Inspiring Excellence

CSE461

Introduction to Robotics

Final Project - Group 8

AI Face Detection Based Autonomous Robot Navigation

Members:

1. **Sihab Sahariar , ID : 20101402**
2. **Golam Dastagir, ID: 20101493**
3. **Faiza Bushra, ID: 20101554**
4. **Mohammad Sultanul Arefin, ID: 20201138**
5. **Rakibul Hassan, ID :**

Table of Contents

Introduction :

We combine advanced computer vision algorithms with real-time motor control in this research to detect human faces with great accuracy and efficiency. This self-driving system is intended to overcome the difficulties of navigation in complicated areas, improve human-robot interaction, and promote inclusion. The capacity of the robot to handle complex situations makes it a great tool for public guiding and logistics. Furthermore, its capacity to detect faces and offer customized experiences promotes diversity by giving alternate communication modalities for people of varying capacities. This project represents an important intersection of robotics and computer vision, changing technology's position in our lives. It bridges the gap between software and hardware, providing a futuristic view of computers that can instinctively adapt to human demands.

Components :

1. Raspberry Pi 4
2. L298N Motor Driver
3. DC Motor (2X)
4. Ultrasonic Sensor
5. 9 Volt Battery
6. Jumper Wires
7. Robot Body
8. Webcam
9. Power Bank

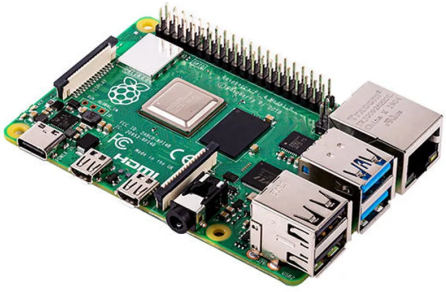

Figure : Raspberry Pi 4

The Raspberry Pi is a diminutive and cost-effective computer board that boasts remarkable processing and connectivity capabilities. It has found extensive employment in a diverse range of applications, spanning from coding exercises to robots and servers. Its GPIO pins facilitate hardware interaction, rendering it a favored instrument for acquiring knowledge in electronics and programming. This petite gadget has democratized technology, empowering innovation and ingenuity at a modest expense.

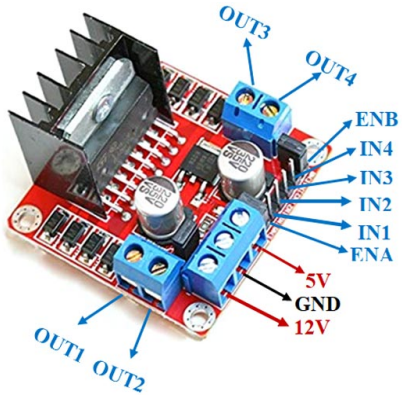

Figure : L298N Motor Driver

The L298N motor driver is a widely used twin H-bridge module for motor control in various applications. It has the capability to control two DC motors or a single stepper motor in both forward and reverse directions. Due to its ability to withstand higher currents and voltages, this module is particularly suitable for robotics, automation, and projects that require precise motor control. The L298N streamlines motor control with its uncomplicated interface, which utilizes digital inputs to regulate motor direction and speed.

Figure : DC Motor

A direct current (DC) motor employs a direct current power supply to convert electrical energy into mechanical motion. It produces rotational movement through the interaction between a magnetic field and current-carrying conductors. DC motors find application in various fields such as robotics, household appliances, automotive systems, and heavy-duty machinery.

Figure : 9 Volt Battery

A 9-volt battery is a tiny, portable power source that is typically seen in small electrical gadgets. It has a nominal voltage of 9 volts and is appropriate for applications such as smoke detectors, remote controls, and low-power devices.

Figure : Jumper Wires

Jumper wires are brief, pliable cables that possess connectors on either end, which are utilized to interconnect components on a breadboard or circuit. They facilitate swift prototyping and testing in electronics applications.

Figure : Ultrasonic Sensor

To assess distances or detect objects, an ultrasonic sensor generates high-frequency sound waves and detects their echo. It is widely utilized in robotics and automation because it provides non-contact sensing and precise distance measurements.

Figure : Robot Body

A simple plastic body for our robot to fix all the components on the rover.

Figure : Webcam

This is a normal webcam we use, in our project we used this module for robot vision to detect faces.

Figure : Power Bank

We have this power bank to power up Raspberry PI and the whole robot.

Connectivity and Code :

Ultrasonic Sensor (HC-SR04):

- The ultrasonic sensor serves as the robot's "eyes" for obstacle detection.
- Its TRIG pin is connected to GPIO 22 and ECHO pin is connected to GPIO27 on the Raspberry Pi, allowing the Pi to receive distance information from the sensor.

L298N Motor Driver:

- The L298N motor driver acts as the "brain" for controlling the robot's motors.
- GPIO pins on the Raspberry Pi are utilized to control the motor driver's inputs:
  IN1: Connected to GPIO 12
  IN2: Connected to GPIO 13
  IN3: Connected to GPIO 16
  IN4: Connected to GPIO 19
- A 9-volt battery powers the motor driver, providing the necessary voltage for the motors' operation.

Power Sources:

- The robot's motor driver is powered by a 9-volt battery, ensuring sufficient power for motor movement.
- The Raspberry Pi is powered by a 5-volt power bank, providing the necessary power for the Pi's computation and control functions.

Webcam for Face Detection:

- A webcam is connected to the Raspberry Pi, enabling face detection capabilities.
- The Raspberry Pi processes the webcam's input to identify and track faces, enhancing the robot's interactivity.

In this connectivity setup, the Raspberry Pi acts as the central control unit, receiving information from the ultrasonic sensor, executing motor commands via the L298N driver, and processing webcam input for face detection. The separate power sources ensure stable operation for both the motors and the Raspberry Pi. Altogether, this configuration enables the robot to intelligently navigate its environment, avoid obstacles, and even interact with faces using its webcam-based vision.

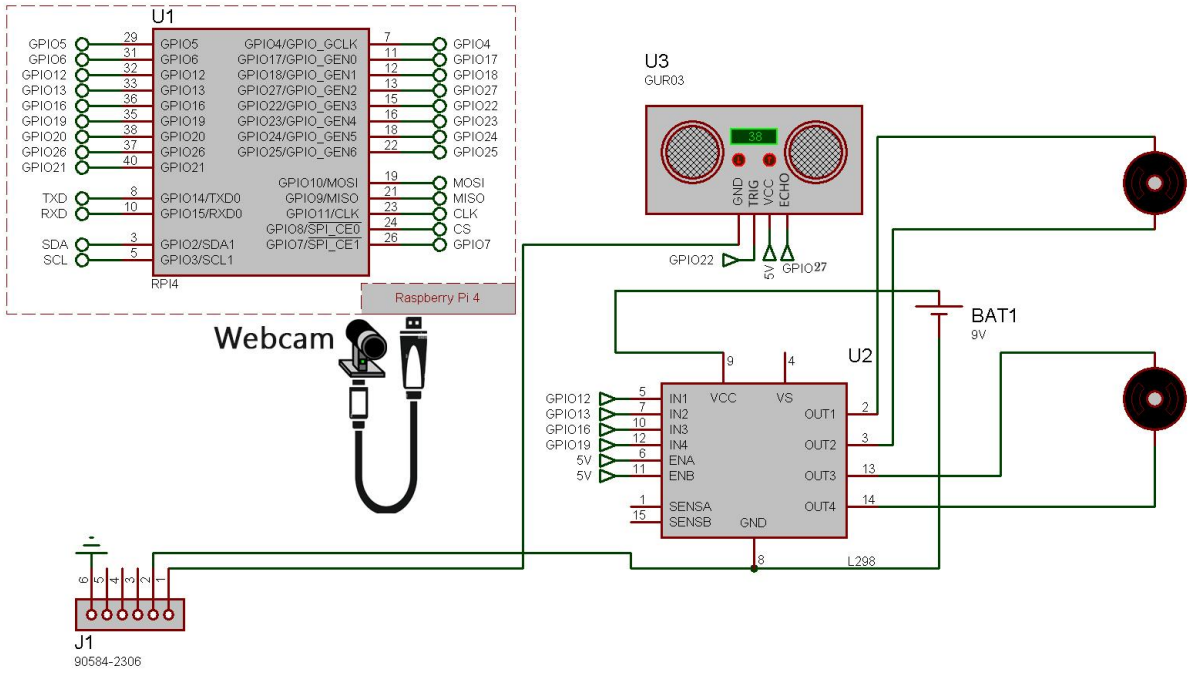

Figure : Circuit of the Robot

Code :

```
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np
```

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

# Ultrasonic sensor pins
TRIG_PIN = 22
ECHO_PIN = 27

# Motor driver pins
IN1_PIN = 12
IN2_PIN = 13
IN3_PIN = 16
IN4_PIN = 19

# Set up GPIO mode
GPIO.setmode(GPIO.BCM)

# Set up ultrasonic sensor pins
GPIO.setup(TRIG_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)

# Set up motor driver pins
GPIO.setup(IN1_PIN, GPIO.OUT)
GPIO.setup(IN2_PIN, GPIO.OUT)
GPIO.setup(IN3_PIN, GPIO.OUT)
GPIO.setup(IN4_PIN, GPIO.OUT)

# Function to move forward
def move_forward():
    GPIO.output(IN1_PIN, GPIO.HIGH)
    GPIO.output(IN2_PIN, GPIO.LOW)
    GPIO.output(IN3_PIN, GPIO.HIGH)
    GPIO.output(IN4_PIN, GPIO.LOW)

# Function to move left
def move_left():
    GPIO.output(IN1_PIN, GPIO.HIGH)
    GPIO.output(IN2_PIN, GPIO.LOW)
    GPIO.output(IN3_PIN, GPIO.LOW)
    GPIO.output(IN4_PIN, GPIO.HIGH)

# Function to move right
def move_forward():
    GPIO.output(IN1_PIN, GPIO.LOW)
    GPIO.output(IN2_PIN, GPIO.HIGH)
    GPIO.output(IN3_PIN, GPIO.HIGH)
    GPIO.output(IN4_PIN, GPIO.LOW)

# Function to stop
def stop():
    GPIO.output(IN1_PIN, GPIO.LOW)
    GPIO.output(IN2_PIN, GPIO.LOW)
    GPIO.output(IN3_PIN, GPIO.LOW)
    GPIO.output(IN4_PIN, GPIO.LOW)
```

```
# Function to measure distance using ultrasonic sensor
def get_distance():
    GPIO.output(TRIG_PIN, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(TRIG_PIN, GPIO.LOW)

    while GPIO.input(ECHO_PIN) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO_PIN) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150  # Speed of sound in cm/s
    return distance

# Function to move my robot with sensor value and face position
# Navigation Algorithm
def move_robot(distance, face_x, frame_width):
    mid_x = frame_width // 2
    if (face_x < mid_x - 50):
        print("Turn Left")
        move_left()
    elif face_x > mid_x + 50:
        print("Turn Right")
        move_right()
    else:
        print("Move Forward")
        move_forward()
        if distance < 20:
            print("Stop")
            stop()

try:
    while True:
        ret, frame = cap.read()
        distance = get_distance()
        if not ret:
            break

        frame_height, frame_width, _ = frame.shape
        mid_x = frame_width // 2
        mid_y = frame_height // 2

        # Draw center rectangle box
        rect_start = (mid_x - 100, mid_y - 100)
        rect_end = (mid_x + 100, mid_y + 100)
        cv2.rectangle(frame, rect_start, rect_end, (0, 0, 255), 2)

        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5,
minSize=(30, 30))

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
```

```
            face_center_x = x + w // 2
            move_robot(distance, face_center_x, frame_width)

        cv2.imshow('BRACU Facebot v1.0', frame)

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

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

To commence, we import the requisite libraries for the purpose of controlling GPIO pins, interacting with computer vision, and managing time. We proceed to load a pre-trained face identification model and configure the camera's video recording. We establish the GPIO mode and define pins for an ultrasonic sensor and motor driver. For robot motions, we define four functions: forward, left, right, and stop. Another function utilizes the ultrasonic sensor to calculate distance. Based on recognized faces and distance data, we create a function to navigate the robot. When a face is recognized, the robot's direction is adjusted to maintain the face centered.

Subsequently, we enter the main loop, where we indefinitely read frames from the camera. Using the cascade classifier, we determine the distance from the ultrasonic sensor and recognize faces in the frame. We construct a rectangle around each identified face and determine its center. The navigation feature enables the robot to modify its moving direction based on the location of the face. The processed frame is displayed on the screen. The loop is terminated if the 'q' key is pressed. We ensure that GPIO cleaning is executed if the loop is manually interrupted (by pressing Ctrl+C). We then release the camera and close the OpenCV windows.

Results :

The project involving the Face Following robot has yielded a cohesive and reactive system. The robot's ultrasonic sensor enables it to gauge facial proximity while navigating autonomously, thereby ensuring a seamless and unobstructed mobility experience. The integration of the L298N motor driver and 9-volt battery ensures accurate motor control and sufficient power supply. Additionally, the robot's ability to identify faces through webcam-based technology underscores its capacity to engage with its surroundings and recognize human faces within its visual range. In summary, this research has culminated in the development of an intelligent robot that can navigate its environment adeptly and even engage in face-to-face interactions.

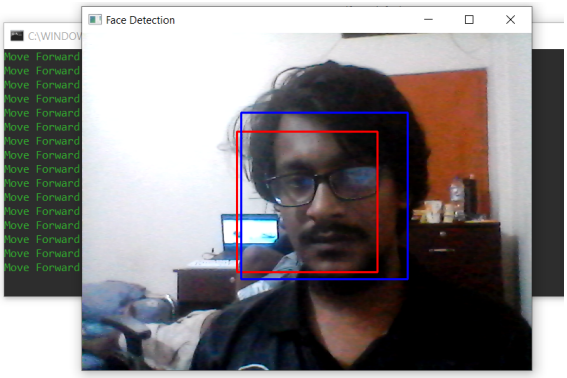

Figure : Move Forward Based on Face Detection

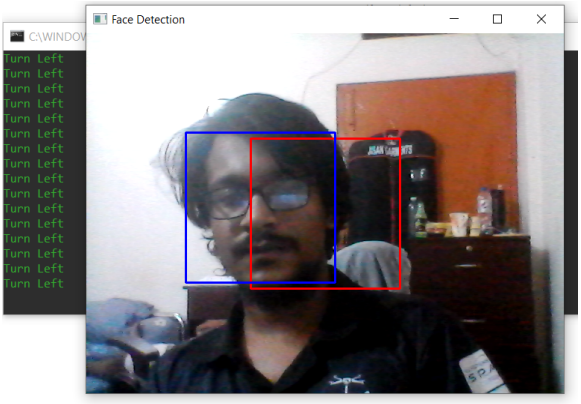

Figure : Move Left Based on Face Detection

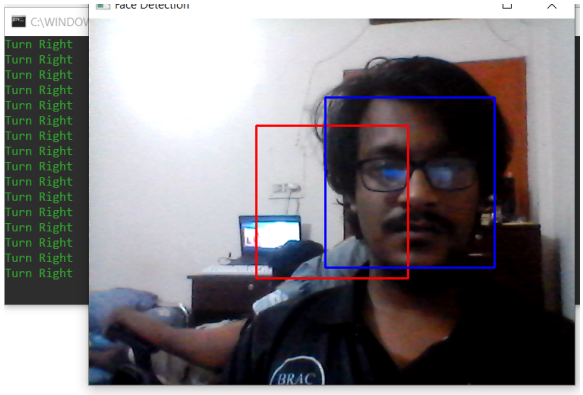

Figure : Move Right Based on Face Detection

Future Improvements :

To enhance the robot's obstacle avoidance capabilities, it is recommended to incorporate advanced algorithms, such as Simultaneous Localization and Mapping (SLAM). This integration would enable the robot to map its surroundings while avoiding obstacles, resulting in more efficient and accurate navigation.

Incorporating machine learning techniques is essential to improve the face identification and tracking process. By training the system with a diverse range of facial expressions, angles, and lighting conditions, the robot's face recognition capabilities can become even more precise and robust, thereby enhancing its interaction.

To facilitate remote control of the robot, it is suggested to implement wireless communication. This would enable individuals to maneuver the robot manually using a smartphone or computer, as required.

References :

1. https://www.raspberrypi.com/documentation/
2. https://towardsdatascience.com/face-detection-in-2-minutes-using-opencv-python-90f89d7c0f81
3. https://raspberrypi-guide.github.io/programming/install-opencv

Contribution of Members :

Name	ID	Contribution
Sihab Sahariar	20101402	Circuit Designing and Robot Navigation Algorithm Implementation
Golam Dastagir	20101493	Components Selection and Face Detection Integration
Faiza Bushra	20101554	Distance Measurement integration and Power Source Selection
Mohammad Sultanul Arefin	20201138	GPIO PIN Selection of Components and Debugging
Rakibul Hassan		Motor Movement Function Implementation and Debugging