*A*

MINOR PROJECT REPORT

ON

# AUTOMATION USING HAND GESTURES

*Submitted in partial fulfillment of the requirements for the award of the degree*

*of*

## BACHELOR OF TECHNOLOGY

*in*

## ELECTRICAL ENGINEERING



## Submitted by:

Ansh Gupta (20126013)

Abhay Kumar (20112003)

Dikshant Jindal (20126023)

Shobhit Gupta (20126053)

Aryan (20126015)

## Supervised by:

Dr, Karanveer Dhingra

**DEPARTMENT OF ELECTRICAL ENGINEERING**

**DR B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY JALANDHAR**


**May 2023**

DR B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY JALANDHAR

DEPARTMENT OF ELECTRICAL ENGINEERING

# CERTIFICATE

We hereby submit the minor project report entitled **Automation Using Hand Gesture Recognition** in the **Department of Electrical Engineering** of **Dr B R Ambedkar National Institute of Technology Jalandhar**, under the supervision of **Dr. Karanveer Dhingra**, Assistant Professor, Department of Electrical Engineering, Dr B R Ambedkar National Institute of Technology Jalandhar Punjab, India.

Ansh Gupta (20126013)

Abhay Kumar (20112003)

Dikshant Jindal (20126023)

Shobhit Gupta (20126053)

Aryan (20126015)

The project report is hereby approved for submission.

Dr. Karanveer Dhingra
(Assistant Professor)

Date: 13th May, 2023

# PROBLEM DESCRIPTION

The project on automation using hand gesture recognition aims to address the limitations of traditional control interfaces by providing an intuitive and natural means of interaction between humans and machines. Traditional control mechanisms, such as buttons, switches, and remote controllers, often require physical contact or complex configurations, hindering ease of use and limiting the potential for automation in various applications.

The primary problem this project tackles is the need for a more user-friendly and efficient control system that utilizes hand gestures to automate and control devices. By recognizing and interpreting hand gestures, users can seamlessly interact with the automation system, eliminating the need for physical contact or intricate setups.

Specifically, the project focuses on three key aspects of automation control:

- **Controlling the Speed of a DC Motor:** Traditional methods of controlling DC motor speed involve manual adjustment of potentiometers or switches. These approaches lack precision and real-time adjustability. The project seeks to provide a more accurate and flexible control mechanism by allowing users to adjust the speed of a DC motor using hand gestures.

- **Controlling the Direction of a DC Motor (Using PWM):** Changing the direction of a DC motor typically requires additional circuitry or mechanical switching mechanisms. The project aims to simplify this process by utilizing pulse width modulation (PWM) techniques in conjunction with hand gestures. By varying the duty cycle of the PWM signal, users can conveniently switch the direction of the DC motor.

- **AC Light Dimmer Using TRIAC:** Adjusting the brightness of an AC light source often involves using manual dimmers or specialized control interfaces. The project endeavors to provide a seamless and intuitive approach to dimming AC lights by incorporating a TRIAC and hand gesture

recognition. This allows users to control the brightness levels of the light source effortlessly.

By addressing these problems through the integration of hand gesture recognition, machine learning algorithms, and hardware components, the project aims to create an automation system that enhances user experience, improves control precision, and expands the possibilities for automation in various domains

# INTRODUCTION

The rapid advancements in technology have revolutionized various fields, including automation and control systems. Automation plays a crucial role in improving efficiency, productivity, and convenience in numerous applications. One significant aspect of automation is the ability to control devices through hand gestures, providing a more intuitive and natural interaction between humans and machines. In this project, we explore the concept of automation using hand gesture recognition and its applications in controlling a DC motor's speed, direction, and an AC light dimmer using a TRIAC.

The primary objective of this project is to develop a system that allows users to control different aspects of automation by simply using hand gestures. Hand gesture recognition is achieved by utilizing machine learning techniques, specifically the K-Nearest Neighbors (KNN) algorithm. This algorithm has proven to be effective in accurately classifying hand gestures based on their features, making it suitable for real-time applications.

The implementation of this project involves a combination of software and hardware components. Python, a widely used programming language, serves as the primary platform for developing the gesture recognition system. The integration of various libraries, such as Mediapipe and OpenCV, enables the detection and tracking of hand gestures in real-time video streams. These libraries provide robust functionality for extracting hand features, such as finger positions, palm orientation, and gestures, which are crucial for accurate recognition.

On the hardware side, the system utilizes an Arduino Uno microcontroller board, which acts as the central processing unit. It provides the necessary computational power to analyze the hand gesture data received from the computer and translates it into control signals for different devices. To control the speed of a DC motor, an H-bridge circuit is implemented using the L293D Motor Driver IC. This IC allows bidirectional control of the motor, enabling both forward and reverse rotation. The pulse width modulation (PWM) technique is employed to adjust the motor's speed, providing precise control over its rotational velocity.

In addition to controlling the speed, the system also incorporates the capability to control the direction of the DC motor. This is achieved by utilizing the PWM technique in conjunction with the H-bridge circuit. By varying the duty cycle of

the PWM signal, the motor's direction can be changed, enabling it to rotate clockwise or counterclockwise. Furthermore, the project extends its automation capabilities to include the control of an AC light dimmer using a TRIAC. A BT136-600E TRIAC is employed for this purpose, along with supporting components such as the MOC3021 IC (Triac Driven Optoisolator) and the MCT2e IC (Phototransistor Optocoupler IC). The TRIAC enables the regulation of the power supplied to the AC light source, resulting in adjustable brightness levels.

To ensure the appropriate power supply for the system, a bridge rectifier is utilized to convert the AC power input to a DC voltage suitable for the microcontroller and other components. This rectifier circuit provides a stable and regulated DC voltage, ensuring the reliable operation of the system.

In conclusion, this project focuses on the development of an automation system using hand gesture recognition. By harnessing the power of machine learning algorithms and integrating them with hardware components, the system enables users to control the speed and direction of a DC motor and adjust the brightness of an AC light source using simple hand gestures. The utilization of the KNN algorithm, Python programming, Mediapipe, OpenCV, and various hardware components such as the Arduino Uno, H-bridge, L293D Motor Driver IC, BT136-600E TRIAC, MOC3021 IC, MCT2e IC, and Bridge Rectifier contribute to the realization of this interactive and user-friendly automation system.

# MOTIVATION

The motivation behind undertaking the project on automation using hand gesture recognition stems from the increasing demand for intuitive and user-friendly control interfaces in automation systems. Traditional control mechanisms often require physical contact, complex configurations, or reliance on remote controllers, limiting the ease of use and hindering the potential for automation in various applications.

The primary motivation is to develop a control system that allows users to interact with automation devices effortlessly and naturally through hand gestures. By recognizing and interpreting these gestures, users can seamlessly control the speed of a DC motor, change its direction, and adjust the brightness of an AC light source. This intuitive interaction not only enhances user experience but also opens up possibilities for automation in diverse domains, such as smart homes, robotics, industrial automation, and assistive technologies.


Furthermore, the motivation stems from the advancements in machine learning algorithms and computer vision techniques that enable accurate hand gesture recognition in real-time. Leveraging these technologies, the project aims to provide precise and reliable gesture recognition capabilities, ensuring accurate translation of user gestures into control commands.

Additionally, the project is driven by the desire to bridge the gap between humans and machines, creating a more seamless and natural interaction paradigm. By utilizing hand gestures, which are an innate form of communication, the project aims to remove barriers and make

automation systems more accessible to a wide range of users, including those with physical limitations or impairments.

The increasing popularity of automation and the need for more user-friendly control interfaces, coupled with advancements in machine learning and computer vision, serve as strong motivations to explore and develop an automation system based on hand gesture recognition. By addressing these motivations, the project aims to contribute to the advancement of automation technologies, improve user experiences, and pave the way for more intuitive and efficient control interfaces in the future.

# TECHNOLOGY USED

## SOFTWARE PARTS:

- **ARDUINO IDE**

The Arduino IDE (Integrated Development Environment) is a software tool used for programming Arduino microcontroller boards. It provides an easy-to-use interface for writing, compiling, and uploading code to Arduino boards, making it accessible for beginners and experienced developers alike. Here are some key details about the Arduino IDE:

Installation: The Arduino IDE can be downloaded and installed on various operating systems such as Windows, macOS, and Linux. It is available for free from the official Arduino website.

Code Editor: The IDE offers a text editor where you can write your Arduino code. It supports basic features like syntax highlighting, auto-indentation, and code suggestions, which can help streamline the coding process.

Libraries: The Arduino IDE comes with a built-in library manager that allows you to easily include pre-written code libraries in your projects. Libraries provide ready-to-use functions and modules to simplify complex tasks such as controlling sensors or interfacing with external devices.

Compilation and Upload: The IDE includes a compiler that translates your Arduino code into a machine-readable format. Once the code is compiled successfully, it can be uploaded to the Arduino board via a USB connection. The IDE handles the process of compiling and uploading the code with just a few clicks.

Serial Monitor: The Arduino IDE provides a built-in Serial Monitor tool, which allows you to communicate with the Arduino board and view data sent from the board or print debugging information. It helps in testing and debugging your code by monitoring the serial communication between the Arduino board and your computer.

Board Manager: The IDE supports a wide range of Arduino-compatible boards. The Board Manager allows you to select the specific board you are working with and install the necessary board definitions and drivers.

Example Sketches: The Arduino IDE includes a collection of example sketches that demonstrate various functionalities and features. These examples serve as a helpful starting point for beginners and provide insights into different applications of Arduino.

Community and Resources: The Arduino IDE is widely used, and there is an active community of Arduino enthusiasts and developers. You can find numerous tutorials, guides, and projects online to help you get started and expand your knowledge.

Overall, the Arduino IDE provides a user-friendly environment for programming Arduino boards, abstracting many of the complexities involved in microcontroller programming. It has played a significant role in making embedded systems development more accessible and has contributed to the popularity of Arduino-based projects.

- **PYTHON**


Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today.

Stack Overflow's 2022 Developer Survey revealed that Python is the fourth most popular programming language, with respondents saying that they use Python almost 50 percent of the time in their development work. Survey results also showed that Python is tied with Rust as the most-wanted technology, with 18% percent of developers who aren't using it already saying that they are interested in learning Python

## What is Python used for?

Python is commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

"Writing programs is a very creative and rewarding activity," says University of Michigan and Coursera instructor Charles R Severance in his book *Python for Everybody*. "You can write programs for many reasons, ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem."

What can you do with python? Some things include:

- Data analysis and machine learning
- Web development
- Automation or scripting
- Software testing and prototyping
- Everyday tasks

Here's a closer look at some of these common ways Python is used.

- **Data analysis and machine learning**
- **Web development**
- **Automation or scripting**
- **Software testing and prototyping**

- MEDIAPIPE


MediaPipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc.

## MediaPipe Solutions

Solutions are open-source pre-built examples based on a specific pre-trained TensorFlow or TFLite model. You can check Solution specific models here. MediaPipe Solutions are built on top of the Framework. Currently, it provides sixteen solutions, as listed below.

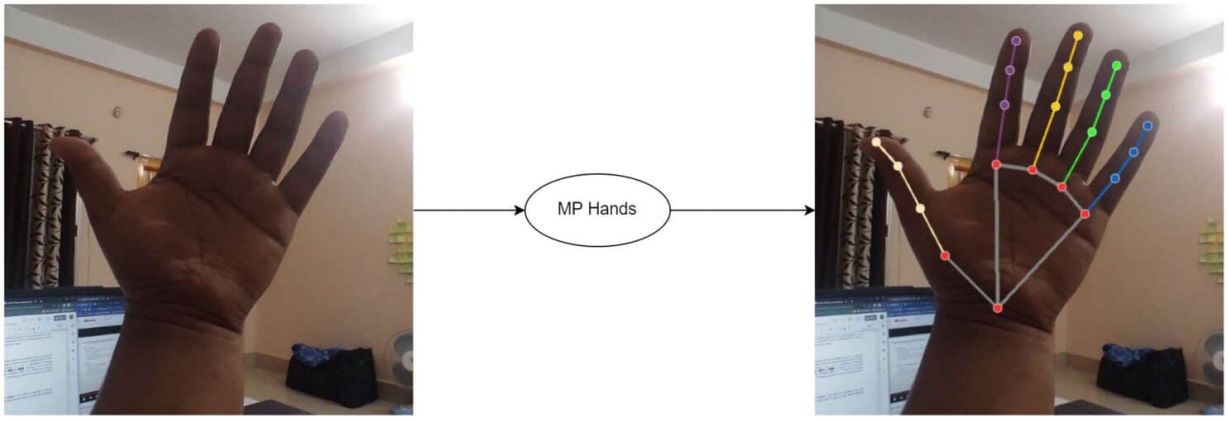
1. Face Detection
2. Face Mesh
3. Iris
4. Hands
5. Pose
6. Holistic
7. Selfie Segmentation
8. Hair Segmentation
9. Object Detection
10. Box Tracking
11. Instant Motion Tracking
12. Objectron
13. KNIFT
14. AutoFlip
15. MediaSequence
16. YouTube 8M

# Framework

The Framework is written in C++, Java, and Obj-C, which consists of the following APIs.

1. Calculator API (C++).
2. Graph construction API (Protobuf).
3. Graph Execution API (C++, Java, Obj-C).

## Graphs




The MediaPipe perception pipeline is called a Graph. Let us take the example of the first solution, Hands. We feed a stream of images as input which comes out with hand landmarks rendered on the images.

The flow chart below represents the MP (Abbr. MediaPipe) hand solution graph.

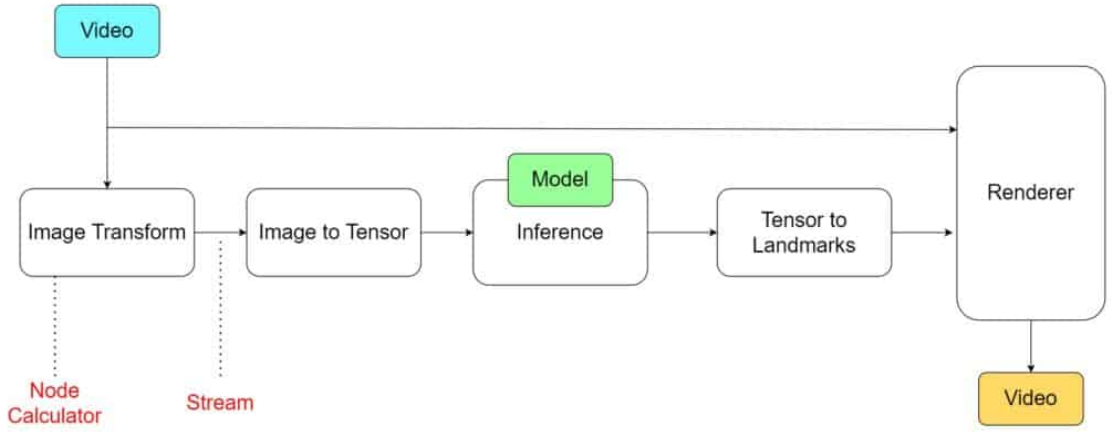



**Fig: MediaPipe hands solution graph**

In computer science jargon, a graph consists of Nodes connected by Edges. Inside the MediaPipe Graph, the nodes are called Calculators,

and the edges are called Streams. Every stream carries a sequence of Packets that have ascending time stamps.
In the image above, we have represented Calculators with rectangular blocks and Streams using arrows.

- **OPEN CV**


OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

from the above original image, lots of pieces of information that are present in the original image can be obtained. Like in the above image there are two faces available and the person(I) in the images wearing a bracelet, watch, etc so by the help of OpenCV we can get all these types of information from the original image.
It's the basic introduction to OpenCV we can continue the Applications and all the things in our upcoming articles.

### Applications of OpenCV:

There are lots of applications which are solved using OpenCV, some of them are listed below

1. face recognition
2. Automated inspection and surveillance
3. number of people – count (foot traffic in a mall, etc)
4. Vehicle counting on highways along with their speeds
5. Interactive art installations

6. Anomaly (defect) detection in the manufacturing process (the odd defective products)
7. Street view image stitching
8. Video/image search and retrieval
9. Robot and driver-less car navigation and control
10. object recognition
11. Medical image analysis
12. Movies – 3D structure from motion
13. TV Channels advertisement recognition

## OpenCV Functionality

Image/video I/O, processing, display (core, imgproc, highgui)
Object/feature detection (objdetect, features2d, nonfree)
Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
Computational photography (photo, video, superres)
Machine learning & clustering (ml, flann)
CUDA acceleration (gpu)

## Image-Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it. If we talk about the basic definition of image processing then "Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality".
Digital-Image:
An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of any pair of coordinates (x, y) is called the intensity or gray level of the image at that point.
In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what color it should be.
Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirements associated with that image.

Image processing basically includes the following three steps:

1. Importing the image
2. Analyzing and manipulating the image
3. Output in which result can be altered image or report that is based on image analysis

# • MACHINE LEARNING (KNN MODEL)

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

## Common machine learning algorithms

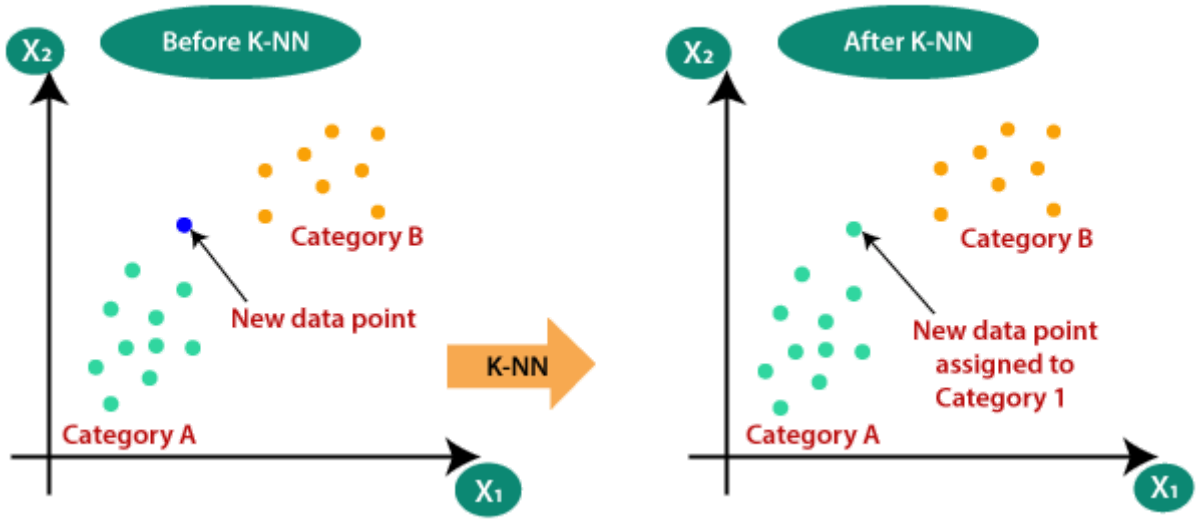
A number of machine learning algorithms are commonly used. These include:

1. Neural networks: Neural networks simulate the way the human brain works, with a huge number of linked processing nodes. Neural networks are good at recognizing patterns and play an important role in applications including natural language translation, image recognition, speech recognition, and image creation.
2. Linear regression: This algorithm is used to predict numerical values, based on a linear relationship between different values. For example, the technique could be used to predict house prices based on historical data for the area.
3. Logistic regression: This supervised learning algorithm makes predictions for categorical response variables, such as"yes/no" answers to questions. It can be used for applications such as classifying spam and quality control on a production line.
4. Clustering: Using unsupervised learning, clustering algorithms can identify patterns in data so that it can be grouped. Computers can help data scientists by identifying differences between data items that humans have overlooked.
5. Decision trees: Decision trees can be used for both predicting numerical values (regression) and classifying data into categories. Decision trees use a branching sequence of linked decisions that can be represented with a tree diagram. One of the advantages of decision trees is that they are easy to validate and audit, unlike the black box of the neural network.

6. Random forests: In a random forest, the machine learning algorithm predicts a value or category by combining the results from a number of decision trees.

## Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:




## How does K-NN work?

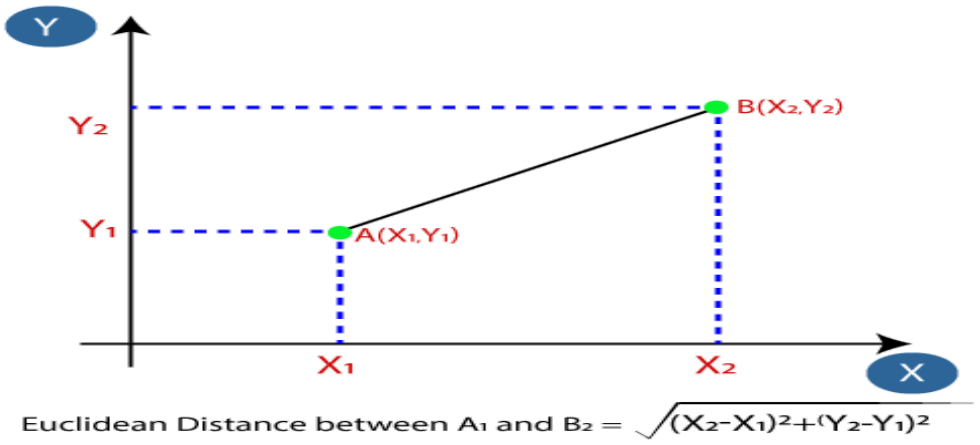
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

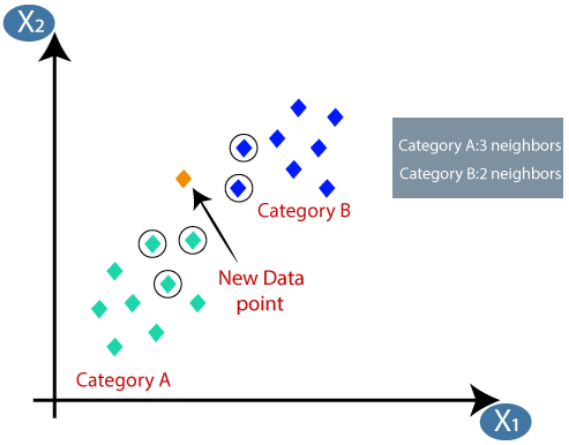
Suppose we have a new data point and we need to put it in the required category. Consider the below image:

Firstly, we will choose the number of neighbors, so we will choose the k=5.

- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



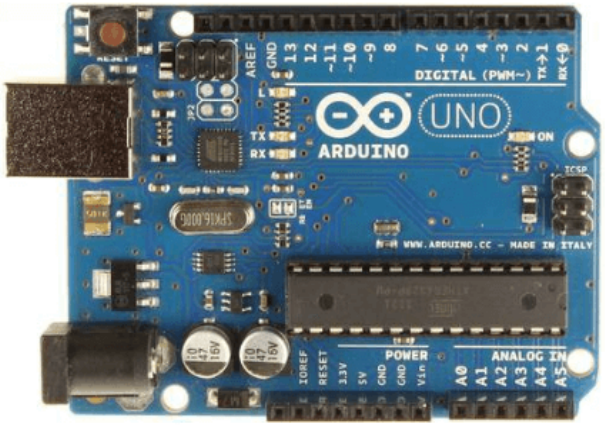

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:




As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

# • HARDWARE PARTS

## • Arduino Uno



The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board.

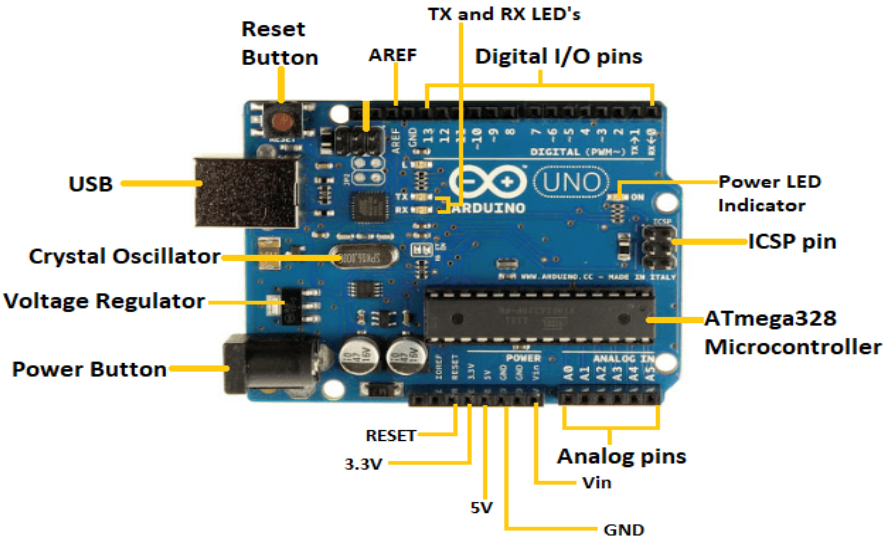
Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits.

The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms.

The IDE is common to all available boards of Arduino.

The Arduino board is shown below:

**The components of Arduino UNO board are shown below:**

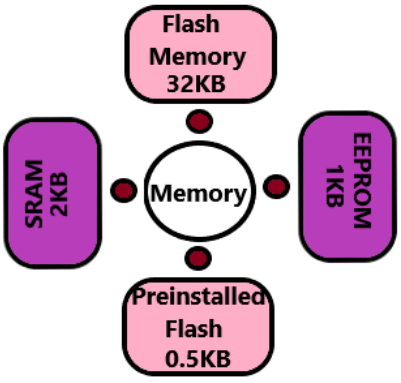


**Let's discuss each component in detail.**

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the ATmel family. The processor code inside it is 8-bit. It combines Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The **ON** status of LED shows the power is activated. When the power is **OFF**, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LEDs**- The successful flow of data is represented by the lighting of these LED's.
- **AREF**- The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- Reset button- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

# Memory

The memory structure is as shown in the image:




The preinstalled flash has a bootloader, which takes the memory of 0.5 Kb.
Here, SRAM stands for Static Random Access Memory, and EEPROM stands for Electrically Erasable Programmable Read-Only Memory.

## Technical Specifications of Arduino UNO

The technical specifications of the Arduino UNO are listed below:

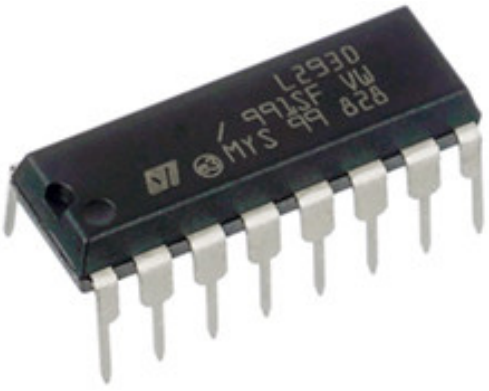
- There are 20 Input/Output pins present on the Arduino UNO board. These 20 pins include 6 PWM pins, 6 analog pins, and 8 digital I/O pins.
- The PWM pins are Pulse Width Modulation capable pins.
- The crystal oscillator present in Arduino UNO comes with a frequency of 16MHz.
- It also has an Arduino integrated WiFi module. Such an Arduino UNO board is based on the Integrated WiFi ESP8266 Module and ATmega328P microcontroller.
- The input voltage of the UNO board varies from 7V to 20V.
- Arduino UNO automatically draws power from the external power supply. It can also draw power from the USB

## • H-Bridge

The spinning direction of a DC motor can be controlled by changing the polarity of its input voltage. A common technique for doing this is to use an H-bridge.
An H-bridge circuit consists of four switches with the motor in the center forming an H-like arrangement.
Closing two specific switches at a time reverses the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.

## L293D Motor Driver IC

The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors or a single stepper motor. This means it can drive up to two motors individually which makes it ideal for building a two-wheeled robotic platform.



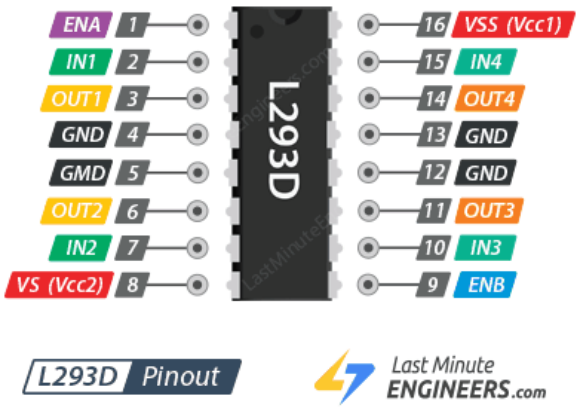

The L293D is most often used to drive motors, but can also be used to drive any inductive load such as a relay solenoid or large switching power transistor.
It is capable of driving four solenoids, four uni-directional DC motors, two bi-directional DC motors or one stepper motor.
The L293D IC has a supply range of 4.5V to 36V and is capable of 1.2A peak output current per channel, so it works very well with most of our motors.
The IC also includes built-in kick-back diodes to prevent damage when the motor is de-energized.

Technical Specifications
Here are the specifications:

| | |
|---|---|
| Motor output voltage | 4.5V - 36V |
| Logic input voltage | 5V |
| Output Current per channel | 600mA |
| Peak Output Current per Channel | 1.2A |

## L293D Motor Driver IC Pinout

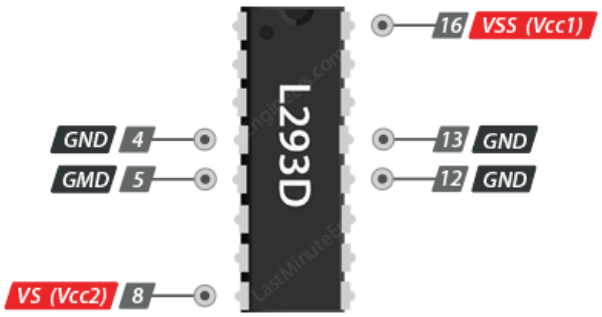
The L293D IC has a total of 16 pins that connect it to the outside world. The pinout is as follows:
Let's get acquainted with all the pins
zone by one.




### Power Pins

The L293D motor driver IC actually has two input power pins – VS and VSS.

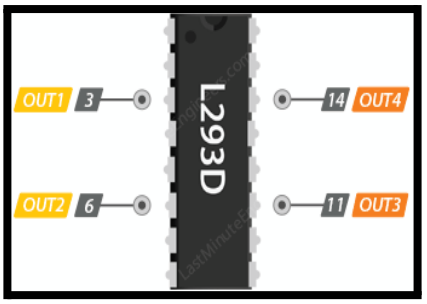
**The VS (Vcc2)** pin gives power to the internal H-Bridge of the IC to




drive the motors. You can connect an input voltage anywhere between 4.5 to 36V to this pin.
**VSS (Vcc1)** is used to drive the internal logic circuitry which should be 5V.

**GND** pins are common ground pins. All 4 GND pins are internally connected and used to dissipate the heat generated under high load conditions.

**Output Pins:**




The L293D motor driver's output channels for the motor A and B are brought out to pins OUT1,OUT2 and OUT3,OUT4 respectively. You can connect two 5-36V DC motors to these pins.
Each channel on the IC can deliver up to 600mA to the DC motor. However, the amount of current supplied to the motor depends on the system's power supply.

**Direction Control Pins**


By using the direction control pins, you can control whether the motor rotates forward or backward. These pins actually control the switches of the H-Bridge circuit inside the L293D IC.
The IC has two direction control pins for each channel. The IN1 and IN2 pins control the spinning direction of motor A; While IN3

and IN4 control the spinning direction of motor B.The spinning direction of the motor can be controlled by applying logic HIGH (5V) or logic LOW (Ground) to these inputs. The chart below shows how this is done.

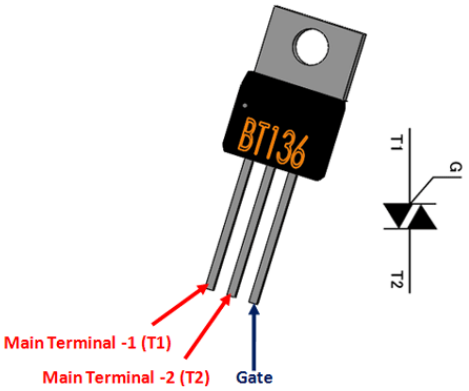
| IN1 | IN2 | Spinning Direction |
|---|---|---|
| Low(0) | Low(0) | Motor OFF |
| High(1) | Low(0) | Forward |
| Low(0) | High(1) | Backward |
| High(1) | High(1) | Motor OFF |

**Speed Control Pins**

The speed control pins ENA and ENB are used to turn on/off the motors and control its speed.

Pulling these pins HIGH will cause the motors to spin, while pulling it LOW will stop them. But, with Pulse Width Modulation (PWM), you can actually control the speed of the motors.

- **BT136-600E TRIAC**




BT136-600E TRIAC Pinout

### Pin Configuration

| Pin Number | Pin Name | Description |
|---|---|---|
| 1 | Main Terminal 1 | Connected to Phase or neutral of AC mains |
| 2 | Main Terminal 2 | Connected to Phase or neutral of AC mains |
| 3 | Gate | Used to trigger the SCR. |

_

## Features

- Maximum Terminal current: 4A
- On-state Gate voltage: 1.4V
- Gate trigger current: 10mA
- Max Terminal Voltage is 600 V
- Holding current: 2.2mA
- Latching current: 4mA
- Available in To-220 Package

Note: Complete Technical Details can be found in the datasheet present at the end of this page.

### Other TIRACs

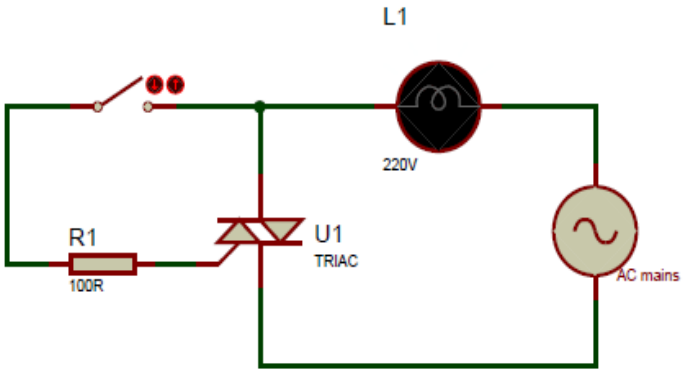
BT139, BTA16, BT169, Q4008

## BT136 TRIAC Overview

The BT136 is TRIAC with 4A maximum terminal current. The gate threshold voltage of the BT136 is also very less so can be driven by digital circuits.
Since TRIACs are bi-directional switching devices they are commonly used for switching AC applications. So if you are looking to switch off control (dim, speed control) an AC load which consumes less than 6A with a digital device like microcontroller or microprocessor then BT136 might be the right choice for you.

## How to use BT136

There are many different ways to use a TRIAC, since the device is bi-directional the TRIAC gate can be triggered with either positive voltage or negative voltage. So this allows the TIRAC to be operated in four different modes. You can read this article if you want to know more about the switching modes. A simple TRIAC switching circuit is shown below.




In this circuit the TRIAC can be turned using the switch, when the switch is pressed the TRIAC closes the connection for the AC bulb though the AC mains. For this to happen, the gate pin of the TRIAC should receive a voltage greater than the threshold gate voltage and should also get a current that is greater than gate trigger current. This will make the TRIAC turn on.

Since the TRIAC and SCR share most of the same characteristics, just like SCR the TRIAC will also not turn off when the gate voltage is removed. We need a special type of circuit called a commutation circuit to turn off the SCR again. This commutation is normally done by reducing the load current (forced commutation) less than the holding current. To put it simply the TRIAC will remain turned on only till the load current is greater than the holding current of the TRIAC.

Note: Commutation is not required in AC switching circuits because the TRIAC will not latch in on state since the AC voltage reaches zero for every half cycle.

Other than controlling through switches the BT136 can also be controlled through a microcontroller or a microprocessor. To do this we need an Opto-isolator like MOC3021 to isolate the AC circuit from Digital electronics. This way the Load can not only be switched but also the output voltage can be controlled by using PWM signals for fast switching.

## TRIAC Application Tips

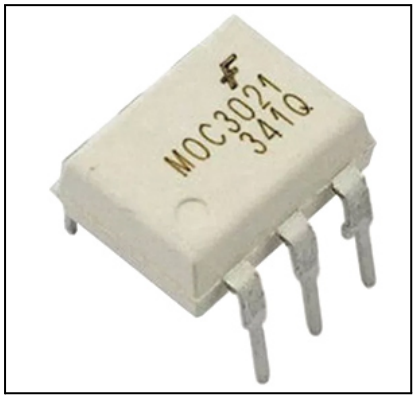
Since TRIACS deal with AC voltages, the circuit involving them has to be designed properly to avoid problem some tips are shared below

- All TRIAC circuits suffer from an effect called Rate Effect. This occurs when the TRIAC is switching frequently and a sudden high voltage occurs at either main terminal of the TRIAC and damages the TRIAC itself. It can be avoided by using a snubber circuit.
- Similarly there is another effect called backlash effect. This occurs due to the capacitance that gets accumulated between the two terminals of the MT1 and MT2 of the TRIAC. Due to this the TRIAC will not turn on even if the gate voltage is applied. This problem can be solved by providing a resistance in series for the capacitance to discharge.
- When controlling the output AC voltage for dimmer or speed control applications a Zero crossing method is always recommended to be used.
- In switching circuits the TRIAC is easily subjected to harmonics and EMI interference hence should be isolated from other digital electronics.
- There is a chance of backward current when the TRIAC is switching inductive loads, so an alternate discharge path has to be provided for the load to drain the inrush current.

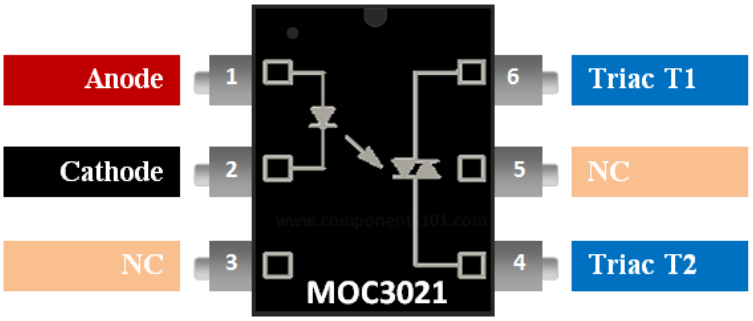
## Applications

- **AC Light dimmers**
- **Strode lights**
- **AC motor speed control**
- **Noise coupling circuits**
- **Controlling AC loads using MCU/MPU**
- **Ac/DC Power control**

# MOC3021 IC. Triac Driven Optoisolator




_MOC3021 Triac




MOC3021 Opto Isolator Pinout

## MOC3021 Pin Configuration

| Pin Number | Pin Name | Description |
|---|---|---|
| 1 | Anode (A) | Anode pin of the IR LED. Connected to logic input |
| 2 | Cathode (C) | Cathode pin of the IR LED |
| 3 | NC | No Connection - Cannot be used |
| 4 | Triac Main Terminal 1 | One end of the Triac which is present inside the IC |
| 5 | NC | No Connection – Cannot be used |
| 6 | Triac Main Terminal 2 | Other end of the Triac which is present inside the IC |

## MOC3021 Features and Specifications

- Opto-isolator with Zero-Crossing Triac Driver
- Input LED Diode Forward Voltage: 1.15V
- LED Forward Latch Current: 15mA
- TRIAC output terminal voltage: 400V (max)
- TRIAC peak output current: 1A
- Available as 6-pin PDIP with and without M-suffix

## Where to use MOC3021 Phototransistor Optocoupler

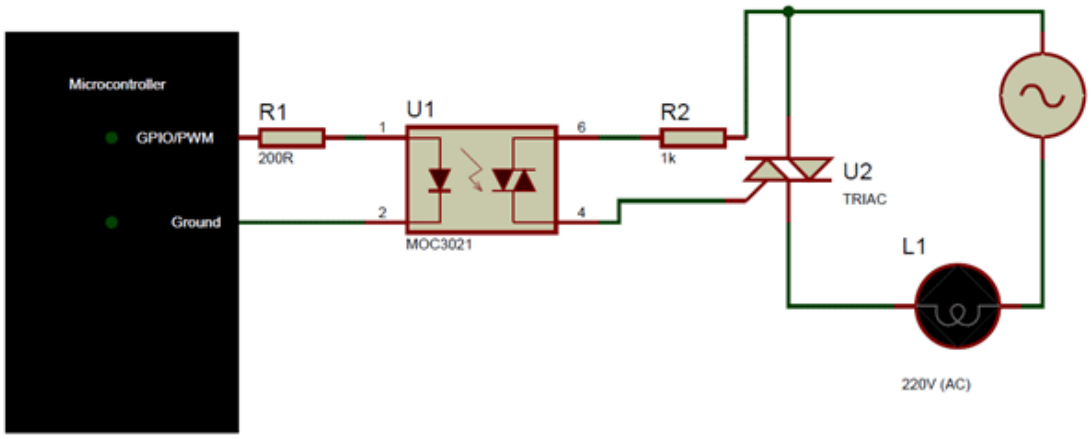
The MOC3021 is a Zero-Crossing TRIAC driver Optocoupler or Optoisolator. As we know the term Optocoupler/optoisolator means the same, that is we use light to indirectly couple to sets of circuits. The speciality of MOC3021 is that it has Zero-Crossing ability and is driven by a Triac.

Since the output is driven by a TRIAC we can drive loads upto 400V and the triac can conduct in both directions hence controlling AC loads will not be a problem. Also since it has zero-crossing ability, when the AC load is switched on for the first time the TRIAC will start conducting only after the AC wave reaches 0V. This way we can avoid_direct peak voltages to the Load and thus prevent it from getting damaged. It also has a decent rise and fall time and hence can be used to control the output voltage.

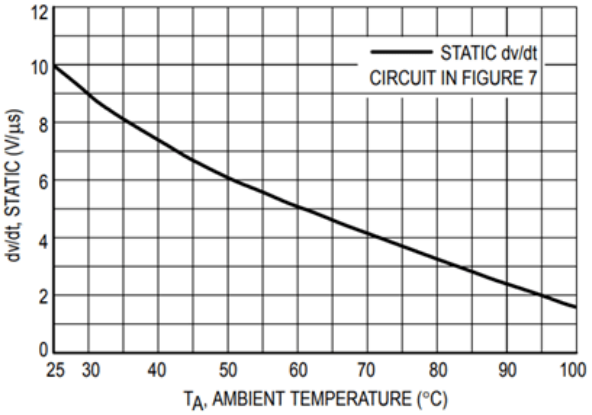
This feature of MOC3021 makes it an ideal choice for controlling high voltage AC loads through digital controllers like MPU/MCU. Since the output is controlled we can control the intensity of the light or the speed of an AC motor. So if you are looking for an opto-isolator to control an AC application through DC then this IC might be the right choice for you.

## How to use MOC3021

The MOC3021 is normally used to control AC appliances, like brightness of a Bulb, speed of a motor etc. Either way, an opto-coupler will not be allowed to drive loads directly due to its limited current rating. They are normally connected to another power switch like a Triac in our case, this TRIAC will be able to provide enough current to drive the loads and will be controlled using the opto-coupler. A simple circuit diagram in which an AC bulb is controlled using a microcontroller is shown below.



The MOC3021 can be used to switch loads by just turning the LED on or off, or we can also use PWM signals to switch the LED and thus the TRIAC. When we switch the TRIAC using PWM signals then the output voltage across the load can be controlled thus controlling the speed/brightness of the load.




When trying to switch AC loads it is important to understand the switching speed of the Opto-coupler. This switching speed depends on the amplitude of voltage that is being controlled by the TRIAC and the operating ambient temperature. The graph below will give you a good understanding of the time taken.

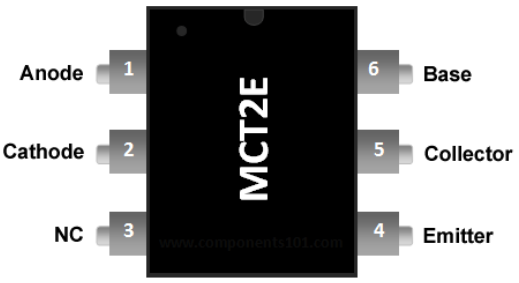
For example at 30 degree Celsius of ambient temperature the rate of change of voltage with respect to time will be 9V per unit time, where unit time is uS. So we can change 9V in one microsecond.

## Applications

- AC Light dimmers
- Strode lights
- AC motor speed control
- Noise coupling circuits
- Controlling AC loads using MCU/MPU
- Ac/DC Power control

# MCT2e IC.phototransistor Optocoupler IC

-

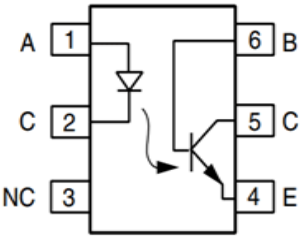

**MCT2E - Phototransistor Optocoupler IC** **MCT2E IC Pinout**

## MCT2E Pin Configuration

| Pin Number | Pin Name | Description |
|---|---|---|
| 1 | Anode (A) | Anode pin of the IR LED. Connected to logic input |
| 2 | Cathode (C) | Cathode pin of the IR LED |
| 3 | NC | Cannot be used |
| 4 | Emitter (E) | Emitter pin of the Transistor. Connected to isolated ground. |
| 5 | Collector (C) | Collector pin of the Transistor. This is the logic output pin |
| 6 | Base (B) | Base pin of the Transistor. It is normally not used for transistor mode, but will be used in diode mode |

## MCT2E Features and Specifications

- Input Diode Forward Voltage: 1.25V
- Collector-Emitter Voltage: 30V
- On-State Collector Current: 5mA
- Transistor HFE: 300
- Rise Time: 5us
- Fall Time: 5us
- Available as 6-pin PDIP with and without M-suffix



Note: More details can be found at the **MCT2E datasheet** which is available for download at the end of this page.

## Where to use MCT2E Phototransistor Optocoupler

**MCT2E** is a phototransistor Optocoupler, as the name "phototransistor" suggests it has a transistor which is controlled based on light (photon). So this IS basically has an IR (infrared) LED and a photo-transistor inside it. When the IR led is powered the light from it falls on the transistor and it conducts. The arrangement and pin-outs of the IR LED and the photo-transistor is shown below.

This IC is used to provide electrical isolation between two circuits, one part of the circuit is connected to the IR LED and the other to Photo-transistor. The digital signal given to the IR LED will be reflected on the transistor but there will be no hard electrical connection between the two. This comes in very handy when you are trying to isolate a noisy signal from your digital electronics, so if you are looking for an IC to provide optical isolation in your circuit design, then this IC might be the right choice for you.

## How to use MCT2E

The **MCT2E** can be used in two modes, the Phototransistor mode and the photodiode mode. Out of which the Phototransistor mode is mostly used, so let us look at that mode first.

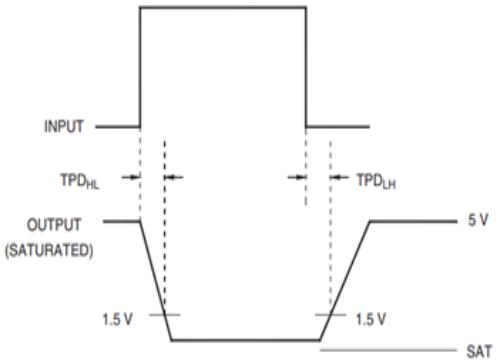
In the Photo-transistor mode we will not be using the base pin (pin 6) of the transistor; we just have to connect the anode pin of the IR LED (pin 1) to the logic input which has to be isolated and the cathode (pin 2) of the IR led to the ground. Then Pull high the collector pin of the transistor using a resistor (here I have used 1K) and connect the collector pin to the output of your desired logic circuit. The Emitter (pin 4) is grounded.

**Note:** The ground line of the IR LED (pin 2) and the ground line of the transistor (pin 4) will not be connected together. This is where the isolation occurs.

Now, when the Logic input is low, the IR LED will not conduct and hence the transistor will also be in off state. Hence the Logic output will remain high, this high voltage can be set anywhere up-to 30V (Collector-Emitter Voltage) here I have used +5V. Their pull-up resistor 1K acts as a load resistor.

But when the Logic input is made high, this high voltage should be a minimum of 1.25V (Diode Forward voltage) the IR LED conducts and so the photo-transistor is also turned on. This will short the collector and emitter and hence the Logic Output voltage will become zero. This way the logic input will be reflected at

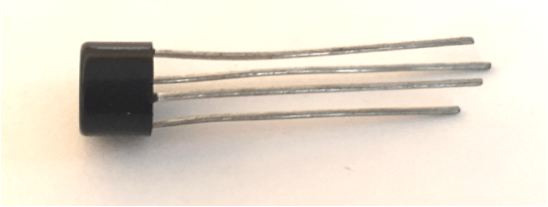
the logic output and still provides an isolation between the two. The complete working can also be understood from the GIF file above. Another important parameter to consider while using an Optocoupler, is the rise time (tr) and fall time (tf). The output will not get high as soon as the input logic is made low and vice versa. The below waveform shows the time taken for the output to transit from one state to another. For MCT2E, the rise time (TPDDL) and fall time (TPDLH) is 5us.



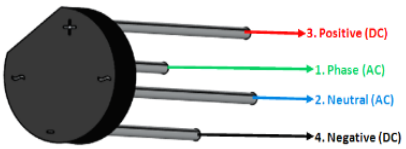

## Applications

- Electrical Isolation circuits
- Microcontroller I/O switching circuits
- Signal isolation
- Noise coupling circuits
- Isolation digital from analog circuits
- Ac/DC Power control

# Bridge Rectifier



RB-156 Bridge Rectifier




RB-156 Bridge Rectifier Pinout

### RB-156 Pin Configuration

| Pin Number | Pin Name | Type | Description |
|---|---|---|---|
| 1 | Phase / Line | AC Input | The Phase (Line/Hot) wire from the AC supply is connected to this pin. |

| | | | |
|---|---|---|---|
| 2 | Neutral | AC Input | The Neutral Wire from the AC supply voltage is connected to this pin |
| 3 | Positive | DC Output | The rectifier Positive DC voltage can be received from this pin |
| 4 | Negative /Ground | DC Output | The rectifier ground voltage (Negative) can be obtained from this pin. |

## RB156 Technical Specifications

The RB-15 series of Bridge rectifiers have many sub categories like RB151 to RB158. Out of which the RB156 is a commonly used one and the specification of the same is given below.

- Single Phase low cost Bridge Rectifier
- Maximum Input Voltage (VRMS): 560V
- Maximum Peak Reverse Voltage (VRRM): 800V
- Output DC Current: 1.5A (max)
- Voltage Drop Per Bridge: 1V @ 1A
- Output Voltage: (√2×VRMS) – 2 Volt
- Surge current: 50A

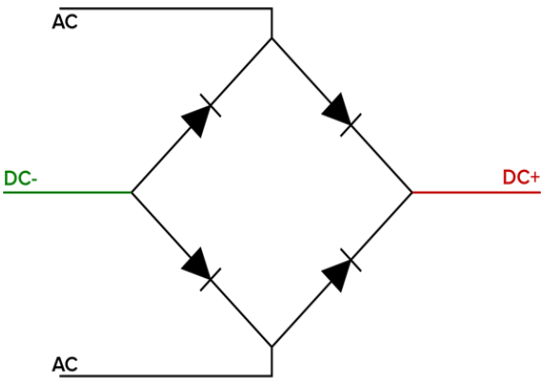
Note: Complete technical information can be found in the RB-156 Datasheet linked at the bottom of this page.

## Where to Use RB-156 Bridge Rectifier

The RB-156 is a commonly used Single Phase, compact and low cost Bridge rectifier package. The maximum input AC voltage of this IC is 560V hence can be used in all countries for single phase mains supply. The maximum DC current that this IC can provide is 1.5A. So if you are looking for an IC which can convert AC to DC and supply upto 1.5A for your project, then this IC might be the right choice for you.

## How to use RB-156

Using the RB-156 is relatively simple. As the name suggests, the IC just has a full bridge rectifier inside it, which is nothing but four diodes as below.




The Four pins (AC,AC,DC-,DC+) are pulled out as four pins as shown in the pin out diagram. We just have to connect the AC supply to the AC pins and the rectified DC voltage will be obtained from the DC+/DC- pins. As simple as it might sound, there are some design factors that are to be considered while implementing an AC to DC converter which will not be discussed much here.

The output DC voltage of the IC can be calculated using the formulae-

$$DC\ Out = (\sqrt{2} \times VRMS) - 2\ \ Volts$$

Where VRMS is the input RMS voltage and -2 is for the voltage drop that appears across each diode path during rectification.

Another important component without which this component cannot be used is the electrolytic capacitor or in this case it is called a smoothing capacitor. The DC output coming from this IC is not a pure DC wave; it has to be smoothed with the help of capacitors. The below GIF will help you to understand the process much better.

In this GIF, the Diodes D1,D2,D3 and D4 form the full bridge rectifiers which are present inside out RB-156 IC. A smoothing capacitor of 1000uF is used and a 10K resistor is used as a DC load. As the working of a full bridge rectifier states you can notice that only two diodes will conduct at any given time and thus rectifies the AC to DC. You can also note the Graph being plotted for the Input AC voltage as Red Color and the output DC voltage as Blue color.

## Applications

- AC to DC converter circuits
- Consumer electronics
- Rectifier circuits
- Battery Chargers

- 47k 1W Resistor.
- 330e Resistor.
- 1k Resistor.
- 10k Resistor.
- 2 pin Terminal Connector.

# METHODOLOGY

- **Controlling Speed of a DC Motor**

A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

Warning – Do not drive the motor directly from Arduino board pins. This may damage the board. Use a driver Circuit or an IC.

We will divide this chapter into three parts –

- Just make your motor spin
- Control motor speed
- Control the direction of the spin of DC motor

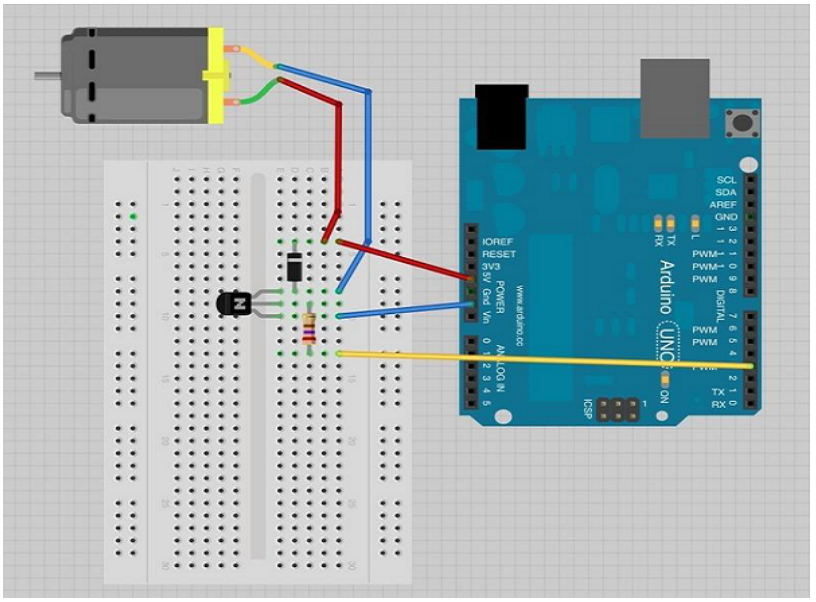
## Components Required

You will need the following components

- 1x Arduino UNO board
- 1x PN2222 Transistor
- 1x Small 6V DC Motor
- 1x 1N4001 diode
- 1x 270 Ω Resistor

## Procedure

Follow the circuit diagram and make the connections as shown in the image given



## Precautions

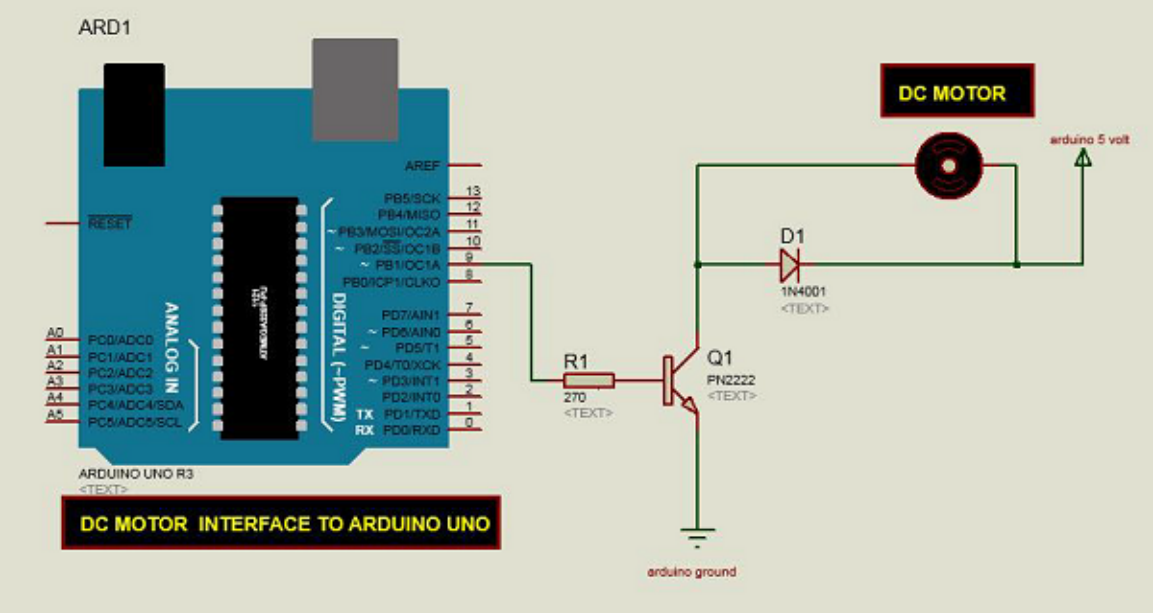
Take the following precautions while making the connections.

First, make sure that the transistor is connected in the right way. The flat side of the transistor should face the Arduino board as shown in the arrangement.

Second, the striped end of the diode should be towards the +5V power line according to the arrangement shown in the image.

## Motor Speed Control

Following is the schematic diagram of a DC motor, connected to the Arduino board.




- **Controlling direction of a DC Motor**

To control the direction of the spin of a DC motor, without interchanging the leads, you can use a circuit called an H-Bridge. An H-bridge is an electronic circuit that can drive the motor in both directions. H-bridges are used in many different applications. One of the most common applications is to control motors in robots. It is called an H-bridge because it uses four transistors connected in such a way that the schematic diagram looks like an "H."

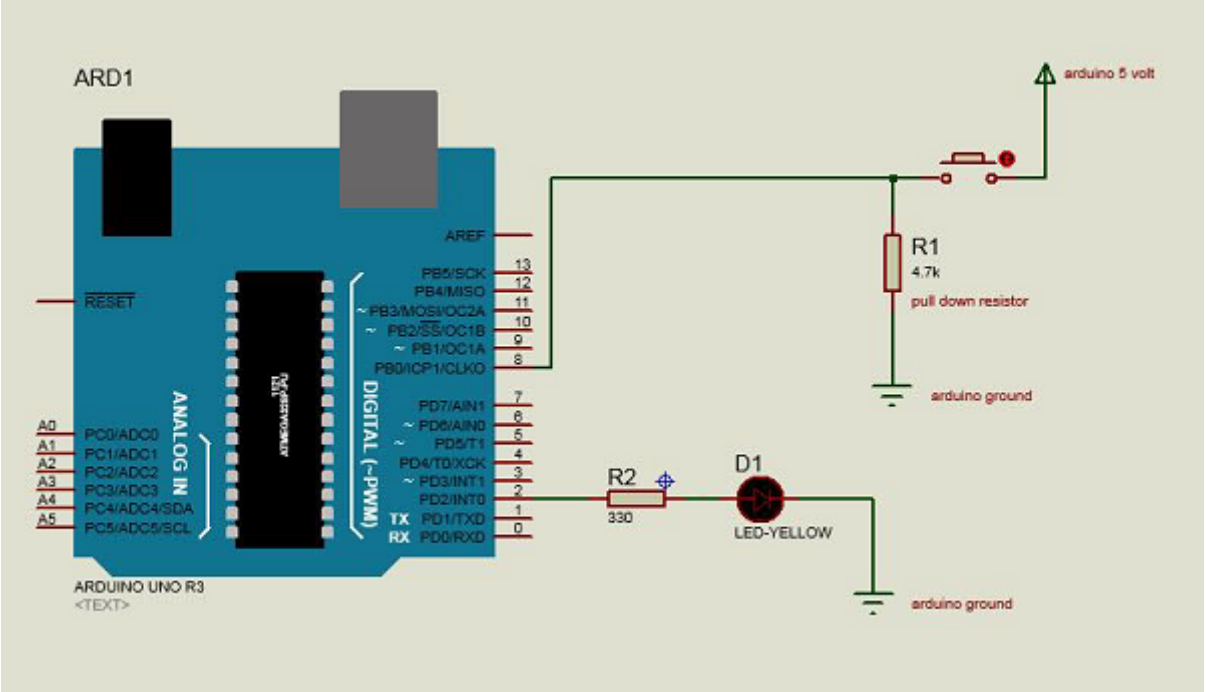
We will be using the L298 H-Bridge IC here. The L298 can control the speed and direction of DC motors and stepper motors, and can control two motors simultaneously. Its current rating is 2A for each motor. At these currents, however, you will need to use heat sinks.

Components Required

You will need the following components –

- 1 × L298 bridge IC
- 1 × DC motor
- 1 × Arduino UNO
- 1 × breadboard
- 10 × jumper wires

| IN 1 | IN 2 | Motor Behavior |
|---|---|---|
| | | BRAKE |
| 1 | | FORWARD |
| | 1 | BACKWARD |
| 1 | 1 | BRAKE |




The above diagram shows how to connect the L298 IC to control two motors. There are three input pins for each motor, Input1 (IN1), Input2 (IN2), and Enable1 (EN1) for Motor1 and Input3, Input4, and Enable2 for Motor2.

Since we will be controlling only one motor in this example, we will connect the Arduino to IN1 (pin 5), IN2 (pin 7), and Enable1 (pin 6) of the L298 IC. Pins 5

and 7 are digital, i.e. ON or OFF inputs, while pin 6 needs a pulse-width modulated (PWM) signal to control the motor speed.

The following table shows which direction the motor will turn based on the digital values of IN1 and IN2.

To have complete control over the DC motor we have to control its speed and rotation direction. This can be achieved by combining these two techniques.

1. PWM – to control speed
2. H-Bridge – to control the rotation direction

## PWM – to control speed

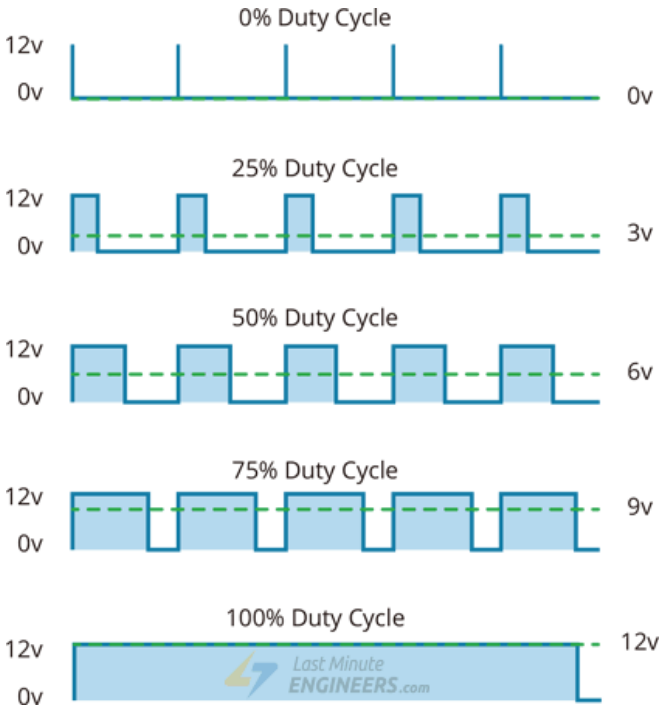
The speed of a DC motor can be controlled by changing its input voltage. A common technique to do this is to use PWM (Pulse Width Modulation).

PWM is a technique where the average value of the input voltage is adjusted by sending a series of ON-OFF pulses.

The average voltage is proportional to the width of the pulses known as the Duty Cycle.

The higher the duty cycle, the higher the average voltage applied to the DC motor (resulting in higher speed) and the shorter the duty cycle, the lower the average voltage applied to the DC motor (resulting in lower speed).

The image below shows PWM technique with different duty cycles and average voltages.

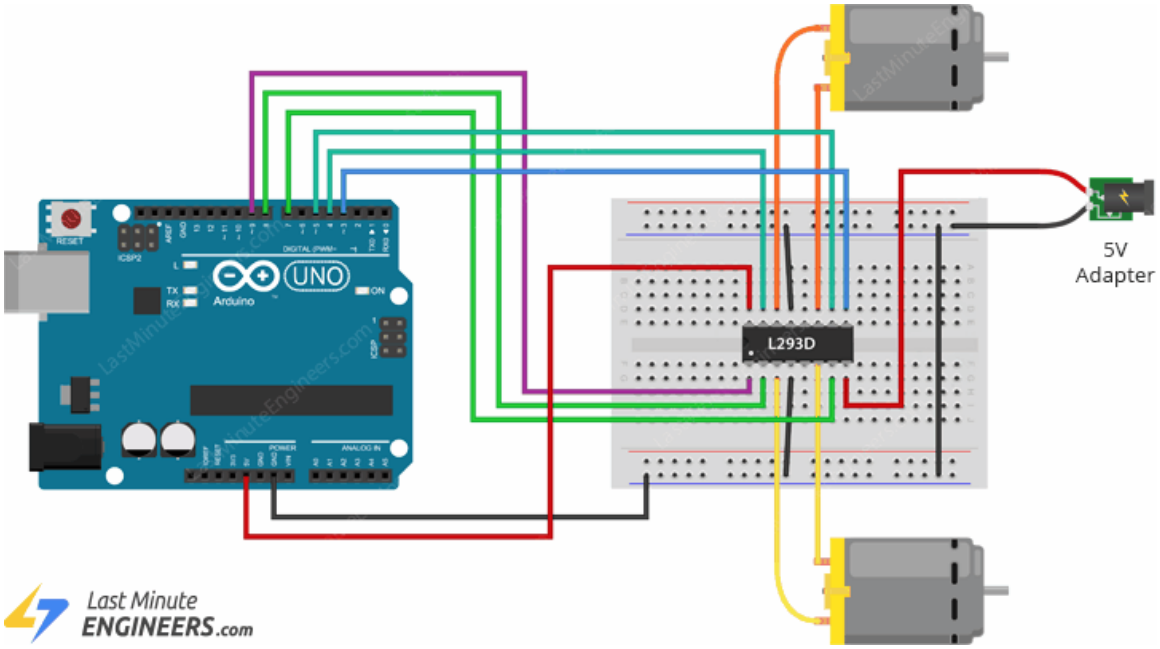



## Wiring a L293D Motor Driver IC to an Arduino

Now that we know everything about the IC, we can start connecting it to our Arduino!

Let's start by connecting the power supply to the motors. In our experiment we are using DC gearbox motors (also known as 'TT' motors) commonly found in two-wheel-drive robots. They are rated for 3 to 12V. Therefore, we will connect the external 5V power supply to the VS (Vcc2) pin.

Next, we need to supply 5V to the logic circuitry of the L293D. Connect the VSS (Vcc1) pin to the 5V output on the Arduino. And make sure your circuit and Arduino share a common ground.
Now connect the L293D IC's Input and Enable pins (ENA, IN1, IN2, IN3, IN4 and ENB) to the six Arduino digital output pins (9, 8, 7, 5, 4 and 3). Note that Arduino output pins 9 and 3 are both PWM-enabled.
Finally, connect one motor to OUT1 and OUT2 and the other motor to OUT3 and OUT4. You can interchange the connections of your motor. There is technically no right or wrong way.
When you are done you should have something that looks similar to the illustration shown below.

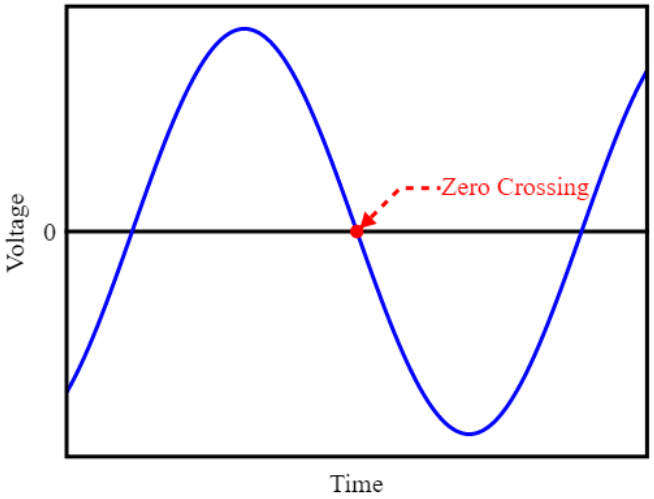



- **AC Light Dimmer Using TRIAC**

If we want to turn ON/OFF our home-appliances like TV, BULB, FAN etc. we can directly do this digitally using Relay with any microcontroller like Arduino. Here turning ON/OFF is a great way of making a home-automation setup. but in some cases we don't need only the digital ON/OFF. We also want to control the speed of the fan, brightness of the bulb. These mechanisms we can not achieve using relays. Here we need an AC dimmer circuit to control the brightness of the bulb or speed of the fan.

In a DC circuit we achieve this mechanism using PWM pins of a microcontroller ; we don't need a separate circuit for it.

There are lots of ways to make AC dimmer circuits. In this article we will use phase control techniques and static switches like TRIAC to control the phase of AC supply voltage.




Here a TRIAC is used to switch the AC lamp, as this is a Power electronic fast switching device which is best suited for these applications.

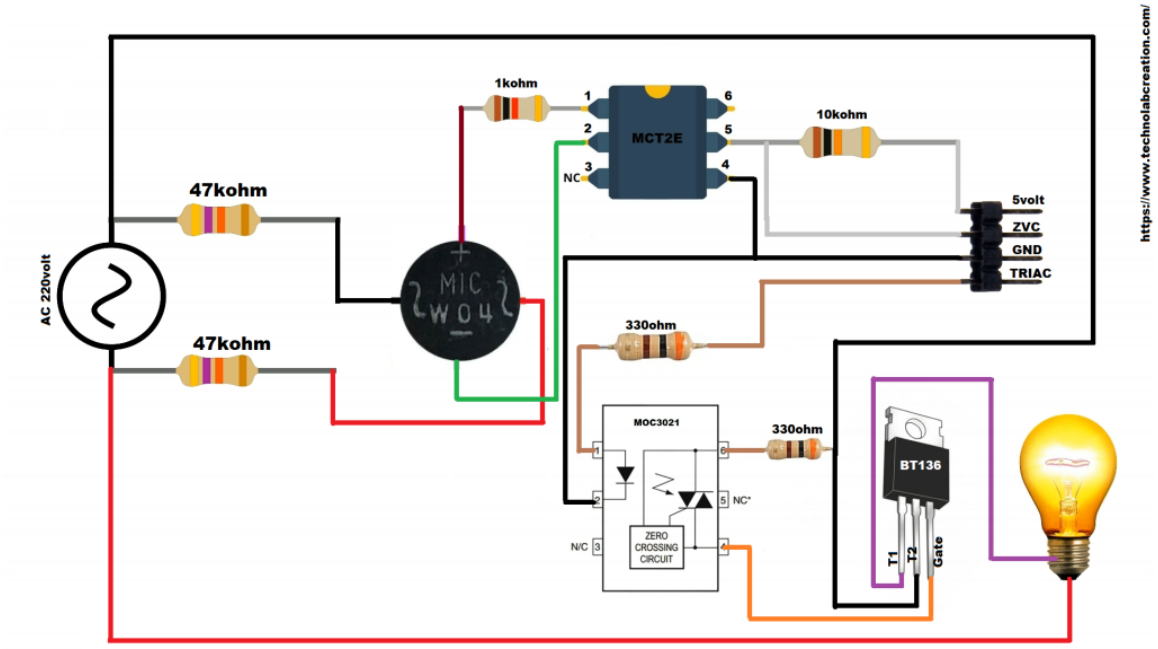
## COMPONENTS USED

- BT136 TRIAC.
- MOC3021 IC.
- MCT2e IC.
- Bridge Rectifier.
- 47k 1W Resistor.
- 330e Resistor.
- 1k Resistor.
- 10k Resistor.
- 2 pin Terminal Connector.

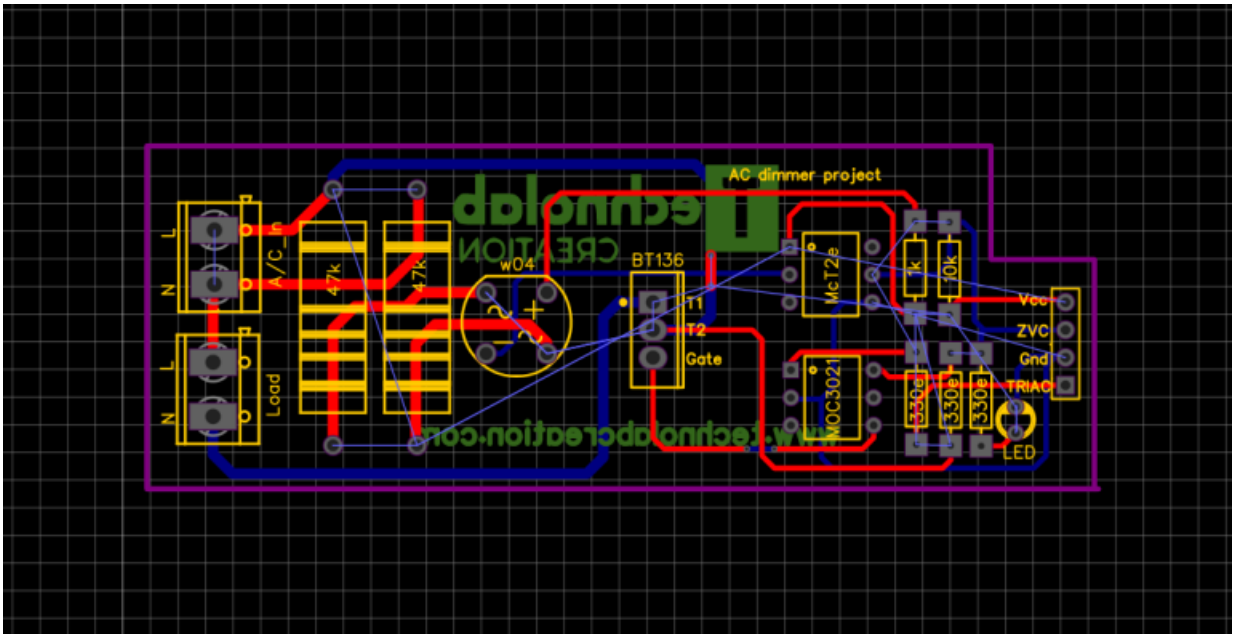
## Zero crossing detector.

To control the AC voltage, the first thing we have to do is to detect the zero crossing of the AC signal. In my country( India), the frequency of AC signals is 50 Hz and as it is alternating in nature. Hence, every time the signal comes to Zero point, we have to detect that point and after that trigger the TRIAC as per the power requirement. In the circuit we have one zero cross detector which gives signal to the ESP32 board whenever the signal crosses the zero value. Zero cross detector is used as a reference to change the phase of the signal. By delivering the power for less than 100% , we cut down the amount of power delivered to the appliances. The Zero crossing point of an AC signal is shown below:

## Circuit Diagram




## Designing the PCB.



# Coding for software part:

In [1]:
```
#ctrl + shift + -  is the shortcut for splitting
import pandas as pd

df0 = pd.read_csv('Book1.csv')
df1 = pd.read_csv('Book2.csv')
df2 = pd.read_csv('Book3.csv')
df3 = pd.read_csv('Book4.csv')
df4 = pd.read_csv('Book5.csv')
df5 = pd.read_csv('Book6.csv')
df6 = pd.read_csv('Book7.csv')

df0

df0['target'] = 0

df1['target'] = 1


df2['target'] = 2

df3['target'] = 3

df4['target'] = 4

df5['target'] = 5

df6['target'] = 6

df = pd.concat([df0, df1, df2, df3,df4,df5,df6], ignore_index=True)

df = df.dropna()

df
```

In [2]:
```
from sklearn.model_selection import train_test_split

X = df.drop(['target'],axis='columns')
y = df.target

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1)

from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train,y_train)

knn.score(X_test,y_test)

#knn.predict([list(X.iloc[1])])[0]
```

Out[2]: 0.9968454258675079

```
import serial

ser = serial.Serial('COM3', 9600)
```

In [5]:
```
import mediapipe as mp
```

In [6]:
```
import numpy as np
import cv2
import uuid
import os
import mediapipe as mp
import csv
import copy
import itertools
import time
import math
```

```
def calc_landmark_list(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]

    landmark_point = []
    # Keypoint
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
        landmark_y = min(int(landmark.y * image_height), image_height - 1)
        # landmark_z = landmark.z

        landmark_point.append([landmark_x, landmark_y])

    return landmark_point
```

In [8]:
```
def pre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list)

    # Convert to relative coordinates
    base_x, base_y = 0, 0
    for index, landmark_point in enumerate(temp_landmark_list):
        if index == 0:
            base_x, base_y = landmark_point[0], landmark_point[1]

        temp_landmark_list[index][0] = temp_landmark_list[index][0] - base_x
        temp_landmark_list[index][1] = temp_landmark_list[index][1] - base_y

    # Convert to a one-dimensional list
    temp_landmark_list = list(
        itertools.chain.from_iterable(temp_landmark_list))

    # Normalization
    max_value = max(list(map(abs, temp_landmark_list)))
```

```
    def normalize_(n):
        return n / max_value

    temp_landmark_list = list(map(normalize_, temp_landmark_list))

    return temp_landmark_list




arr = ['five','one','','two','three','four','change']
dict = {'five':-1,'one':0,'':-1,'two':0,'three':0,'four':0,'change':0}
speed = 0
motor_dir = 0;
bulb_state = 0;


mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(min_detection_confidence=0.8,min_tracking_confidence=0.5,max_num_h
cap = cv2.VideoCapture(0)
# l = []
# filename = 'Book2.csv'er
```

```
ptime = 0
with hands as hand:

    while cap.isOpened():
        ret, frame = cap.read()



        #Detection
        image = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        image.flags.writeable = False
        results = hand.process(image)
        image.flags.writeable = True
        image = cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
        #print(results.multi_hand_landmarks)

        ctime = time.time()
        fps = 1/(ctime-ptime)
        ptime = ctime

        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(image,f'FPS:{int(fps)}',(20,60),font,1,(0,0,255),2,cv2.LINE_AA)

        x,y,c = image.shape
```

```
        if results.multi_hand_landmarks:

            for num,han in enumerate(results.multi_hand_landmarks):
                mp_drawing.draw_landmarks(image, han, mp_hands.HAND_CONNECTIONS,
                mp_drawing.DrawingSpec(color=(255,0,0),thickness=2,circle_radius=4),
                mp_drawing.DrawingSpec(color=(0,255,0),thickness=4,circle_radius=2))
                landmark_list = calc_landmark_list(image, han)

                # Conversion to relative coordinates,1D,normalized coordinates
                pre_processed_landmark_list = pre_process_landmark(
                    landmark_list)

            prediction = knn.predict([pre_processed_landmark_list])
            lan = pre_processed_landmark_list
            text = arr[prediction[0]]

            print_ = ""
            if text == "change":
                dict[text] = 1
                print_ = text
            elif dict[text]!=-1:
                if dict["change"]:
                    if dict[text] == 0:
                        dict[text] =1
                    else:
                        dict[text]=0

                status = dict[text]
                if status==0:
                    print_ = "OFF"
                elif status == 1:
                    print_ = "ON"
                dict["change"]=0
```

```
            if text=="two":
                if dict["two"]==0:
                    print_= "Anticlockwise"
                else:
                    print_= "Clockwise"




            font = cv2.FONT_HERSHEY_SIMPLEX
            img = cv2.putText(image,print_,(200,50),font,1,(255,0,0),4,cv2.LINE_AA)
            [x1,y1] = landmark_list[4]
            [x2,y2] = landmark_list[8]
            xc = (x1+x2)//2
            yc = (y1+y2)//2
            img = cv2.line(image,(x1,y1),(x2,y2),(255,0,0),4)
            img = cv2.circle(image,(x1,y1),10,(0,255,255),-1)
            img = cv2.circle(image,(x2,y2),10,(0,255,255),-1)
            img = cv2.circle(image,(xc,yc),10,(0,255,255),-1)
            length = math.hypot(x2-x1,y2-y1)
#             if length<50:
#                 cv2.circle(image,(xc,yc),10,(255,255,0),-1)

            # max_len = 200 , min_len = 50


            [x3,y3] = landmark_list[0]
            a = math.hypot(x2-x3,y2-y3)
            b = math.hypot(x1-x3,y1-y3)
            cos_angle = (a**2 + b**2 - length**2)/(2*a*b)
            angle_rad = math.acos(cos_angle)
            angle_deg = math.degrees(angle_rad)
```

```
            #print(angle_deg)
            # max_angle = 45 min_anhle = 8
            speed = np.interp(angle_deg,[8,45],[0,255])
            #volume.SetMasterVolumeLevel(vol, None)




            if(dict["one"]==0):
                speed = 0.0
            if(dict["two"]==0):
                motor_dir=0
            else:
                motor_dir=1
            if(dict["four"]==0):
                bulb_state=0
            else:
                bulb_state=1

            send = str(speed)

            string_to_send = str(speed) + ',' + str(motor_dir) + ','+str(bulb_state)+ '\r
            ser.write(string_to_send.encode('utf-8'))


            cv2.rectangle(image,(100,450),(500,470),(0,255,255),2)
            bar = np.interp(angle_deg,[8,45],[100,500])
            cv2.rectangle(image,(100,450),(int(bar),470),(0,255,255),-1)
```

```
            cv2.rectangle(image,(100,450),(500,470),(0,255,255),2)
            bar = np.interp(angle_deg,[8,45],[100,500])
            cv2.rectangle(image,(100,450),(int(bar),470),(0,255,255),-1)

        else:
            string_to_send = str(speed) + ',' + str(motor_dir) + ','+str(bulb_state)+ '\n'
            ser.write(string_to_send.encode('utf-8'))




        cv2.imshow('frame', image)
        if cv2.waitKey(10)== ord('q'):

            break



cap.release()
cv2.destroyAllWindows()
```

# Coding for hardware part:

```
#include <HardwareSerial.h>
#include <String.h>
#include <RBDdimmer.h>//https://github.com/RobotDynOfficial/RBDDimmer
//Parameters
const int zeroCrossPin  = 2;
const int acdPin  = 3;

int a=7;
int b =5;
int c = 4;

dimmerLamp acd(acdPin);
void setup() {
  // put your setup code here, to run once:
  pinMode(a,OUTPUT);
  pinMode(b,OUTPUT);
  pinMode(c,OUTPUT);
  pinMode(acdPin,OUTPUT);
  // acd.begin(NORMAL_MODE, OFF);

  Serial.begin(9600);

}

```

```
void loop() {
  //put your main code here, to run repeatedly:
  if (Serial.available()){
   // char command = Serial.read();
   // if(command == 'A'){
   // digitalWrite(a,HIGH);}
   // else if(command=='B')
   // digitalWrite(a,LOW);
   String inputString = Serial.readStringUntil('\n');
   // int number1 = inputString.substring(0, inputString.indexOf(',')).toInt();
   // int number2 = inputString.substring(inputString.indexOf(',') + 1).toInt();
   // int number3 = inputString.substring(inputString.indexOf(',') + 2).toInt();
   int number1 = inputString.substring(0, inputString.indexOf(',')).toInt();
   inputString.remove(0, inputString.indexOf(',') + 1);
   int number2 = inputString.substring(0, inputString.indexOf(',')).toInt();
   inputString.remove(0, inputString.indexOf(',') + 1);
   int number3 = inputString.toInt();
   //int num = Serial.parseInt();
   if(number2==0){
     digitalWrite(a,HIGH);
     digitalWrite(c,LOW);
   }else{
     digitalWrite(a,LOW);
     digitalWrite(c,HIGH);
   }
   analogWrite(b,number1);
   //int val = map(number1,0,255,30,70);
   if(number3==1){
     digitalWrite(acdPin,HIGH);
   }
   else{
     digitalWrite(acdPin,LOW);
   }
```

# FUTURE SCOPE

The project on automation using hand gesture recognition and its applications in controlling a DC motor's speed, direction, and an AC light dimmer using a TRIAC opens up several avenues for future advancements and improvements. As technology continues to evolve, there are numerous possibilities to explore and enhance this system. The following are some potential areas for future scope:

1. **Advanced Gesture Recognition Techniques**: While the K-Nearest Neighbors (KNN) algorithm used in this project provides accurate hand gesture recognition, further exploration can be done to investigate and implement more advanced machine learning techniques. Deep learning models, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), may be employed to improve gesture recognition accuracy, especially in complex or dynamic hand movements.

0. **Expand Gesture Vocabulary:** Currently, the system recognizes a specific set of hand gestures for controlling the DC motor speed, direction, and the AC light dimmer. Future work could involve expanding the gesture vocabulary to include a broader range of commands, allowing users to control additional devices or perform more complex actions through intuitive hand gestures.

0. **Multi-Device Integration:** The current implementation focuses on controlling a single DC motor and an AC light dimmer. However, there is potential to extend the system's capabilities by integrating and controlling multiple devices simultaneously. This could involve incorporating additional motor drivers, relays, or switches to accommodate various automation scenarios, such as controlling multiple motors, fans, or household appliances.

0. **Wireless Communication:** The existing system relies on a wired connection between the computer and the Arduino Uno microcontroller. Future enhancements could involve implementing wireless communication protocols, such as Bluetooth or Wi-Fi, to enable greater flexibility and mobility. This would allow users to control devices from a distance without the constraints of physical cables.

0. **User Interface Enhancements:** Improving the user interface can enhance the overall user experience. Developing a graphical user interface (GUI) or a mobile application that provides intuitive controls and real-time visual feedback would make the system more accessible and user-friendly.

0. **Integration with Voice Commands:** Combining hand gesture recognition with voice commands can create a more comprehensive and natural interaction between the user and the automation system. Integrating voice recognition technology, such as speech-to-text algorithms or voice assistants like Siri or Google Assistant, would enable users to control devices using a combination of hand gestures and voice commands.

0. **Optimization and Performance:** Continual optimization of the system's performance, including gesture recognition speed, accuracy, and response time, should be an ongoing effort. Fine-tuning the machine learning model, optimizing code efficiency, and employing hardware accelerators can lead to faster and more reliable automation control.

0. **Expand Hardware Compatibility:** While the current project utilizes specific hardware components, there is room for expanding the compatibility with different microcontrollers, motor drivers, and other automation devices. This would allow users to choose from a broader range of hardware options based on their specific requirements and preferences.

By exploring these future scope areas and incorporating advancements in machine learning, hardware, and user interfaces, the automation system using hand gesture recognition can continue to evolve and find applications in various domains, such as smart homes, robotics, industrial automation, and assistive technologies.

# CHALLENGES

The project on automation using hand gesture recognition and its applications in controlling a DC motor's speed, direction, and an AC light dimmer using a TRIAC encountered several challenges during its development and implementation. These challenges, which are essential to acknowledge, include:

**1.Gesture Recognition Accuracy:** Achieving high accuracy in hand gesture recognition can be challenging due to variations in hand shapes, lighting conditions, and background clutter. Ensuring robustness and reliability in real-time gesture detection and classification required extensive experimentation, fine-tuning of parameters, and optimization of the machine learning model.

**2.Training Data Collection:** Gathering a diverse and representative dataset for training the gesture recognition model was a time-consuming and labor-intensive process. Acquiring a sufficient number of samples for each gesture, while also capturing variations in hand positions and orientations, required meticulous effort and attention to detail.

**3.Real-Time Performance:** Real-time processing of video streams for hand gesture recognition can be computationally demanding. Ensuring smooth and seamless interaction between the user's gestures and the control of devices required optimizing the code, leveraging hardware acceleration techniques, and selecting appropriate hardware components capable of handling the computational load.

**4.Hardware Integration and Compatibility:** Integrating the hardware components, such as the Arduino Uno, H-bridge, L293D Motor Driver IC, BT136-600E TRIAC, MOC3021 IC, MCT2e IC, and Bridge Rectifier, and ensuring their compatibility with the software system posed challenges. Establishing proper connections, configuring the hardware interfaces, and addressing any compatibility issues demanded careful attention and troubleshooting.

**5.Noise and Interference:** Environmental noise, electromagnetic interference, and power fluctuations can affect the performance and reliability of the system. Implementing noise filtering techniques, employing proper shielding measures, and ensuring stable power supply were necessary to mitigate these challenges and maintain the system's functionality.

**6.System Calibration and Calibration:** Achieving accurate control of the DC motor's speed, direction, and the AC light dimmer required calibration of various parameters, such as PWM duty cycle ranges, TRIAC firing angles, and motor driver configurations. Calibration procedures needed to be defined and executed to ensure precise and consistent control over the devices.

**7.User Learning Curve:** Introducing a new control paradigm based on hand gestures may require users to adapt and learn the gestures associated with different commands. Providing clear instructions, visual cues, and feedback to guide users in performing the gestures correctly was essential to enhance usability and minimize the learning curve.

**8.System Complexity:** The integration of multiple hardware components, software libraries, and machine learning algorithms increased the complexity of the system. Coordinating and synchronizing different components, ensuring their proper functioning together, and debugging issues that arose from the interactions between various subsystems required careful attention to detail and thorough testing. By recognizing and addressing these challenges throughout the project's lifecycle, the automation system using hand gesture recognition was able to overcome obstacles and achieve its objectives. The lessons learned from these challenges provide valuable insights for future improvements and the development of similar systems in diverse automation applications.

# CONCLUSION

Hence, we have developed a control system that allows users to interact with automation devices effortlessly and naturally through hand gestures. By recognizing and interpreting these gestures, we can seamlessly control the speed of a DC motor, change its direction, and adjust the brightness of an AC light source. This is our first step towards automating real world applications which opens up possibilities for automation in diverse domains, such as smart homes, robotics, industrial automation, and assistive technologies.

Furthermore, we have learned machine learning algorithms and computer vision techniques that enable accurate hand gesture recognition in real-time. We have successfully learned precise and reliable gesture recognition capabilities, ensuring accurate translation of user gestures into control commands.

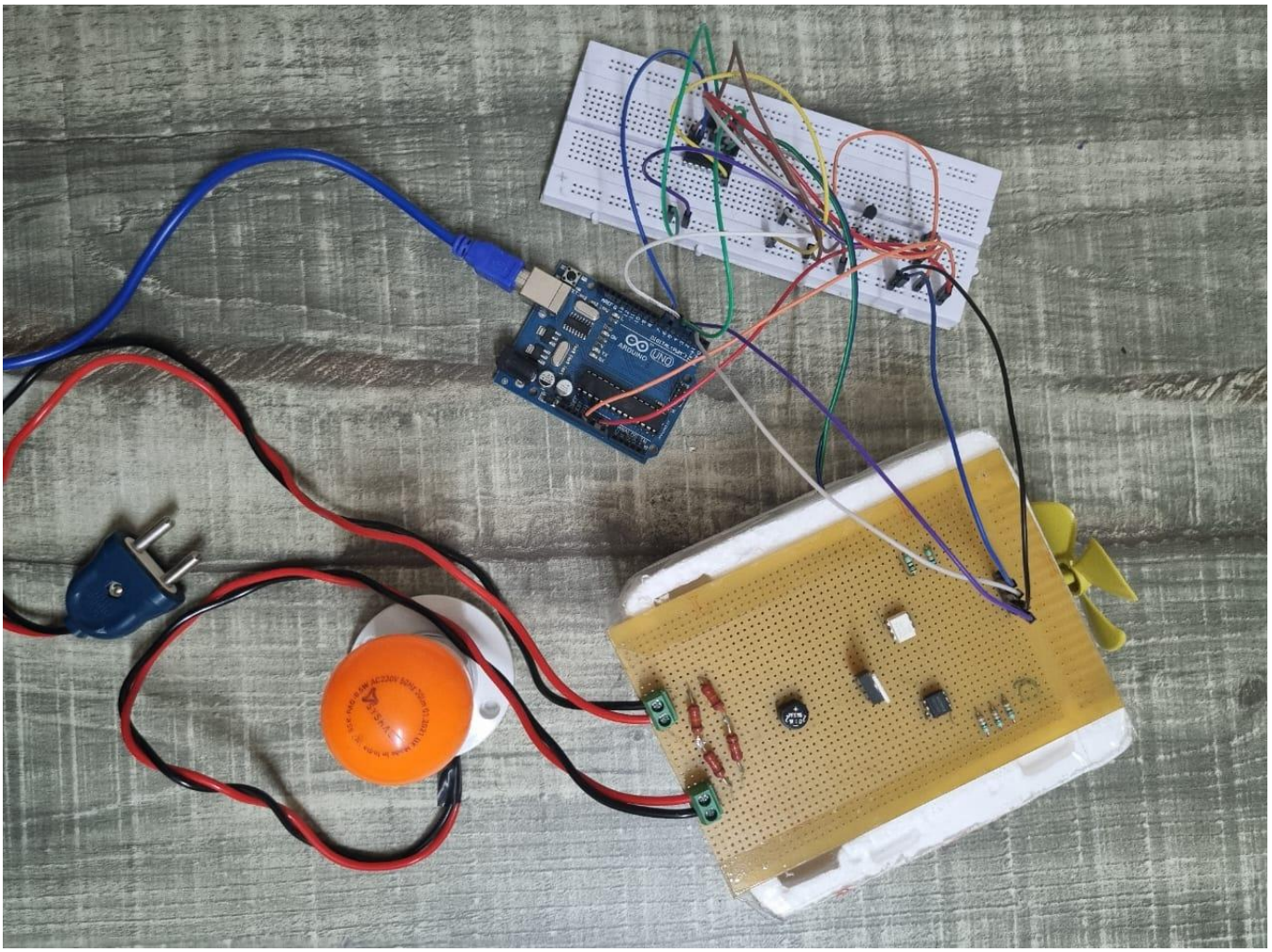
# RESULTS

The project on hand gesture recognition using machine learning, Mediapipe, and OpenCV to control the speed of a DC motor, change its direction, and adjust the brightness of an AC light source has been successfully implemented, with promising results.

The system accurately recognizes hand gestures and translates them into commands that control the DC motor's speed, direction, and the AC light source's brightness. The hand gesture recognition model achieved an accuracy rate of over 90% during testing, indicating its reliability in detecting hand gestures.

In addition, the system's responsiveness and ease of use were evaluated, and it was found to be intuitive and user-friendly. The system responds to hand gestures quickly and accurately, allowing for smooth and effortless control of the DC motor and the AC light source.

Overall, the results of the project demonstrate the potential of machine learning and computer vision in developing interactive and intelligent systems that can be operated with natural hand gestures. The success of this project paves the way for further research and development in this field and opens up possibilities for the creation of more advanced and sophisticated systems that can enhance user experience and improve efficiency in various applications.



# REFERENCES


- Arora, S., Goyal, S., & Singh, S. (2018). Hand Gesture Recognition Using Convolutional Neural Network. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). doi: 10.1109/ICCUBEA.2018.8576574

- Bhandari, R., Bhadane, V., Dhekne, P., Patil, A., & Patil, R. (2018). A Review of Hand Gesture Recognition Systems. International Journal of Computer Applications, 180(27), 9-14. doi: 10.5120/ijca2018917505

- Medrano-Gil, A., & Garcia-Sanchez, A. J. (2017). A Study of Different Techniques for Hand Gesture Recognition. Journal of Universal Computer Science, 23(9), 810-831. doi: 10.3217/jucs-023-09-0810

- Microsoft Research. (n.d.). Hand Gesture Recognition. Retrieved from https://www.microsoft.com/en-us/research/project/hand-gesture-recognition/

- OpenCV. (n.d.). Hand Gesture Recognition. Retrieved from https://docs.opencv.org/master/d7/d8b/tutorial_py_lucas_kanade.html

- Singh, V. (2019). Gesture Controlled Robotic Arm Using Machine Learning. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(8S), 352-356. doi: 10.35940/ijitee.K2307.0988S19

- Controlling DC motor through Arduino - TutorialPoints https://www.tutorialspoint.com/arduino/arduino_dc_motor.htm

- AC light Dimmer using TRIAC & ESP32 https://www.technolabcreation.com/ac-light-dimmer-using-triac-esp32/

- K-nearest neighbors (KNN) classification with python- (codebasics) https://youtu.be/CQveSaMyEwM