The high level structure of the application is summarized in Figure 1. The three functions readOPT3001(), readMpu9250(), and PWMLED() will be initialized as threads. The task readOPT3001() reads the light intensity. The task readMpu9250() reads the acceleration. The task PWMLED() takes the maximum of mpuDc or optDc and sets the duty cycle of the LED.

float mpuDc = 0; // Enables OPT sensor

float optDc = 0; // Enables OPT sensor

void readOPT3001(UArg arg0, UArg arg1) {

// Enables OPT sensor

while(1){

// Reads light intensity from light sensor

}

}

void readMpu9250() {

// Enables MPU sensor

while (1) {

// Reads acceleration from accelerometer

}

}

void PWMLED(UArg arg0, UArg arg1) {

// Enables PWM

while (1) {

// Set PWM according to the max of mpuDC or optDc

}

}

int main(void) {

// Enables board peripherals

// Initializes tasks as threads

}

Figure 1: High Level Structure

Table 1 shows the duration to sleep given to each task. The task transitions from running to a blocked state. The value of sleep in Table 1 is passed to the function Task\_sleep() and delays its execution for the number of ticks of the system clock. The time elapsed per tick is determined by Clock\_tickPeriod, which is set to 10us per tick inside the file main.cfg. 10000 ticks allows CC2650 sensor tag enough time to read and set data from different peripherals and time to context switch between threads during round robin.

|  |  |  |  |
| --- | --- | --- | --- |
| Task | readOPT3001() | readMpu9250() | PWMLED() |
| Sleep (nticks) | 10000 | 10000 | 10000 |
| Sleep (us) | 100000 | 100000 | 100000 |

Table 1: Sleep Period of Task