ELEC 424 Project 3: Motor Control

Team Members: John David Villarreal, Liam Waite, Ulises Moreno, Devin Gonzalez

April 25th, 2025

Project Goal and Steps

The goal of this project was to implement PWM-based motor control for an autonomous RC car using a Raspberry Pi. We began by initializing the LCD to display the Pi's IP address, which required switching from Raspberry Pi 5 to Pi 4 due to compatibility issues. Next, we tested PWM signals with LEDs on GPIO pins 12 (BCM 18) and 13 (BCM 19) at 50Hz, validating duty cycles of 5% (reverse), 7.5% (neutral), and 10% (forward). Finally, we calibrated the ESC by holding the calibration button until confirmation beeps, which deviated from the written instructions but proved necessary for proper functionality.

Challenges and Learnings

We faced three significant challenges during this project. First, the LCD failed to initialize on the Raspberry Pi 5 but worked perfectly on a Pi 4, revealing unexpected hardware compatibility issues. Second, GPIO conflicts required us to reboot the Pi to clear residual PWM states, emphasizing the importance of proper GPIO cleanup. Third, the ESC calibration process required holding the button rather than pressing it as instructed. These experiences taught us valuable lessons about hardware troubleshooting and documentation verification. This motor control approach could be effectively applied to robotic arms for precise servo control or drones for ESC-driven propulsion systems.

Screenshots

Note: The screenshots are located in the pages below.

Citations

This work draws from the following sources:

- User raja_961, "Autonomous Lane-Keeping Car Using Raspberry Pi and OpenCV". Instructables. `https://www.instructables.com/Autonomous-Lane-Keeping-Car-Using-R`

- Gradude, "ELEC 424 BeagleBone Project". Hackster. https://www.hackster.io/439751/gradudes-9d9a6e

Code [redacted] 34 lines (29 loc) · 716 Bytes [redacted]

```python
import RPi.GPIO as GPIO
import time
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18, GPIO.OUT)

def blink():
    p = GPIO.PWM(18, 1)
    p.start(1)
    input('Press return to stop:')   # use raw_input for Python 2
    p.stop()
    GPIO.cleanup()

def intensity():

    p = GPIO.PWM(18, 50)  # channel=12 frequency=50Hz
    p.start(0)
    try:
        while 1:
            for dc in range(0, 101, 5):
                p.ChangeDutyCycle(dc)
                time.sleep(0.1)
    #         for dc in range(100, -1, -5):
    #             p.ChangeDutyCycle(dc)
    #             time.sleep(0.1)
    except KeyboardInterrupt:
        pass

    p.stop()
    GPIO.cleanup()

blink()
intensity()
```

Figure 1: Key PWM initialization code from graDudes_script.py showing motor control setup. The code demonstrates PWM configuration at 50Hz with neutral duty cycle initialization.

main / main.py

Create main.py

Code Blame 21 lines (17 loc) · 310 Bytes

```
1   import RPi.GPIO as GPIO
2   import time
3
4   GPIO.cleanup()
5   GPIO.setmode(GPIO.BCM)
6   GPIO.setwarnings(False)
7   GPIO.setup(19, GPIO.OUT)
8
9   p = GPIO.PWM(19, 50)
10
11  p.start(1.5)
12  time.sleep(5)
13  p.ChangeDutyCycle(1.2)
14  time.sleep(5)
15  p.ChangeDutyCycle(1.8)
16  time.sleep(5)
17  p.ChangeDutyCycle(1.2)
18  time.sleep(5)
19
20  p.stop()
21  GPIO.cleanup()
```

Figure 2: Key code for the centralized programming intended to allow the RC to perform the conditions necessary for the demonstration.

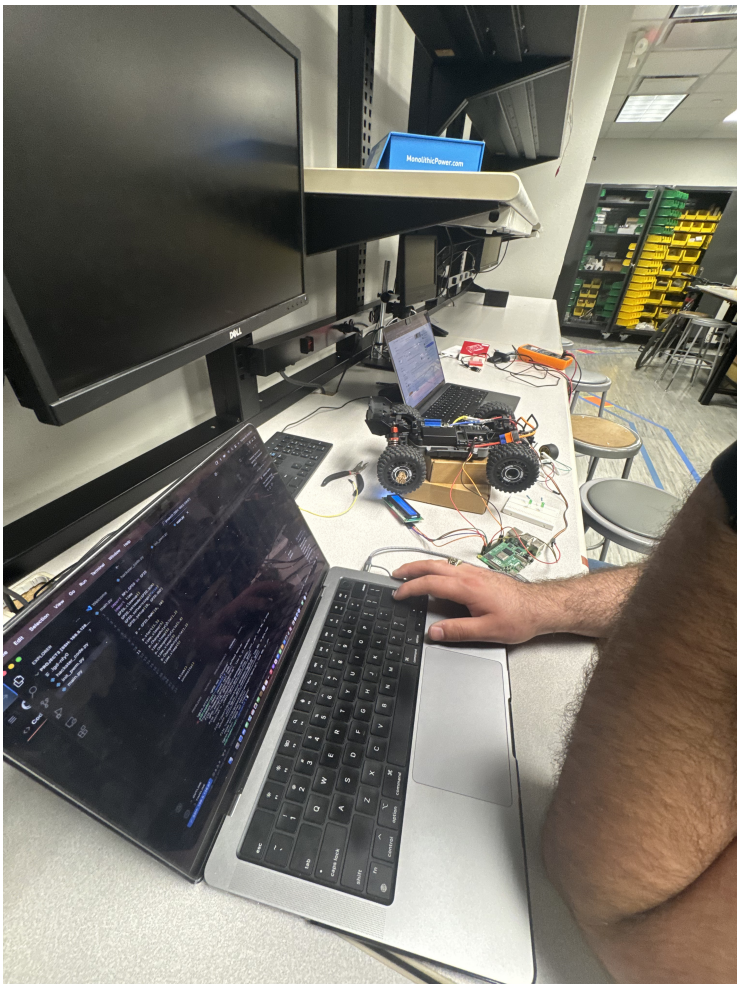

Figure 3: Complete hardware setup featuring Raspberry Pi 4 connected to RC car components, LCD display, and LED test circuits. The image shows proper wiring of motor control pins and power connections.