Real-Time Systems

Lab 6: Reentrancy

Due: April 12, 2018 (Before Class)

Introduction

In this lab we will address reentrancy in our RTS. So far we've been using whatever functions, libraries, etc. were helpful to achieve our objectives. But these functions likely are not reentrant, indicating they cannot be guaranteed to operate properly under all timing conditions. So to complete our Ringo software stack we need to remove our dependence on Plum Geek's software and ensure all our code is reentrant. As a result, in this lab you will:

1. remove **all** Plum Geek and any other external support libraries. The only libraries/files you may keep are `Adafruit_NeoPixel`, and `Arduino_FreeRTOS` library as long as any functions you use from those libraries are reentrant. It is safe to assume that functions in `Arduino_FreeRTOS` are reentrant.
2. rewrite all the additional functionality you need as reentrant functions that can be called by your tasks
3. demonstrate that you can still accomplish the mission from Lab 4 and Lab 5 with all your reentrant functions

Possible Approaches

As an example, consider the function in the Plum Geek API that sets the motor speed `Motors(int LeftMotorSpeed, int RightMotorSpeed)` in the `RingoHardware` library. This function accesses a shared resource and a shared variable and hence is not reentrant. Here are a few possible approaches you could take:

- Copy this function into your code and modify it to be reentrant.
  - You would also need to do that with any functions called by the `Motors(int LeftMotorSpeed, int RightMotorSpeed)` since a reentrant function can't call a non-reentrant function.
- You can write new (improved) functions that do whatever you want them to do as long as any functions in the code are reentrant.
- You can more aggressively reconsider your code design. That is, maybe in your code the motors are only called by one task, which means they don't really need to be in a function at all. You could move that functionality directly into the task. You can move functionality between "functions" and "tasks" as you like. Recognize, however, this increases your odds of introducing new bugs.

Requirements

No matter which method you use, here are the requirements for the lab:

- The code you've written this semester using FreeRTOS, that is, the controller, planner, and guidance, all needs to run without any external libraries except
  1. `Adafruit_NeoPixel`

2. `Arduino_FreeRTOS`

- Ringo will need to accomplish its mission from the past two labs without using any Plum Geek libraries.
- You need to have **at least** 5 reentrant functions even if you refactor your code to move some functionality previously in a function into a task.
  - This means you can't just move all the code from the various Plum Geek libraries into your existing tasks (though this wouldn't help with the shared data problem anyway).

Protips

- Start this lab early and come up with a plan of attack. This lab could be very difficult since you will likely need to drill down the function call chain all the way to the functions that interact with the hardware directly.
- I think the best route to success will be to modify existing functions (rather than write all new ones) as this will minimize debugging. However, we know there are some suboptimal design choices in the existing functions, so you may want to fix some of them as well to make your life easier.

Reminder on Reentrancy

If a function can be safely executed concurrently, then it's called reentrant. In class we narrowed this down to a few requirements. A function:

- Must not hold or return pointer any static (or global) non-constant data.
- Must work only on local data and the arguments provided by the caller.
- Must not call non-reentrant functions.
- Must not rely on locks to singleton resources.

Remember that timing bugs don't always show up predictably. A non-reentrant function might be called millions of times without a problem, but then one time the timing of a task preemption could lead to a shared data bug. This is why it's important to follow real-time programming best-practices and write your functions to be reentrant in the first place.

What to submit

1. **(35 points)** Zip the entire Arduino project so that we can just unzip and execute the source code.
   - Please include a diff of the source code between this lab and previous one. Please use a "diff" tool of some kind that highlights the changes from previous files (e.g. `https://www.diffchecker.com/`, meld, github, etc.).
   - Your code cannot use any Plum Geek libraries. If I see them in the code or zipped folder I will dock points.
2. **(30 points)** Documentation of the task(s) you have implemented for this lab. (0.5-1 page long)

- Provide a one paragraph explanation of how you made your functions reentrant as well as your overall approach for this lab in removing Plum Geek's libraries. That is, what functionality did you move into tasks vs. writing new functions, etc.
- Please list in your writeup all the new reentrant functions you wrote so I can identify them in the code.
- If for some reason you can't finish the lab and you still need some functionality from Plum Geek's libraries, tell me what functions you use and I will give you partial credit. Make sure you turn something in for partial credit at least!

3. **(35 points)** Record a short video of Ringo achieving the same mission you did in the video for Lab 4 and 5. To help me know you aren't just submitting the same video include some audio or timestamp in the video with the date you recorded the video.
    - Be sure to describe or narrate the test that Ringo is doing that demonstrates that you have accomplished the objectives of the lab.
    - Please upload the video to YouTube, Vimeo, or something similar and then provide me with a link. Just put the link somewhere in the writeup you did in #2 above.
4. Upload all of this into "handin" at `https://cse-apps.unl.edu/handin`