VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

FACULTY OF COMPUTER ENGINEERING

GROUP 4: **ĐỖ LA QUỐC TUẤN – 21521621**

TRẦN GIA KIỆT – 21522264

LÊ NGUYÊN ANH - 21521822

CLASS: CE232.O21.MTCL

CAPSTONE PROJECT

THE CAR IS CONTROLLED BY WIFI THROUGH HAND GESTURES

MENTOR:
ĐOÀN DUY

HO CHI MINH City – 2023

TABLE OF CONTENTS

Contents

Chapter 1: INTRODUCTION.....3
1.1. Question.....3
1.2. Directions for researching the topic.....3
1.3. Target of the project.....3
Chapter 2: THEORY BASIS.....4
2.1. Python programming language.....4
2.2. MediaPipe.....4
2.3. Hand landmark.....5
2.4. Esp32.....5
2.5. Camera.....7
2.6. L298N Motor Driver Module.....8
2.7. Car.....10
Chapter 3: SYSTEM IMPLEMENTATION PROCESS.....13
3.1. Circuit map.....13
3.2. Code.....14
3.3.2. CarRobot.....17
3.3.3. DataParser.cpp.....22
3.3.4. DataParser.h.....23
Chapter 4: EXPERIMENTIAL RESULT.....24
4.1. Overview of the results.....24
4.2. Detail results.....24
Chapter 5: EVOLUTION AND CONCLUSION.....25
5.1. Evolution.....25
5.2. Conclusion.....25
REFERENCE.....25

Chapter 1: INTRODUCTION

1.1. Question

- In the digital age, where information technology is at the forefront, the fact that machines can replace humans is unquestionable. Instead of us having to work, we can order machines. The topic "Hand recognition through landmarks" will also be a solution to help people control machines. Through hand gesture recognition, specific commands will be issued to perform the function corresponding to that gesture, specifically controlling the car via wifi. This report will talk about the topic of how to control the car with hand gestures via wifi.

1.2. Directions for researching the topic

- Learn about image recognition and processing technologies and devices such as faces, hands, ... on the market.
- Develop a hand recognition system through landmarks, then upgrade to mobile app.
- Research and develop control and communication techniques between parts of the system.

1.3. Target of the project

- Develop a hand recognition system through landmarks, thereby exporting and controlling vehicle functions via wifi.
- Optimize system stability and accuracy through research and development of control and communication techniques through cameras.
- Evaluate the effectiveness and feasibility of the system through tests.

Chapter 2: THEORY BASIS

2.1. Python programming language

Python is a high-level programming language for general-purpose programming purposes, created by Guido van Rossum and first released in 1991. Python is designed with the strong advantage of being easy to read, easy to learn and easy to remember. . Python is a language with a very bright appearance, clear structure, convenient for beginners to learn programming and is an easy programming language to learn; widely used in artificial intelligence development.

Python is completely dynamically typed and uses automatic memory allocation; Therefore it is similar to Perl, Ruby, Scheme, Smalltalk, and Tcl.

Initially, Python was developed to run on the Unix platform. But over time, Python gradually expanded to all operating systems from MS-DOS to Mac OS, OS/2, Windows, Linux and other operating systems of the Unix family.

2.2. MediaPipe

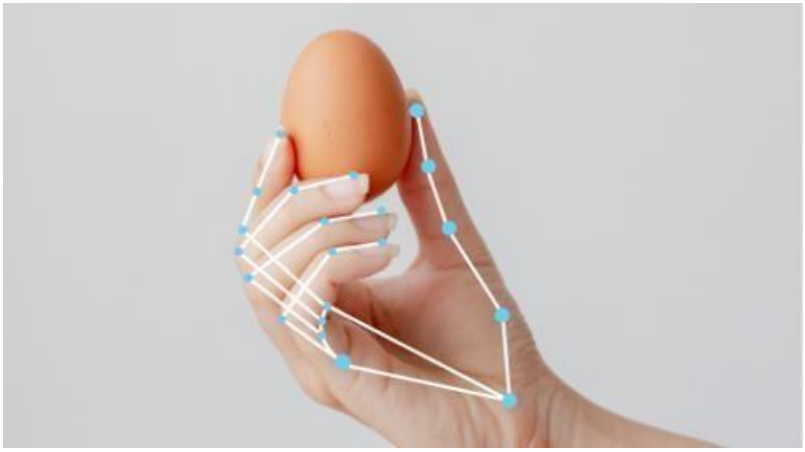
Mediapipe is a very accurate and lightweight physical gesture detection library.

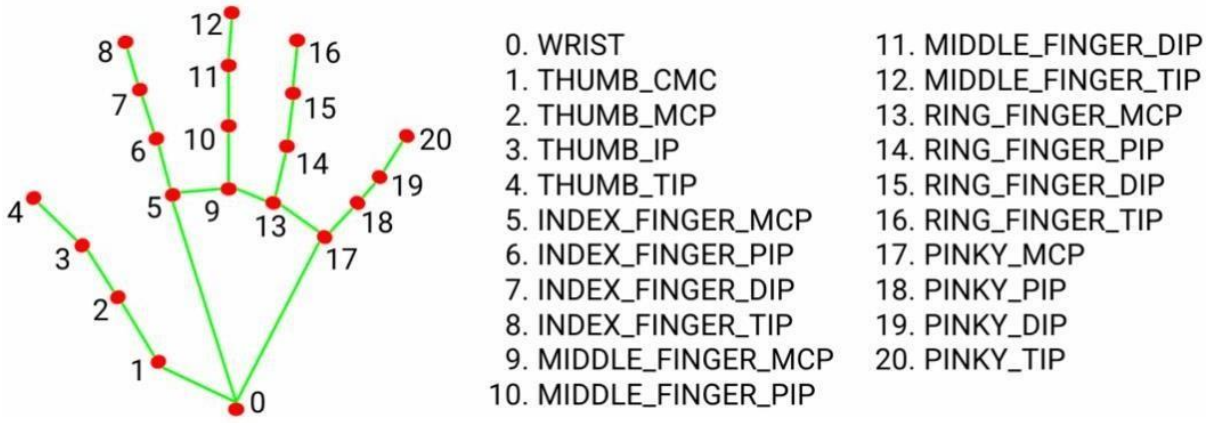
Function:

1. Face detection
2. Find the face mesh
3. Iris: find the distance from the eye's pupil to the camera without needing a Depth webcam
4. Detect hand gestures
5. Find your body shape
6. Change hair color
7. Object detection & Box tracking: find objects
8. Track the movement of objects
9. Objectron: find the cube containing the object
10. KNIFT: Find objects using known features

2.3. Hand landmark

The MediaPipe Hand Landmarker task lets you detect the landmarks of the hands in an image. You can use this task to locate key points of hands and render visual effects on them. This task operates on image data with a machine learning (ML) model as static data or a continuous stream and outputs hand landmarks in image coordinates, hand landmarks in world coordinates and handedness(left/right hand) of multiple detected hands.





2.4. Esp32

ESP32 NodeMCU LuaNode32 BLE Wifi transceiver RF kit is developed on the basis of the central module ESP32 with the latest Wifi, BLE technology and integrated ARM SoC core, the kit has similar hardware design, firmware and usage. Kit NodeMCU ESP8266, with the advantages of easy use, full pin output, integrated charging circuit and UART CP2102 communication, Wifi BLE Kit ESP32 NodeMCU LuaNode32 is the top choice in Wifi research and applications, BLE and IoT.

Specifications:

- Power used: 5VDC from Micro USB port.

- Integrated charging citcuit and UART CP2102 interface.

- Full output of ESP32 module, standard 2.54mm plug.

- Integrated Led Status, BOOT and ENABLE buttons.

- Dimensions: 28.33x51.45mm

2.5. Camera

A camera or photo camera is a device used to record images into a still image or into a series of moving images (called film or video). The function of the camera is similar to the human eye. Cameras can work in the visible light spectrum or in other regions of the electromagnetic radiation spectrum.

2.6. L298N Motor Driver Module

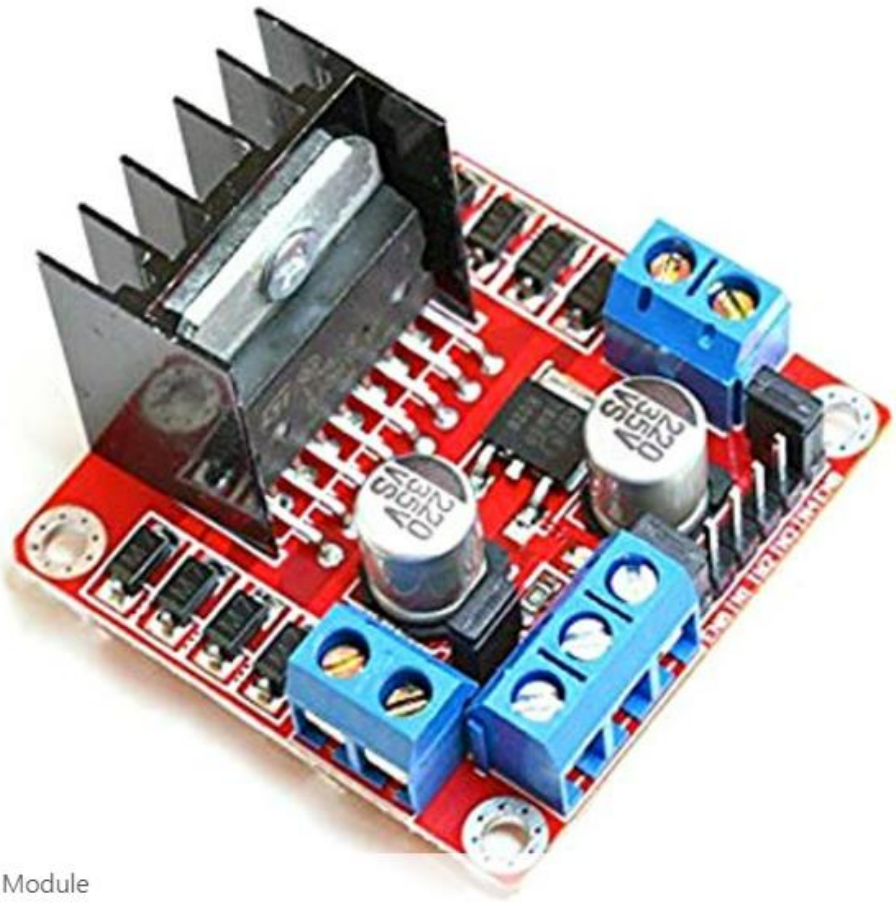
L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M055 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

L298N Module Pinout Configuration

Pin Name	Description
IN1 & IN2	Motor A input pins. Used to control the spinning direction of Motor A
IN3 & IN4	Motor B input pins. Used to control the spinning direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 & OUT2	Output pins of Motor A
OUT3 & OUT4	Output pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground pin

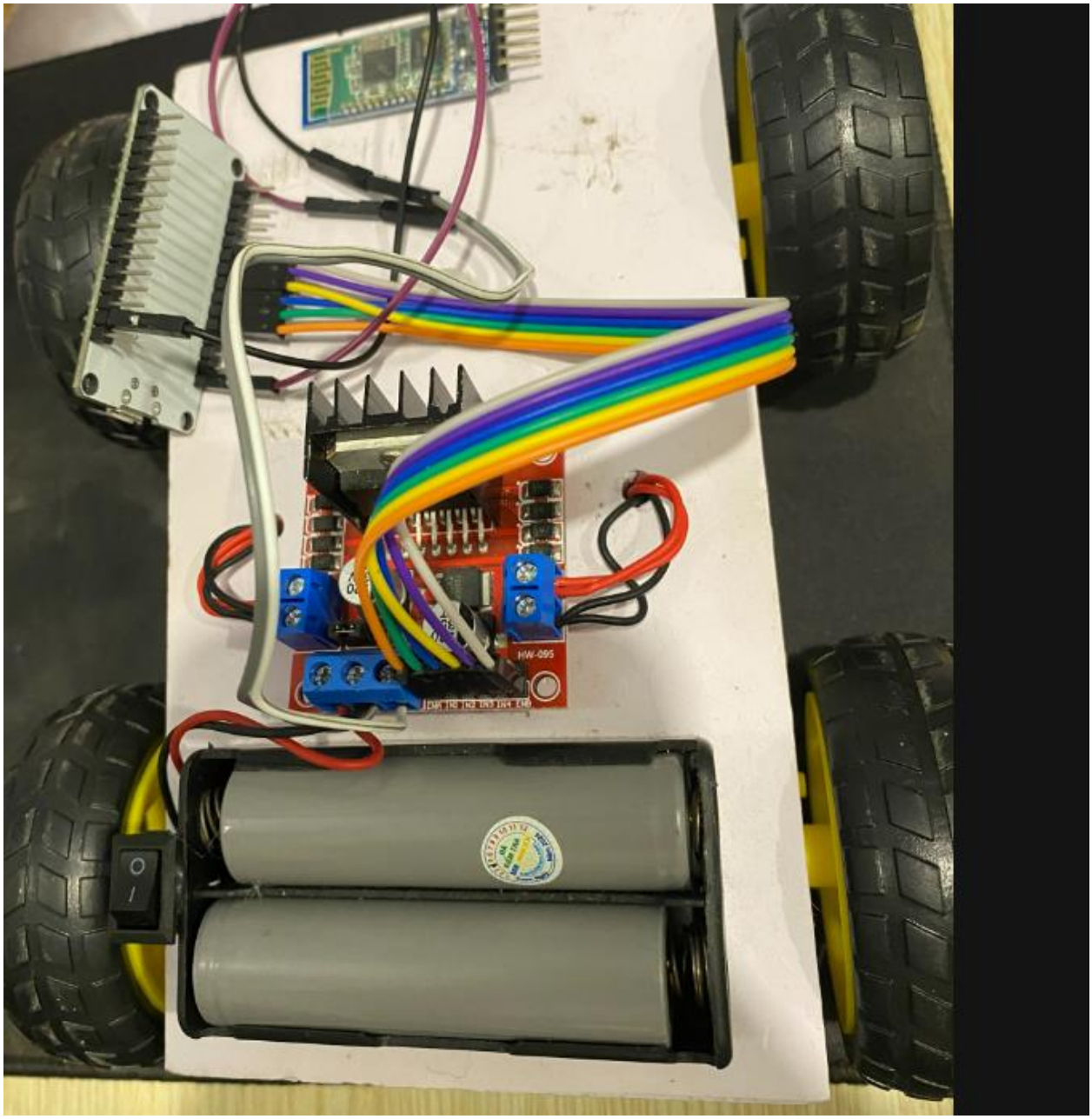
Features & Specifications

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Motor Supply Voltage(Max): 46V
- Motor Supply Current(Max): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Logical Voltage: 0-36mA
- Maximum Power(W): 25W
- Current Sense for each motor
- Heatsink for better performance
- Power-On led indicator



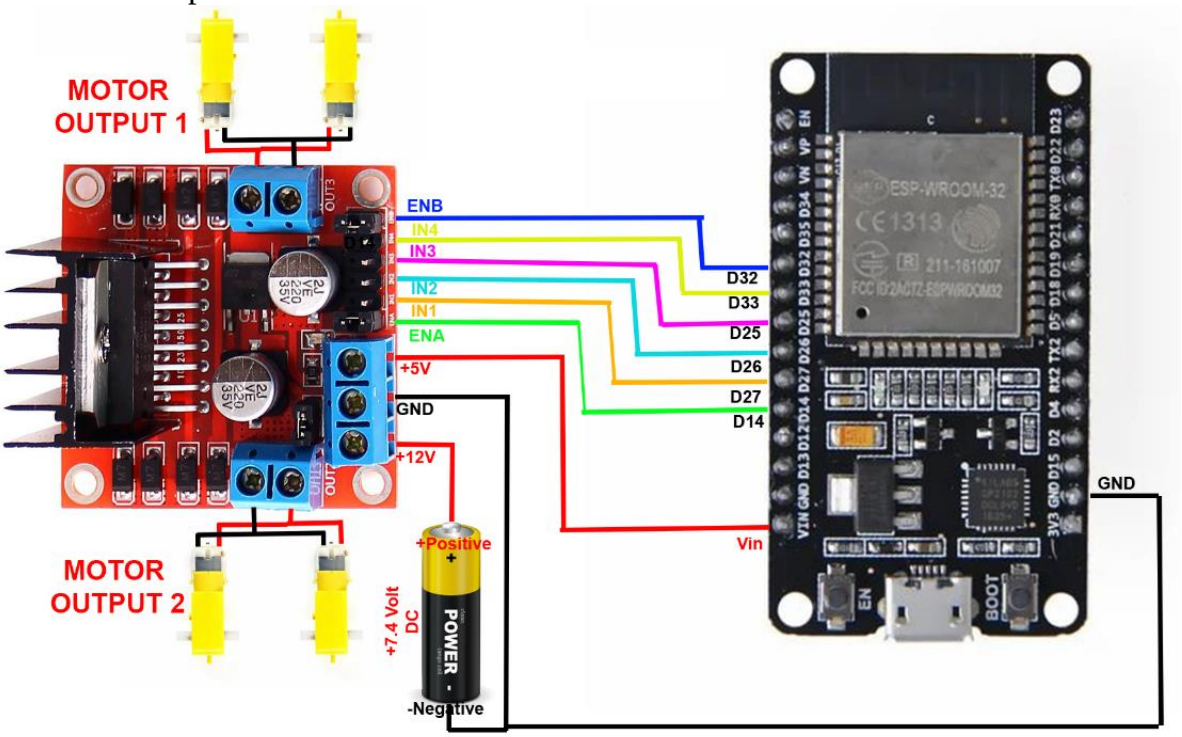
L298N Motor Driver Module

2.7. Car


HW-095

Chapter 3: SYSTEM IMPLEMENTATION PROCESS

3.1. Circuit map



3.2. Code

3.2.1. Code description

My system is capable of recognizing hand gestures via camera using the OpenCV library. Then encrypt the data and transmit it by Serial communication to the Esp32 wifi module. From the encrypted data, the vehicle will be controlled according to the coded data according to the correct function.

3.2.2. Hand gesture recognizition

```
import cv2
import mediapipe as mp
import math
import socket

robotAddressPort = ("192.168.1.9", 12345)

mp_hands = mp.solutions.hands
hands = mp_hands.Hands()

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

cv2.namedWindow('Hand Tracking', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Hand Tracking', 1280, 720)

height = 720
width = 1280

center_x = width // 2
center_y = height // 2

top_box_top_left = (center_x - 100, 0)
top_box_bottom_right = (center_x + 100, 200)

bottom_box_top_left = (center_x - 100, height - 200)
bottom_box_bottom_right = (center_x + 100, height)

while cap.isOpened():
    global turn
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.flip(frame, 1)
    image = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
```

```
    results = hands.process(image)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            mp_drawing = mp.solutions.drawing_utils
            mp_drawing.draw_landmarks(frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)

            landmarks = []
            for landmark in hand_landmarks.landmark:
                x, y, _ = image.shape
                landmarks.append((int(landmark.x * y), int(landmark.y *
x)))

            thump_tip = landmarks[4]
            index_tip = landmarks[8]
            middle_tip = landmarks[12]
            pinky_tip = landmarks[20]

            dist_thump_index = thump_tip[0] - index_tip[0]
            dist_index_middle = index_tip[0] - middle_tip[0]
            dist_middle_pinky = middle_tip[0] - pinky_tip[0]

            if dist_thump_index < 0 and dist_index_middle < 0 and
dist_middle_pinky < 0:
                hand_label = "Right Hand"
            else:
                wrist = landmarks[0]
                middle_mcp = landmarks[9]
                cv2.line(frame, wrist, middle_mcp, (0, 255, 0), 2)

                angle = math.atan2(wrist[1] - middle_mcp[1], wrist[0] -
middle_mcp[0])
                angle = math.degrees(angle)
                angle -= 90
                if angle <= -180:
                    angle += 360
                elif angle > 180:
                    angle -= 360

                cv2.putText(frame, f"Rotation Angle: {angle:.2f}", (50,
50), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2, cv2.LINE_AA)

                if angle > 20:
                    hand_label = "Right"
                    turn = True
                    sendTorobot("r")
                elif angle < -20:
                    hand_label = "Left"
                    turn = True
                    sendTorobot("l")
                else:
                    turn = False
                    hand_label = ""

                if (top_box_top_left[0] < landmarks[9][0] <
top_box_bottom_right[0] and
```

```
                    top_box_top_left[1] < landmarks[9][1] <
top_box_bottom_right[1]) and \
                        (top_box_top_left[0] < landmarks[13][0] <
top_box_bottom_right[0] and
                         top_box_top_left[1] < landmarks[13][1] <
top_box_bottom_right[1]):
                    # Display rotation angle on the left side of the
screen
                    cv2.putText(frame, f"Forward", (50, 100),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2,
                                cv2.LINE_AA)
                    sendTorobot("f")
                    print("Forward")
                elif (bottom_box_top_left[0] < landmarks[9][0] <
bottom_box_bottom_right[0] and
                      bottom_box_top_left[1] < landmarks[9][1] <
bottom_box_bottom_right[1]) and \
                        (bottom_box_top_left[0] < landmarks[13][0] <
bottom_box_bottom_right[0] and
                         bottom_box_top_left[1] < landmarks[13][1] <
bottom_box_bottom_right[1]):
                    cv2.putText(frame, f"Back", (50, 100),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2,
                                cv2.LINE_AA)
                    sendTorobot("b")
                    print("Back")
                elif turn != True:
                    cv2.putText(frame, f"Stop", (50, 100),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2, cv2.LINE_AA)
                    sendTorobot("s")
                    print("Stop")

        cv2.putText(frame, hand_label, (landmarks[0][0], landmarks[0][1]),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2, cv2.LINE_AA)

    cv2.rectangle(frame, top_box_top_left, top_box_bottom_right, (0, 0,
255), 2)
    cv2.rectangle(frame, bottom_box_top_left, bottom_box_bottom_right, (0,
0, 255), 2)

    def sendTorobot(move):
        msg4robot = ','.join([move, '1000, 0, 0, 0'])
        print(msg4robot)
        bytesToSend = str.encode(msg4robot)
        bufferSize = 1024
        UDPClientSocket = socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)
        UDPClientSocket.sendto(bytesToSend, robotAddressPort)

    cv2.imshow('Hand Tracking', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyWindow()
```

3.3.2. CarRobot

```
#include <WiFi.h>
#include <AsyncUDP.h>
#include <Arduino.h>
#include "DataParser.h"

const char* ssid = "Tuan Do La";
const char* password = "0943632346A@";

DataParser dataParser;

const int udpPort = 12345;

int in1 = 27;
int in2 = 26;
int ena = 14;
int in3 = 25;
int in4 = 33;
int enb = 32;

int DataIndex;
int Speed = 50;
```

```
int Right_speed = 0;
int Left_speed = 0;

AsyncUDP udp;

void setup() {
  Serial.begin(115200);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(ena, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(enb, OUTPUT);

  WiFi.begin(ssid, password);

  while(WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
    }

   Serial.println("Connectited to WiFi");

   if(udp.listen(udpPort)) {
    Serial.print("UDP listening on IP: ");
```

```
    Serial.println(WiFi.localIP());
    udp.onPacket( [](AsyncUDPPacket packet) {
      String IncomingData = (char*)packet.data();
      dataParser.parseData(IncomingData, ',');
      Speed = (dataParser.getField(1)).toInt();
      Left_speed = Speed;
      Right_speed = Speed;
      });
    }
}

void loop() {
  if(dataParser.getField(0) == "f") {
    forward(Left_speed, Right_speed);
    Serial.println("fwd");
  }
  if(dataParser.getField(0) == "b") {
    backward(Left_speed, Right_speed);
    Serial.println("bck");
  }
  if(dataParser.getField(0) == "l") {
    left(Left_speed, Right_speed);
    Serial.println("left");
  }
  if(dataParser.getField(0) == "r") {
```

```
    right(Left_speed, Right_speed);
    Serial.println("right");
  }
  if(dataParser.getField(0) == "s") {
    Stop();
  }
}

void forward(int left_speed, int right_speed) {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(ena, left_speed);
  analogWrite(enb, right_speed);
}

void backward(int left_speed, int right_speed) {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(ena, left_speed);
  analogWrite(enb, right_speed);
}
```

```
void left(int left_speed, int right_speed) {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(ena, left_speed);
  analogWrite(enb, right_speed);
}

void right(int left_speed, int right_speed) {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(ena, left_speed);
  analogWrite(enb, right_speed);
}

void motor_speed(int Right_Speed, int Left_Speed) {
  Left_speed = Left_Speed;
  Right_speed = Right_Speed;
}

void Stop() {
```

```
 digitalWrite(in1, LOW);
 digitalWrite(in2, LOW);
 digitalWrite(in3, LOW);
 digitalWrite(in4, LOW);
 analogWrite(ena, 0);
 analogWrite(enb, 0);
}
```

3.3.3. DataParser.cpp

```
#include "DataParser.h"
DataParser::DataParser() {
  _delimiter = ',';
  _fieldCount = 0;
}

void DataParser::parseData(const char* data, char delimiter) {
  _delimiter = delimiter;
  _fieldCount = 0;
  char* mutableData = strdup(data);
  char* token = strtok(mutableData, &_delimiter);
  while (token != NULL) {
    _fields[_fieldCount++] = String(token);
    token = strtok(NULL, &_delimiter);
  }
  free(mutableData);
}

void DataParser::parseData(const String& data, char delimiter) {
  parseData(data.c_str(), delimiter);
}

String DataParser::getField(int index) {
  if (index >= 0 && index < _fieldCount) {
    return _fields[index];
  } else {
    return "";
  }
```

```
}

int DataParser::getFieldCount() {
  return _fieldCount;
}
```

3.3.4. DataParser.h

```
#ifndef DataParser_h

#define DataParser_h


#include <Arduino.h>


class DataParser {

public:

  DataParser();

  void parseData(const char* data, char delimiter);

  void parseData(const String& data, char delimiter);

  String getField(int index);

  int getFieldCount();


private:

  char _delimiter;

  String _fields[50]; // Adjust the array size as needed

  int _fieldCount;

};
```

Chapter 4: EXPERIMENTIAL RESULT

4.1. Overview of the results

The system has been tested 10 times and all 10 times the request ran correctly, with no unexpected errors.

4.2. Detail results

Link video: https://drive.google.com/file/d/1hTZOvcdrwKRpea-2b9a-MVLkDVwdUy0b/view?usp=sharing

Chapter 5: EVOLUTION AND CONCLUSION

5.1. Evolution

The possibility: The system operates stably in real environments.

The stabilization: The system operates normally, functions properly, no errors occur

The improvement: The system can be developed into topics such as gesture recognition to control household devices, language translation for mute people.

5.2. Conclusion

The system operates stably, recognizes gestures correctly, and has the ability to develop into a large system based on the old system. However, the recognition accuracy is about 80%.

REFERENCE

1) https://glints.com/vn/blog/ngon-ngu-lap-trinh-python-la-gi/