Raspberry Pi and Arduino – Comparison and Applications (Robot Car Project: Face Identification)

Jazzal Kandiel, Liza Yousef, Mawa Hijji, Maha blawi

S21107396, S23108546, S22107735, S

Effat University

ECE 342L: Analog Control Systems

Dr. Nema Salem

April 22 2025

Table of contents

Abstract

This project focuses on building a smart robot car capable of identifying faces using a 5 Megapixel camera module and a Raspberry Pi 4B. The robot uses a motor driver board controlled directly by the Raspberry Pi's GPIO pins and processes video streams using OpenCV and Python to detect human faces in real time. When a face is detected, the robot moves forward; otherwise, it stops. This system demonstrates the computational power of the Raspberry Pi in handling both hardware control and complex image processing tasks simultaneously.

Introduction

The Raspberry Pi, a powerful single-board computer, allows for advanced robotics projects that require both hardware interfacing and intensive computation like image processing. Unlike microcontrollers, which are limited in processing capabilities, Raspberry Pi can perform real-time video analysis while controlling motors and sensors. This project leverages the Raspberry Pi's capabilities to create a 4WD robot car that moves intelligently based on face detection using Python and OpenCV libraries. This application highlights the Raspberry Pi's suitability for embedded AI and robotics tasks without needing a secondary microcontroller like Arduino.

Objective

The objectives of this project are as follows:

- To design and build a 4WD robot car using only Raspberry Pi.

- To implement real-time face identification using a camera module and OpenCV.

- To directly control DC motors through a motor driver board using Raspberry Pi GPIO pins.

- To demonstrate the Raspberry Pi's ability to handle both hardware control and image processing.

Components

Each component used in the project and its role:

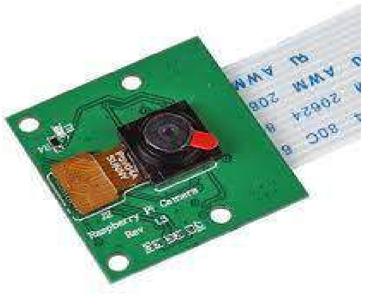
1. Raspberry Pi 4B

- Function: Central processing unit of the robot.
- Role: Runs Python code to detect faces, and controls motor movement based on detection results.
- Why: Combines hardware control with real-time video processing.

2. 5MP Camera Module

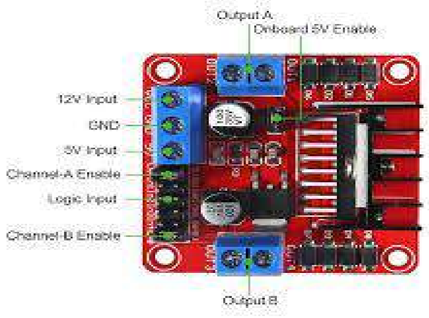
- Function: Captures live video feed.
- Role: Provides input images for face detection algorithms.

- Why: Essential for implementing computer vision.



3. Motor Driver Board (L298N or Motor HAT)

- Function: Interfaces between Raspberry Pi and DC motors.
- Role: Amplifies control signals from Raspberry Pi GPIOs to drive four DC motors.
- Why: Raspberry Pi GPIO pins alone cannot supply the necessary current for motors.



4. 4 DC Motors and 4 Wheels

- Function: Provides mobility to the robot.
- Role: Driven by the motor driver board to allow forward motion when a face is detected.
- Why: Necessary for movement based on detected faces.

5. Chassis

- Function: Structural platform.
- Role: Holds Raspberry Pi, motor driver board, motors, camera, and batteries.
- Why: Provides support and alignment for components.

6. Battery Pack

- Function: Power supply for motors and Raspberry Pi.
- Role: Delivers stable voltage and current to ensure uninterrupted operation.
- Why: Required to make the robot mobile and untethered.

7. Jumper Wires and Connectors

- Function: Electrical connectivity.

- Role: Connect Raspberry Pi GPIO pins to motor driver inputs and power lines.
- Why: Enables communication and control between Raspberry Pi and hardware components.

Connections

The system uses a direct connection between the Raspberry Pi, the motor driver board, the camera module, the DC motors, and the power supply.

The connections are explained below :

1. Camera Module to Raspberry Pi

The 5MP camera module is connected to the Raspberry Pi through the CSI (Camera Serial Interface) port.

To connect it:

- Gently pull up the plastic clip on the CSI port of the Raspberry Pi.
- Insert the camera's ribbon cable with the shiny metallic side facing toward the HDMI ports.
- Push the clip back down to lock the cable in place.

  No GPIO pins are used for the camera, and it communicates with the Raspberry Pi through a dedicated high-speed interface.

2. Motor Driver Board to Raspberry Pi GPIO Pins

The motor driver board (such as L298N or a Motor HAT) connects to the Raspberry Pi's GPIO pins to control the movement of the motors.

The control pins of the motor driver are wired to the Raspberry Pi GPIO pins as follows:

- The IN1 pin of the motor driver is connected to GPIO17 on the Raspberry Pi (physical pin 11).
- The IN2 pin is connected to GPIO18 (physical pin 12).
- The IN3 pin is connected to GPIO22 (physical pin 15).
- The IN4 pin is connected to GPIO23 (physical pin 16).

These connections allow the Raspberry Pi to control the rotation direction of the motors by setting the GPIO pins HIGH or LOW through Python programming.

The GND (Ground) pin of the motor driver board is connected to one of the Raspberry Pi's GND pins (for example, pin 6) to ensure a common ground between the two devices.

This common ground is essential for the motor control signals to be properly recognized.

3. Motors to Motor Driver Board

Each of the four DC motors is connected directly to the output terminals of the motor driver board:

- The left-side motors are connected to the first motor output channels (OUT1 and OUT2).
- The right-side motors are connected to the second motor output channels (OUT3 and OUT4).

If the motors run in the wrong direction (for example, backward when they should move forward), the two motor wires on that side can simply be swapped to correct the direction.

4. Power Connections

The motors require higher current than the Raspberry Pi's GPIO pins can provide.

Therefore, a separate battery pack (such as 7V to 12V) is used to supply power directly to the motor driver board.

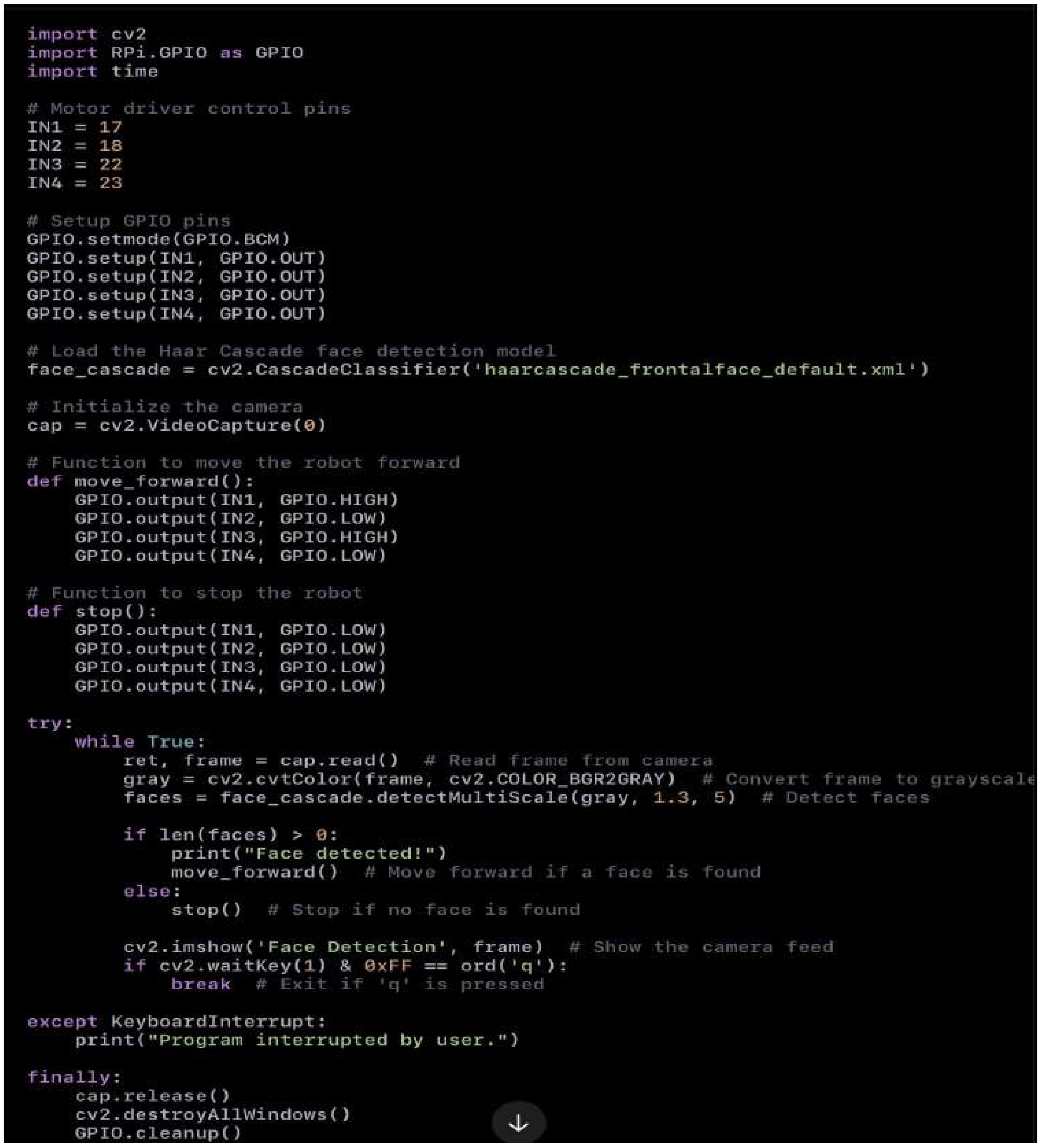
The positive terminal of the battery pack is connected to the VCC (or +12V) input on the motor driver board.

The negative terminal of the battery pack is connected to the GND on the motor driver board, and this ground is also shared with the Raspberry Pi to maintain a common ground reference.

The Raspberry Pi itself is powered separately using a 5V portable power bank connected to its USB-C power input port

.

Code Implementation



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

# Motor driver control pins
IN1 = 17
IN2 = 18
IN3 = 22
IN4 = 23

# Setup GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(IN3, GPIO.OUT)
GPIO.setup(IN4, GPIO.OUT)

# Load the Haar Cascade face detection model
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Initialize the camera
cap = cv2.VideoCapture(0)

# Function to move the robot forward
def move_forward():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)

# Function to stop the robot
def stop():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)

try:
    while True:
        ret, frame = cap.read()  # Read frame from camera
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # Convert frame to grayscale
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)  # Detect faces

        if len(faces) > 0:
            print("Face detected!")
            move_forward()  # Move forward if a face is found
        else:
            stop()  # Stop if no face is found

        cv2.imshow('Face Detection', frame)  # Show the camera feed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break  # Exit if 'q' is pressed

except KeyboardInterrupt:
    print("Program interrupted by user.")

finally:
    cap.release()
    cv2.destroyAllWindows()
    GPIO.cleanup()
```

Code Explanation

1. Import libraries

```
import cv2

import RPi.GPIO as GPIO

import time
```

- cv2 is OpenCV library for face detection.
- RPi.GPIO is used to control the motor driver from the Raspberry Pi GPIO pins.
- time is used for small delays if needed.

2. Set up motor driver control pins

```
IN1 = 17

IN2 = 18

IN3 = 22

IN4 = 23
```

- These four GPIO pins will control the direction of two motors.
- Each motor needs two pins to decide forward or backward.

3. Initialize GPIO pins

```
GPIO.setmode(GPIO.BCM)

GPIO.setup(IN1, GPIO.OUT)

GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(IN3, GPIO.OUT)

GPIO.setup(IN4, GPIO.OUT)
```

- Set the Raspberry Pi pin numbering style to BCM (based on GPIO numbers).
- Configure the motor control pins as outputs.

4. Load Face Detection Model

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

- Load a pre-trained Haar Cascade model that knows how to detect faces inside an image.

5. Open Camera

```
cap = cv2.VideoCapture(0)
```

- Start capturing video from the Raspberry Pi camera or a connected USB camera.

6. Define functions to move and stop

```
def move_forward():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
```

- Turns the GPIO pins HIGH/LOW to spin motors forward.

```
def stop():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)
```

- Turns all motor driver inputs OFF to stop the robot.

7. Main program loop

```
try:
    while True:
        ret, frame = cap.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

- Capture a frame from the camera.
- Convert the frame to grayscale (face detection works better in gray).
- Detect any faces in the frame.

8. Decide movement based on detection

```
        if len(faces) > 0:
            print("Face detected!")
            move_forward()
        else:
            stop()
```

- If faces are detected → move forward.
- If no faces are detected → stop.

9. Show camera feed and exit option

```
        cv2.imshow('Face Detection', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
        break
```

- Display the camera feed in a window.
- Allow exiting by pressing the ‘q’ key.

10. Clean up

```
except KeyboardInterrupt:
    print("Program interrupted by user.")
finally:
    cap.release()
    cv2.destroyAllWindows()
    GPIO.cleanup()
```

- Safely close the camera.
- Close the OpenCV window.
- Reset GPIO pins back to safe state.

Testing and results

Conclusion

The robot successfully demonstrates the capabilities of the Raspberry Pi in managing both real-time computer vision and hardware control. By detecting faces through live video feeds and responding by moving accordingly, the system showcases the power of embedded AI applications. The project highlights that using Raspberry Pi alone is sufficient for complex robotic systems that require significant computational resources, thus eliminating the need for external microcontrollers like Arduino.

References

1. Raspberry Pi Foundation. Raspberry Pi Documentation: General Purpose Input Output (GPIO). Raspberry Pi Ltd., https://www.raspberrypi.com/documentation/computers/raspberry-pi.html. Accessed 24 Apr. 2025.

2. STMicroelectronics. L298 Dual Full-Bridge Driver Datasheet. STMicroelectronics, https://www.st.com/resource/en/datasheet/l298.pdf. Accessed 24 Apr. 2025.

3. Python Software Foundation. RPi.GPIO Library Documentation. https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/. Accessed 24 Apr. 2025.

4. OpenCV Team. OpenCV-Python Tutorials. OpenCV.org, https://docs.opencv.org/master/d6/d00/tutorial_py_root.html. Accessed 24 Apr. 2025.

5. Raspberry Pi Foundation. PiCamera Python Library Documentation. https://picamera.readthedocs.io/en/release-1.13/. Accessed 24 Apr. 2025.
PDF