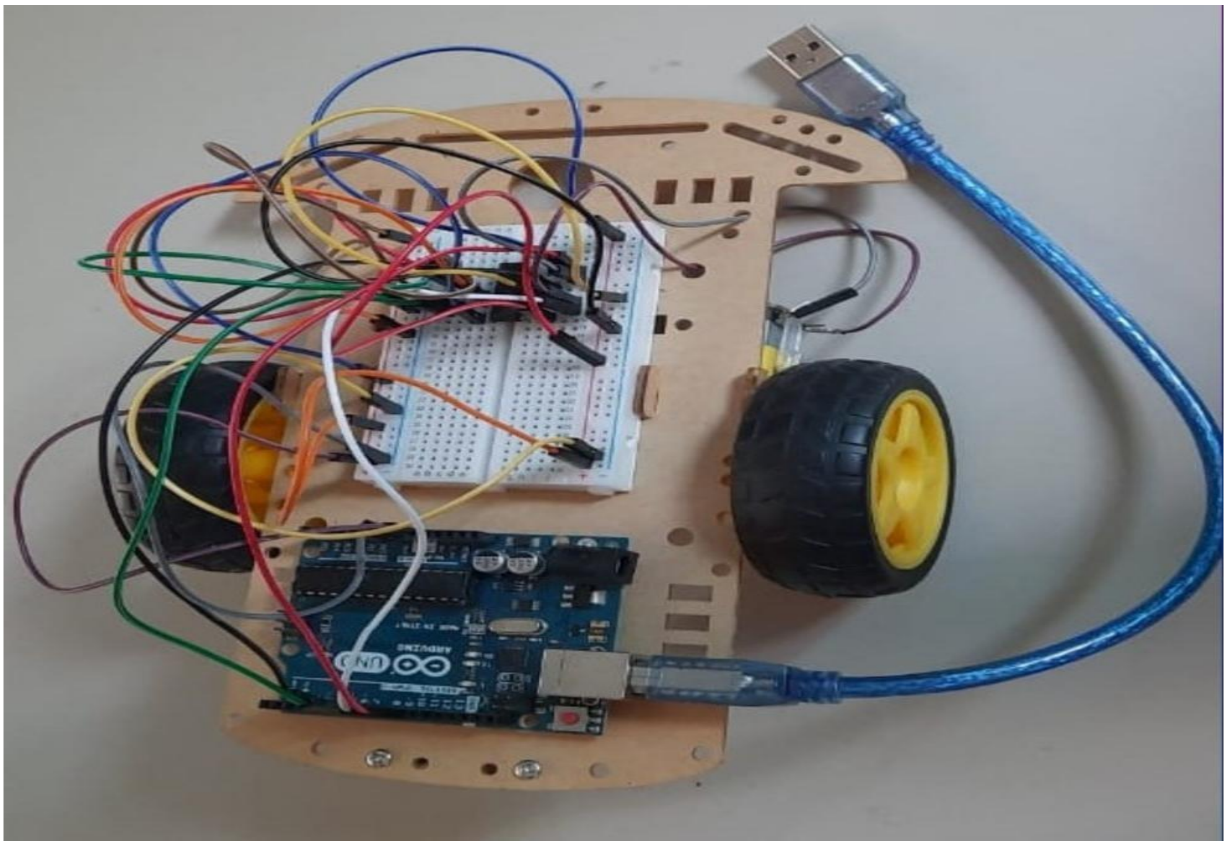
GESTURE CONTROLLED ROBOT USING IMAGE PROCESSING (OpenCV)



This project is a real time monitoring system by which humans interact with robots through gestures. While earlier works on this theme have focused primarily on issues such as manipulation and navigation in the environment, few robotic systems are used with user friendly interfaces that possess the ability to control the robot by natural means.

Gesture recognition consists of **three** stages: **capturing of image, image processing and navigation of bot**. By giving commands to a robot, the user can control or navigate the robot using gestures of his/her palm, thereby interacting with the robotic system. Using Image processing, the command signals which are generated from these gestures are then passed to the robot to navigate it in the specified direction.

Step 1: Things required

To build this, we will need

- Chassis Kit
- Battery and its cap, switch
- Motor Driver IC L293D
- Breadboard
- Microcontroller Arduino Atmega328P
- Jumper wires
- USB cable to establish serial communication

Additionally, we'll will need some tools like screw driver and spanner. All these components are easily available at any local store.

Step 2: Components

9V Battery

A battery is one that converts chemical energy to electrical energy by a chemical reaction. Usually, the chemicals are kept inside the battery. It is used in a circuit to power the other components.

Battery Cap

This serves two purposes. They permit the checking and maintenance of water and acid levels. Provides a vent for the escape of gases formed when the battery is charging.

Switch

It is a device to make or break the connection in an electric circuit.

Breadboard

It is the construction base for prototyping of electrons. Used to build and test circuits quickly before finalizing any circuit design.

Jumper Wires

A Jumper wire is an electrical wire or any group of them in a cable with a connector or pin at each end which is normally used to interconnect the components of a breadboard/other prototype/test circuit, internally / with other equipment/ components without soldering.

Chassis Kit

This is the two-wheel smart car robot chassis DIY kit. It comes with the 2 pairs of BO Motors and Wheels. All the products included in this car kit are quality products. The chassis used in this kit is transparent so as to create dynamic handling of the components mounted on our robotic vehicle.

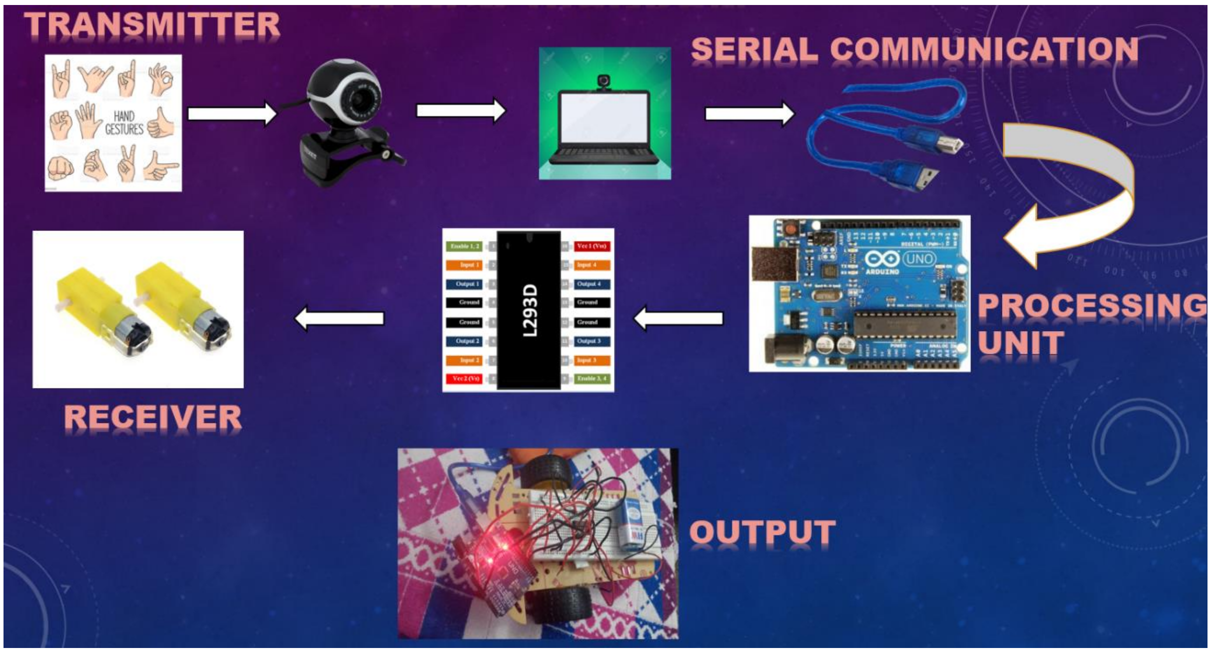
Motor Driver IC L293D

The L293D is a popular 16-Pin **Motor Driver IC**. As the name suggests it is mainly used to drive motors. A single **L293D IC** is capable of running two DC motors at the same time; also, the direction of these two motors can be controlled independently.

Arduino

Arduino is an open-source hardware and software company that designs and manufactures single-board microcontroller and microprocessor kits for building digital devices. Arduino UNO is an open-source microcontroller board on microchip ATmega328P microcontroller. Arduino projects can be stand-alone or used to communicate with software running on a computer.

Step 3: Work Flow



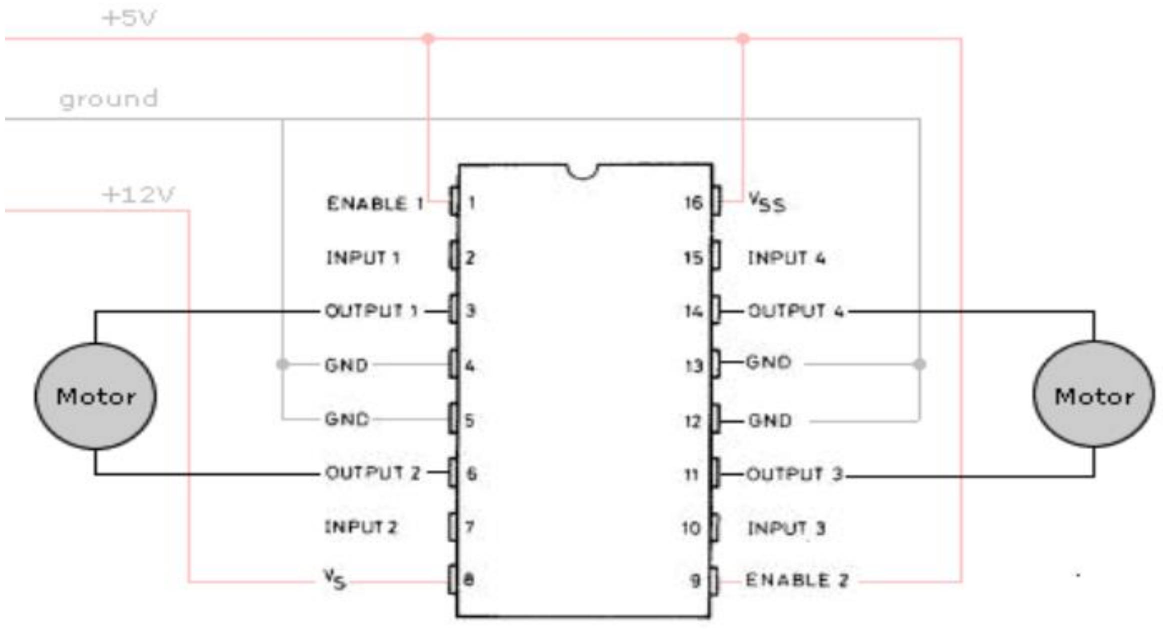
The whole project is divided into two sections, transmitter and receiver sections and in between a processing unit is present where required code is performed to arrive at the output.

In the transmitter side (our laptop), which captures gesture (hand as gesture) through webcam. Then, the gesture is captured which detects the hand palm. This detected portion is instructed by various commands using image processing techniques and then it is communicated with the bot through serial communication. And the microcontroller Arduino processes the information and passes it to L293D motor driver IC. From the motor driver IC, the digital signal moves to the motors of the bot which finally navigates according to our commands.

Step 4: Mechanical Assembly

Since our chassis is ready made, we don't have much work in assembling the robot. It has provisions for everything.

Placing all the motors in their respective places gets the job done.



Step 5: Circuit Connection

Below are the connections to be made using Arduino, L293D motor IC and breadboard :

- IC PINS : 1,8,9,16 → positive of breadboard
- IC PIN 2 → 5th pin of Arduino
- IC PIN 3 → Right motor (top)
- IC PINS 4,5 → negative of breadboard
- IC PIN 6 → Right wheel (bottom)
- IC PIN 7 → 6th pin of Arduino
- IC PIN 10 → 9th pin of Arduino
- IC PIN 11 → left wheel (top)
- IC PINS 12,13 → negative of breadboard
- IC PIN 14 → Left wheel (bottom)
- IC PIN 15 → 10th pin of Arduino
- Arduino 5V pin → positive of breadboard
- Arduino ground → negative of breadboard

The negative and positive strips of the breadboard on the top and bottom also need to be interconnected using jumper wires to establish closed connection.

If connection to the motor is altered, i.e., if the positive and negative terminals are interchanged then the movement of robot also changes accordingly.

ARDUINO CODE

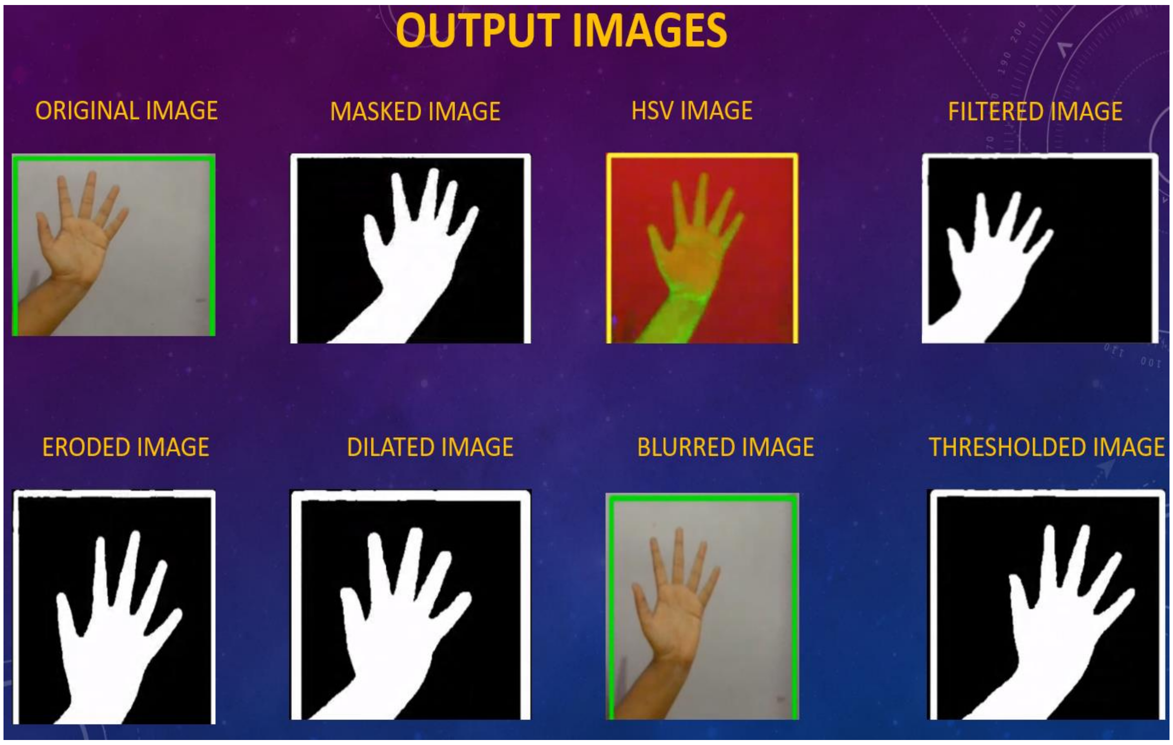
```
int motorA1 = 5; // Pin 2 of L293

int motorA2 = 6; // Pin 7 of L293

int motorB1 = 9; // Pin 10 of L293

int motorB2 = 10; // Pin 14 of L293

int vel = 255; // Speed Of Motors (0-255)
```

```
char state = '0'; // Initialise Motors

#define rxpin 7

#define txpin 8

//SoftwareSerial mySerial(rxPin, txPin); // RX, TX

void setup() { Serial.begin(115200); // Initialize serial communication at 115200
bits per second // Set pins as outputs

   pinMode(motorA1, OUTPUT);

   pinMode(motorA2, OUTPUT);

   pinMode(motorB1, OUTPUT);

   pinMode(motorB2, OUTPUT);}

void loop() {

  if(Serial.available()>0){ // Reads from bluetooth and stores its

        state = Serial.read(); }Serial.println(state);

        if(state==5 || state=='5'){ // Forward

      Serial.println(state);


      digitalWrite(motorA1, 1);

      digitalWrite(motorA2, 0);

      digitalWrite(motorB1, 1);

      digitalWrite(motorB2, 0); }


       if(state==2 || state=='2'){ // Reverse

      Serial.println(state);
```

```
digitalWrite(motorA1, 0);

digitalWrite(motorA2, 1);

digitalWrite(motorB1, 0);

digitalWrite(motorB2, 1);

}

 if(state==3 || state=='3'){ // Right

Serial.println(state);

digitalWrite(motorA1, 1);

digitalWrite(motorA2, 0);

digitalWrite(motorB1, 0);

digitalWrite(motorB2, 0); }


 if(state==4 || state=='4'){ // Left

Serial.println(state);

digitalWrite(13,1);

digitalWrite(motorA1, 0);

digitalWrite(motorA2, 0);

digitalWrite(motorB1, 1);

digitalWrite(motorB2, 0); }

 if(state==1 || state=='1'){ // Stop

Serial.println(state);

digitalWrite(motorA1, 0);

digitalWrite(motorA2, 0);
```

```
        digitalWrite(motorB1, 0);

        digitalWrite(motorB2, 0); } }

//      delay(2000);}
```

Step 6: Image-Processing techniques involved



- First, hand gesture is captured through webcam.
- Gaussian blur technique is used in this process to remove noise i.e. it removes high frequency content like edges noise etc.
- The masked image observed is directly converted from bgr format to hsv format.
- For the obtained hsv (Hue-Saturation-Value) image masking technique is applied again to get filtered image.
- Next morphological transformation technique (erosion followed by dilation) is applied to filter out the background noise.
- Gaussian blur is applied to the eroded image.

- Threshold operation performed in the blurred image which is performed last step here necessity of thresholding is to select a region or area of interest i.e., ignoring the area where we are not concerned with and segmenting only the portion of our hand.
- Then for the region we are going to find the contours for max area of our hand.
- Then we are going to form convex hull and then using cosine theorem we are going to find the defects with the count of defect the no of fingers will be printed
- This output is then generated as a signal and communicated to the bot. Once if u see the code and observe the each command we will arrive at final output.

- **<u>OPENCV CODE</u>**

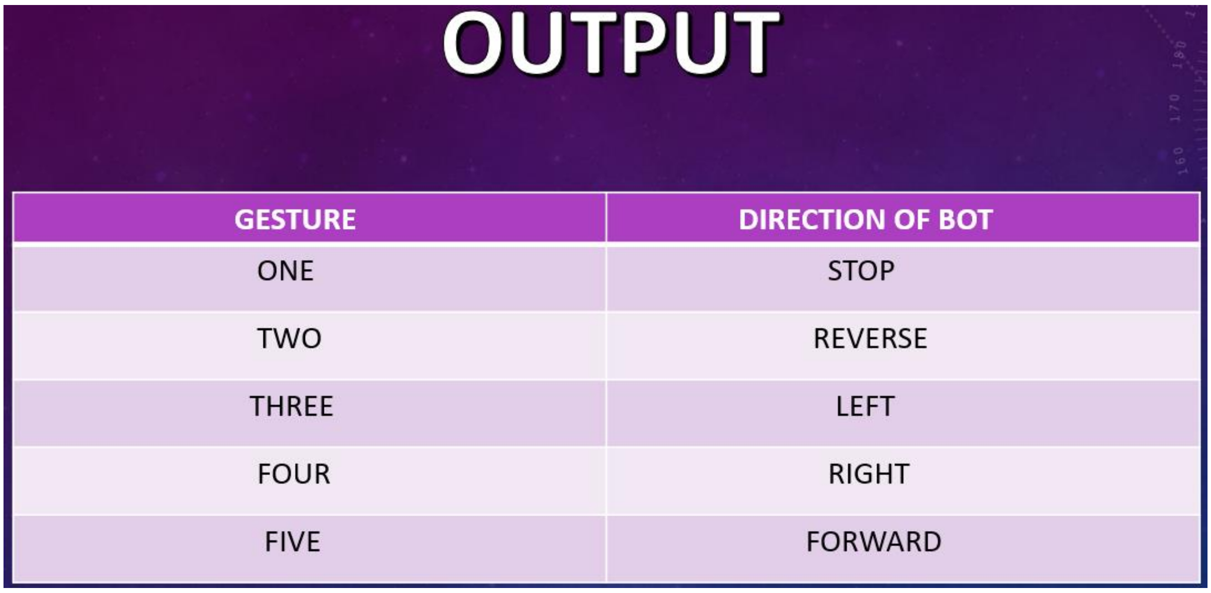
```
# Imports

import numpy as np

import cv2

import math

import serial

import time

# Open Camera

capture = cv2.VideoCapture(0)

finger=0

Arduino_Serial = serial.Serial('COM3', 115200)

time.sleep(2)

while capture.isOpened():
    # Capture frames from the camera

    ret, frame = capture.read()


    # Get hand data from the rectangle sub window

    cv2.rectangle(frame, (100, 100), (500, 500), (0, 255, 0), 5)

    # crop_image = frame[100:300, 100:300]


    # Apply Gaussian blurq

    blur = cv2.GaussianBlur(frame, (3, 3), 0)


    # Change color-space from BGR -> HSV

    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)


    # Create a binary image with where white will be skin colors and rest is
black

    mask2 = cv2.inRange(hsv, np.array([2, 20, 0]), np.array([30, 210, 210]))


    # Kernel for morphological transformation

    kernel = np.ones((5, 5))


    # Apply morphological transformations to filter out the background
noise

    dilation = cv2.dilate(mask2, kernel, iterations=1)
    erosion = cv2.erode(dilation, kernel, iterations=1)

    # Apply Gaussian Blur and Threshold
    filtered = cv2.GaussianBlur(erosion, (3, 3), 0)
    ret, thresh = cv2.threshold(filtered, 127, 255, 0)

    # Show threshold image
    #cv2.imshow("Thresholded", thresh)

    # Find contours
    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    try:
        # Find contour with maximum area
        contour = max(contours, key=lambda x: cv2.contourArea(x))

        # Create bounding rectangle around the contour
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 0)

        # Find convex hull
        hull = cv2.convexHull(contour)
```

```
# Draw contour

drawing = np.zeros(frame.shape, np.uint8)

cv2.drawContours(drawing, [contour], -1, (0, 255, 0), 0)

cv2.drawContours(drawing, [hull], -1, (0, 0, 255), 0)

  # Find convexity defects

hull = cv2.convexHull(contour, returnPoints=False)

defects = cv2.convexityDefects(contour, hull)


# Use cosine rule to find angle of the far point from the start and
end point i.e. the convex points (the finger

# tips) for all defects

count_defects = 0


for i in range(defects.shape[0]):

   s, e, f, d = defects[i, 0]


   start = tuple(contour[s][0])

   end = tuple(contour[e][0])

   far = tuple(contour[f][0])


   a = math.sqrt((end[0] - start[0]) ** 2 + (end[1] - start[1]) ** 2)

   b = math.sqrt((far[0] - start[0]) ** 2 + (far[1] - start[1]) ** 2)
```

```
            c = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)

            angle = (math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c)) * 180) /
3.14


            # if angle > 90 draw a circle at the far point

            if angle <= 90:

                count_defects += 1

                cv2.circle(frame, far, 1, [0, 0, 255], 3)


            cv2.line(frame, start, end, [0, 255, 0], 2)


        # Print number of fingers

        if count_defects ==1:

            finger=2

            cv2.putText(frame, "TWO", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)

        elif count_defects == 2:

            finger=3

            cv2.putText(frame, "THREE", (5, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)

        elif count_defects == 3:

            finger=4

            cv2.putText(frame, "FOUR", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)
```

```
        elif count_defects == 4:

            finger=5

            cv2.putText(frame, "FIVE", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2,(0,0,255), 2)

        elif count_defects == 0:

                aspectRatio = w/h



                #print(aspectRatio)

                if aspectRatio < 0.65:

                    finger=1

                    cv2.putText(frame, "ONE", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 3)



                else:

                    finger=0

                    cv2.putText(frame, "ZERO", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 3)



        else:

            pass

    except:

        pass



    # Show required images
    #cv2.imshow("Gesture", frame)

    all_image = np.hstack((drawing,frame))



    if(finger!=0):


         # Create Serial port object called arduinoSerialData

        # print(Arduino_Serial.readline())

        print('Number of finger', finger)

        value = str(finger)

        Arduino_Serial.write(value.encode('utf-8'))

        #Arduino_Serial.write(b"finger")

        #print('Arduino_Serial',Arduino_Serial.readline())

        #time.sleep(0.5)


    cv2.imshow('Contours', all_image)


    # Close the camera if 'q' is pressed

    if cv2.waitKey(1) == ord('q'):

        break


capture.release()

cv2.destroyAllWindows()
```

Step 7 : Output



OUTPUT

GESTURE	DIRECTION OF BOT
ONE	STOP
TWO	REVERSE
THREE	LEFT
FOUR	RIGHT
FIVE	FORWARD

THANK YOU !