CE 4DS4 - Project 1: Towards a More Realistic System

Professor: Dr. Mohamed Hassan

Section L04

Khaled Hassan, hassak9, 400203796
Nikola Petrevski, petrevsn, 400198379
Zekai Yang, yangz203, 400308646

Due: Thursday, March 14, 2024

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: Nikola Petrevski (petrevsn, 400198379), Khaled Hassan (hassak9, 400203796) & Zekai Yang (yangz203, 400308646)

Declaration of Contributions

Nikola and Khaled predominantly collaborated on the RC and Motor components, and had contributions in the LED component. Zekai mainly focused on the LED component. All three of us worked on the bonus Accelerometer component together. The code that we compiled for this project can be found under the “repo/” directory in our group’s github repository.

RC

The RC component is responsible for decoding the data coming from the remote controller and sending that data to the appropriate module, whether that be the motor or LED components. In addition to the provided skeleton code, we relied heavily on the exercises and experiments conducted in previous labs. In this case, we relied on the code in Experiment 5 from Lab 2 to allow for UART communication between the remote controller and the FMU board. In our implementation, we created a struct of RC_Values containing 7 16-bit unsigned integers, one for the header and 6 for the 6 channels we are reading from the controller. Channels 1 to 4 correspond to the horizontal and vertical values of the two joysticks. Channels 5 and 6 correspond to the two switches on the top right of the controller, which we decided to use for the speed mode and direction, respectively. The task that was created for this component, rcTask, receives and decodes the necessary RC values, then uses them to calculate the required motor speed (DC motor PWM) and motor angle (Servo motor PWM). These values are then sent to the motor queue and the angle queue, respectively, to be received by the Motor component. The speed mode, represented by the current value of channel 5, is also sent to the LED component the same way, via a queue.

Motor

The main function of this component is to receive the values from the RC component and update the speed of the DC motor and the angle of the servo accordingly.

To do this, there are two external queues that are used to receive the values from the RC components. There is motor_queue which is used to handle the DC motor speed, and angle_queue which is used to handle the angle of the servo motor. This component only receives the final values of the speed and angle. That is, any calculations that are done to account for the direction or speed mode are already done in the RC component before they are put into the queues. The motor component receives these values as floats.

In terms of setup, there are separate setupServo() and setupDCMotor() functions to set up the PWM for each motor. There is a task for the DC motor and a task for the servo motor, and all these functions do is update the PWM duty cycle to each motor as received from the queues.

The values that are received from the RC component are updated continuously. That is, the motor component does not wait for a change in value to update the PWM.

LED

The led serves as an indicator for three speed modes: slow, medium, and fast, represented by the colors red green blue respectively. Similar to the previous lab0, The LED's functionality is implemented in the Led_component.c file, where the LED pins are initialized and configured. A dedicated LED queue receives data from RC channel 5, which corresponds to the speed modes, with values of 1000, 1500, and 2000 indicating slow, medium, and fast speeds, respectively. A while loop continuously monitors the incoming data from channel 5, using if statements to check for color change and adjust the LED color based on the selected speed mode. Additionally, we keep track of the previous color by storing it in a value called lastcolor. This allows us to change the color of the LED dynamically by invoking the togglePin() function.

Accelerometer

Similar to the above, we used the code in Experiment 4B from Lab 1 as the setup for this component to set up SPI communication between the FMU and the accelerometer sensor. In our implementation, the accelerometer sensor measures the tilt of the rover in both the x and y axes. We use the x_angle value to determine the incline the rover is going in, since we only care about the rover's tilt in that direction affecting speed. Similar to the above, an external queue is initialized in the motor component's header file called tilt_queue. The accelerometer component calculates the x angle and sends it to the motor component, which factors the x angle into the PWM angle value it had received from the RC component before updating the DC motor's duty cycle accordingly.