Lab 7: OS and Peripherals

ECSE 444 - Microprocessors

Fall 2018

Introduction

In this lab, you will learn how to use some of the peripheral sensors via drivers that have been provided by STM, and learn how to use a real time operating system (RTOS). In particular, you will:

- Study the driver documentation in order to enable and acquire sensor data.
- Learn how to use FreeRTOS.
- Create individual OS threads to sample the different peripherals and perform I/O.
- Optimize power usage by using OS thread synchronization tools to suspend unnecessary threads.

Grading

- 30 % Able to use the sensor drivers (print data over UART)
- 40 % Complete RTOS application with sensors and user IO
- 30 % Power reduction by shutting down unnecessary threads.

Peripheral sensors and drivers

The board has a variety of different sensors at your disposal, and they are listed below:

- HTS221: Temperature and humidity
- LIS3MDL: Magnetometer
- LPS22HB: Pressure
- LSM6DSL: Accelerometer and gyroscope

When you open the base-project in Keil, you will find a source-group *Drivers/B-L475E-IOT01* which contains drivers that STM has provided to use these peripherals. The documentation for using the drivers can be found here, and you will also find the same documentation (in a much easier to read format) along with the base-project in the *Drivers/B-L475E-IOT01* directory.

In this task, you must sample raw data from the sensors at a rate of 5 Hz (using timers or osDelay, the choice is yours) and print the values over USART. Print which value is being read, followed by the data. When the button is long pressed (at least 3 sec), the system should stop working by turning off every unnecessary threads and sensors to optimize the power consumption. By pressing the button again, the system should starts again.

To help you get started, the base-project contains example code on how to use the accelerometer driver. Please note that sometimes the same chip is used for multiple data values (eg. accelerometer and gyroscope), so the chip needs not be initialized twice. You will have to study the driver documentation/code to fully understand how it works.

FreeRTOS

The FreeRTOS website contains many useful links, including a Reference Manual and a hands-on tutorial. From this lab onward, all of your tasks will be contained within OS threads. The base-project contains an example of declaring and instantiating the default thread in the OS, and you may create more threads as needed. You should have unique threads for each of the followings:

- USART printing
- User input (push-button)
- Data acquisition from each sensor

Your application should only display a single type of data from a sensor (e.g. temperature). When the push-button is pressed, display a different type of data (e.g. humidity). Pressing the push-button should continuously loop through displaying the different data fields. As before, each sensor should be sampled at 5 Hz.

There are some design choices to be made, one of them for instance being the shared resource that the threads will have to use for USART printing. For instance, there could be a global character array that the an individual sensor thread will write data to, and the USART thread will read for printing. In this case, you will have to use thread synchronization techniques to ensure only one thread accesses the resource (a read from this shared variable should be after a write). *Note: This is not the only way to implement the printing (an alternate: message passing).*

Power Optimization

Finally, you will need to optimize your application for power. When only one sensor is printing its data, the other sensor threads need to be deactivated. So, the relevant devices/sensors should be turned off (not stopped). You can save power by suspending the unnecessary threads as well. This can be simply achieved via thread synchronization techniques like semaphores.