Installation of RTOS on Pololu A-Star 32U4

by John Undersander

The boards we are working with this semester contain the ATmega 32U4 microcontroller, which has 32KB of flash program memory and 2.5KB SRAM. We have been working on projects with increasing complexity, such as our scheduling lab and projects, but they are only utilizing a few C files at most and a small main loop to achieve their purpose. A Real Time Operating System is by no means comparable to an OS such as Windows or Unix, but provides a more sophisticated system to schedule and run different tasks on any given hardware. I have chosen an industry standard solution, the FreeRTOS kernel, to port onto our boards as my project. This is not a straightforward task, as the project only currently has a working port published for the AVR 32U3, but with some modifications to the interfacing file specific to the board, the kernel will run with the 32U4. I anticipate this port to be quite a time consuming process, but in the event that I complete it (with documentation) and get some tasks running within it quickly, I will identify one of the free expansion products available for FreeRTOS and complete a small example of its use as an extension to my project.

I believe I can learn a lot from this project. In my CS career thus far, I have learned various operating system concepts and terminology, but have never really dug into any kernel code to see how they work under the hood. FreeRTOS is much more approachable than looking through a large open source project such as the Linux project, and I will most likely learn a lot more about our board than I would have otherwise. I am sure I will realize the extent of our 32KB storage resources, as the kernel itself (depending on configuration) could use up around 25KB of that! More importantly though, I will endeavor to understand the advantage that a RTOS offers over our approach to real time projects in this course. What if I need to schedule 50 tasks? 150? What does this mean for our task array + main loop approach and how might it compare to how one might solve this problem in an industry application? These are questions I seek answers for.

I have OS experience from recent courses (including 4061 this fall) but not RTOS. I have installed major operating systems such as Linux, Windows onto machines and also have played around with flashing different versions of firmware onto my dashcam that I use in my car. I don’t anticipate these experiences to carry much weight going into this specific project, but I think my knowledge and experience with how particular (maybe stubborn) certain software/hardware combos can be will certainly influence how I approach this work.

- Resources
    - A-star 32U4 datasheet
    - FreeRTOS quick start guide
    - AVRFreaks
        - Any source code help pertaining to the port used from this source will be fully cited and explained in my own words, as to understand why the solution works.