Progress Report

Automation using Hand Gestures

Attendance Details

Mentor :-

Dr. Karanveer Dhingra

Team Members -

Abhay Kumar

Ansh Gupta

Aryan

Dikshant Jindal

Shobhit Gupta

Dr B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR
Department of Electrical Engineering

Attendance/Meeting Sheet Format for Minor Project

Minor Project Title: Home Automation using Hand Gestures Number of Students in Group: 5

Meeting S.No	Roll Numbers Present	Roll Number Absent	Date	Signature of Mentor
1	20112003 20126013, 15, 23, 53		09/02/2023	Karanveer Dhingra
2	20112003 20126013, 15, 23, 53		22/02/23	Karanveer Dhingra
3	20112003 20126013, 15, 23, 53		15/03/23	Karanveer Dhingra
4	20112003 20126013, 15, 23, 53		12/04/23	Karanveer Dhingra
5	20112003 20126013, 15, 23, 53		10/05/23	Karanveer Dhingra
6				
7				

Tech Stack Used

Arduino

Mediapipe

OpenCV

Python

Code 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
```

Out[1]:

data import
containing hand landmarks
preprocessed coordinates with
defined classes

In [2]:
```python
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]
```

knn model train

```python
import serial

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

serial port library
import and intialize

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

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

other useful libraries

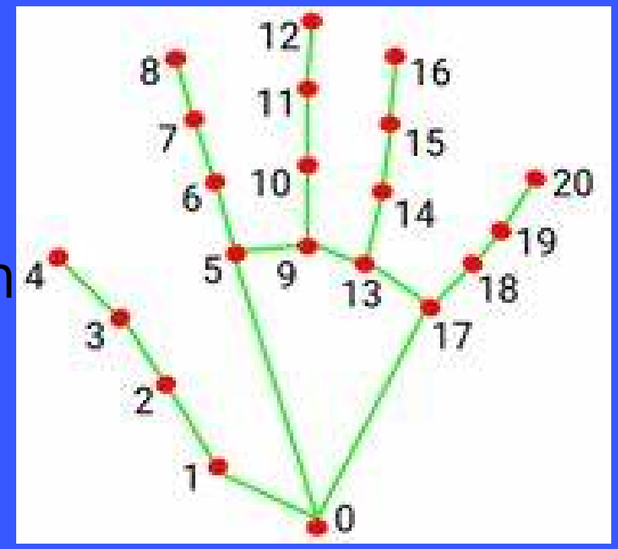
```python
def calc_landmark_list(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]

    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
```

actual coordinate calculation

In [8]:
```python
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)))
```

preprocessing coordinates for optimizing solution

Representation of Pre-processing



coordinates breakdown

Actual X, Y coordinates

0	1	2	3	4	….	…	…	…	20
[45,65]	[50,65]	[64,88]	[99,45]	[86,33]	….	…	…	….	[150,250]

Relative X, Y coordinates

0	1	2	3	4	….	…	…	…	20
[0,0]	[5,0]	[19,13]	[44, -20]	[41, -23]	….	…	…	….	[105,185]

1D array

0	1	2	3	4	….	…	…	40	41
0	0	5	0	19	….	…	…	105	185

1D normalize array

0	1	2	3	4	….	…	…	40	41
0/max	0/max	5/max	0/max	19/max	….	…	…	105/max	185/max

importing hand gesture solutions and utilities

```
    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_
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
```

storing frames from camera and then applying solution on it

```
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
```

defining state for ON and OFF of devices using condtional statements.

```
            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)
```

marking the state in real time

degree measurement between index finger and thumb for intensity control

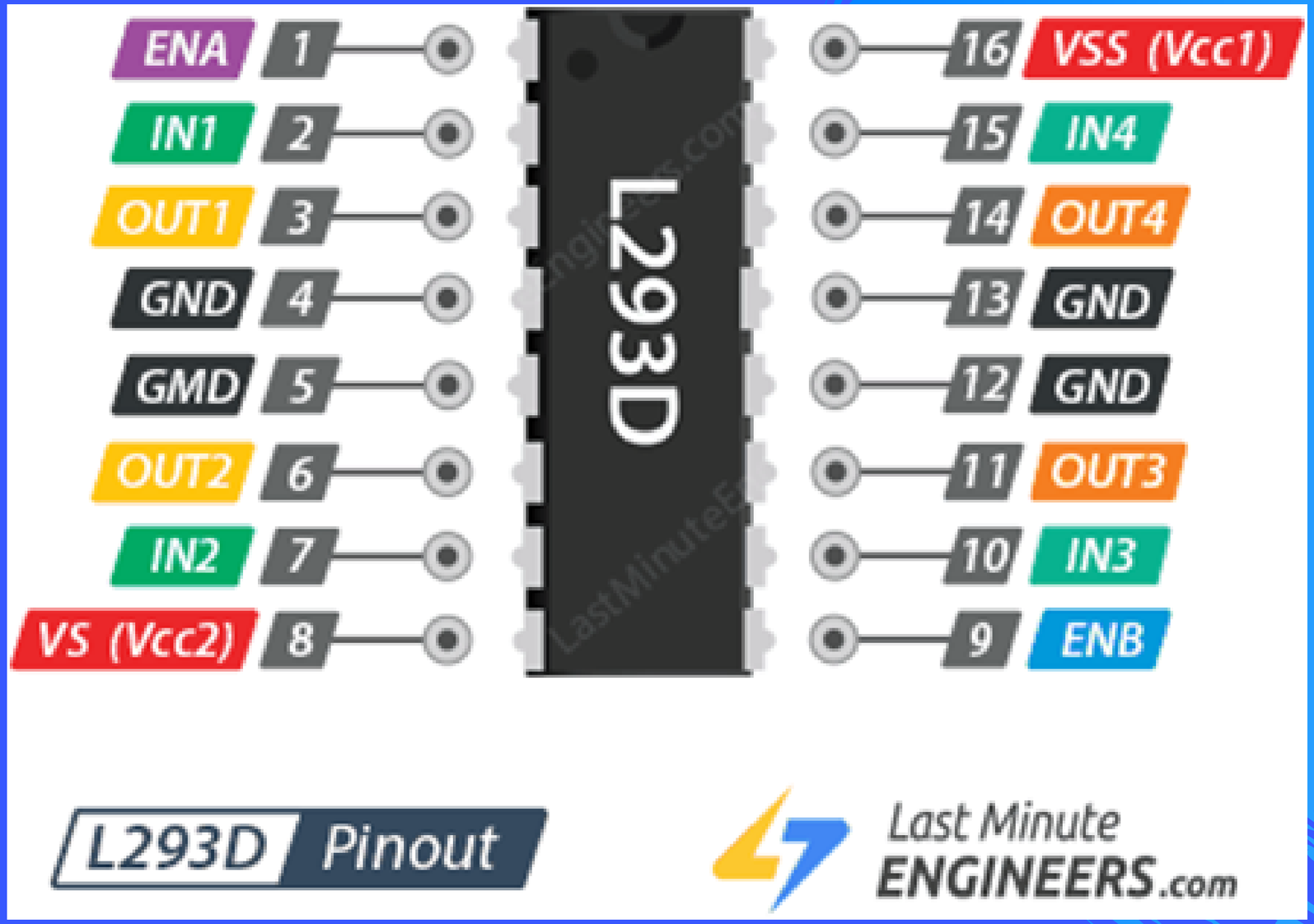
```
#print(angle_deg)
# max_angle = 45 min_anhle = 8
speed = np.interp(angle_deg,[8,45],[0,255])
#volume.SetMasterVolumeLevel(vol, None)
```

```
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)
```

collecting the values and encoding in bits which then sent to Arduino.

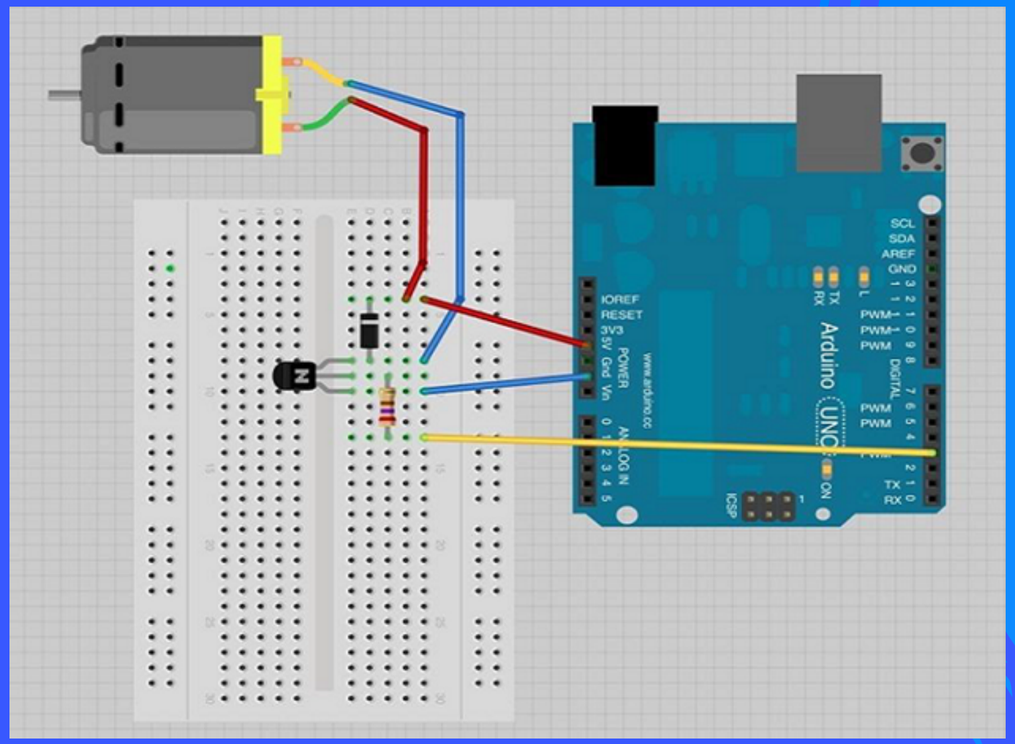
```
            cv2.rectangle(image,(100,450),(500,470),(0,255,255),2)
            bar = np.interp(angle_deg,[8,45],[100,500])
            cv2.rectangle(image,(100,450),(int(bar),470),(0,255,255),-1)

        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()
```

H-bridge


ENA 1
IN1 2
OUT1 3
GND 4
GMD 5
OUT2 6
IN2 7
VS (Vcc2) 8
L293D
16 VSS (Vcc1)
15 IN4
14 OUT4
13 GND
12 GND
11 OUT3
10 IN3
9 ENB
L293D Pinout
Last Minute
ENGINEERS.com

Hardware Part

Driver circuit



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.
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 system's power supply.

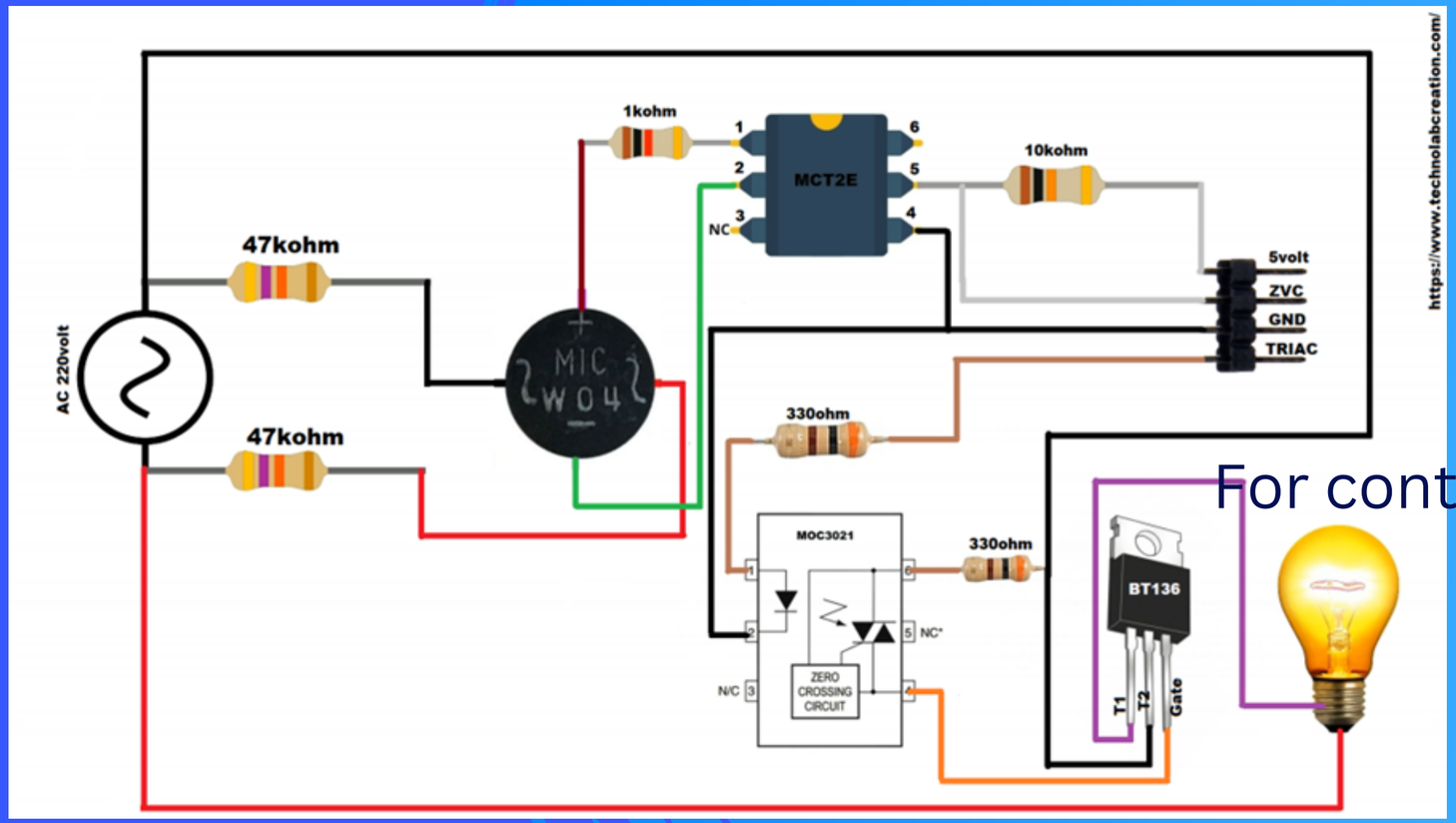
The IC has two direction control pins for each channel. The *IN1* and *IN2* pins control the spinning direction of motor A; While *IN3* and *IN4* control the spinning direction of motor B.

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.

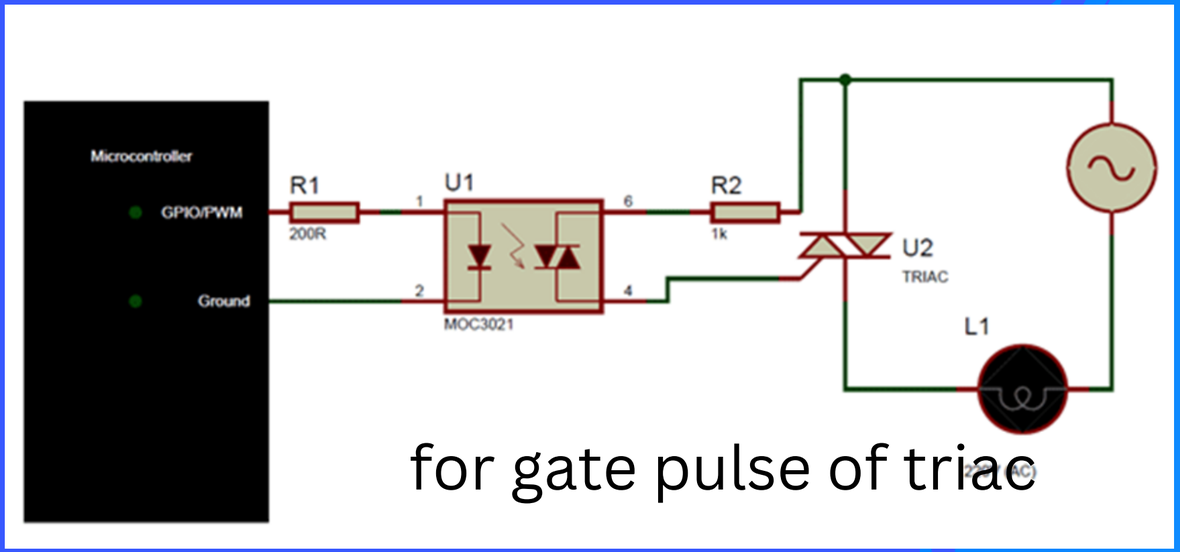
IN1	IN2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

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

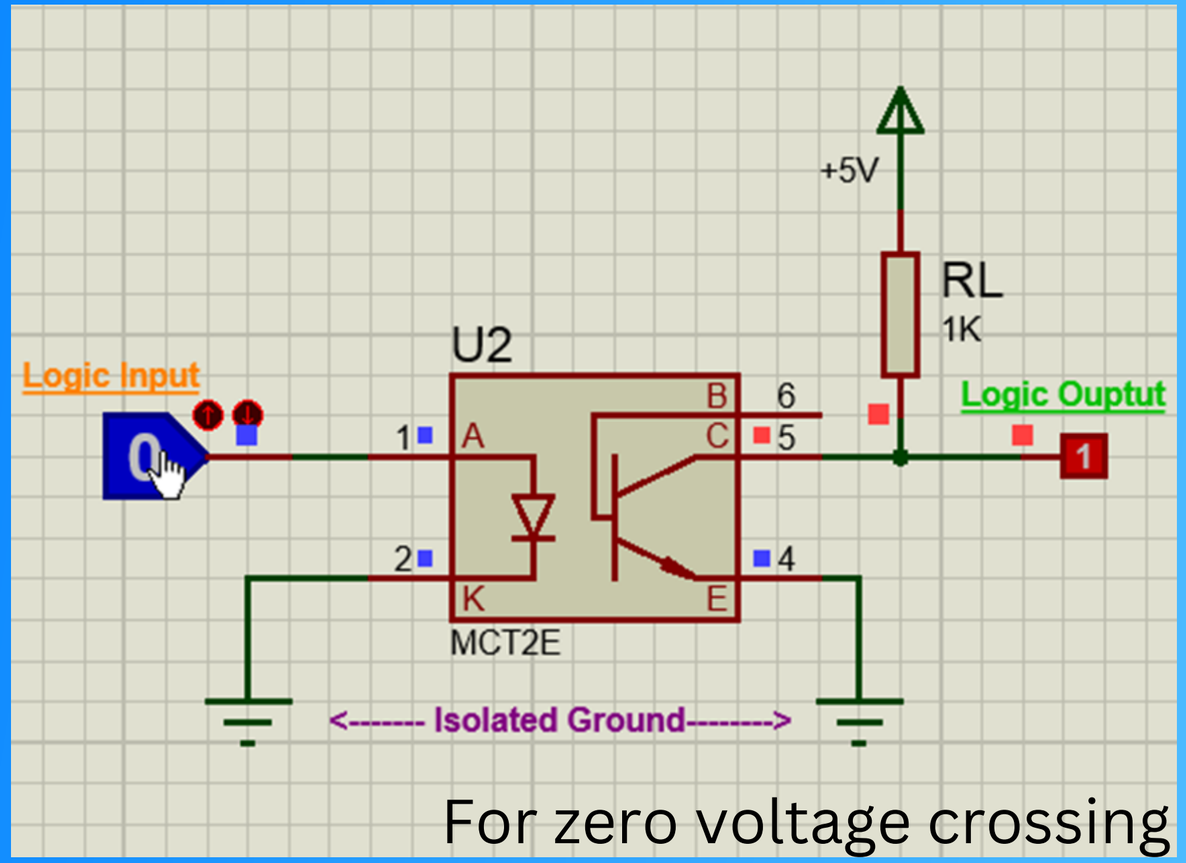
AC Dimmer Circuit



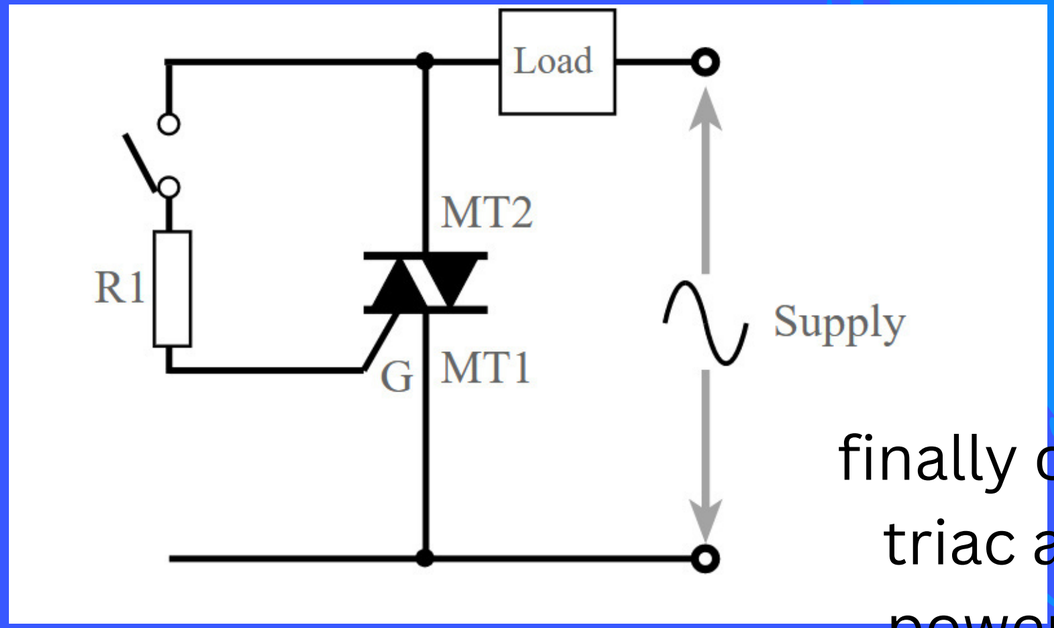
For controlling ac load



for gate pulse of triac



For zero voltage crossing



finally output from triac as variable power intensity

Code for Arduino

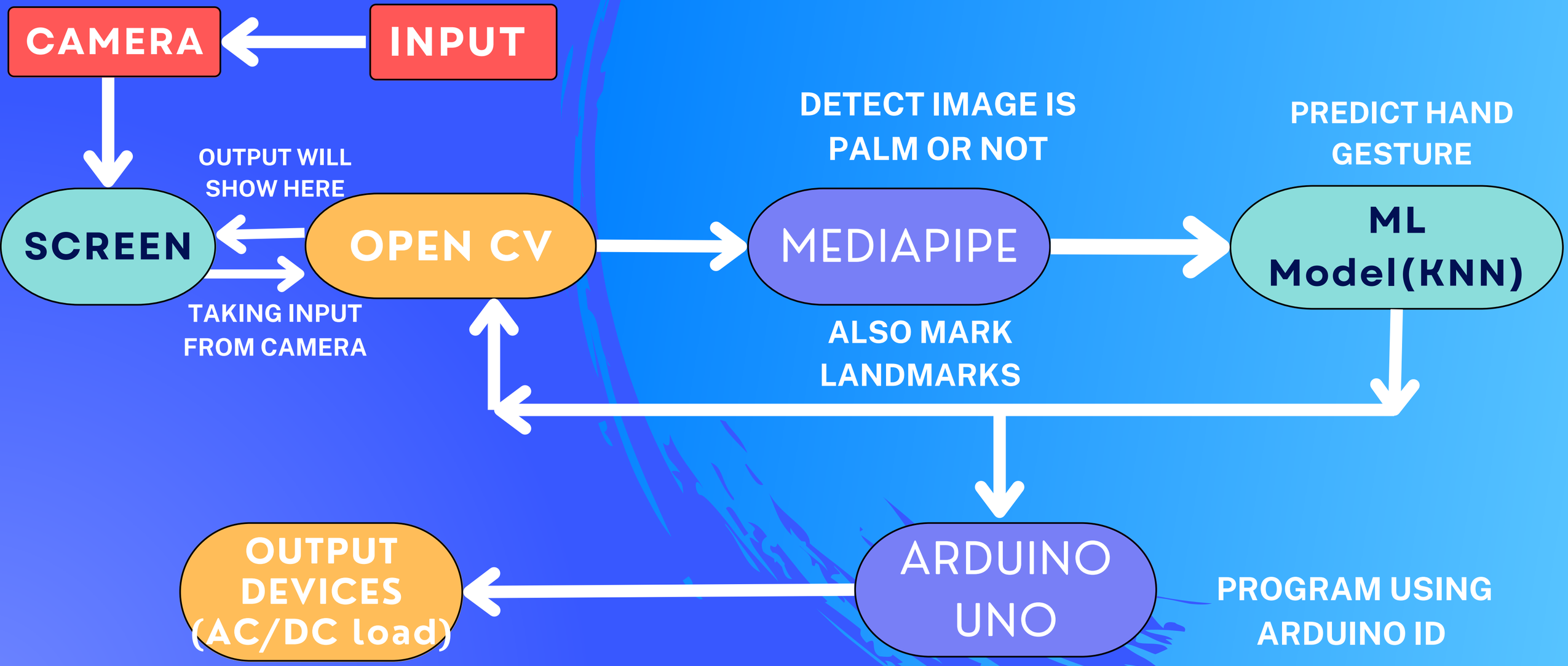
```
#include <HardwareSerial.h>
#include <String.h>
#include <RBDdimmer.h>//https://github.com/RobotDynOfficial/RBDDimmer
//Parameters
const int zeroCrossPin  = 2;
const int acdPin  = 3;

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);

}

```

defining and intializing pins

```
25  void loop() {
26    //put your main code here, to run repeatedly:
27    if (Serial.available()){
28      // char command = Serial.read();
29      // if(command == 'A'){
30      // digitalWrite(a,HIGH);}
31      // else if(command=='B')
32      // digitalWrite(a,LOW);
33      String inputString = Serial.readStringUntil('\n');
34      // int number1 = inputString.substring(0, inputString.indexOf(',')).toInt();
35      // int number2 = inputString.substring(inputString.indexOf(',') + 1).toInt();
36      // int number3 = inputString.substring(inputString.indexOf(',') + 2).toInt();
37      int number1 = inputString.substring(0, inputString.indexOf(',')).toInt();
38      inputString.remove(0, inputString.indexOf(',') + 1);
39      int number2 = inputString.substring(0, inputString.indexOf(',')).toInt();
40      inputString.remove(0, inputString.indexOf(',') + 1);
41      int number3 = inputString.toInt();
42      //int num = Serial.parseInt();
43      if(number2==0){
44        digitalWrite(a,HIGH);
45        digitalWrite(c,LOW);
46      }else{
47        digitalWrite(a,LOW);
48        digitalWrite(c,HIGH);
49      }
50      analogWrite(b,number1);
51      //int val = map(number1,0,255,30,70);
52      if(number3==1){
53        digitalWrite(acdPin,HIGH);
54      }
55      else{
56        digitalWrite(acdPin,LOW);
57      }
```

applying conditions to control output devies

FLOWCHART


CAMERA
INPUT
SCREEN
OUTPUT WILL SHOW HERE
OPEN CV
TAKING INPUT FROM CAMERA
DETECT IMAGE IS PALM OR NOT
MEDIAPIPE
ALSO MARK LANDMARKS
PREDICT HAND GESTURE
ML Model(KNN)
OUTPUT DEVICES (AC/DC load)
ARDUINO UNO
PROGRAM USING ARDUINO ID

DEMOSTRATION