnrun() used in the sketch.(as described above)

5. Using pinMode keyword in the setup() function, define which of the arduino pin is input and which is output.
6. In the loop() function , set the arduino pins from 8 to 13 representing yup, ydn, xup, xdn, zup and zdn as high i.e. 1.
7. After this we keep checking the status of the key 4. If keystate1 is 1, then monitor the status of other keys which would be pressed in the combinations. Thus with this the motors will work accordingly as mentioned above. When the value of the keystate1 changes to 0, then again we monitor the status of keys (which now would be pressed individually) and it will start plotting the image which is assigned to the particular key.

This sketch is compiled and is stored into the Arduino module using a USB cable.

CHAPTER 4 : RESULT

Here given below is the snapshot of the output of the project. This picture is the plotted output of the ice-cream.



CHAPTER 5: FUTURE SCOPE

There are certain future possibilities in our project. Those are as following:

1. Increasing the efficiency by using slotted disk instead of the single interrupt.

By using the slotted disk in our project, the number of interrupts per unit time will be increased and so does the number of pixels covered per rotation of the motor. And hence over all plotting speed will be improved. Also for this we have to replace our existing motors by higher rpm motors.

2. Developing a laser Engraver (or laser cutter) by using laser^[40]. For this the basic requirements are:

- laser diode,
- laser housing,
- Laser driver a simple LM317 based circuit can be used.

Certainly there are some limitations with the use of laser if is used in our project. As in case if we will be using a ~200mW red laser. It may not cut through chunks of wood but if not handled with care the user can even loose his/her vision and go blind. So it is very important to remember not to look into the beam ever, even the reflections can be very dangerous if focused.

Laser engraver, as by its name is used for engraving through laser. This have many creative uses some of them shown below:

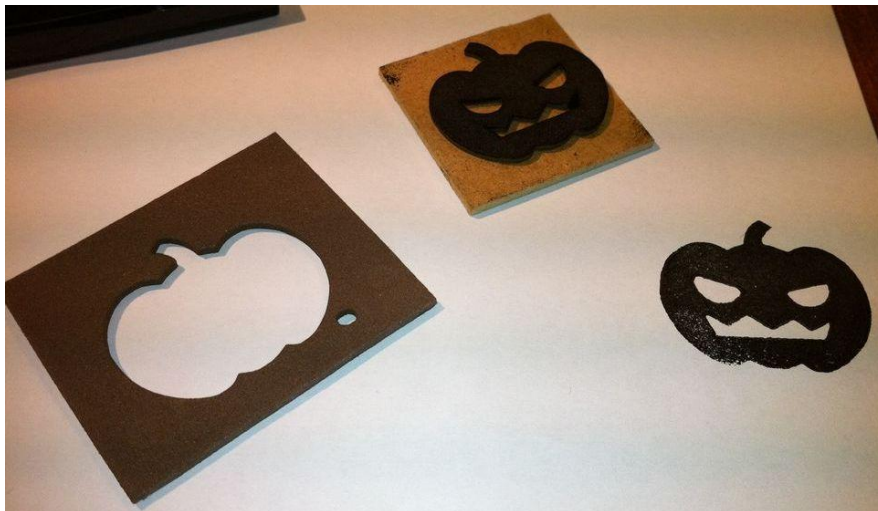
a). Key chain.

Simply put a wooden piece under the laser engraver. In the figure shown below, there is a key chain made of wood engraved with the Binford logo.



b). Cloth and the Foam cutter

Laser acts as cutter for the cloth and foam sheet. As shown below we can get small cut outs of the foam sheet in the required shape.



APPENDIX

APPENDIX A: Sketch of arduino

```
#include <glcd.h>
#include <fonts/allFonts.h>
#include "home.h"
#include "icecream.h"
#include "smiley.h"
#include "star.h"
```

```
unsigned int key1 = 2;
unsigned int key2 = 3;
unsigned int key3 = 4;
unsigned int key4 = 5;
unsigned int key5 = 6;
unsigned int key6 = 7;
```

```
unsigned int xint = 0;
unsigned int yint = 1;
```

```
unsigned int yup = 8;
unsigned int ydn = 9;
unsigned int zup = 11;
unsigned int zdn = 10;
unsigned int xup = 13;
unsigned int xdn = 12;
```

```
int keystate1 = 0;
int keystate2 = 0;
int keystate3 = 0;
int keystate5 = 0;
```

```
unsigned int pixel=0;
unsigned int xaxis=0;
unsigned int ycut=1;
```

```
unsigned int xcut=1;
unsigned int z=0;
unsigned int i=0,j=0,k=0;
```

```
void mark()
{
digitalWrite(zdn, LOW);
delay(700);
digitalWrite(zdn, HIGH);
delay(20);
digitalWrite(zup, LOW);
delay(700);
digitalWrite(zup, HIGH);
}
```

```
void checkycut()
{
ycut=digitalRead(yint);
ycut=digitalRead(yint);
while(ycut==1)
{ycut=digitalRead(yint);}
}
```

```
void checkxcut()
{
xcut=digitalRead(xint);
xcut=digitalRead(xint);
while(xcut==1)
{xcut=digitalRead(xint);}
}
```

```
void yuprun()
{
digitalWrite(yup,LOW);
delay(200);
checkycut();
digitalWrite(yup,HIGH);
delay(10);
}
```

```
void ydnrun()
{
digitalWrite(ydn,LOW);
delay(200);
checkycut();
digitalWrite(ydn,HIGH);
delay(10);
}
```

```
void xuprun()
{
digitalWrite(xup,LOW);
delay(200);
checkxcut();
digitalWrite(xup,HIGH);
delay(10);
}
```

```
void xdnrun()
{
digitalWrite(xdn,LOW);
delay(200);
checkxcut();
digitalWrite(xdn,HIGH);
delay(10);
}
```

```

void setup()
{
  pinMode(key1, INPUT);
  pinMode(key2, INPUT);
  pinMode(key3, INPUT);
  pinMode(key4, INPUT);
  pinMode(key5, INPUT);
  pinMode(key6, INPUT);
  pinMode(xint, INPUT);
  pinMode(yint, INPUT);

  pinMode(yup, OUTPUT);
  pinMode(ydn, OUTPUT);
  pinMode(xup, OUTPUT);
  pinMode(xdn, OUTPUT);
  pinMode(zup, OUTPUT);
  pinMode(zdn, OUTPUT);

}

```

```

void loop()
{
  digitalWrite(yup, HIGH);
  digitalWrite(xup, HIGH);
  digitalWrite(zup, HIGH);
  digitalWrite(xdn, HIGH);
  digitalWrite(ydn, HIGH);
  digitalWrite(zdn, HIGH);

```

```

keystate1=digitalRead(key4);
  while(keystate1==1)
  {

    keystate1=digitalRead(key1);
    keystate2=digitalRead(key5);

```

```

while(keystate1==0&&keystate2==0)
{
keystate1=digitalRead(key1);
keystate2=digitalRead(key5);

digitalWrite(yup, LOW);
}
digitalWrite(yup, HIGH);

```

```

keystate1=digitalRead(key1);
keystate2=digitalRead(key6);
while(keystate1==0&&keystate2==0)
{
keystate1=digitalRead(key1);
keystate2=digitalRead(key6);
digitalWrite(ydn, LOW);
}
digitalWrite(ydn, HIGH);

```

```

keystate1=digitalRead(key2);
keystate2=digitalRead(key5);
while(keystate1==0&&keystate2==0)
{
keystate1=digitalRead(key2);
keystate2=digitalRead(key5);
digitalWrite(xup, LOW);
}
digitalWrite(xup, HIGH);

```

```

keystate1=digitalRead(key2);
keystate2=digitalRead(key6);
while(keystate1==0&&keystate2==0)
{
keystate1=digitalRead(key2);

```

```

        keystate2=digitalRead(key6);

        digitalWrite(xdn, LOW);
    }
digitalWrite(xdn, HIGH);

keystate1=digitalRead(key3);
keystate2=digitalRead(key5);
    while(keystate1==0&&keystate2==0)
    {
        keystate1=digitalRead(key3);
        keystate2=digitalRead(key5);
        digitalWrite(zup, LOW);
    }
digitalWrite(zup, HIGH);

keystate1=digitalRead(key3);
keystate2=digitalRead(key6);
    while(keystate1==0&&keystate2==0)
    {
        keystate1=digitalRead(key3);
        keystate2=digitalRead(key6);

        digitalWrite(zdn, LOW);
    }
digitalWrite(zdn, HIGH);

keystate1=digitalRead(key4);
}

digitalWrite(zup,LOW);
delay(700);
digitalWrite(zup,HIGH);
xaxis=0;
keystate1=digitalRead(key1);

```



```
keystate2=digitalRead(key2);
keystate3=digitalRead(key3);
keystate5=digitalRead(key5);
```

```
while(keystate1==1 && keystate2==1 && keystate3==1 && keystate5==1)
{
keystate1=digitalRead(key1);
keystate2=digitalRead(key2);
keystate3=digitalRead(key3);
keystate5=digitalRead(key5);
}
```

```
if(keystate1==0)
{
for( i=0;i<5;i++)
{
```

```
for( j=0;j<96;j++)
{
yuprun();
```

```
if(star[pixel]>0)
{
```

```
xaxis=star[pixel]&0x80;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=star[pixel]&0x40;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=star[pixel]&0x20;
```

```
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=star[pixel]&0x10;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=star[pixel]&0x08;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=star[pixel]&0x04;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=star[pixel]&0x02;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=star[pixel]&0x01;
if(xaxis>0)
mark();
xuprun();
xuprun();
```

```
for(z=0;z<9;z++)
{
xdnrun();
}
```

```
}
```

```

        pixel++;
    }

    yuprun();

    for(j=97;j>0;j--)
    {
        ydnrun();
    }
    for(z=0;z<9;z++)
    {
        xuprun();
    }
}

```

```

if(keystate2==0)

```

```

{
    for( i=0;i<5;i++)
    {

```

```

        for( j=0;j<96;j++)

```

```

        {
            yuprun();

```

```

        if(icecream[pixel]>0)

```

```

        {

```

```

            xaxis=icecream[pixel]&0x80;

```

```

            if(xaxis>0)

```

```

            mark();
            xuprun();

```

```

            xaxis=icecream[pixel]&0x40;

```

```
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=icecream[pixel]&0x20;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=icecream[pixel]&0x10;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=icecream[pixel]&0x08;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=icecream[pixel]&0x04;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=icecream[pixel]&0x02;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=icecream[pixel]&0x01;
if(xaxis>0)
mark();
xuprun();
```

```
for(z=0;z<9;z++)
```

```

{
xdnrun();
}
}
pixel++;
}

yuprun();

for(j=97;j>0;j--)
{
ydnrun();
}
for(z=0;z<10;z++)
{
xuprun();
}
}
}

if(keystate3==0)
{
for( i=0;i<5;i++)
{

for( j=0;j<96;j++)
{
yuprun();

if(home1[pixel]>0)
{

xaxis=home1[pixel]&0x80;

```

```
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=home1[pixel]&0x40;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=home1[pixel]&0x20;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=home1[pixel]&0x10;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=home1[pixel]&0x08;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=home1[pixel]&0x04;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=home1[pixel]&0x02;
if(xaxis>0)
mark();
xuprun();
```

```
xaxis=home1[pixel]&0x01;
```

```

if(xaxis>0)
mark();
xuprun();

for(z=0;z<9;z++)
{
xdnrun();
}
}
pixel++;
}

yuprun();

for(j=97;j>0;j--)
{
ydnrun();
}
for(z=0;z<10;z++)
{
xuprun();
}
}
}

if(keystate5==0)
{
for( i=0;i<5;i++)
{
for( j=0;j<96;j++)
{
yuprun();

if(smiley[pixel]>0)

```

```
{  
  
    xaxis=smiley[pixel]&0x80;  
    if(xaxis>0)  
        mark();  
    xuprun();  
  
    xaxis=smiley[pixel]&0x40;  
    if(xaxis>0)  
        mark();  
    xuprun();  
  
    xaxis=smiley[pixel]&0x20;  
    if(xaxis>0)  
        mark();  
    xuprun();  
  
    xaxis=smiley[pixel]&0x10;  
    if(xaxis>0)  
        mark();  
    xuprun();  
  
    xaxis=smiley[pixel]&0x08;  
    if(xaxis>0)  
        mark();  
    xuprun();  
  
    xaxis=smiley[pixel]&0x04;  
    if(xaxis>0)  
        mark();  
    xuprun();  
  
    xaxis=smiley[pixel]&0x02;  
    if(xaxis>0)  
        mark();
```



```
xuprun();
```

```
xaxis=smiley[pixel]&0x01;
```

```
if(xaxis>0)
```

```
mark();
```

```
xuprun();
```

```
    for(z=0;z<9;z++)
```

```
    {
```

```
        xdnrun();
```

```
    }
```

```
    }
```

```
    pixel++;
```

```
}
```

```
yuprun();
```

```
    for(j=97;j>0;j--)
```

```
    {
```

```
        ydnrun();
```

```
    }
```

```
    for(z=0;z<10;z++)
```

```
    {
```

```
        xuprun();
```

```
    }
```

```
    }
```

```
    }
```

```
}
```

REFERENCES

1. <http://en.wikipedia.org>
2. <https://www.flickr.com/photos/anachrocomputer/>
3. <http://www.testequipmentconnection.net/Plotters.asp>
4. <http://www.premiersign.co.uk/proddetail.asp?prod=MimakiCG-60SLProfessional>
5. <http://www.textileweb.com/doc/m9000-digital-cutting-system-ets-m9000-0001>
6. <http://scientificcuriosity.blogspot.in/2006/09/how-is-digital-photo-stored.html>
7. http://en.wikipedia.org/wiki/Vector_graphics
8. http://en.wikipedia.org/wiki/Raster_graphics
9. http://pippin.gimp.org/image_processing/chap_dir.html
10. <http://bitmap2lcd.com>
11. <http://www.mikroe.com/visualglcd/>
12. <http://www.electrical4u.com/working-or-operating-principle-of-dc-motor/>
13. <http://www.engineersgarage.com/electronic-components/7805-voltage-regulator-ic>
14. <http://arduino.cc/en/Main/ArduinoBoardUno>
15. <http://www.circuitstoday.com/arduino-for-beginners>
16. <http://www.circuitstoday.com/interfacing-led-using-push-button-switch-to-8051>
17. <http://www.arunet.co.uk/tkboyd/ec/ec1optoiso.htm>
18. <http://www.electroncomponents.com/MOC-7811-Encoder-Sensor>
19. <http://www.circuitstoday.com/wpcontent/uploads/2009/08/reflective-slotted-optocoupler.jpg>
20. <http://www.circuitstoday.com/optocoupler-devices-and-application>
21. <http://energiaalternativa.forumcommunity.net/?t=30137183&st=540>
22. <http://playground.arduino.cc/CommonTopics/PullUpDownResistor>

23. <http://ugpro143.blogspot.in/2010/08/lm358-dual-opamp-features.html>
24. <http://www.engineersgarage.com/tutorials/understanding-op-amp-2?page=5>
25. <http://www.dummies.com/how-to/content/electronics-components-how-to-use-an-op-amp-as-a-v.html>
26. <https://learn.sparkfun.com/tutorials/pull-up-resistors/what-is-a-pull-up-resistor>
27. <http://playground.arduino.cc/CommonTopics/PullUpDownResistor>
28. <http://at-sky.com.vn/san-pham/1648-rna05-a103j.html> 2NDCKT
29. <http://ronelex.com/optoisolator-ic-power-section/196/>
30. <http://www.mcmanis.com/chuck/Robotics/tutorial/h-bridge/>
31. http://www.societyofrobots.com/schematics_h-bridgedes.shtml
32. <http://www.instructables.com/id/H-Bridge-on-a-Breadboard/>
33. <http://www.engineersgarage.com/insight/how-dc-motor-works>
34. <http://books.google.co.in/books?id=tBdKnLcf2oEC&pg=PA88&dq=dc+geared+motor&hl=en&sa=X&ei=kYd2U53jDtOOuATzoGgDw&ved=0CD8Q6AEwAA#v=onepage&q=dc%20geared%20motor&f=false>
35. <http://www.engineersgarage.com/insight/how-geared-dc-motor-works?page=6>
36. <http://www.electrical4u.com/working-or-operating-principle-of-dc-motor/>
37. http://img.frbiz.com/nimg/35/0c/db7447fa8c6c1da263468e70d891-0x0-0/strong_style_color_b82220_dc_gear_motor_strong.jpg
38. <http://electricalquestionsguide.blogspot.in/2011/05/dc-motors-advantages-disadvantages-ac.html>
39. <http://arduino.cc/en/Reference/HomePage>
40. <http://www.instructables.com/id/Pocket-laser-engraver/step9/Final-results/>
41. <http://www.embed4u.com/tag/optocoupler-working-principleembed4u-com/>
42. <https://wiki.analog.com/university/courses/electronics/electronics-lab-22>