Connor Cimowsky
20427800

Below is my implementation of the timing function. From experimentation in the lab, I determined that the outer loop should iterate `10*delay_ms` times. To fine-tune the timing, I allowed the timer to run for 10 minutes and then scaled the inner loop accordingly. The value I arrived at is 1536.

```
void sleep(uint32_t delay_ms)
{
    int i, j;
    int noop_counter;

    __disable_irq();

    for (i = 0; i < (10 * delay_ms); i++) {
        for (j = 0; j < 1536; j++) {
            noop_counter += 1;
        }
    }

    __enable_irq();
}
```

After running the timer for 10 minutes, the software approach yielded a time of 09:58, while the hardware approach yielded a time of 10:00. Although the software approach could have been further calibrated to exactly 10:00, it is still not fully predictable. For reasons including variations in clock speed, branch prediction, and caching, the nested loops may not always take the same amount of time to execute. This means that even on the same CPU, the software timer's accuracy can't be guaranteed, even after calibration. However, the hardware timer can be configured by simply setting the prescale and match control registers. The result is a timer that will fire interrupts at the correct interval, regardless of the CPU (ignoring negligible variations due to interrupt latency, etc.).