Lab 6: Arduino Real Time Operating System

November 2019

1 Objective

In this lab, you will learn how to use the Arduino Real Time Operating System (RTOS) library to flash LEDs at different rates.

2 Introduction

Most processors allow multiple programs or tasks running at the same time, which is called multi-tasking. However, in the actual processor, only one program can be run at a time. How it achieves parallelism is through scheduling. A part of the operating system called "scheduler" is responsible for deciding which program to run and switching between each program to provide an illusion of simultaneous execution.

The scheduler in RTOS is designed to provide a predictable (normally described as deterministic) execution pattern. This is particularly interesting for embedded systems, like Arduino devices, as embedded systems often have real-time requirements.

There are different types of schedulers depending on the operating system needs. The most basic one is called “Run to completion” in Fig.1. It means that one task runs until it completes and the scheduler would start to run the next task and so on. A more complex one uses the concept of priority in Fig.2. The idea is that each task has a priority and is either “ready” [to run] or “suspended”. The scheduler runs the task with the highest priority that is “ready”. When that task suspends, it runs the one with the next highest priority. If an event occurs, which may have readied a higher priority task, the scheduler will stop the current task and start to run the higher priority task. You will be learning the details of RTOS in the lectures later this semester.

3 FreeRTOS in Arduino

In this lab, we will be using FreeRTOS in Arduino to schedule the LED blinks. FreeRTOS is a priority based scheduler. It uses user-defined priorities to schedule each task's execution. This should be a simple lab but hopefully you will learn the concept of a scheduler.
3.1 Install FreeRTOS

In your Arduino IDE, select "Manage Libraries" under "Tools". Inside the pop up window, look for the FreeRTOS Library under the Type: “Contributed” and the Topic: “Timing”. Then under the Sketch->Include Library menu, ensure that the FreeRTOS library is included in your sketch. You can see ” #include <Arduino_FreeRTOS.h>” in your window.

3.2 Using FreeRTOS

There are two functions we are going to use in this lab: xTaskCreate and vTaskDelayUntil. xTaskCreate creates a new task and adds it to the ready list. It also has a field called priority to specify the priority of this task where "3" means the highest priority and "0" means the lowest priority. vTaskDelayUntil delays a task for a given number of ticks. vTaskDelayUntil specifies an absolute time at which the task wishes to unblock, which provides an deterministic sequence of task execution. Note that vTaskDelayUntil is different from delay(), where the latter one stalls the CPU, while vTaskDelayUntil only delays a specific task and puts the task in blocked mode. You can learn more details about the two functions in the link below.

1https://www.freertos.org/a00125.html
2https://www.freertos.org/vtaskdelayuntil.html
**Task:** Blink three LEDs at different frequencies using FreeRTOS. First try using the same priority for all three tasks and then try changing the priority for each task and observe any changes in the LED flashing pattern.