Real Time System
Lab 1: Getting Started

Deadline: 1/19/17 (Before Class)

Setting Up

To set up your computer for use with the Ringo robot, visit: *http://www.plumgeek.com/getting-started.html*

Follow the instructions there before you plug the Ringo in - they will guide you through installing drivers, the Arduino IDE, and Ringo's software libraries. I recommend watching the Getting Started video (use 1.5x speed if you are comfortable with the material).

After finishing the setup process, scroll up to the navigation bar on the Getting Started page, and navigate to "Robots" $>$ "Ringo" $>$ "Getting Smart".

Find the "Reference Code" section and download the "Ringo Base Sketch". This will be your project folder. You can also have some fun with their preloaded behaviors to check out how they coded the Ringo for use without an RTOS.

Implementing FreeRTOS

Because we are not interested in using the Arduino's default "`loop()`" functionality, we will need to find a way to implement tasks. Visit: *https://github.com/feilipu/Arduino_FreeRTOS_Library*

Download the library and move to your Libraries folder where you installed the Ringo libraries.

Now open up your Ringo Base Sketch project (mine is "`Ringo_Base_Sketch_Rev`*XX_XX*`.ino`"). In the "Sketch" menu of the Arduino IDE, click "Include Library" and click on "FreeRTOS" under the Contributed libraries. This should add a large block (or just one) of "`#include`" statements to the beginning of your code.

Building Tasks

In your FreeRTOS library folder navigate to "`/examples/Blink_AnalogRead`". Open the "`.ino`" file and study how it builds its tasks, and use that knowledge to build your own task in your Ringo Base Sketch file. Some lines in the Analog Read file aren't relevant to what we're currently interested in, so focus on the syntax of task creation and definition. This may call for some trial and error - just keep trying out new ideas, and you'll be able to get it.

Once you're confident that you have your task laid out correctly in the Base Sketch, try adding some functionality using some of the functions from the other files in your project folder like "`FunStuff.h`". When your project verifies, connect the Ringo via the programming chip (check out the Plum Geek Getting Started video if you aren't sure how to do this) and a light will activate on the robot.

Before you upload your code to the Ringo MAKE SURE that in the Arduino "Tools" menu, "Board" is set to "Arduino Fio" and "Port" is set to whatever new port pops up when you connect the Ringo. Then upload your code to the Ringo and test out your task!

What to submit

Send an email with all your artifacts as attachments. If required, videos can be uploaded somewhere else and then send a link to the video.

- Zip the entire Arduino project folder so that we can directly unzip and execute it. **(20 points)**
  - Provide a diff of source code. It can be a simple text-diff (ex: *meld*) or something more detailed like a diff from a version control system (assuming base sketch from PlumGeek as base revision). **(10 points)**
- A documentation of your experiences with the setup process. Particularly focus on the challenges you faced during the setup and how you resolved them. Also, describe the task that you have implemented. Clearly state the expected behavior and any shortcoming that you are aware of. **(40 points)**
- Record a short video of the robot performing the implemented task. **(30 points)**

Note:- If for some reason, you are not able to implement a task of your own but were able to setup Ringo with the IDE, demonstrate the robot with one of the predefined behaviors and record a video of that. **Partial Credit**