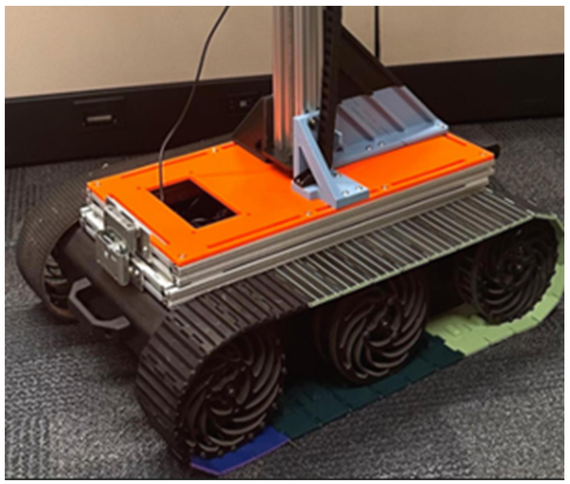
A New Approach to Robot Motor Control

Jacob Lopez, Wilbur Wright College – CCI Intern

FERMILAB-POSTER-24-0227-STUDENT

Introduction

The Remote Viewing Robot (RVR) is a robot designed to investigate issues that arise within the accelerator tunnels at Fermilab. These tunnels can emit harmful radiation, necessitating a remote solution to minimize human exposure. The RVR originally used a single Raspberry Pi for all functions, leading to potential performance bottlenecks and reliability concerns.



The RVR is designed to remotely investigate issues that arise within accelerator tunnels. The robot captures and collects information while minimizing human exposure to radiation.

The purpose of this project was to reallocate the RVR's motor control system by utilizing the Raspberry Pi Pico W. This aimed to achieve precise speed and torque control using Pulse Width Modulation (PWM). This would improve the robot's navigation reliability and overall system performance, while also reducing the risk of single-point failures.

- This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.
- This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Community College Internship (CCI)

The Pico W is a small, affordable, and versatile microcontroller that was chosen for its ability to handle Pulse Width Modulation (PWM) at a compact size. It allowed for precise motor control and the off loading of the motor functions from the main Raspberry Pi.

Materials & Methods

This project made use of the Raspberry Pi Pico W, Pittman Lo-Cog DC Servo Motors, MicroPython, the VS Code editor, jumper wires, a power supply, and a motor driver. Motor control was offloaded to the Pico W to reduce the main Raspberry Pi's processing load. This was done via programming the Pico W with the help of MicroPython. The software's modular approach separates initialization, control, and stopping functions. This helps ensure program maintainability and readability. PWM was implemented to manage motor speed and torque, with the ability to output adjusted duty factors for precise control.

Brushed DC Motors (Pittman Lo-Cog DC Servo Motors) used in the RVR. These motors are selected for their affordability, reliability, and responsiveness to PWM control, which is crucial for the RVR's precise navigation and task execution.

Results & Conclusion

The new system successfully tested precise speed and torque control on prototype motors. The Pico W will now handle motor control, freeing the main Raspberry Pi for higher-level functions. This setup minimizes single-point failures, ensuring continuous operation even if one component fails.

```python
def control_motor(motor, direction, duty_factor):
    #Use global variables to keep track of PWM instances for motor pairs A and B
    global pwm_A, pwm_B
    if motor == "A":
        #Initialize PWM for motor pair A if not already initialized
        if pwm_A is None:
            pwm_A = PWM(Pin(MOTOR_A_PIN))
            pwm_A.freq(1000)
            pwm_A.duty_u16(0)
        #Set up direction control pin for motor pair A
        dir_pin = Pin(MOTOR_A_DIR_PIN, Pin.OUT)
        pwm = pwm_A
    elif motor == "B":
        #Initialize PWM for motor B if not already initialized
        if pwm_B is None:
            pwm_B = PWM(Pin(MOTOR_B_PIN))
            pwm_B.freq(1000)
            pwm_B.duty_u16(0)
        #Set up direction control pin for motor B
        dir_pin = Pin(MOTOR_B_DIR_PIN, Pin.OUT)
        pwm = pwm_B
    else:
        print("Invalid motor selection. 'A' or 'B'.")
        return
    if direction == "forward":
        #Set direction to forward
        dir_pin.value(1)
    elif direction == "reverse":
        #Set direction to reverse
        dir_pin.value(0)
    else:
        print("Invalid direction. Choose 'forward' or 'backward'.")
        return
    #Convert duty factor (0-100%) to the range (0-65535) for PWM and set the duty cycle
    pwm.duty_u16(int(duty_factor * 65535 / 100))
```

The control_motor function initializes PWM, sets motor direction, and adjusts speed using a duty factor for precise control.

The new motor control system enhances the RVR's performance and reliability. Offloading control to the Pico W prevents overloads and failures. PWM and duty factor adjustments allow precise control, crucial in accelerator tunnels. The modular structure improves code readability and maintainability. This project improves the RVR's ability to diagnose and address issues in tunnels, minimizing human exposure to hazards.