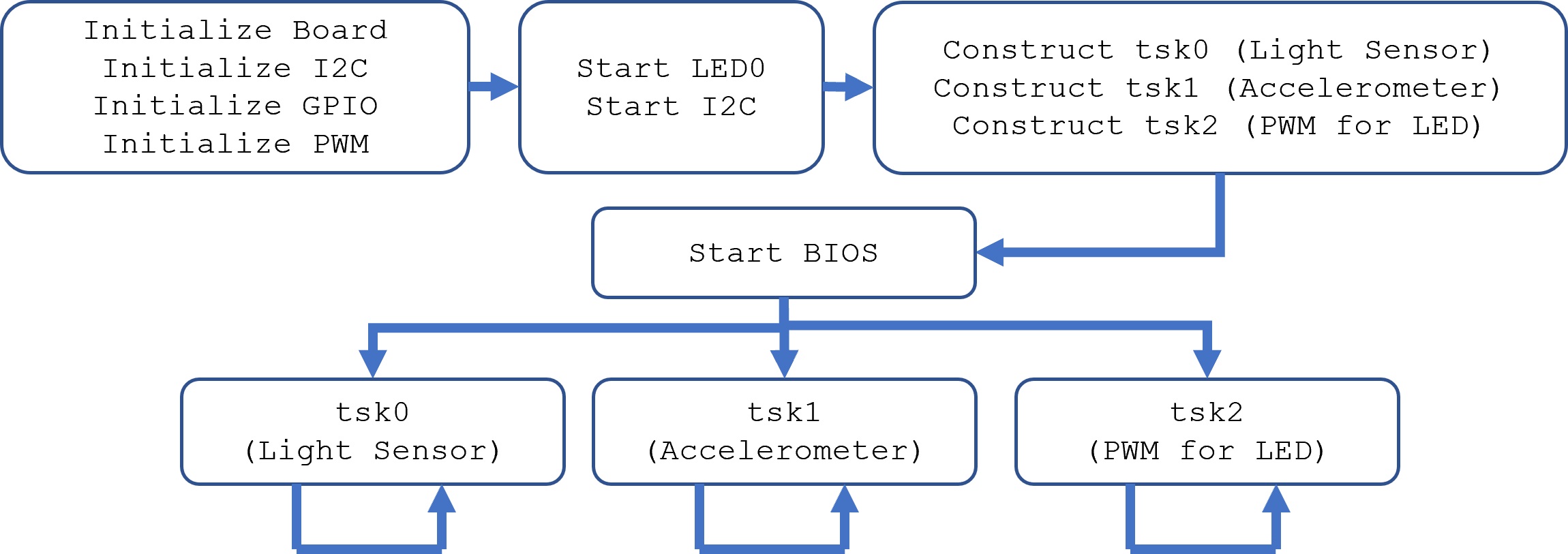
**CSW3237 Introduction to Internet of Things  
Lab Assignment: Application Development on Sensor Tag with RTOS**

Mitchell Kok Ming En  
A0242607J

I implemented TI’s RTOS on the CC2650 SensorTag with the following features: 1) LED brightness is changed proportionally to light sensor readings, and 2) LED brightness is changed proportionally to accelerometer readings. To achieve this, I came up with the flow diagram:



After initializing and starting the necessary resources, the first thing to do was construct the tasks. For this I used 3 separate tasks, one each for the Light Sensor, Accelerometer, and PWM.

Task 0 (OPT3001 Light Sensor) and Task 1 (MPU-9250 Accelerometer) were written to initialize the respective sensors, verify their functionality, then go into an endless loop that reads from the sensor and writes that value into a global variable. Between loops, Task\_sleep() is called in order to: 1) reduce power consumption by the sensor from reading unnecessarily fast, and 2) allow the other tasks to run. Task 0 sleeps for 10ms in between readings, and Task 1 sleeps for 20ms between readings.

Task 2 (PWM) controls the PWM for the LED’s brightness. This task configures the initializes the PWM with a period of 3ms, before starting an endless loop. In the loop, the task reads the values of the global variables used by Task 0 and Task 1 to store readings. It then scales these readings to be between 0 and 3000 so they can be used as the duty value for the PWM, since 3000 is the PWM period and therefore maximum brightness. For the scaling, I used the general formula, such that the given duty value is proportional to the sensor reading:

duty\_value = (PWM\_period \* (sensor\_curr – sensor\_min)) / sensor\_max

where sensor\_curr is the current reading, sensor\_min is the minimum reading found through testing, and sensor\_max is the maximum reading found through testing. Based on initial testing/calibration, I obtained the following formulas to scale the light sensor and accelerometer readings accurately:

duty1 = (3000 \* opt3001data) / 45000; // Task 0 (Light Sensor)

duty2 = (3000 \* (mpu9250data - 4500)) / 6000; // Task 1 (Accelerometer)

Between duty1 and duty2, the larger value is selected as the current duty value for the PWM. Finally, Task 2 sleeps for 20ms between loops.  
  
The RTOS schedules the 3 tasks according to their respective periods until the program is terminated by the user.