A New Approach to Robot Motor Control

Jacob Lopez

(Dated: 31 July 2024)

This essay details the motor control improvements to Fermilab's Remote Viewing Robot (RVR). It is a robot tasked with remotely investigating issues within the accelerator tunnels at Fermilab. Initially controlled by a single Raspberry Pi that housed all the robot's operations, the RVR will now use a Raspberry Pi Pico W for its motor control. This was achieved using Pulse Width Modulation (PWM) to allow for precise speed and torque control for the robot. This enhances the robot's reliability. By addressing the challenge of navigating high radiation environments, the upgrade helps assist the RVR's goal of decreasing the need for human intervention in accelerator tunnels.

I. INTRODUCTION AND BACKGROUND

Within the particle accelerator tunnels at Fermilab, issues may arise. Equipment may malfunction, water leaks may occur, and routine maintenance may be needed. When problems like these occur, fixing them as soon as possible is essential to minimize downtime. However, technicians addressing these issues may be unnecessarily exposed to radiation in the process.

The Remote Viewing Robot (RVR), displayed in figure 1, is designed to circumvent the human necessity to investigate problems within the tunnels. When a problem arises, the RVR enters the tunnels to investigate and collect information, such as what went wrong and what might have been the root of the problem. This allows technicians and engineers to view the errors remotely and safely. From there, a plan may be made to optimally fix this issue, helping to decrease potential hazard exposure time.

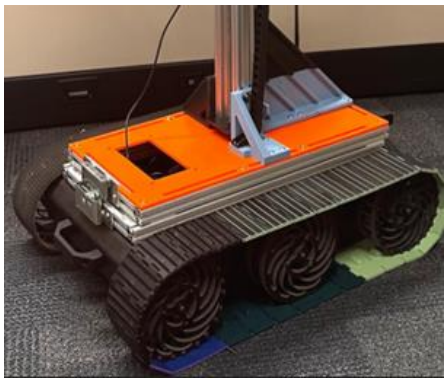

FIG 1. Remote Viewing Robot

II. SYSTEM ARCHITECTURE

All of the Remote Viewing robot's functions, such as motor control, data collection, and communication, are handled via a single Raspberry Pi. Although the Raspberry Pi is capable of handling these tasks, limitations exist. Performance bottlenecks may occur due to the Raspberry Pi becoming overburdened when handling multiple tasks. To help mitigate these potential issues, the current plan is to offload the robot's function handling from the main Raspberry Pi onto other microcontrollers, specifically, the Raspberry Pi Pico W (figure 2). These Pico W microcontrollers would take on dedicated tasks of their own and would, therefore, reduce the processing burden on the main Raspberry Pi. The main Raspberry Pi would remain the central command unit to orchestrate the robot's operations. In contrast, the necessary Pico W units would be the driving force that executes them. This makes the system with which the RVR runs on, more reliable as this distribution approach reduces the risk of a single point of failure.

FIG. 2: Raspberry Pi Pico W

III. CHOICE OF MICROCONTROLLER

The Raspberry Pi Pico W was chosen as it is a small and affordable microcontroller accessible for various applications that are useful for the RVR. Its small size allows for multiple units to be used for the robot's various functions without taking up much space. The microcontroller also has a range of input and output options that include support for Pulse Width Modulation, which is the technique that will be used to control the speed and torque output of the motors on the RVR [1]. MicroPython, a simple and highly readable programming language, is used to program the Pico W. To do this, the Pico W must be able to receive information from the Raspberry Pi, such as the pair of motors that are desired to be controlled, the direction with which they are desired to move, and the duty factor they are desired to have. After taking in this information, the Pico W would handle this input and perform the desired motor controls.

IV. MOTOR SPECFIFCATIONS

The RVR's movement uses two pairs of Brushed Direct Current (DC) Motors (figure 3) that allow the robot to traverse the accelerator tunnels. The RVR uses brushed DC motors as their affordability aligns with making the RVR an economically viable solution to the danger of hazard exposure within the tunnels. Brushed DC Motors also respond well to Pulse Width Modulation, which is crucial for our system as it will allow for speed control on the RVR.

FIG. 3: Pittman Lo-Cog DC Servo Motors used on the RVR

V. PULSE WIDTH MODULATION (PWM)

To allow precise motor control of the RVR, the robot uses pulse width modulation (PWM). PWM controls the amount of power delivered to the motors. In this case, the amount of power delivered affects the speed and torque of the robot. PWM rapidly switches between the on and off signal states [2]. The duration the signal stays on versus off per cycle is called the duty factor. The Pico W must be able to take in a duty factor and output it onto the motor accordingly. PWM is advantageous in that it allows us to finely control the power delivery to the motors, allowing for precise adjustments to the speed and torque of the robot. If there were no PWM involved, there would be poor control over the robot's speed and movement, as the simple fully on-or-off approach would cause the robot to run at its maximum speed when it is on and not run at all when off.

VI. DUTY FACTOR

A duty factor is a parameter of PWM that represents the proportion of time a signal is on within a given period. The formula for duty factor is the duration of the PWM cycle when the signal is on divided by the entire duration of the PWM cycle, all multiplied by one hundred. A complete PWM cycle includes the time the signal is on and off. A duty factor of zero would mean that the motor would be entirely off, as no power would be delivered. A duty factor of fifty percent would mean that the signal is on for half of the period and off for the other. The motor would run at moderate power, balancing power delivery and energy efficiency. This would put the robot's movement at approximately half its maximum speed. A Duty Factor of one hundred percent would mean that the signal would continuously be on and delivering full power, running the motor at maximum speed. This would be as if the motor was fully on and supplied with a constant direct current. With a Duty Factor, we can make precise adjustments to the motor speed and torque for specialized situations by setting the Duty Factor to various states as needed. Figures 4 and 5 illustrate these concepts.

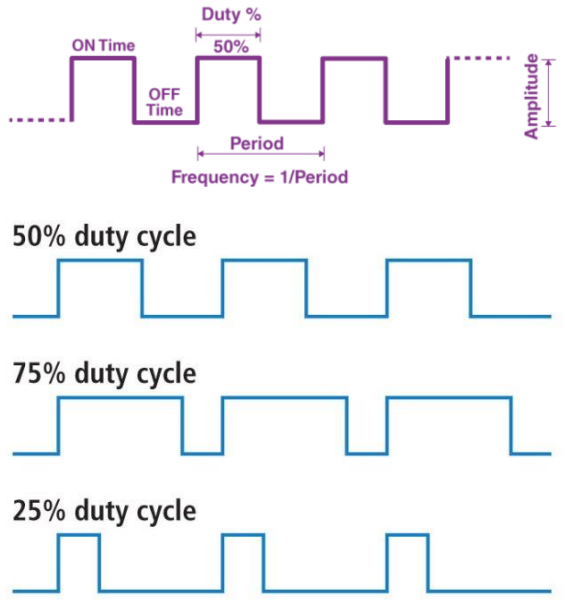

FIG. 4 & 5: Duty Factor concept & 50%, 75%, and 25% Duty Factor Examples

VII. IMPLEMENTATION

The processes of setting up the PWM channels, defining control logic, and implementing stop functions were established to implement the motor control program within the Pico W. Initially, PWM channels on the Pico W are set up for each pair of motors on the RVR. Any two pins on the Pico W can be initialized for the two pairs of RVR motors (A and B), where each pair follows the same direction. Each pin corresponds to one pair of motors, making this a binary decision. Likewise, any two pins can be chosen as the motor's directional pins, with each directional pin corresponding to one pair of the RVR's motors. The state of the pin (on or off) would dictate whether the motors would move in a forward or reverse direction.

The control function sets the direction and speed of the chosen motor pair using a provided duty factor. This provided duty factor determines how long the PWM signal remains during each cycle. The function first checks to see if the provided motor pair is either motor pair A or B. After checking, the specified motor pair is initialized to be controlled with the corresponding PWM channel PIN on the Pico W. This initialization of matching a Pico W PIN with a motor pair only occurs if motor control has not already been initialized, if it has, the directional control pin that corresponds to the chosen motor pair is then set on or off based on the desired movement direction (forward or reverse). The provided duty factor is then used to adjust the motor's speed.

The program includes stopping functions to shut down the motors safely. These functions set the duty factor to zero, cutting off power to the motors and stopping them. For individual motor control, the stop_motor function checks if the specified motor pair (A or B) is turned on. If so, the duty factor of the specified motor is set to zero, stopping the motor. For stopping all motors simultaneously, the stop_all_motors function sets the duty factor of both motors to zero, even if only one motor is in use. These functions ensure that all motors are safely stopped, crucial for maintaining motor control and preventing damage to the robot in its operating environment.

VIII. SOFTWARE AND COMMUNICATION

The Pico W and its program written in Micropython will be connected to the main Raspberry Pi system. The Pico W will receive commands as to which motor to control, the desired direction, and the duty factor for the speed. The main Raspberry Pi would output the commands, which would then be processed and received by the Pico W to carry out. The modular structure of this program was chosen for the sake of simplicity and effectiveness. Separating the initialization, control, and stopping functions allowed each system part to be tested and modified individually.

IX. TESTING

Testing was conducted on a pair of prototype motors to validate the new motor control system for the RVR. This included ensuring that both motor pairs could operate concurrently without conflicts. Forward and reverse directional controls were verified for each robot's motor pair. Various duty factors were tested to see their effects on the motor's speed and torque. Edge case testing ensured

boundary values, such as a zero or one hundred percent duty factor, were correctly managed. Unit testing of individual functions, such as PWM initialization and duty factor adjustment, was conducted to verify that each functioned correctly before implementation. These tests help ensure that the RVR's motor control system is robust in its reliability and capable of performing the required operations.

X. CONCLUSION AND OUTCOMES

With the successful system testing, moving the motor control of the RVR from the Raspberry Pi to the Pico W should now help enhance the robot's performance and reliability. The new setup of the RVR will now allow the robot to operate even if individual parts fail. Using PWM and the ability to adjust duty factors, precise motor control for speed and torque is possible. The programming of the Pico W was designed using a structure that separated the motor's initialization, control, and stopping functions. The main Raspberry Pi is liberated to focus on more complex and critical functions by delegating the low-level motor control tasks to the Pico W. This enhancement not only helps to increase the RVR's efficiency but also boosts the robot's operational capabilities. With this more reliable and efficient motor control system, the RVR's ability to diagnose and address issues within the accelerator tunnels will help minimize human exposure to hazardous radiation. Ultimately, this innovation will reduce the time and effort needed to resolve accelerator issues, improving safety and efficiency.

XI. ACKNOWLEDGMENTS

I extend my profound gratitude to Brian Hartsell, Jason Morin, Adam Watts, and the AD Robotics Initiative. Your mentorship, assistance, and guidance throughout this summer have been truly invaluable and are deeply appreciated.

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).

[1] Raspberry Pi Foundation, "Raspberry Pi Documentation: Raspberry Pi Pico and Pico W", (2012-2024), https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

[2] Rohini College of Engineering and Technology, "Pulse Width Modulation", Research Center for Educational Technology (RCET), Kent State University, (2001), https://www.rcet.org.in/uploads/academics/rohini_58510678608.pdf

[3] Micropython, "class PWM – pulse width modulation", (July 25, 2024), https://docs.micropython.org/en/latest/library/machine.PWM.html