# ENCE361
# Fitness Tracker Project

28/05/2020

Sam Fraser 869665592

Kyle Johnson 68232416

Christopher Dewhurst

# Introduction

The goal for this project was to design and implement a fitness tracker prototype for a TIVA microcontroller with an attached Orbit BoosterPack peripheral board. The full requirements for the project were given in the program specifications. Some of the main features were:

- It should estimate the number of steps the user has taken using the accelerometer data
- The number of steps should be used to calculate distance in kilometres and miles
- Pushing the left and right buttons should switch between distance and step view on the display
- Step count and distance should be able to be reset independently by long holding the down button
- Switching one of the switches should cause the program to enter a test mode where the accelerometer data is ignored, and the steps/distance is controlled by pushing the up and down buttons.

# Design Summary

Much of the functionality of the program can be defined as foreground or background tasks. Background tasks are ongoing throughout the program and foreground tasks only occur when an external input, such as a button push, is received. The overall behaviour of the user interface can be modelled with the following state diagram.

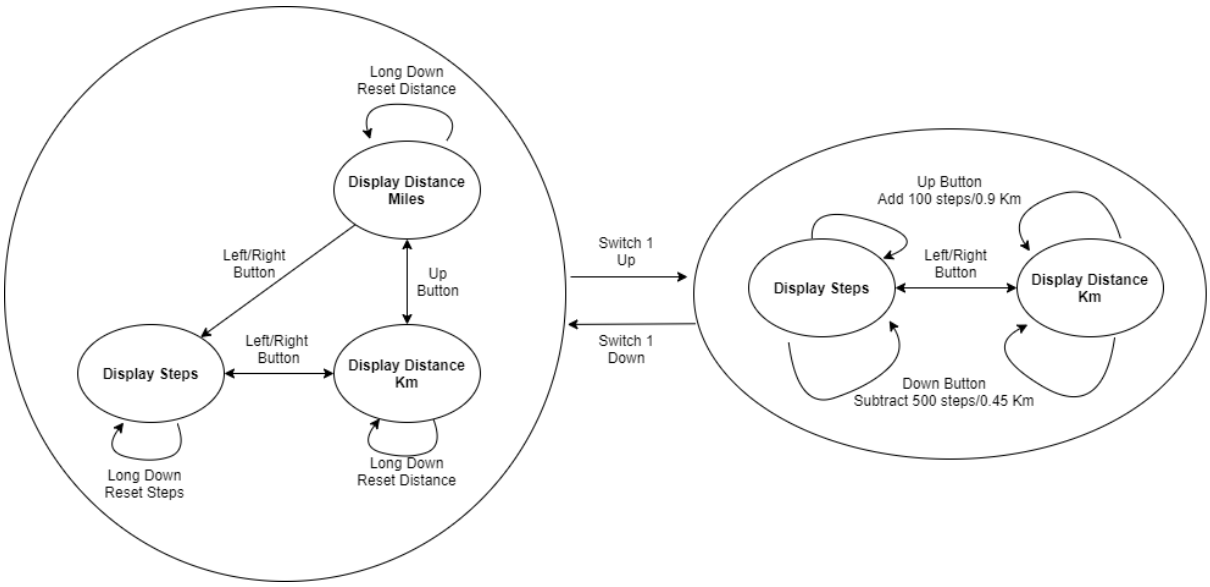


*Figure 1 State diagram modelling the behaviour of the program*

## Scheduling

| Task | Clock cycles | Duration (microseconds) | Frequency Hz | Period ms |
|---|---|---|---|---|
| SysTick ISR | 6 | 0.150 | 1000 | 1 |
| Sw1 ISR | 33 | 0.825 | NA | NA |
| Get accelerometer data & Circular buff | 78248 | 1900 | 100 | 10 |
| Display | 284974 | 7100 | NA | NA |
| Buttons | 403 | 10 | 50 | 20 |
| Accel norm | 140 | 3.5 | 33 | 30 |

**System Clock:** A 40MHz system clock was used. This clock rate was chosen so that get accelerometer data and display function which required high processing times would not interfere with scheduling of other tasks.

## Foreground tasks

**SW1 Interrupt:** Called when there is a change to the state of the switch. An interrupt was used on SW1 mainly due the criteria of this assignment to use and learn about interrupts. A background implementation of SW1 would not affect or change the performance of the Fitness tracker.

**SysTick Timer:** The SysTick interrupt controls the scheduling of each task such as taking data from the accelerometer and polling the inputs.

## Background Tasks

**Circular Buffer/storing accelerometer data:** This task was chosen to be a background task as it requires a high processing time due to I2C communication.

**Buttons**: As button presses occur for a relatively long period, they were chosen to be background task. Buttons states are checked every 20ms as this ensures high responsiveness.

**Display**: The display function requires high processing time therefore, this function only runs when the display needs to updated. This occurs when either a step or button press has occurred.

**Step Incrementor:** Steps are calculated from circular buffers containing the raw acceleration data this data in circular buffers is averaged and the norm of x, y and z components is calculated. If the norm exceeds an acceleration 1.5g the step count is incremented by one. When a step has been taken, a flag is enabled to ensure that the step count only increments by 1 each step. This is disabled once the norm drops below a low threshold so the next step can be counted.

## Code structure and modules

**Accelerometer Module:** This module contains the code required for the operation of the accelerometer. The data from the accelerometer is written into three circular buffers to enable signal averaging and the calculation of steps.

**Display Module:** The display module defines several functions for updating different display states. These functions use the step count to calculate the distance travelled in both miles and kilometres. These are only called when required, such as when a button is pushed or a step is taken, so that there is no display jitter and the high processing time display functions are not run unnecessarily.

**Button Module:** The button module consists of functions to initialise the button on orbit boost pack and to determine if button presses have occurred.

**Switch Module:** This module contains a non-interrupt alternative for checking the SW1 state. The Switch state function in this module is run once on initialization to determine the state of SW1 on start up.

## Calculating Steps

Calculation of steps can be broken up into three steps. These are retrieving the accelerometer data, buffering/processing that data, and calculating the steps.

The accelerometer data at 100 HZ by calling getAcclData(). This stores the three different axis values into a struct. The values from the struct are then all written into three circular buffers, each for a different axis. We used a buffer size of 14 for each of the circular buffers.

In the processing stage, the circular buffer is averaged, and the norm of the 3-dimensional acceleration vector is stored. This is done every 33HZ by calling the AccelNorm function which sums up all the values in the buffer passed in and calculates the mean using this formula

```
mean = (2 *sum + buffer_size) / 2 / buffer_size;
```

Calculating the mean by including buffer_size helps round the mean to a more accurate value by adding 0.5. Once the means have been calculated for each axis, the vector norm is calculated. The norm calculation converts each average acceleration value from its raw unit to g scale. The norm is stored as an int in the main function.

The steps are then determined by checking whether the norm value exceeds the 1.5g threshold. This is done at the frequency of SysTick which is 1000HZ. If the norm does exceed the threshold, 1 is added to the step count and a flag is raised. This flag prevents another step to be added unless the norm decreases below the bottom frequency of 0.95g. On each step added to the count, the display will update the step count. Figure 2 on the following page shows the acceleration norm waveform while steps are being simulated.

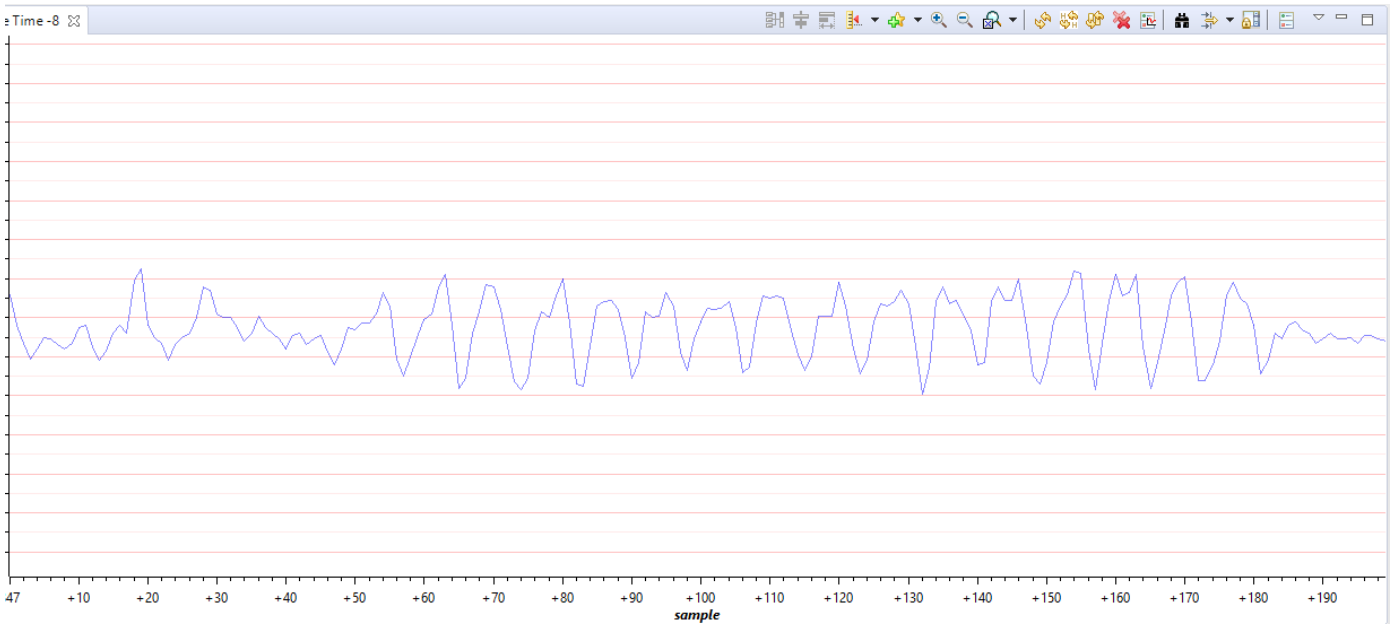


*Figure 2: waveform of acceleration norm*

## Design decisions

An average person walks at 5km/h. This is approximately 100 steps per minute or 0.6 steps per second. This means that a single step cycle takes 600ms to complete. We read the accelerometer data into the circular buffers every 10ms which means we have 60 values of data per step. This is more than enough to show all the information of a step without sampling too much noise. Our norm is calculated every 30ms which gives us 20 averaged norm samples per step. From vigorous testing by walking with the accelerometer, we found 20 norm samples provided very accurate results when calculating steps.

The size of the circular buffer was set to 14 to accommodate the faster sampling of 100hz. A smaller buffer would create wasted values which would get removed before being averaged. Having this buffer size allowed any noise to be filtered out through averaging, giving more consistent results.

## Inter-Task Communication

One of the key learning points for this project was how to deal with shared data problems or inter-task communication. The main issue that can occur with shared data problems is if an interrupt occurs during a critical section of code, modifying the shared data. Fortunately, this was not much of an issue with our project as we did not identify any critical sections, due to the mailbox method used for the SW1 ISR and 32bit global variables used to store the sysTick count which could be executed on in single clock cycle of the 32-bit ARM Cortex . There were several functions that did require access to global variables, most notably the display module that reads and resets the steps and steps_distance variables. This is achieved with the mutator method (get and set functions) which updates variables local to the module and then sets the global variables if local one is modified.

Other modules used static local variables and typedefs to help readability of code.

## Conclusion

The code outlined in this report successfully implements a prototype fitness tracker that meets the desired specifications for the TIVA microcontroller and ORBIT Booster pack peripheral. Some of the key design issues that were encountered in the development of the project included timing of the program, the implementation of interrupts and project management. If the project were to be repeated, creating a robust and modular plan for the program before starting the code would be a great improvement as it would allow separate modules of code to be developed simultaneously instead of breaking one large program into modules in the later stages of development. Many of the smaller problems that were encountered were generally due to the lack of familiarity with concepts and were able to be ironed out with perseverance and more practice.

## References

https://www.healthline.com/health/exercise-fitness/average-walking-speed#average-speed-by-age