COMP ENG 4DS4

Project 1 - Towards A More Realistic System

Professor: Mohamed Hassan

Submitted on *March 14th, 2025* by: GROUP 07

Aidan Mathew - mathea40 - 400306142

Aaron Rajan - rajana8 - 400321812

Sameer Shakeel - shakes4 - 400306710

Chris Jiang - jiangc46 - 400322193

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. **Submitted by Aidan Mathew, Aaron Rajan, Sameer Shakeel, and Chris Jiang.**

Declaration of Contributions:

All students contributed to the lab report and did the write-up.

Student	Contributions
Aidan Mathew	Worked on RC Component
Aaron Rajan	Worked on Motor Component
Chris Jiang	Worked on LED Component
Sameer Shakeel	Worked on RC Component

Project Details

5.1 - Controlling the Car using the Controller

The control of the car's movement in this project is achieved through a joystick interface, with the firmware developed to manage the speed and direction of a DC motor and the steering angle of a servo motor. The system uses FreeRTOS, allowing real-time task scheduling and communication through queues. The firmware architecture is modular, with separate components handling motor speed control, servo steering, and LED indication, each designed to respond dynamically to user inputs.

The speed of the DC motor is regulated using pulse-width modulation (PWM) signals. The joystick's vertical axis is mapped to three discrete speed modes: fast, moderate, and slow, each corresponding to a predefined PWM duty cycle. These duty cycles are used to control the effective voltage supplied to the DC motor, thereby adjusting its rotational speed. The firmware includes a task named motorTask, which is responsible for applying the appropriate PWM signal based on the received speed mode. This task runs continuously and retrieves speed commands from a FreeRTOS queue named motor_queue. The use of a queue allows for asynchronous communication between the input processing module and the motor control logic, ensuring smooth and immediate response to user actions.

In addition to speed control, the firmware provides a visual indication of the current speed mode using the onboard RGB LED. This functionality is implemented through the LED component module. Within the motor control logic, the speed mode is linked to specific RGB values. For instance, when the fast speed mode is selected, the LED emits a red light; in moderate speed mode, it turns orange or yellow; and in slow mode, another color such as green or blue is used. This real-time visual feedback aids the user in quickly identifying the current operational state of the car without the need for additional display hardware. The LED control is seamlessly integrated into the main control loop, with the color state updated whenever the speed mode changes.

Directional control is implemented via a physical switch connected to the controller, allowing the user to toggle between forward and reverse motion. The state of the switch is polled by the firmware, and depending on its position, the direction of the motor's rotation is adjusted by reversing the polarity of the PWM signal applied to the motor driver. This feature enhances maneuverability and provides flexibility in navigation, particularly in confined or obstacle-rich environments.

The car's steering mechanism is handled by a servo motor, which adjusts the angle of the front wheels. The joystick's horizontal axis is mapped to discrete angle values, which are sent to the servo motor via a second FreeRTOS queue named angle_queue. A dedicated task, positionTask, monitors this queue and updates the servo motor's position accordingly. The servo motor angle is

calculated to ensure smooth transitions and precise control of the car's heading direction. The use of a queue ensures that steering commands do not interfere with motor speed control, thereby maintaining modularity and stability within the system.

The setup functions in the code, such as setupMotorComponent, initialize the necessary hardware interfaces and create the FreeRTOS queues and tasks. Error handling is included to ensure that the system halts execution if any queue or task fails to initialize, thereby preventing undefined behavior. The structure of the firmware emphasizes reliability and responsiveness, ensuring that each component—speed control, direction control, and steering—operates efficiently and without interference.

In summary, the car control system is designed with a focus on real-time responsiveness and modularity. Through the use of FreeRTOS tasks and queues, the system achieves effective separation of concerns, enabling simultaneous management of speed, direction, and steering. The inclusion of RGB LED feedback further enhances usability by providing intuitive visual cues. The implementation reflects a robust embedded systems approach, combining real-time operating system principles with precise motor control techniques.

5.2 - Controlling the Car using the Terminal (Optional)

Not Attempted.

5.3 - Adapting the Car Speed Dynamically using The Accelerator (Optional)

Not Attempted.