Project 1: Towards a More Realistic System

1 Project Rules

- You have to stick to your Lab slot assigned to you on Mosaic.
- For the rest of labs (Project 1 and Project 2/Lab3), we will join forces to have groups of 4.
- Prepare a demonstration for all the Lab experiments, and get ready to be asked in any part of the experiments.
- The demonstrations of Project 1 will be held starting from **Mar 11th at the first hour of each lab slot**.
- Each team needs to submit one report for all the members, and the first page of the report should contain the team number and the names of its members.
- The submission should be through github classroom at 12pm on the day of your demo. Put the report in a PDF format. submission will be through github classroom.
- The first page (After the title page) of the report must include a **Declaration of Contributions**, where each team member writes his own individual tasks conducted towards the completion of the lab.
- You also need to submit all source files that you modify or add through out the lab.

General Note:

- Make sure to push all your code to the assignment repository from the lab computer before leaving the lab room because your saved work may be deleted after the lab slot.

2 Github Classroom

You should accept the lab assignment through this link:
https://classroom.github.com/a/Q-t2bjOr

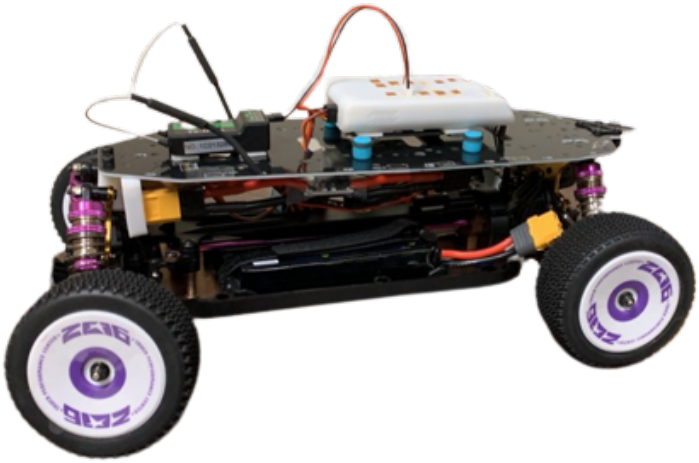
3 Project Skeleton

Please note that a project skeleton is released for you to help you get started. This skeleton dictates for each task the required functions to be implemented such that the tasks are executing as expected towards the full functionality of the project. Adding more functions is permitted as needed. However, you cannot remove a function from the one given in the skeleton. **The code skeleton is already provided in the starter code in your assignment repository.**

4 Project Components

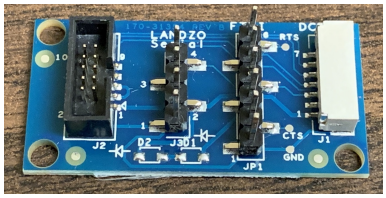
You will use the following modules in this project.

1. 1x Assembled car (Please note that the following picture does not have the uppermost layer of the car, as it is not required for this project).



2. 1x Segger J-Link EDU Mini debugger + 1x micro USB cables.

3. 1x Debug breakout board + 1x 7-wire cable.



4. 1x FlySky controller (radio transmitter).
   .

5. 2x Telemetry modules + 1x 6-wire cable + 1x micro USB cables.

5 Project Details

The main goal of this project is to fully control the car using several ways as follows:

1. The radio controller (**Mandatory**)
2. The terminal (**Bonus**)
3. Adaptively using the accelerometer (**Bonus**)

5.1 Controlling the Car using The Controller

The controller should be able to control the speed of the DC motor using one of its joysticks. In addition, the firmware of the FMU should define 3 modes of speed, and the controller will switch between them. The firmware, also, should utilize the on-board RGB LED to indicate the selected speed mode, for example, the LED turns red if the fastest speed mode is selected, and it turns orange or yellow for the moderate speed mode and so on. Besides the speed modes, there should be a switch to select forward or reverse motion for the car. Lastly, the front wheels position will be controlled as well through the controller by changing the angle of the servo motor.

5.2 Controlling the Car using The Terminal

On the other hand, the terminal will be used to control the car similar to the controller but with limited functionalities. The keyboard keys 'w' and 's' are used for moving the car forward and backward, respectively, with a fixed speed, while the keys 'a' and 'd' are for turning the wheels to the left and right, respectively, with the maximum angle. The action of the keys will remain as long as one of the keys is pressed, and once all the keys are released (during a small period of time) the car stops moving and the wheels return to their original position. Additionally, the terminal will be used for printing the log messages to keep track of the car state, and these messages can be used for debugging.

5.3 Adapting the Car Speed dynamically using The Accelerometer

The last component in this project is the accelerometer, which is needed to determine the terrain's inclination and provide speed adjustment accordingly. For instance, if the car moves downhill, the accelerometer will calculate the angle of the slope and decrease the speed of the motor, according to the angle, so at the end the speed of the car remains constant as chosen from the controller.

6 Project Software Components

In order to fulfill the project requirements in a modular way, the project can be broken down into a number of components, where each component serves a certain purpose. The component should contain at least one task, and it can utilize the synchronization and data sharing techniques to work in harmony with the other components. Figure 1 shows a possible breakdown for the project's components and how they communicate with each other, and the description of the roles of the task are listed in Table 1.

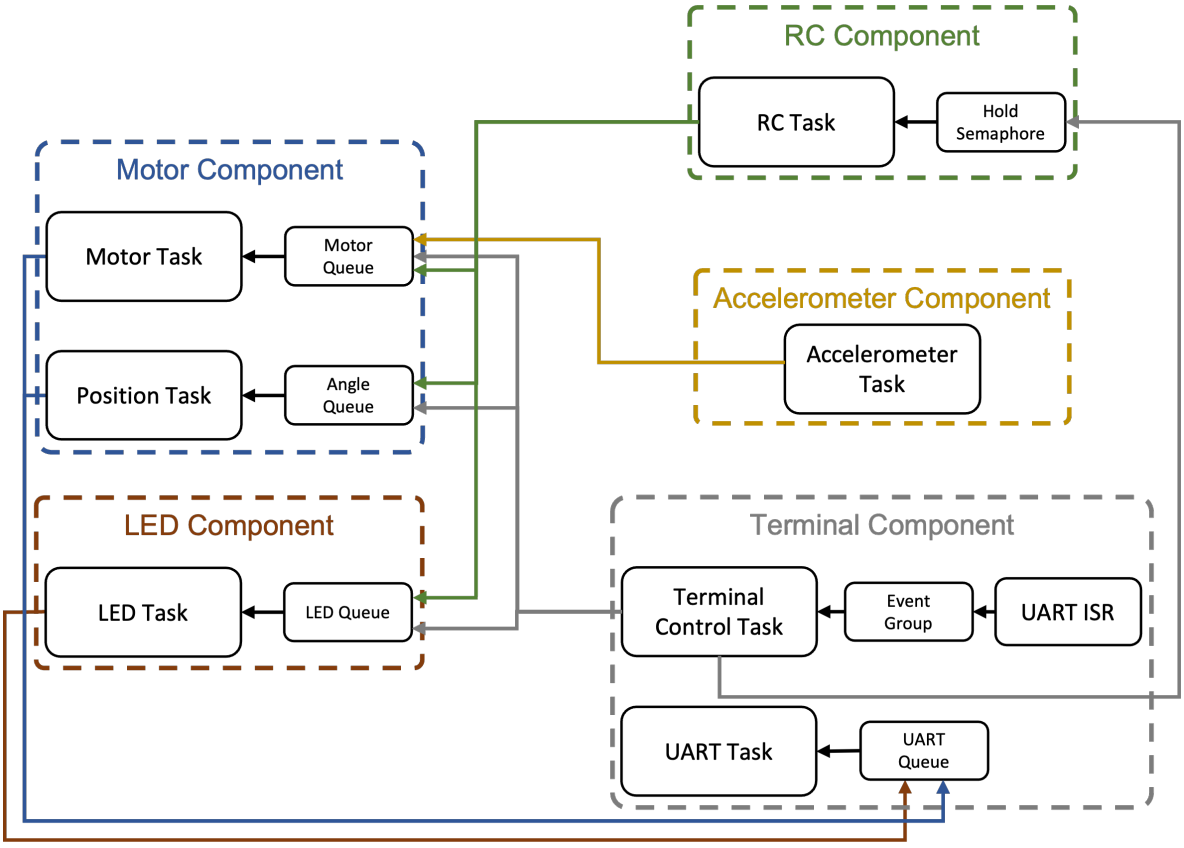


Figure 1: An overview of the project's components. **Accelerometer and Terminal components are bonus deliverables for extra marks**.

Table 1: The roles of the tasks in the project

Task name	Description
Motor Task	This task is responsible for controlling the DC motor's speed by calculating the PWM duty cycle and control the FTM module. The task waits for messages that come from Motor queue and change the speed accordingly. The message can contain speed, speed mode, and/or speed compensation (speed adjustment) depending on which task sends this message.
Position Task	This task controls the position of the front wheels by controlling the servo motor. It waits for an angle value to arrive at Angle queue and change the PWM signal based on the received value.
LED Task	This task is responsible for changing the LED color according to the chosen speed mode. The speed mode value is sent to the task via LED queue.
RC Task	This task is responsible for decoding the data coming from the radio receiver, and it sends the corresponding values to Motor, Angle, and LED queues. The task can be paused if it receives a signal from Hold semaphore.
Accelerometer Task	This task keeps reading the tilt angle from the accelerometer and sends a speed compensation to Motor queue.
Terminal Task	This task receives events from the event group and act if any of 'w', 'a', 's', or 'd' is pressed. The task responds to the events by sending a fixed value to Motor, Angle, and/or LED queues. Additionally, the task signals Hold semaphore to pause RC task as long as it receives events, and after the events stop for a certain period, it allows RC task to resume.
UART Task	This task waits for strings to arrive at UART queue and prints it on the terminal (not Semihost console.)