

AI GESTURE BASED AUTOMATION WITH MESH DETECTION THROUGH OPENCV ON PYTHON AND PYFIRMATA PROTOCOL

Submitted in partial fulfilment of the requirements for the award of Bachelor of
Engineering Degree in Electronics and Communication Engineering

by

ROY CARLOUS C (38130188)

VASANTH MATHEW B (38130232)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119

APRIL 2022



SATHYABAMA
INSTITUTE OF SCIENCE AND
TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600 119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ROY CARLOUS C (38130188)** and **VASANTH MATHEW B (38130232)** who carried out the project entitled **"AI GESTURE BASED AUTOMATION WITH MESH DETECTION THROUGH OPENCV ON PYTHON AND PYFIRMATA PROTOCOL"** under our supervision from **September 2021 to April 2022**.

Internal Guide

Dr. T. RAVI, M.E., P.hD.

Head of the Department, ECE

Head of the Department

Dr. T. RAVI, M.E., P.hD.

Submitted for Viva Voce held on _____

Internal Examiner

External Examiner

DECLARATION

We, **ROY CARLOUS C (38130188)** and **VASANTH MATHEW B (38130232)** hereby declare that the Project Report entitled **“AI GESTURE BASED AUTOMATION WITH MESH DETECTION THROUGH OPENCV ON PYTHON AND PYFIRMATA PROTOCOL”** done by us under the guidance of **Dr. T. RAVI, M.E., Ph.D.** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering Degree in Electronics and Communication engineering.

DATE:

PLACE:

1)

2)

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D. Dean, School of Electrical and Electronics Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr. T. RAVI, M.E., Ph.D. Head of Department, ECE** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of the project work.

We wish to express our thanks to all teaching and non-teaching staff members of the **Department of Electronics and Communication Engineering** who were helpful in many ways for the completion of the project.

We express our gratitude to our parents for their constant encouragement and support for the completion of the project.

ABSTRACT

Automation is an evolving field providing immense aid in various fields. Types of automation include Artificial Intelligence, home automation, industrial automation, etc. Smart gesture based automation systems provides application in home automation which can support disabled and elderly persons. Home automation is the use and control of home appliances remotely or automatically. Every day the gap between humans and machines is being reduced. Mouse, keyboards have been used as input devices for many decades now in computers. However, with the rising popularity of gesture based devices and virtual technology, which allows users to interact with virtual objects, hand or body gestures are becoming essential. Recently introduced Metaverse and video game consoles like PlayStations gives us the insight of how the future of communication between human and computer would look like. Most of the electronic manufacturers are now focussing on hand gesture based automation. This proposed work uses the tool for the recognition and processing of hand gesture. Deep forest algorithm is used for image processing. At first, the image is captured by the camera and it is processed by OpenCV. If the preloaded gesture is matched with the existing gesture, the data will be sent to the microcontroller. This in turn controls the switching of home appliances. The hardware module consists of a camera, AVR ATmega 328p microcontroller, relay module, power supply and bulbs for demonstration purpose. Supported by the pyFirmata protocol, the microcontroller and the appliances as connected seem less manner. The scope of application for this project is versatile, not limiting its functionality to a fixed application. This gesture based technology helps disabled and elderly people to use smart systems more effectively, reducing cost, manpower and time to perform simple tasks for their comfort.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	viii
1	INTRODUCTION	1
	1.1. ARTIFICIAL INTELLIGENCE	1
	1.2. GESTURE BASED AUTOMATION	6
	1.3. ORGANIZATION OF THE REPORT	7
2	LITERATURE REVIEW	8
	2.1. LITERATURE SURVEY	10
	2.2. LITERATURE SUMMARY	11
	2.3. OBJECTIVE	
3	AIM AND SCOPE OF THE WORK	12
	3.1. PROPOSED METHOD	12
	3.2. AREA OF PROJECT WORK AND APPLICATION	12
4	MATERIALS AND METHODS USED	13
	4.1. BLOCK DIAGRAM	13
	4.2. HARDWARE	14
	4.2.1. AVR ATMEGA 328P WITH ARDUINO UNO	14
	4.2.2. FOUR CHANNEL RELAY MODULE WITH OPTOCOUPLER	17
	4.2.2.1. RELAY DRIVER CIRCUIT USING ULN 2003	21
	4.2.3. BULBS/LOAD	21

4.3.	CIRCUIT DIAGRAM	22
4.4.	SOFTWARE	23
4.4.1.	PYTHON	23
4.4.2.	LIBRARIES	25
	4.4.2.1. OPENCV	25
	4.4.2.2. PYFIRMATA	27
	4.4.2.3. MEDIAPIPE	28
4.4.3.	DEEP FOREST ALGORITHM	31
4.4.4.	STEPS INVOLVED IN CODING	32
	4.4.4.1. TO CREATE A WINDOW AND DETECT POSE USING OPENCV	32
	4.4.4.2. CONVERT BGR TO RGB	33
	4.4.4.3. DRAW LANDMARK ANNOTATIONS	34
	4.4.4.4. OPEN/CLOSE WHEN POSE IS DETECTED	35
	4.4.4.5. DETECT NUMBER OF FINGERS	36
	4.4.4.6. CONTROLLER PROGRAM	38
	4.4.4.7. IMPLEMENTING CODE	39
5	CONCLUSION AND FUTURE WORKS	43
	5.1. CONCLUSION	43
	5.2. FUTURE WORKS	43
	REFERENCES	44

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1.	ARTIFICIAL INTELLIGENCE	1
1.2.	GESTURE BASED AUTOMATION	6
4.1.	BLOCK DIAGRAM	13
4.2.1.	AVR ATMEGA 328P WITH ARDUINO UNO	14
4.2.2.	ARDUINO UNO PIN DIAGRAM	15
4.3.1.	FOUR CHANNEL RELAY MODULE	17
4.3.2.	PINOUT DIAGRAM RELAY	17
4.3.3.	MAINS VOLTAGE CONNECTION OF RELAY	18
4.3.4.	INTERNAL CIRCUIT DIAGRAM OF RELAY	18
4.3.5.	OPTOCOUPLER CIRCUIT	19
4.3.6.	FOUR WAY RELAY USING OPTOCOUPLER	20
4.4.	RELAY DRIVER CIRCUIT	21
4.5.	BULB/LOAD CIRCUIT	21
4.6.	CIRCUIT DIAGRAM	22
4.4.1.	SEGMENT OF HAND REGION	25
4.4.2.	21 3D HAND KNUCLEE CO-ORDINATES	30

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
4.2.1.	ARDUINO UNO TECHNICAL SPECIFICATIONS	14
4.2.2.	ARDUINO UNO PINOUT CONFIGURATION	16
4.3.	FOUR CHANNEL RELAY POINT	19

CHAPTER 1

INTRODUCTION

With an aim to help the disabled, elderly and children communicate with the devices directly through gestures instead of the conventional switchboards; we propose a system to make human hand tracking and controlling the household appliances possible in real time with minimal latency. The automated system developed until now was limited in the gestures made, not much supportive of the gestures made by the individual fingers and also slow in recognizing and computing the input parameters.

1.1. ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is a wide-ranging tool that enables people to rethink how we integrate information, analyse data, and use the resulting insights to improve decision making—and already it is transforming every walk of life. AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analysing large numbers of legal documents to ensure relevant fields are filled in a properly, AI tools often complete jobs quickly and with relatively few errors.

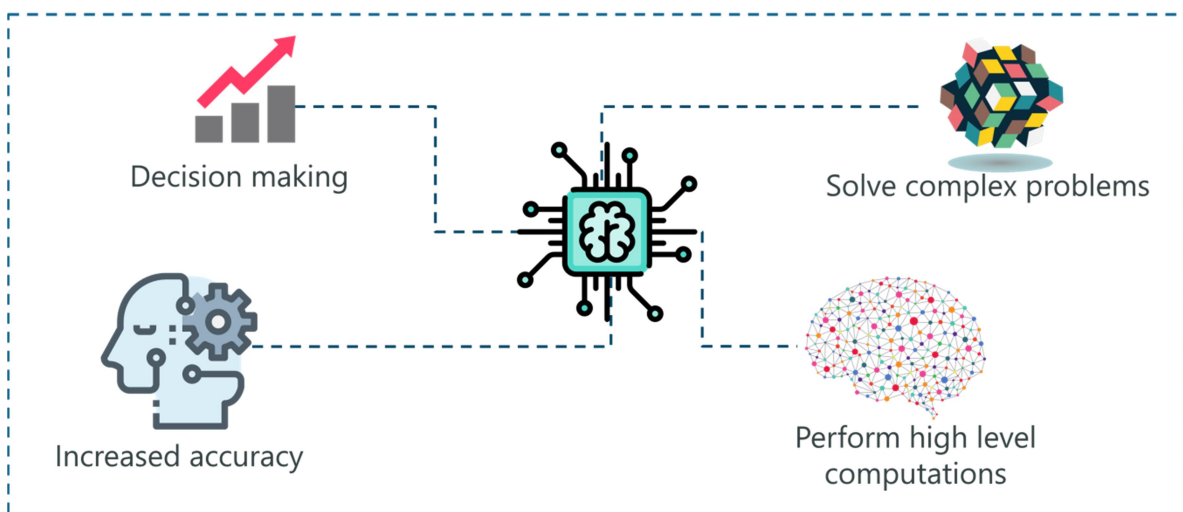


Fig 1.1: **ARTIFICIAL INTELLIGENCE**
(Source: [1])

AI is incorporated into a variety of different types of technology. Here are some examples:

- **Automation:** When paired with AI technologies, automation tools can expand the volume and types of tasks performed. An example is robotic process automation (RPA), a type of software that automates repetitive, rules-based data processing tasks traditionally done by humans. When combined with machine learning and emerging AI tools, RPA can automate bigger portions of enterprise jobs, enabling RPA's tactical bots to pass along intelligence from AI and respond to process changes.
- **Machine learning:** This is the science of getting a computer to act without programming. Deep learning is a subset of machine learning that, in very simple terms, can be thought of as the automation of predictive analytics. There are three types of machine learning algorithms:
 - Supervised learning. Data sets are labelled so that patterns can be detected and used to label new data sets.
 - Unsupervised learning. Data sets aren't labelled and are sorted according to similarities or differences.
 - Reinforcement learning. Data sets aren't labelled but, after performing an action or several actions, the AI system is given feedback.
- **Machine vision:** This technology gives a machine the ability to see. Machine vision captures and analyses visual information using a camera, analog-to-digital conversion and digital signal processing. It is often compared to human eyesight, but machine vision isn't bound by biology and can be programmed to see through walls, for example. It is used in a range of applications from signature identification to medical image analysis. Computer vision, which is focused on machine-based image processing, is often conflated with machine vision.
- **Natural language processing (NLP):** This is the processing of human language by a computer program. One of the older and best-known examples of NLP is spam detection, which looks at the subject line and text of an email and decides if it's junk. Current approaches to NLP are based on machine learning. NLP tasks include text translation, sentiment analysis and speech recognition.

- *Robotics*: This field of engineering focuses on the design and manufacturing of robots. Robots are often used to perform tasks that are difficult for humans to perform or perform consistently. For example, robots are used in assembly lines for car production or by NASA to move large objects in space. Researchers are also using machine learning to build robots that can interact in social settings.
- *Self-driving cars*: Autonomous vehicles use a combination of computer vision, image recognition and deep learning to build automated skill at piloting a vehicle while staying in a given lane and avoiding unexpected obstructions, such as pedestrians.

Some components of AI are discussed here:

Its application include,

- Image recognition
- Speech recognition
- Chatbots
- Natural language generation
- Sentiment analysis

And some of the types of models are,

- Deep learning
- Machine learning
- Neural networks

Software/hardware for training and running these models,

- GPUs
- Parallel processing tools (like spark)
- Cloud data storage and compute platforms

Programming languages for building models,

- Python
- TensorFlow
- Java
- C, C++

AI is not just one technology. Artificial intelligence has made its way into a wide variety of markets. Here are nine examples.

AI in healthcare: The biggest bets are on improving patient outcomes and reducing costs. Companies are applying machine learning to make better and faster diagnoses than humans. One of the best-known healthcare technologies is IBM Watson. It understands natural language and can respond to questions asked of it. The system mines patient data and other available data sources to form a hypothesis, which it then presents with a confidence scoring schema. Other AI applications include using online virtual health assistants and chatbots to help patients and healthcare customers find medical information, schedule appointments, understand the billing process and complete other administrative processes. An array of AI technologies is also being used to predict, fight and understand pandemics such as COVID-19.

AI in business: Machine learning algorithms are being integrated into analytics and customer relationship management (CRM) platforms to uncover information on how to better serve customers. Chatbots have been incorporated into websites to provide immediate service to customers. Automation of job positions has also become a talking point among academics and IT analysts.

AI in education: AI can automate grading, giving educators more time. It can assess students and adapt to their needs, helping them work at their own pace. AI tutors can provide additional support to students, ensuring they stay on track. And it could change where and how students learn, perhaps even replacing some teachers.

AI in finance: AI in personal finance applications, such as Intuit Mint or TurboTax, is disrupting financial institutions. Applications such as these collect personal data and provide financial advice. Other programs, such as IBM Watson, have been applied to the process of buying a home. Today, artificial intelligence software performs much of the trading on Wall Street.

AI in law: The discovery process -- sifting through documents -- in law is often overwhelming for humans. Using AI to help automate the legal industry's labor-intensive processes is saving time and improving client service. Law firms are using machine learning to describe data and predict outcomes, computer vision to classify and extract information from documents and natural language processing to interpret requests for information.

AI in manufacturing: Manufacturing has been at the forefront of incorporating robots into the workflow. For example, the industrial robots that were at one time programmed to perform single tasks and separated from human workers, increasingly function as cobots: Smaller, multitasking robots that collaborate with humans and take on responsibility for more parts of the job in warehouses, factory floors and other workspaces.

AI in banking: Banks are successfully employing chatbots to make their customers aware of services and offerings and to handle transactions that don't require human intervention. AI virtual assistants are being used to improve and cut the costs of compliance with banking regulations. Banking organizations are also using AI to improve their decision-making for loans, and to set credit limits and identify investment opportunities.

AI in transportation: In addition to AI's fundamental role in operating autonomous vehicles, AI technologies are used in transportation to manage traffic, predict flight delays, and make ocean shipping safer and more efficient.

Advantages:

- Good at detail-oriented jobs;
- Reduced time for data-heavy tasks;
- Delivers consistent results.

1.2. GESTURE BASED AUTOMATION

The invention of the touch screen brought some major improvements to the man-computer interaction, making it more natural and tactile. Soon, users could use not just one finger, but multiple fingers simultaneously – multi-touch technology enabled us to use more than one point of contact with the surface (i.e. a trackpad or touchscreen) at the same time. This prompted the emergence of multi-finger gestures allowing the user to zoom in, zoom out, scroll, rotate, and toggle between the elements of the user interface.

Much progress has been made, but we believe that people can communicate even more effectively when given a richer lexicon of gestures. Hand gestures can add another dimension to communication. Just like verbal communication involves enriching voice with subtle facial expressions and gestures, so does the machine interface benefit from an additional layer of interactivity provided by gestures.

Gesture control technology is based on gesture recognition. Gesture recognition can be seen as a way for computers to begin to understand human body language. Compared to the primitive user interfaces, such as keyboard and mouse, it builds a richer bridge between the computers and humans. Gesture control devices recognize and interpret human body movements, allowing the user to interact with a computer system.



Fig 1.2: **GESTURE BASED AUTOMATION**

(Source: [2])

Gesture tracking technologies can be broadly divided into perceptual (utilizing, e.g., vision and audio) and non-perceptual (based on, e.g., mouse or touch surfaces). The taxonomy proposed by Karam and schraefel divides gesture styles between deictic (pointing), manipulative (tight coupling between the controlled object

and hand/arm movement), semaphoric (symbolic), gesticulation (combination of gesture and speech), and language gestures (e.g., sign languages). Due to the prevalence of symbolic gesturing in the real world (e.g., police and military hand signals), we chose it as the starting point for our design. Empty-handed, or touchless, gestural interaction has been proposed to be advantageous in various real world contexts, such as medical environments and environments where physical input devices could be vandalized. In our context of use, the situation is similar to the latter, as introducing input devices into the factory environment is not viable. For example, touchscreens could be smeared with residues, and remote controls could easily be damaged or misplaced. Examples of perceptual gesture-based interaction in a context of use resembling ours can be found in human-robot interaction. However, to our knowledge, our work is one of the first user-centered studies of designing gesture-based control to operate factory automation systems.

1.3. ORGANIZATION OF THE REPORT

The project report is organized into six chapters.

Chapter 1 focuses on introduction about artificial intelligence which would help system proposed. The advantages of the gesture based automation and artificial intelligence, its mechanism and the uses are discussed.

Chapter 2 provides detailed literature survey related to the project and objectives of the project.

Chapter 3 describes about proposed method and area of project work and its applications.

Chapter 4 explains about the materials and methods used in detail. The hardware functionalities and software algorithm are discussed in this chapter.

Chapter 5 provides the simulation results and performance analysis of the proposed methods and techniques. The performances of proposed mechanism which is displayed at the PC output screen has been discussed.

Chapter 6 altogether concludes the project. It also gives the scheme of the future work that is how the project could be enhanced further. Finally the references referred to precede the project are given in brief.

CHAPTER 2

LITERATURE REVIEW

In recent years, several remote hand-gesture control systems for home-media systems have become commercially available. Such systems aim to augment the living-room media experience and enhance user enjoyment. In this regard, scenarios that call for user identification include interface customization (i.e., facilitating a personalized gesture vocabulary), content adaptation, and parental control. Hand gesture recognition is implemented for various systems. Thresholding Algorithm is used for recognition purpose. This chapter provides detailed literature survey related to the project and summary of Literature.

2.1. LITERATURE SURVEY

Kishor P Jadhav et al. (2016) –[3] proposed system for Virtual Switch Control to remove the need to look into the hand held remote and to search for a specific key for specific function. Firstly, camera captures the image of user's hand gesture & then sends it to PC for further image processing. Images are captured by using camera and are processed using MATLAB. The circuit consists of AVR microcontroller, relay driver IC, relay, camera, PC, USB to TTL converter and load. But it is costly and not user friendly.

A. Hayder et al. (2010) –[4] implemented Soapbox method of Sensing, Operating and Activating Peripheral Box, that has low power consumption. It is equipped with 3-axis accelerometer, an illumination sensor, an electronic compass and an optical proximity sensor. For communication purpose, it is wireless with RF technology and code is developed in C.

Archana S Ghotkar et al. (2012) –[5] designed an in glove based device controlling approach, the exact location and orientation of the finger, coordinates of palm and hand configuration are proposed (Dan et al, 2014). The glove uses flex sensor that bends whenever the hand finger makes particular gesture and it gives variety of resistance. Then the accelerometer measures the motion of the hand, the flex sensor produces analogue output which are converted into digital output using ADC converter. The converted digital signals are then sent to computer to process further the gestures into more meaningful information. Even though the glove based system

is expensive it has advantages over the other systems because of its high accuracy and fast speed reaction.

Derpanis K G et al. (2004) –[6] pointed that traditional interfaces, keyboards and mice present a bottleneck in application that rely on heavy interaction of the user with the machine due to the unnaturalness of the interaction. From reading lots of related articles, we have learnt that recent efforts have attempted to eliminate this bottleneck by developing different ways of interacting with computers, for example: speech, handwriting and gesture recognition.

Bhuiyan M et al. (2004) –[7] thought that gesture technology can provide more entertainment opportunity for any type of users. According to the observation, we found that in the game industry, game products have been successfully using this technology, such as Sony Eyetoy, Wii, Microsoft's Xbox, etc. Bhuiyan and Picking also presented that physical simulation can improve the realism of the resulting gestural animation in several ways. Besides, this technology can be developed for training and education purpose

Garg P et al. (2009) –[8] described many successful applications of gesture recognition in varied fields including virtual environment, sign language translation, medical system, etc. However, they also discussed some open issues. This technology is suitable in a controlled lab setting but does not generalize to arbitrary settings. If there are no high contrast stationary backgrounds and ambient lighting conditions, the recognition is very likely to make mistakes. And the machine cannot recognize the start and end points of meaningful gestures from continuous motion of the hands.

Kruti Goyal et al. (2017) –[9] wrote “Face Detection and Tracking Using OpenCV” which helped us understand use cases of OpenCV and its benefits over other plug-ins for the purpose of image processing purposes .

Munir Oudah et al. (2020) –[10] with their “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques” opened eyes on many techniques used for Hand gesture techniques and we finally were able to conclude the preferred technique for the purpose of hand recognition.

Shivasankara S et al. (2018) –[11] in their “American Sign Language Recognition System: An Optimal Approach” explained how to find landmark annotations of the human hand from which can be loaded to the software for finding out each finger.

Omkar Vedak et al. (2020) –[12] wrote in “Sign language Interpreter using Image Processing and Machine Learning” about medaipipe in detail which greatly help in the recognition system.

Hans-Christoph Steiner (2009) –[13] “Firmata: Towards making microcontrollers act like extensions of the computer” aided in choosing the firmware for controlling then appliances in the most efficient and easy way possible.

2.2. LITERATURE SUMMARY

On reviewing various papers and articles, we have found that Early home automation began with labor-saving machines. Self-contained electric or gas powered home appliances became viable in the 1900s with the introduction of electric power distribution and led to the introduction of washing machines (1904), water heaters (1889), refrigerators, sewing machines, dishwashers, and clothes dryers.

We have read some articles which would be explaining about the different opinions on advantages and disadvantages of the gesture-based interaction technology. That helps us to make further understanding, enlarge our thinking, and guide us in our later related work.

In 1975, the first general purpose home automation network technology, X10, was developed. It is a communication protocol for electronic devices. It primarily uses electric power transmission wiring for signalling and control, where the signals involve brief radio frequency bursts of digital data, and remains the most widely available. By 1978, X10 products included a 16 channel command console, a lamp module, and an appliance module. Soon after came the wall switch module and the first X10 timer. By 2012, in the United States, according to ABI Research, 1.5 million home automation systems were installed. There are three generations of home automation: First generation: wireless technology with proxy server, e.g. Zigbee

automation. Second generation: artificial intelligence controls electrical devices, e.g. Amazon Echo. Third generation: robot buddy who interacts with humans, e.g. Robot Rovio, Roomba. These findings have led to defining the objective of the project work.

2.3. OBJECTIVE

We propose a method to solve the issues discussed in previous sections and from the findings obtained from literature review, a simple yet very effective system is designed with an objective to:

- To design an AI based automation system using mesh detection.
- To develop python code using Deep Forest Algorithm to process images obtained from webcam.
- To use Mesh algorithm for image processing, and to extract gestures using OpenCV.
- To perform switching operation of appliances with the help of AVR AT mega 328P by using PyFirmata Protocol

CHAPTER 3

AIM AND SCOPE OF THE WORK

The previous chapter explains about the description about the methodologies used in the existing concepts. While this chapter gives a detailed description about the methodologies used in the literature survey.

3.1. PROPOSED METHOD:

Conventionally, Deep Neural Networks (DNN) method using TensorFlow and TensorFlow Lite Framework have been used. This increased the temperature of the microcontrollers utilized for the project. A high amount of accuracy can be obtained using such complex and high computation power drawing techniques. We have used OpenCV for Object Detection and Recognition. Using this method, desired output can be obtained with less computation power consumed. High Speed with a moderate accuracy can be obtained. A simple yet effective hardware consisting of just an AVR Atmega 328p microcontroller and 4-Channel Relay module is used for the working of the system.

3.2. AREA OF PROJECT WORK AND APPLICATION

Hand gestures are used to operate the devices and home gadgets such as lights, fans, and many more. The main tool used for capturing the motions and gestures of hand is OpenCV. It is required to rerun the program each time subsequent to catching and handling one motion and appliance is controlled. Research is going ahead to control all the home machines once program is effectively run.

These applications include home automation, industry, sign language recognition, navigation and manipulation in virtual environments, automotive manufacturing, and control for robotics. Industrial actuators such as industrial or cooperative robots are a perfect example of devices that can be controlled by gestures.

The hand is the most easily used interface and is a medium capable of various formations due to its high degree of freedom. Therefore, the hands are a logical choice for gesture recognition because of their dexterity and intricate movements.

CHAPTER 4

MATERIALS AND METHODS USED

In this chapter we discuss various materials and methods used for the project:

- AVR ATmega 328P with Arduino UNO
- Four Channel relay module with optocoupler
- Bulbs/Load
- Camera integrated in a computer
- Software Tools: Arduino IDE, PyCharm for coding in Python

4.1. BLOCK DIAGRAM

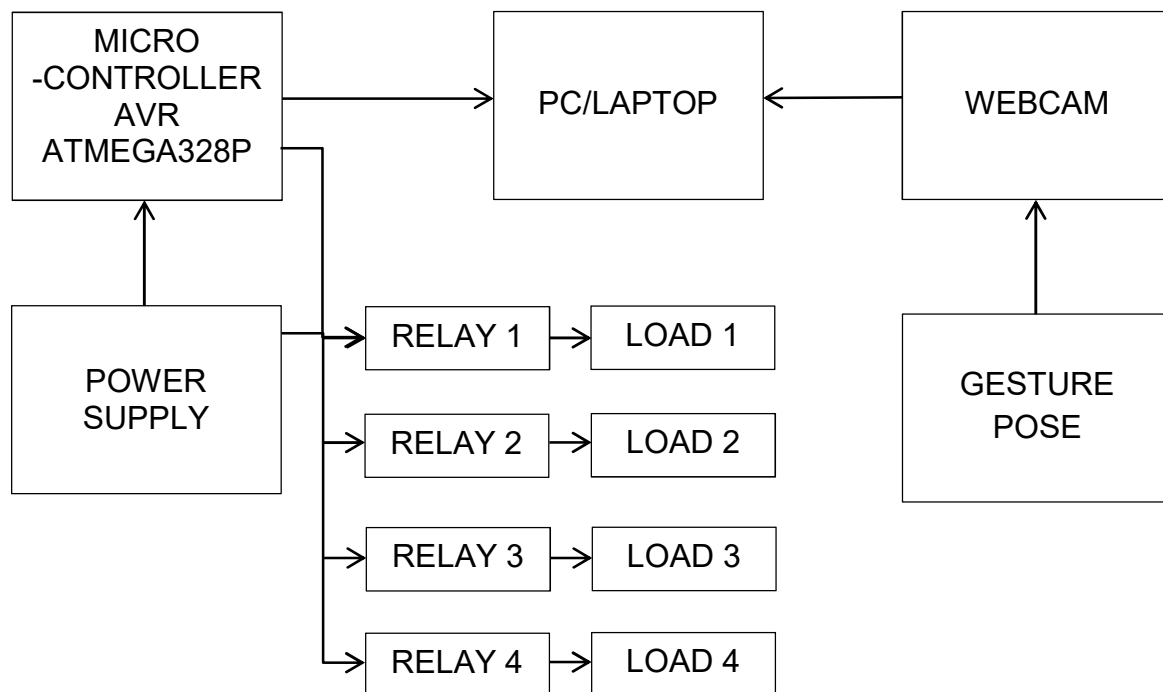


Fig 4.1: **BLOCK DIAGRAM**

4.2. HARDWARE

All hardware parts used in the proposed system are explained in detail in upcoming topics. This provides deeper understanding on the application of the parts used for various purposes.

4.2.1. AVR ATMEGA 328P WITH ARDUINO UNO

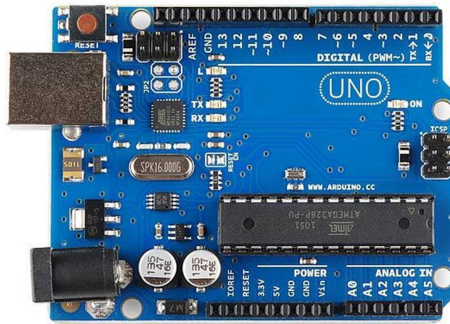


Fig 4.2.1: **AVR ATMEGA 328P WITH ARDUINO UNO**
(Source: [14])

Arduino/Genuino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller.

Microcontroller	ATmega328P – 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)

Table 4.2.1: **ARDUINO UNO TECHNICAL SPECIFICATIONS**

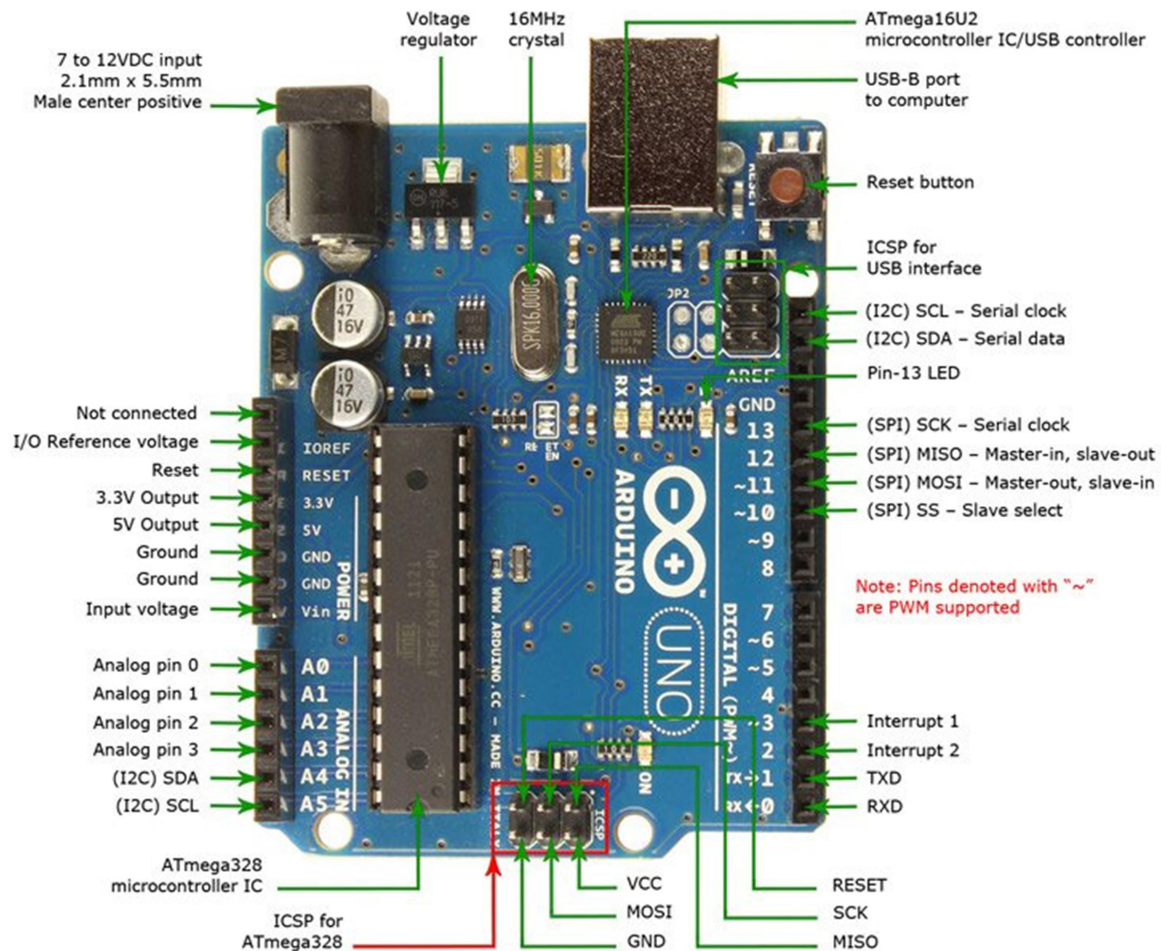


Fig 4.2.2: **ARDUINO UNO PIN DIAGRAM**

(Source: [15])

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip.

This setup has other implications. When the Uno is connected to a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the UNO. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened

The ATmega328 is a single –chip microcontroller created by Atmel in the megaAVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core

ATmega328 is commonly used in many projects and autonomous systems where a simple, low-powered, low-cost micro-controller is needed. Perhaps the most common implementation of this chip is on the popular Arduino development platform, namely the Arduino Uno, Arduino Nano models.

Table 4.2.2: **ARDUINO UNO PINOUT CONFIGURATION**

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

4.2.2. FOUR CHANNEL RELAY MODULE WITH OPTOCOUPLER



Fig 4.3.1: **FOUR CHANNEL RELAY MODULE**
(Source: [16])

A relay operating on the principle of opto-isolation, i.e. there is no direct or hardwire connection between the controller and the relay. And because there is no hardwire, the controller will remain protected against the high voltages that might harm the controller.

It is designed to switch up to 4 high currents (10A) or high voltage (250V) loads with the help of microcontroller. It is designed to interface with microcontroller such as Arduino, PIC and etc. The relays terminal (COM, NO and NC) is being brought out with screw terminal.

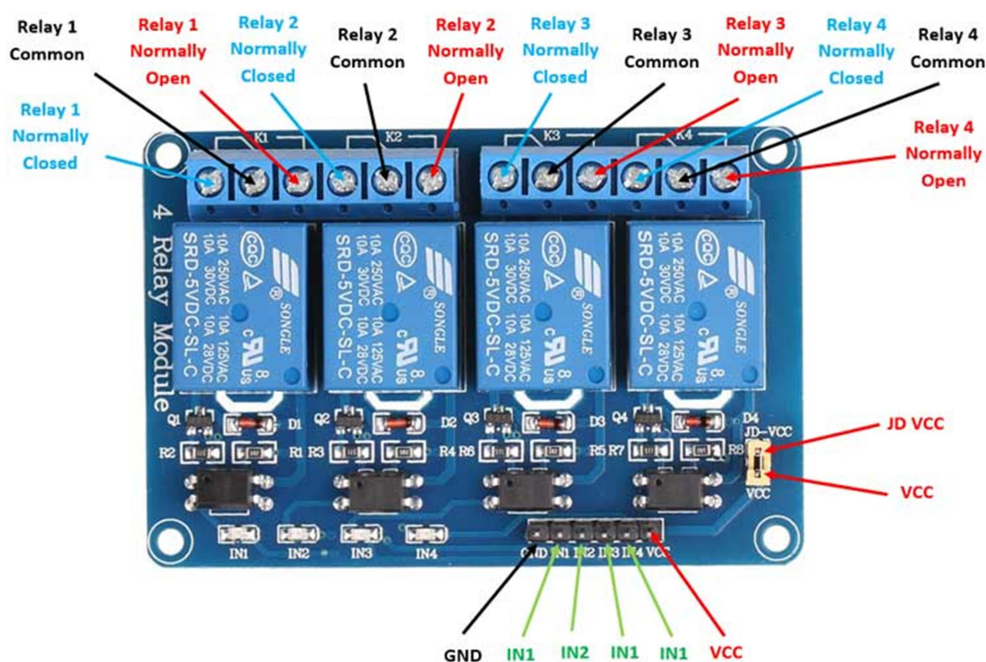


Fig 4.3.2: **PIN OUT DIAGRAM OF RELAY**
(Source: [17])

This relay module has two channels (those blue cubes). There are other models with one, four and eight channels. This module should be powered with 5V, which is appropriate to use with an Arduino. The high-voltage side has two connectors, each with three sockets.

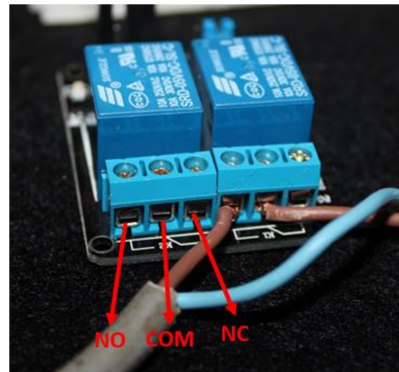


Fig 4.3.3: **MAINS VOLTAGE CONNECTION FOR RELAY**
(Source: [18])

- COM: Common pin
- NC (Normally Closed): the normally closed configuration is used when the relay is to be closed by default, meaning the current is flowing unless a signal is sent from the Arduino to the relay module to open the circuit and stop the current.
- NO (Normally Open): the normally open configuration works the other way around.

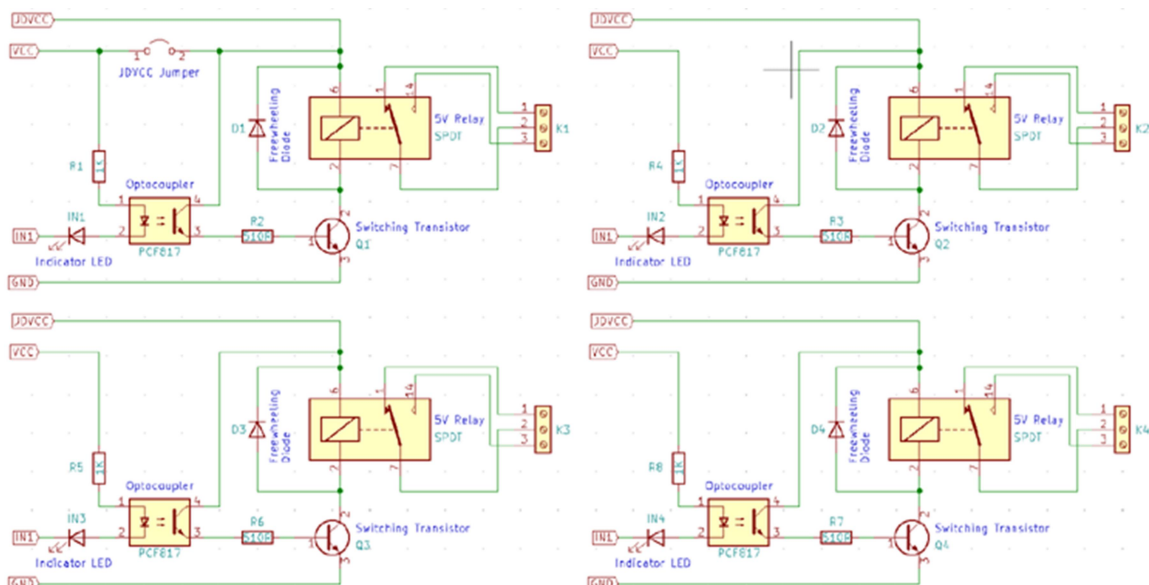


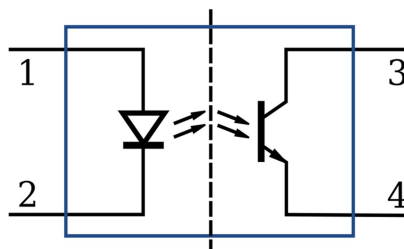
Fig 4.3.4: **INTERNAL CIRCUIT DIAGRAM OF FOUR CHANNEL RELAY**
(Source: [19])

Table 4.3: **FOUR CHANNEL RELAY PINOUT**

Pin Number	Pin Name	Description
1	GND	Ground reference for the module
2	IN1	Input to activate relay 1
3	IN2	Input to activate relay 2
4	IN3	Input to activate relay 3
5	IN4	Input to activate relay 4
6	V_{CC}	Power supply for the relay module
7	V_{CC}	Power supply selection jumper
8	JD-V_{CC}	Alternate power pin for the relay module

OptoCouplers:

OptoCouplers are also known as Opto Isolators. Optocoupler consists of one light-emitting diode and one photosensitive semiconductor which is used to detect the light emitted by led.

Fig 4.3.5: **OPTOCOUPLER CIRCUIT**
(Source: [20])

OPTOCOUPLER

The figure above shows the diagram of the optocoupler Which consists of an LED and a phototransistor. When the LED is in an off state, the transistor works as an off switch. When the LED is turned on, the light falls on the base terminal of the transistor, causing the V_{be} Junction forward bias. When the transistor's (NPN transistor) V_{be} junction becomes a forward bias then it acts as a closed switch. Excess of light on the base terminal will flow more electrons from the transistor.

Opto-isolators prevent high voltages from affecting the system receiving the signal. Commercially available opto-isolators with stand input-to-output voltages up to 10 kV and voltage transients with speeds up to 25 kV/ μ s.

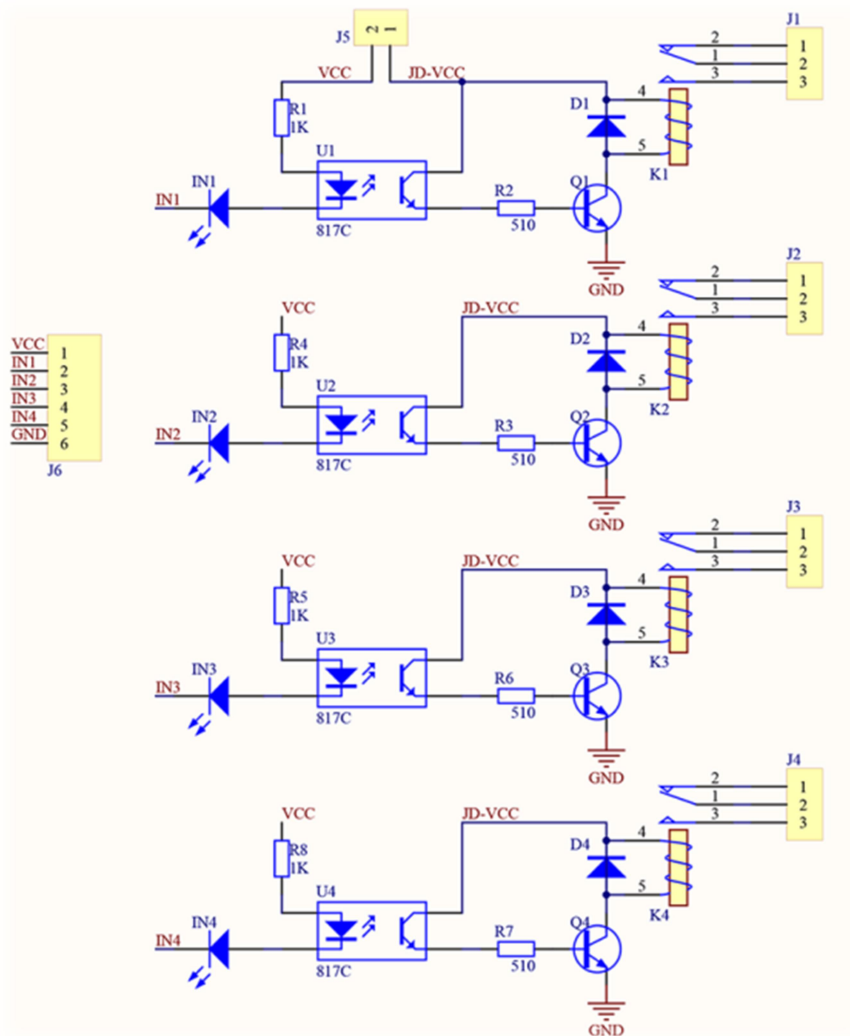


Fig 4.3.6: **FOUR WAY RELAY USING OPTOCOUPLER CIRCUIT**
(Source: [21])

A common type of opto-isolator consists of an LED and a phototransistor in the same opaque package. Other types of source-sensor combinations include LED-photodiode, LED-LASCR, and lamp-photoresistor pairs. Usually opto-isolators transfer digital (on-off) signals, but some techniques allow them to be used with analog signals.

A common type of opto-isolator consists of an LED and a phototransistor in the same opaque package. Other types of source-sensor combinations include LED-photodiode, LED-LASCR, and lamp-photoresistor pairs. Usually opto-isolators transfer digital (on-off) signals, but some techniques allow them to be used with analog signals.

4.2.2.1. RELAY DRIVER CIRCUIT USING ULN 2003

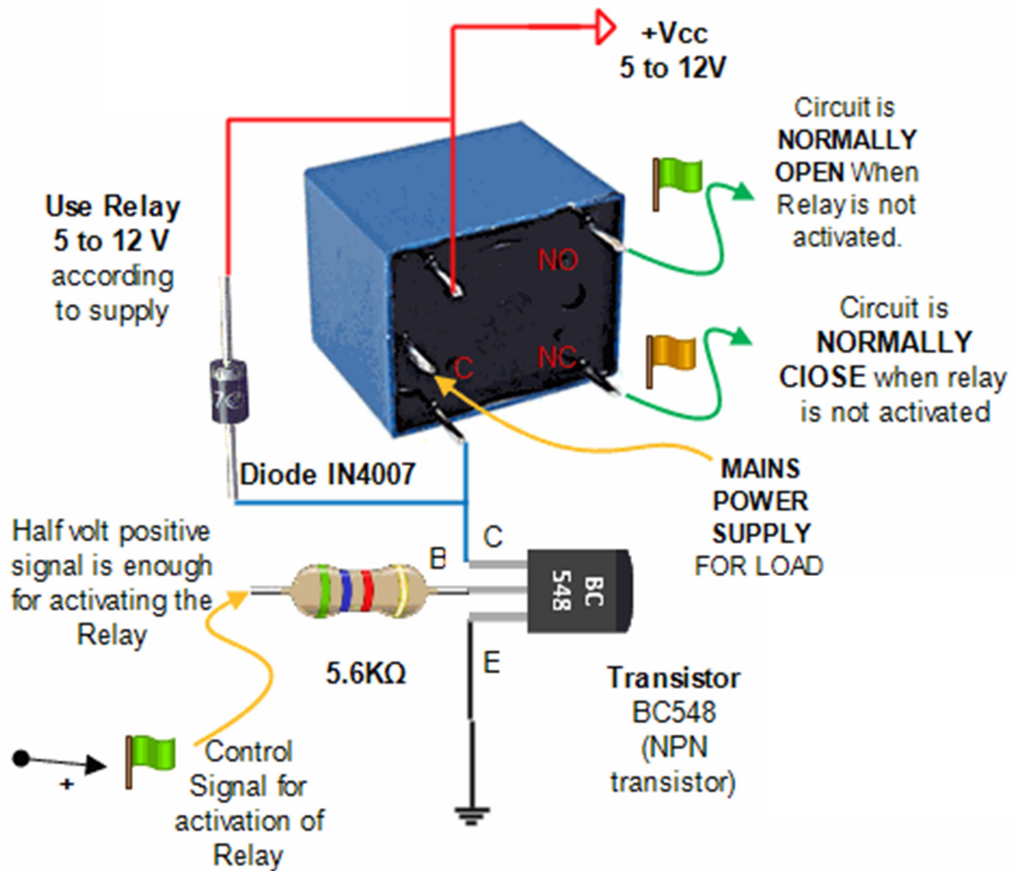


Fig 4.4: **RELAY DRIVER CIRCUIT**

(Source: [22])

Typical usage of the ULN2003A is in driver circuits for relays. ULN2003 is a 16 Pin IC, consisting of 7 Darlington pairs (each pair protected with suppression diode) and thus has the capacity to handle a maximum of 7 loads (could be inductive).

4.2.3. BULBS/LOAD

Bulbs are used for demonstration purpose.

The system has four loads to which any house appliance can be connected.

The socket connection is shown in the figure.

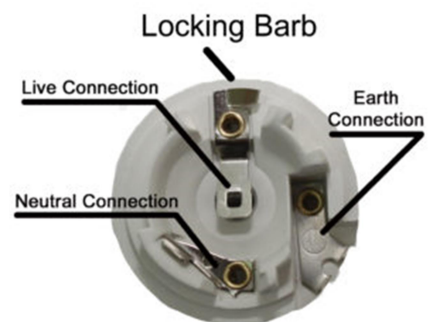


Fig 4.5: **BULB/LOAD CIRCUIT**

(Source: [23])

4.3. CIRCUIT DIAGRAM

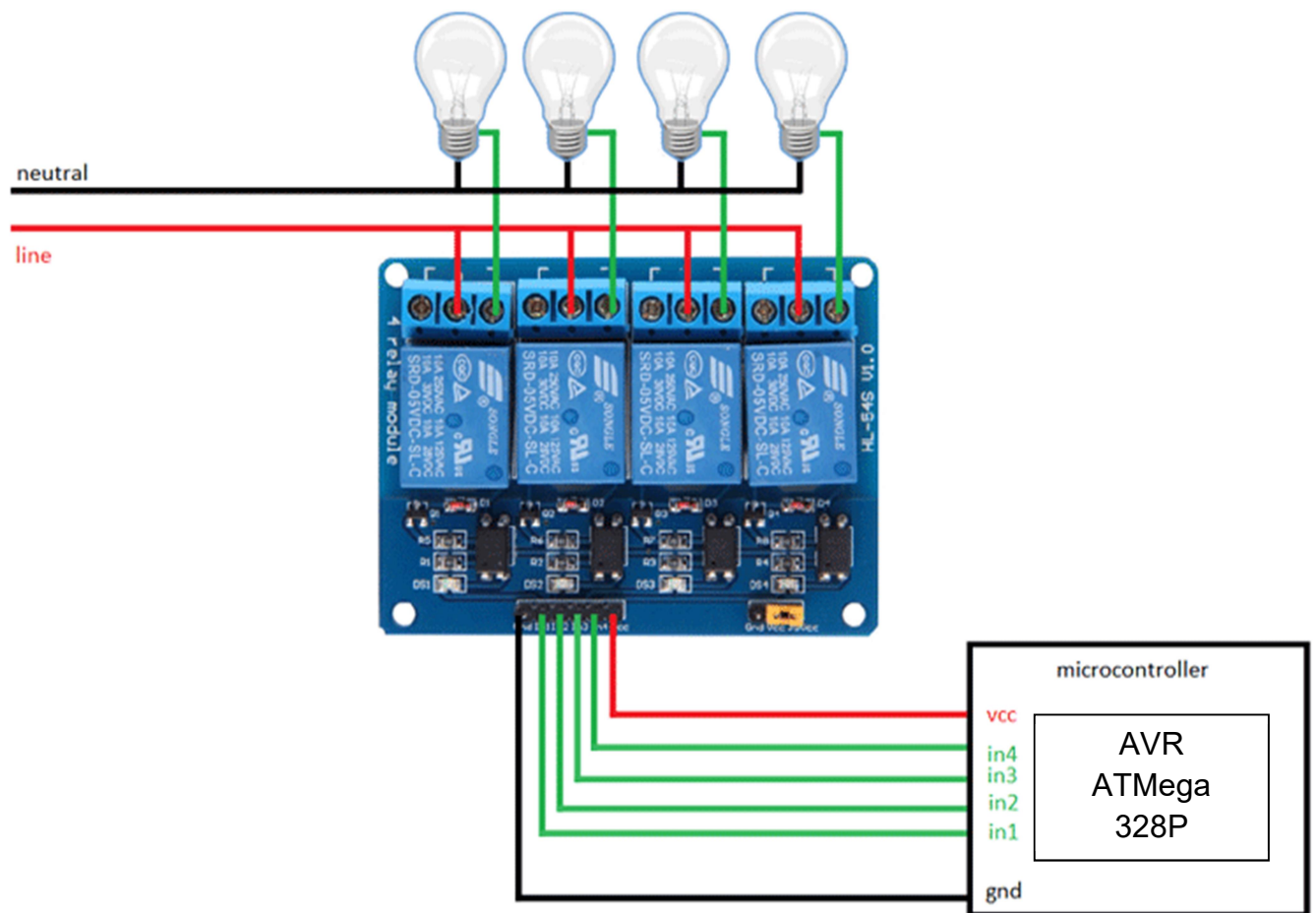


Fig 4.6: **CIRCUIT DIAGRAM**

4.4. SOFTWARE

Various software tools used, its algorithm and methods of implementation are discussed in detail in the following section. For better understanding, python, arduino IDE and object detection methods are explained.

4.4.1. PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name “Python” is nod to the British comedy group Monty Python has a reputation as a beginner-friendly language, replacing Java as the most widely used introductory language because it handles much of the complexity for the user, allowing beginners to focus on fully grasping programming concepts rather than minute details.

Python is used for server-side web development, software development, mathematics, and system scripting and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structure, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python’s support of modules and packages facilitates modular programs and reuse of code.

Python Use Cases

- Creating web application on a server
- Building workflows that can be used in conjunction with software
- Connecting to database systems
- Reading and modifying files
- Performing complex mathematics
- Processing big data
- Fast prototyping
- Developing production-ready software

Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Developers also use Python to build productivity tools, games, and desktop apps.

Features and benefits of Python

- Compatible with a variety of platforms including Windows, Mac, Linux, Raspberry Pi, and others
- Uses a simple syntax comparable to the English language that lets developers use fewer lines than other programming languages
- Operates on an interpreter system that allows code to be executed immediately, fast-tracking prototyping
- Can be handled in a procedural, object-orientated, or functional way

Python Syntax

- Somewhat similar to the English language, with a mathematical influence, Python is built for readability
- Unlike other languages that use semicolons and/or parentheses to complete a command, Python uses new lines for the same function
- Defines scope (i.e., loops, functions, classes) by relying indentation, using whitespace, rather than braces (aka curly brackets)

Python and AI

AI researchers are fans of Python. Google TensorFlow, as well as other libraries (scikit-learn, Keras), establish a foundation for AI development because of the usability and flexibility it offers Python users. These libraries, and their availability, are critical because they enable developers to focus on growth and building.

4.4.2. LIBRARIES

Libraries used in the coding process are explained below. It includes OpenCV, Pyfirmata, mediapipe. Each has specific application for various purposes i.e., for object recognition, etc.,

4.4.2.1. OPENCV

OpenCV is Open Computer Vision Library. It was initially launched in 1999 by Intel. With more updates, it has been modified since then to aim for the real-time computer vision. This library has been written under programming languages like C and C++. It can be easily run on operating systems Windows and Linux. This library can be easily interface with programming languages like Python, MATLAB, Ruby and others as well. Along with NumPy and Python image processing (shape & color detection) can be performed at ease. Read an Image First, a sample image in which processing is to be applied is to be read. It's done using a pre-defined Python function: `CV2.imread()`. The sample image should be available in current folder or the full location of the image is to be mentioned as an argument. For reading an image, below stated functions can be used:

`imread_color, imread_grayscale, imread_unchanged` etc.

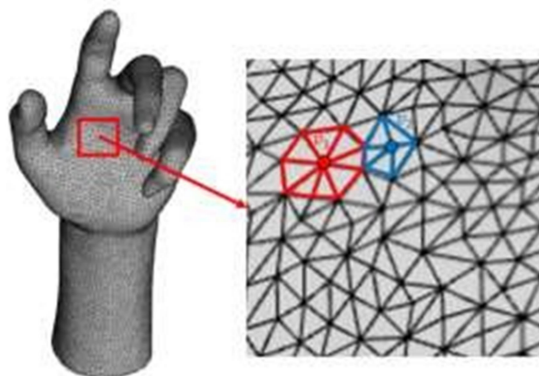


Fig 4.4.1: **SEGMENT OF A HAND REGION**
(Source: [24])

Steps to process hand region:

1. Find and segment the hand region from the video sequence.
2. Count the number of fingers from the segmented hand region in the video sequence.

3. The first step in hand gesture recognition is to find the hand region by eliminating all the other unwanted portions in the video sequence.

Note: Video sequence is just a collection of frames or collection of images that runs with respect to time.

Finding out the hand region:

Background Subtraction: First, an efficient method to separate foreground from background is needed. Concept of running averages is used. The system looks over a particular scene for 30 frames. During this period, the running average over the current frame and the previous frames is computed.

After figuring out the background, the system understand that the hand is a new entry into the background, which means it becomes the foreground object. After figuring out the background model using running averages, current frame is used which holds the foreground object (hand) in addition to the background. The absolute difference between the background model (updated over time) and the current frame (which has our hand) to obtain a difference image that holds the newly added foreground object (which is our hand) is calculated.

Motion Detection and Thresholding: To detect the hand region from this difference image, thresholds the difference image, so that only the hand region becomes visible and all the other unwanted regions are painted as black. Thresholding is the assignment of pixel intensities to 0's and 1's based a particular threshold level so that the object of interest alone is captured from an image.

Contour Extraction: After thresholding the difference image, contours in the resulting image is found out. The contour with the *largest area* is assumed to be the hand.

Note: Contour is the outline or boundary of an object located in an image.

4.4.2.2. PYFIRMATA

Firmata is a protocol for communicating with microcontrollers from software on a computer (or smartphone/tablet, etc). The protocol can be implemented in firmware on any microcontroller architecture as well as software on any computer software package (see list of client libraries below).

Firmata is based on the midi message format in that commands bytes are 8 bits and data bytes are 7 bits. For example the midi Channel Pressure (Command: 0xD0) message is 2 bytes long, in Firmata the Command 0xD0 is used to enable reporting for a digital port (collection of 8 pins). Both the midi and Firmata versions are 2 bytes long, but the meaning is obviously different. In Firmata, the number of bytes in a message must conform with the corresponding midi message. Midi System Exclusive (Sysex) messages however, can be any length and are therefore used most prominently throughout the Firmata protocol.

This repository contains documentation of the Firmata protocol. The core of the protocol is described in the `protocol.md` file. Feature-specific documentation is described in individual markdown files (`i2c.md`, `accelStepperFirmata.md`, `servos.md`, etc). Files added to the `proposals` directory are proposals for new features that have not yet been finalized. See `feature-registry.md` for the full list of documented firmata features.

The Firmata protocol could theoretically be implemented for any microcontroller platform. Currently however, the most complete implementation is for Arduino (including Arduino-compatible microcontrollers).

Updating Firmata in the Arduino IDE - Arduino 1.6.4 and higher

To update to the latest stable version:

1. Open the Arduino IDE and navigate to: Sketch > Include Library > Manage Libraries
2. Filter by "Firmata" and click on the "Firmata by Firmata Developers" item in the list of results.

3. Click the Select version dropdown and select the most recent version (note you can also install previous versions)
4. Click Install.

From your computer, connect to your Arduino UNO.

Execute this command on a terminal inside your Arduino UNO using PyCharm:

```
pip install pyfirmata
```

4.4.2.3. MEDIAPIPE

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, our method achieves real-time performance on a mobile phone, and even scales to multiple hands. We hope that providing this hand perception functionality to the wider research and development community will result in an emergence of creative use cases, stimulating new applications and new research avenues.

ML Pipeline: MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together: A palm detection model that operates on the full image and returns an oriented hand bounding box. A hand landmark model that operates on the cropped image region defined by the palm detector and returns high-fidelity 3D hand keypoints. This strategy is similar to that employed in our MediaPipe Face Mesh solution, which uses a face detector together with a face landmark model.

Providing the accurately cropped hand image to the hand landmark model drastically reduces the need for data augmentation (e.g. rotations, translation and scale) and instead allows the network to dedicate most of its capacity towards coordinate prediction accuracy. In addition, in our pipeline the crops can also be generated based on the hand landmarks identified in the previous frame, and only when the landmark model could no longer identify hand presence is palm detection invoked to relocalize the hand.

The pipeline is implemented as a MediaPipe graph that uses a hand landmark tracking subgraph from the hand landmark module, and renders using a dedicated hand renderer subgraph. The hand landmark tracking subgraph internally uses a hand landmark subgraph from the same module and a palm detection subgraph from the palm detection module.

Models:

Palm Detection Model: To detect initial hand locations, a single-shot detector model is designed and optimized for mobile real-time uses in a manner similar to the face detection model in MediaPipe Face Mesh. Detecting hands is a decidedly complex task: lite model and full model have to work across a variety of hand sizes with a large scale span ($\sim 20\times$) relative to the image frame and be able to detect occluded and self-occluded hands. Whereas faces have high contrast patterns, e.g., in the eye and mouth region, the lack of such features in hands makes it comparatively difficult to detect them reliably from their visual features alone. Instead, providing additional context, like arm, body, or person features, aids accurate hand localization.

This method addresses the above challenges using different strategies. First, a palm detector is trained instead of a hand detector, since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using square bounding boxes (anchors in ML terminology) ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3-5. Second, an encoder-decoder feature extractor is used for bigger scene context awareness even for small objects (similar to the RetinaNet approach). Lastly, the focal loss is minimized during training to support a large amount of anchors resulting from the high scale variance.

With the above techniques, an average precision of 95.7% in palm detection can be achieved. Using a regular cross entropy loss and no decoder gives a baseline of just 86.22%.

Hand Landmark Model: After the palm detection over the whole image, the subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

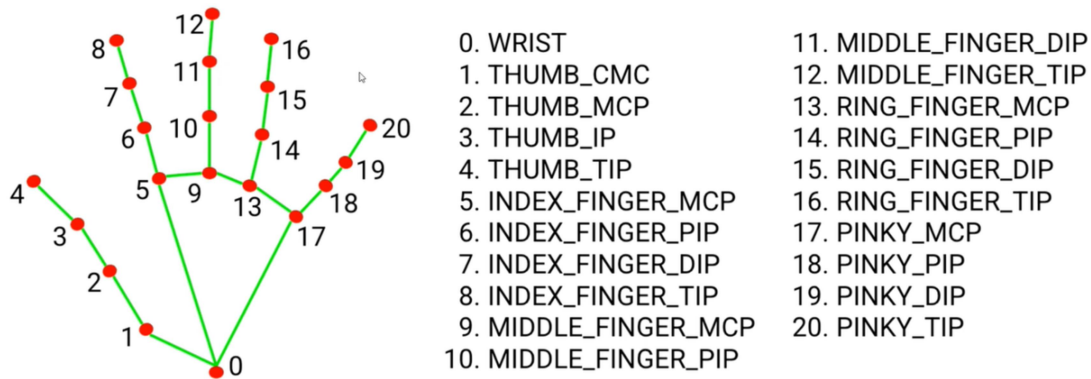


Fig 4.4.2: **21 3D HAND KNUCKLE COORDINATES**

(Source: [25])

To obtain ground truth data, ~30K real-world images are manually annotated with 21 3D coordinates, as shown above (Z-value is taken from image depth map, if it exists per corresponding coordinate). To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, a high-quality synthetic hand model is rendered over various backgrounds and map it to the corresponding 3D coordinates.

4.4.3. DEEP FOREST ALGORITHM

DeepForest is a python package for training and predicting individual tree crowns from airborne RGB imagery. DeepForest comes with a prebuilt model trained on data from the National Ecological Observatory Network. Users can extend this model by annotating and training custom models.

The original project was written in tensorflow based on keras-retinanet. When tensorflow updated to 2.0 there were breaking changes and the authors of keras-retinanet decided to not attempt to recover the project. The rapid development of open machine learning community resources means that tensorflow 1.14.0, which is required for deepforest, will rapidly become out of date. Fear not, starting in 1.0.0, the model now depends on the pytorch and torchvision and will be stable for the foreseeable future.

DeepForest uses deep learning object detection networks to predict bounding boxes corresponding to individual trees in RGB imagery. DeepForest is built on the retinanet model from the torchvision package and designed to make training models for tree detection simpler.

Remote sensing can transform the speed, scale, and cost of biodiversity and forestry surveys. Data acquisition currently outpaces the ability to identify individual organisms in high resolution imagery. Individual crown delineation has been a long-standing challenge in remote sensing and available algorithms produce mixed results. DeepForest is the first open source implementation of a deep learning model for crown detection. Deep learning has made enormous strides in a range of computer vision tasks but requires significant amounts of training data. By including a trained model, we hope to simplify the process of retraining deep learning models for a range of forests, sensors, and spatial resolutions.

4.4.4. STEPS INVOLVED IN CODING

In order to compensate for the high computation and increasing processor heating condition, a simple and fast Object Recognition Technique code has been written to be implemented in the system.

4.4.4.1. TO CREATE A WINDOW AND DETECT POSE USING OPENCV

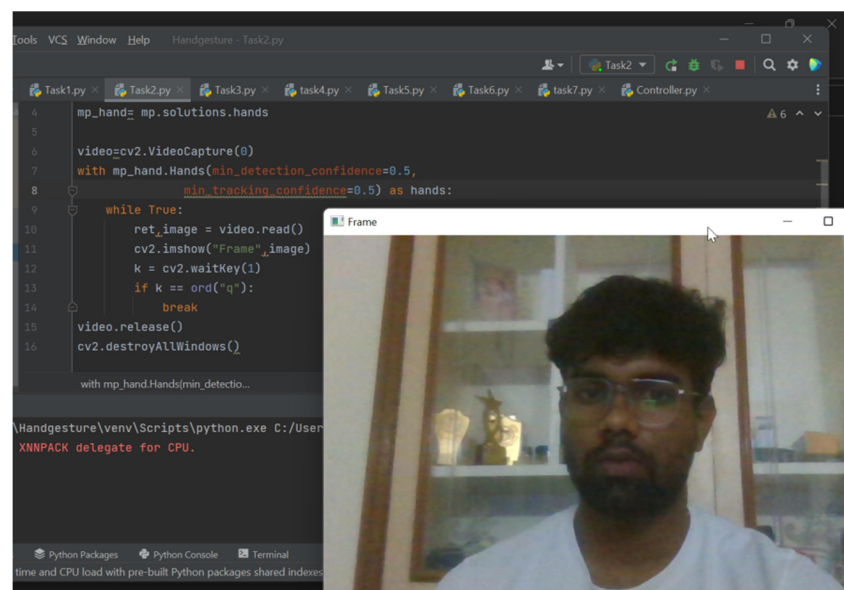
The following code is written in order to create a window to display video captured by the camera. Then inout is detected by the hand pose from the user.

CODE:

```
import cv2
import mediapipe as mp
mp_draw = mp.solutions.drawing_utils
mp_hand= mp.solutions.hands

video=cv2.VideoCapture(0)
with mp_hand.Hands(min_detection_confidence=0.5,
                   min_tracking_confidence=0.5) as hands:
    while True:
        ret,image = video.read()
        cv2.imshow("Frame",image)
        k = cv2.waitKey(1)
        if k == ord("q"):
            break
video.release()
cv2.destroyAllWindows()
```

OUTPUT:



4.4.4.2. CONVERT BGR TO RGB

Converting BGR and RGB with OpenCV function is achieved using the following code.

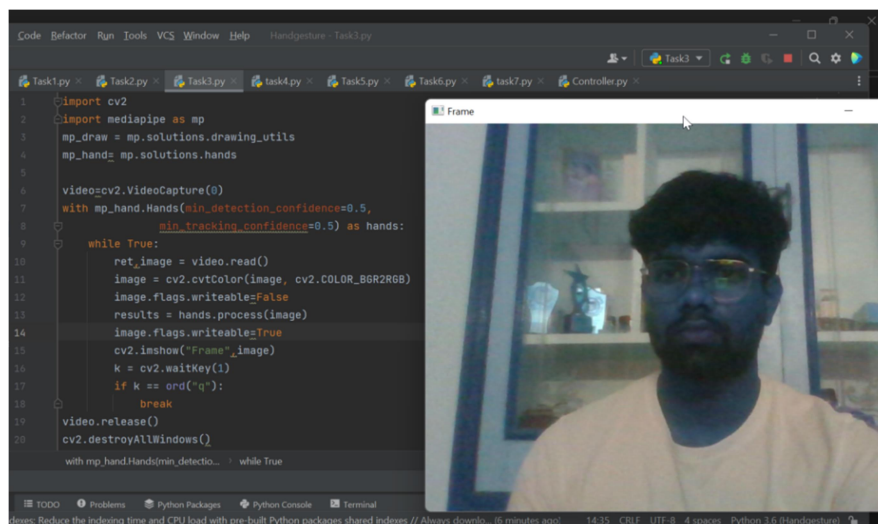
When reading a color image file, OpenCV `imread()` reads as a NumPy array `ndarray` of `row (height) x column (width) x color (3)`. The order of color is BGR (blue, green, red).

CODE:

```
import cv2
import mediapipe as mp
mp_draw = mp.solutions.drawing_utils
mp_hand= mp.solutions.hands

video=cv2.VideoCapture(0)
with mp_hand.Hands(min_detection_confidence=0.5,
                   min_tracking_confidence=0.5) as hands:
    while True:
        ret,image = video.read()
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image.flags.writeable=False
        results = hands.process(image)
        image.flags.writeable=True
        cv2.imshow("Frame",image)
        k = cv2.waitKey(1)
        if k == ord("q"):
            break
video.release()
cv2.destroyAllWindows()
```

OUTPUT:



4.4.4.3. DRAW LANDMARK ANNOTATIONS

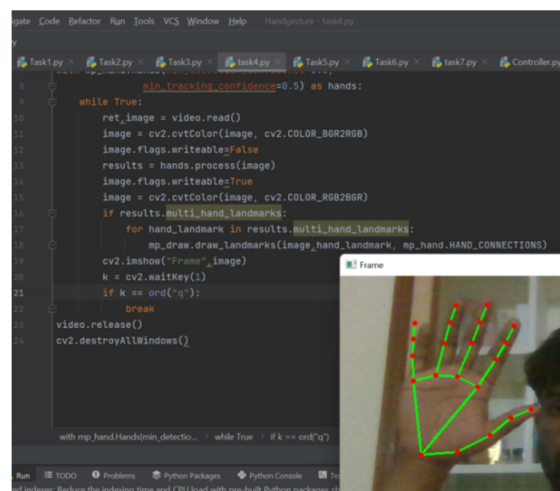
Drawing landmark annotations on the hand further eases understanding the pose shown by the user. The model learns a consistent internal hand pose representation.

CODE:

```
import cv2
import mediapipe as mp
mp_draw = mp.solutions.drawing_utils
mp_hand= mp.solutions.hands

video=cv2.VideoCapture(0)
with mp_hand.Hands(min_detection_confidence=0.5,
                   min_tracking_confidence=0.5) as hands:
    while True:
        ret,image = video.read()
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image.flags.writeable=False
        results = hands.process(image)
        image.flags.writeable=True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        if results.multi_hand_landmarks:
            for hand_landmark in results.multi_hand_landmarks:
                mp_draw.draw_landmarks(image,hand_landmark,
mp_hand.HAND_CONNECTIONS)
                cv2.imshow("Frame",image)
                k = cv2.waitKey(1)
                if k == ord("q"):
                    break
    video.release()
cv2.destroyAllWindows()
```

OUTPUT:



4.4.4.4. OPEN/CLOSE WHEN POSE IS DETECTED

The following code is written in order to control the opening and closing function of the relay module.

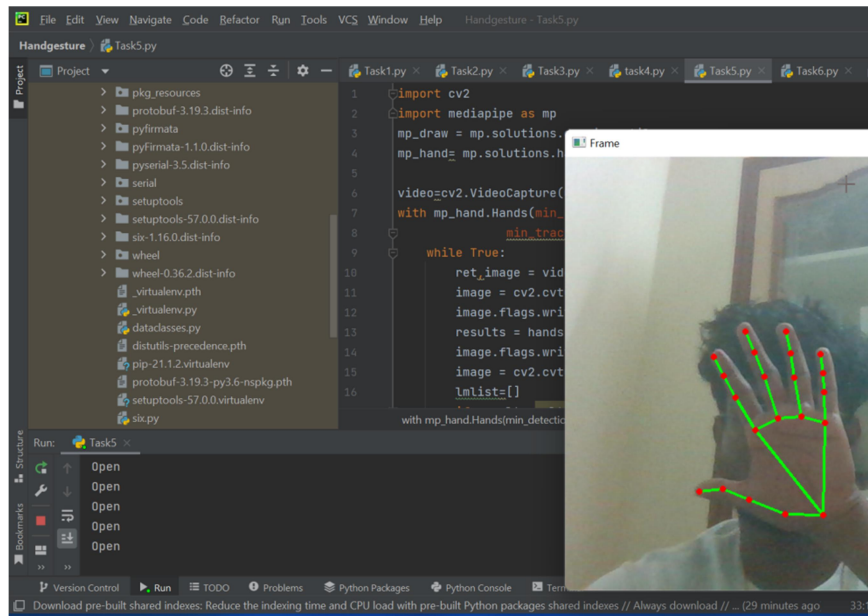
This allows the microcontroller to allow the switching operation of the appliances to be initiated. If a pose is detected, further operation is allowed i.e., tracking the number of fingers showed by the user-explained in the next subtopic.

CODE:

```
import cv2
import mediapipe as mp
mp_draw = mp.solutions.drawing_utils
mp_hand= mp.solutions.hands

video=cv2.VideoCapture(0)
with mp_hand.Hands(min_detection_confidence=0.5,
                   min_tracking_confidence=0.5) as hands:
    while True:
        ret,image = video.read()
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image.flags.writeable=False
        results = hands.process(image)
        image.flags.writeable=True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        lmlist=[]
        if results.multi_hand_landmarks:
            for hand_landmark in results.multi_hand_landmarks:
                myHands=results.multi_hand_landmarks[0]
                for id, lm in enumerate(myHands.landmark):
                    h,w,c = image.shape
                    cx,cy = int(lm.x*w), int(lm.y*h)
                    lmlist.append([id,cx,cy])
                mp_draw.draw_landmarks(image,hand_landmark,
mp_hand.HAND_CONNECTIONS)
            if len(lmlist) != 0:
                if lmlist[8][2] < lmlist[6][2]:
                    print("Open")
                else:
                    print("close")
            cv2.imshow("Frame",image)
            k = cv2.waitKey(1)
            if k == ord("q"):
                break
    video.release()
cv2.destroyAllWindows()
```

OUTPUT:



4.4.4.5 DETECT NUMBER OF FINGERS

Number of fingers in the recorded hand is determined by using their respective tip IDs. The tip IDs include : 4,8,12,16,20.

Using this tip ID data, the number posed using the hand can be understood by the system

CODE:

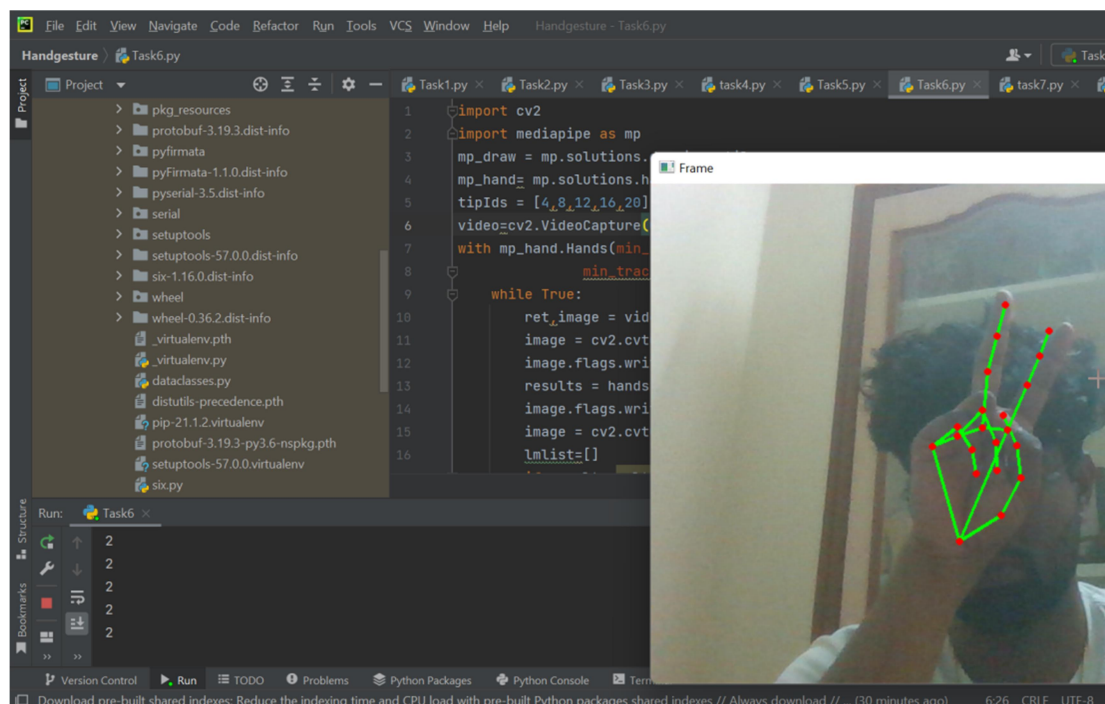
```
import cv2
import mediapipe as mp
mp_draw = mp.solutions.drawing_utils
mp_hand= mp.solutions.hands
tipIds = [4,8,12,16,20]
video=cv2.VideoCapture(0)
with mp_hand.Hands(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as hands:
    while True:
        ret,image = video.read()
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image.flags.writeable=False
        results = hands.process(image)
        image.flags.writeable=True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        lmlist=[]
        if results.multi_hand_landmarks:
            for hand_landmark in results.multi_hand_landmarks:
```

```

        myHands=results.multi_hand_landmarks[0]
        for id, lm in enumerate(myHands.landmark):
            h,w,c = image.shape
            cx,cy = int(lm.x*w), int(lm.y*h)
            lmlist.append([id,cx,cy])
        mp_draw.draw_landmarks(image,hand_landmark,
mp_hand.HAND_CONNECTIONS)
        fingers=[]
        if len(lmlist) != 0:
            if lmlist[tipIds[0]][1] > lmlist[tipIds[0]-1][1]:
                fingers.append(1)
            else:
                fingers.append(0)
            for id in range(1,5):
                if lmlist[tipIds[id]][2] < lmlist[tipIds[id]-
2][2]:
                    fingers.append(1)
                else:
                    fingers.append(0)
            total = fingers.count(1)
            print(total)
            cv2.imshow("Frame",image)
            k = cv2.waitKey(1)
            if k == ord("q"):
                break
        video.release()
        cv2.destroyAllWindows()

```

OUTPUT:



4.4.4.6. CONTROLLER PROGRAM

```
import pyfirmata
import time
comport = 'COM6'
board = pyfirmata.Arduino(comport)
led_1=board.get_pin('d:7:o')
led_2=board.get_pin('d:6:o')
led_3=board.get_pin('d:5:o')
led_4=board.get_pin('d:4:o')

def led(total):
    if total == 0:
        led_1.write(0)
        led_2.write(0)
        led_3.write(0)
        led_4.write(0)
    elif total == 1:
        led_1.write(1)
        led_2.write(0)
        led_3.write(0)
        led_4.write(0)
    elif total == 2:
        led_1.write(0)
        led_2.write(1)
        led_3.write(0)
        led_4.write(0)
    elif total == 3:
        led_1.write(0)
        led_2.write(0)
        led_3.write(1)
        led_4.write(0)
    elif total == 4:
        led_1.write(0)
        led_2.write(0)
        led_3.write(0)
        led_4.write(1)
    elif total == 5:
        led_1.write(1)
        led_2.write(1)
        led_3.write(1)
        led_4.write(1)
    else:
        led_1.write(0)
        led_2.write(0)
        led_3.write(0)
        led_4.write(0)
```

4.4.4.7. IMPLEMENTING CODE

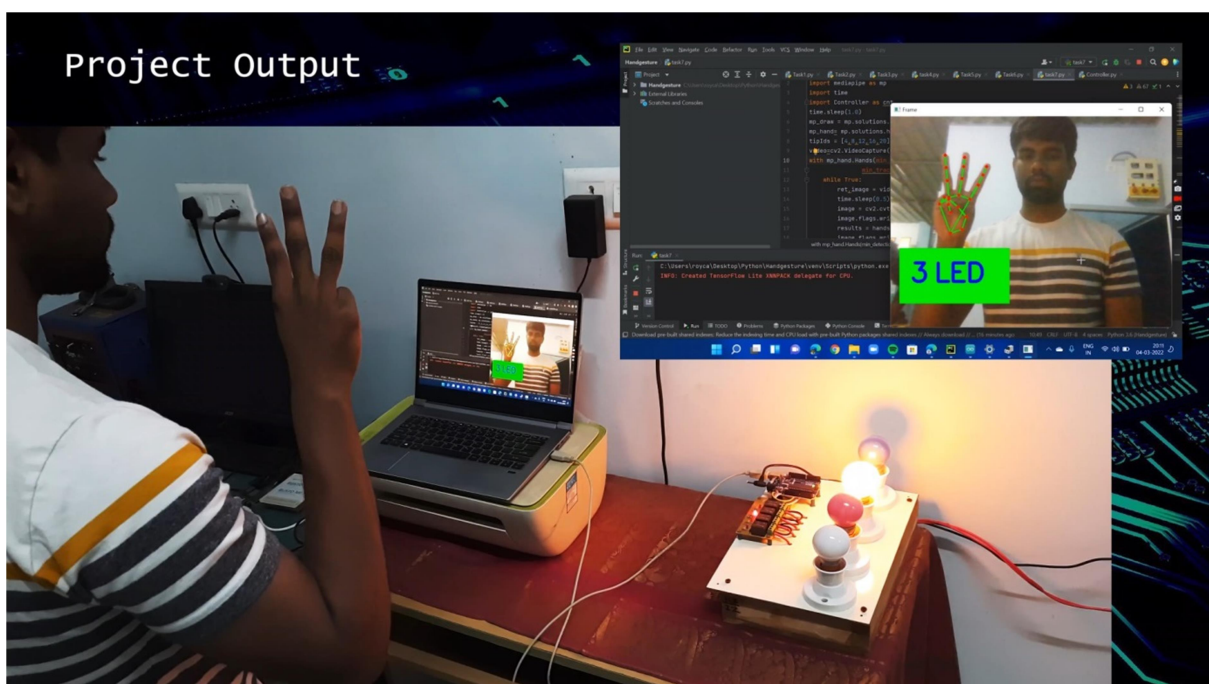
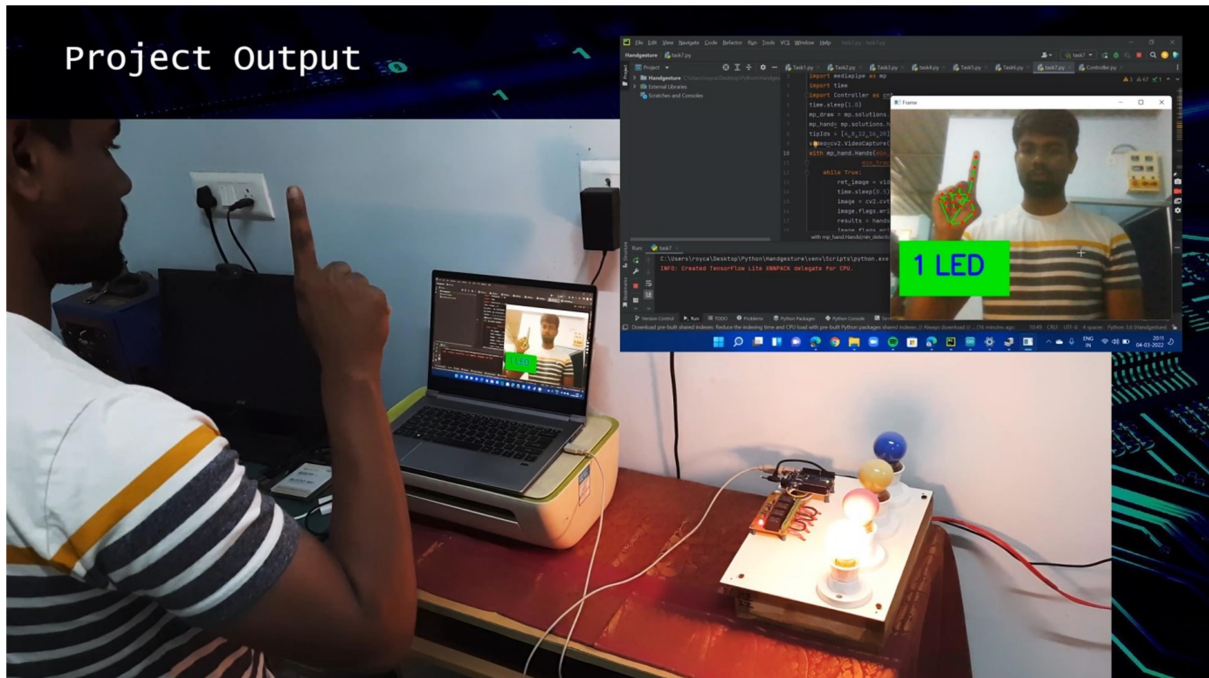
```
import cv2
import mediapipe as mp
import time
import Controller as cnt
time.sleep(1.0)
mp_draw = mp.solutions.drawing_utils
mp_hand= mp.solutions.hands
tipIds = [4,8,12,16,20]
video=cv2.VideoCapture(0)
with mp_hand.Hands(min_detection_confidence=0.5,
                   min_tracking_confidence=0.5) as hands:
    while True:
        ret,image = video.read()
        time.sleep(0.5)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image.flags.writeable=False
        results = hands.process(image)
        image.flags.writeable=True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
        lmlist=[]
        if results.multi_hand_landmarks:
            for hand_landmark in results.multi_hand_landmarks:
                myHands=results.multi_hand_landmarks[0]
                for id, lm in enumerate(myHands.landmark):
                    h,w,c = image.shape
                    cx,cy = int(lm.x*w), int(lm.y*h)
                    lmlist.append([id,cx,cy])
                mp_draw.draw_landmarks(image,hand_landmark,
mp_hand.HAND_CONNECTIONS)
                fingers=[]
                if len(lmlist) != 0:
                    if lmlist[tipIds[0]][1] > lmlist[tipIds[0]-1][1]:
                        fingers.append(1)
                    else:
                        fingers.append(0)
                    for id in range(1,5):
                        if lmlist[tipIds[id]][2] < lmlist[tipIds[id]-
2][2]:
                            fingers.append(1)
                        else:
                            fingers.append(0)
                    total = fingers.count(1)
                    cnt.led(total)
                    if total == 0:
                        cv2.rectangle(image, (20,300), (270,425),
(0,255,0), cv2.FILLED)
                        cv2.putText(image, "0", (45,375),
```

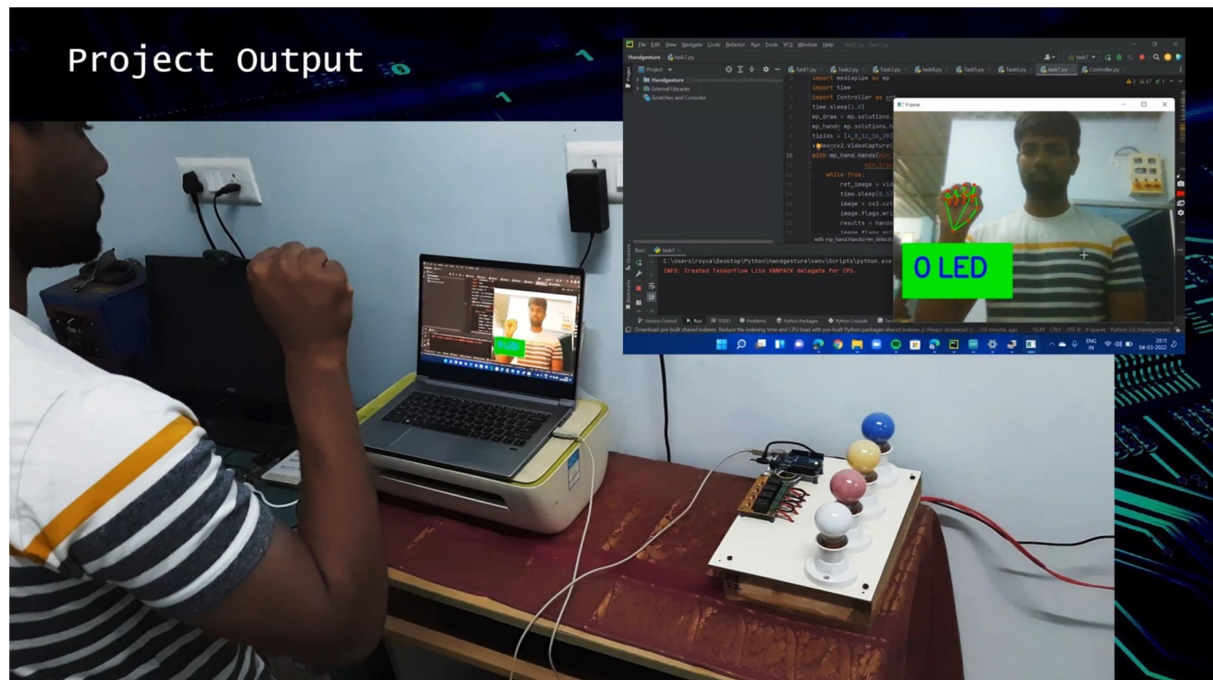
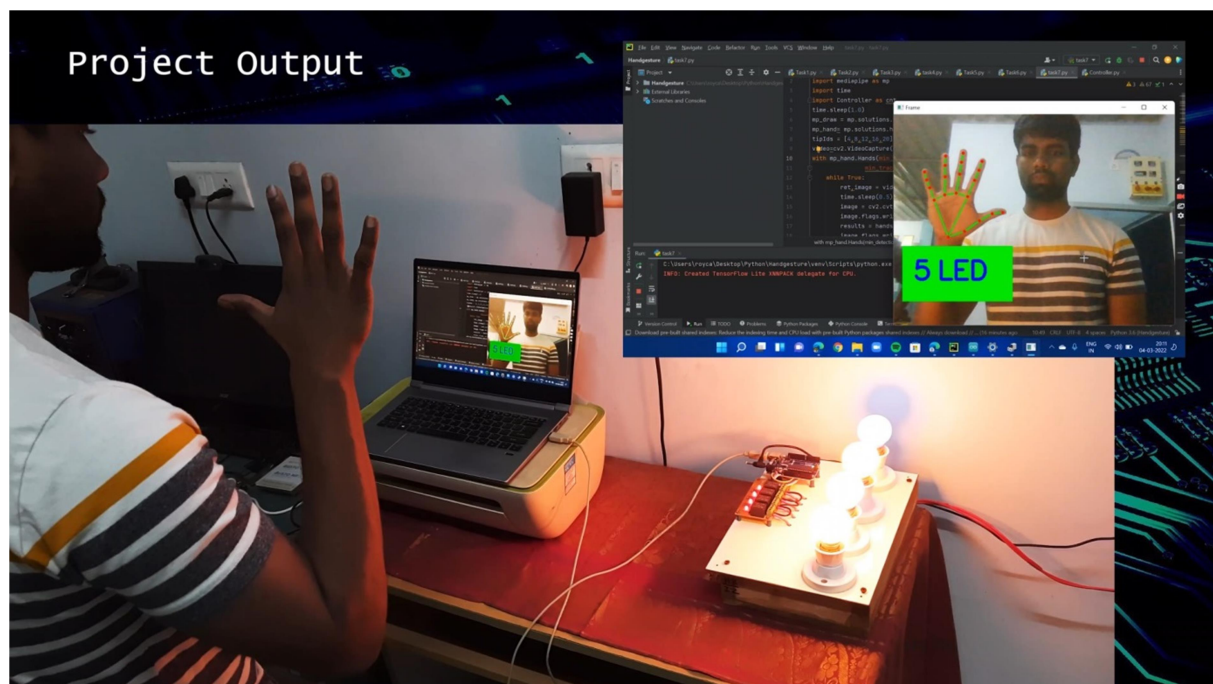
```

cv2.FONT_HERSHEY_SIMPLEX, 2, (255,0,0), 5)
    cv2.putText(image, "LED", (100, 375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 5)
    elif total == 1:
        cv2.rectangle(image, (20,300), (270,425),
(0,255,0), cv2.FILLED)
        cv2.putText(image, "1", (45,375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255,0,0), 5)
        cv2.putText(image, "LED", (100, 375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 5)
        elif total == 2:
            cv2.rectangle(image, (20,300), (270,425),
(0,255,0), cv2.FILLED)
            cv2.putText(image, "2", (45,375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255,0,0), 5)
            cv2.putText(image, "LED", (100, 375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 5)
            elif total == 3:
                cv2.rectangle(image, (20,300), (270,425),
(0,255,0), cv2.FILLED)
                cv2.putText(image, "3", (45,375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255,0,0), 5)
                cv2.putText(image, "LED", (100, 375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 5)
                elif total == 4:
                    cv2.rectangle(image, (20,300), (270,425),
(0,255,0), cv2.FILLED)
                    cv2.putText(image, "4", (45,375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255,0,0), 5)
                    cv2.putText(image, "LED", (100, 375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 5)
                    elif total == 5:
                        cv2.rectangle(image, (20,300), (270,425),
(0,255,0), cv2.FILLED)
                        cv2.putText(image, "5", (45,375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255,0,0), 5)
                        cv2.putText(image, "LED", (100, 375),
cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 5)
                        cv2.imshow("Frame",image)
                        k = cv2.waitKey(1)
                        if k == ord("q"):
                            break
video.release()
cv2.destroyAllWindows()

```

OUTPUT SCREENSHOTS:





CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1. CONCLUSION

An AI based automation system using mesh detection has been designed and implemented using Deep Forest Algorithm to process images obtained from webcam. Image has been processed using Mesh algorithm to extract gestures using OpenCV. Switching operation of appliances with the help of AVR AT mega 328P by using PyFirmata Protocol has been implemented successfully.

Further research and innovation in the field of gesture based home automation can greatly improve user interaction between devices, also providing easier access to people with disabilities.

5.2. FUTURE WORKS

The next trend in home automation may not require you to utter a word or touch a thing. While still in development, gesture-based technologies are emerging on the market. Collectively, we are more wary of touching communal surfaces and accustomed to sharing work/live/play spaces with our family or pod; a silent, non-tactile command could be beneficial to prevent disturbing work, sleep, or even meditation as well as prevent germ-spreading.

A RaspberryPI module can be used instead of Arduino UNO to interface a camera with microcontroller and avoid the usage of a laptop/pc for the usage of camera.

To replace or aid automated systems in the future. The future improvement will be founded on AI. Other than machines, these gesture based systems can be utilized for speed control, volume and channel tuning, and so forth. In future, these systems can be utilized for controlling even autos or programming applications. By the future headway advancements signals can be utilized to control automobiles and even programming applications. It can give brilliant pictures, superior, high exactness and high solid approach to control the gadgets.

REFERENCES

- [1]<https://fuel4media.com/blog/wp-content/uploads/2019/12/AI.jpg>
- [2]<https://www.semanticscholar.org/paper/Using-gestures-to-interact-with-home-automation-Villanueva/01b2146eb42251bc4d092d284a8cf4bb9453fdaa/figure/2>
- [3] *Kishor P Jadhav, Santosh G Bari*- **“Hand Gesture Based Switching Using MATLAB”**, 2016
- [4] *A. Hayder, J. Dargham, A. Chekima, J. kela, P. Korpiaa*- **“Gait Recognition using Radon Transform with Principal Component Analysis”**, 2010
- [5] *Archana S. Ghotkar, Rucha Khatal, Sanjana Khupase, Surbhi Asati, Mithila Hadap*- **“Hand gesture recognition for Indian Sign Language”**, 2012
- [6] *Konstantinos G. Derpanis, Richard P. Wildes, and John K. Tsotsos*- **“Hand Gesture Recognition within a Linguistics-Based Framework”**, 2004
- [7] *Bhuiyam M, Picking R*- **“A Gesture Controlled User Interface for Inclusive Design and Evaluative Study of Its Usability”**, 2011
- [8] *Pragati Garg, Naveen Aggarwal, Sanjeev Sofat Garg P*- **“Vision Based Hand Gesture Recognition”**, 2009
- [9] *Kruti Goyal, Kartikey Agarwal, Rishi Kumar*- **“Face Detection and Tracking Using OpenCV”**, 2017
- [10] *Munir Oudah, Ali AL-Naji, Javaan Chahi*- **“Hand Gesture Recognition Based on Computer Vision: A Review of Techniques”** , 2020
- [11] *Shivasankara S, Srinath S*- **“American Sign Language Recognition System: An Optimal Approach”**, 2018
- [12] *Omkar Vedak, Prasad Zavre, Abhijeet Todkar, Manoj Patil*- **“Sign language Interpreter using Image Processing and Machine Learning”**, 2020
- [13] *Hans-Christoph Steiner*- **“Firmata: Towards making microcontrollers act like extensions of the computer”**, 2009
- [14]https://i0.wp.com/www.elecrom.com/wp-content/uploads/2017/01/intro-to-arduino-Arduino_Uno_006.jpg?resize=300%2C197&ssl=1
- [15]https://www.etechnophiles.com/wp-content/uploads/2020/11/test_3-1.png?ezimgfmt=ng:webp/ngcb40
- [16]<https://www.flyrobo.in/image/cache/catalog/product/5v-Four-Channel-Relay-Module/5v-4-channel-relay-module-550x550.jpg>

- [17]https://components101.com/sites/default/files/component_pin/Four-Channel-Relay-Module-Pinout.jpg
- [18]<https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2016/11/relay-labeled.jpg?resize=372%2C342&quality=100&strip=all&ssl=1>
- [19]<https://components101.com/sites/default/files/inline-images/Four-Channel-Relay-Module-Circuit-Diagram.png>
- [20]https://upload.wikimedia.org/wikipedia/commons/thumb/0/02/Optoisolator_Pinout.svg/1200px-Optoisolator_Pinout.svg.png
- [21]https://img.electronicdesign.com/files/base/ebm/electronicdesign/image/2013/10/electronicdesign_com_sites_electronicdesign.com_files_uploads_2013_09_fig_7.png?auto=format,compress&w=1300&h=730&fit=max
- [22]https://static.wixstatic.com/media/1a5536_9e9736811cd740c8b362c9ddb58e2e60~mv2.gif
- [23]<https://lightbulbsocket.com/wp-content/uploads/2017/10/wire-E27-lamp-holder.jpg>
- [24]https://static.turbosquid.com/Preview/2014/05/26_00_46_24/002.jpgeb23abdd-571d-4281-8ce7-899a599024d6Larger.jpg
- [25]https://google.github.io/mediapipe/images/mobile/hand_landmarks.png