Trail Counting With Vibration & Cell Data

Maggie Oates and Vinayak Vedantam

Introduction

Public and private trail systems can benefit from having hiker data to help manage land conservation, prioritize resource allocation, and predict maintenance needs. Our team prototyped a trail counter designed to be mounted on footbridges that counts pedestrians using bridge vibration.

Objectives

- Count groups of hikers using vibration read by an accelerometer
- Automatically transmit count data periodically via cell service
- Counter should be small enough to be discrete and easily-installed

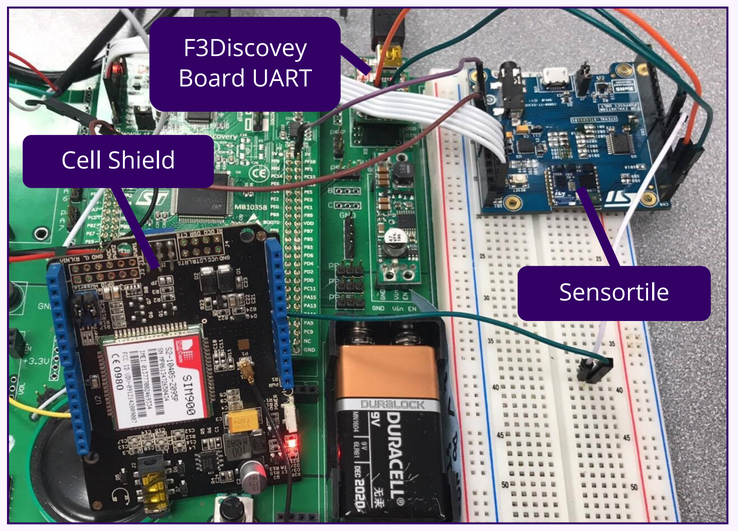

Fig. 1: Trail counter physical proof of concept. A more finished prototype would just use the cell shield and sensortile.

System Hardware

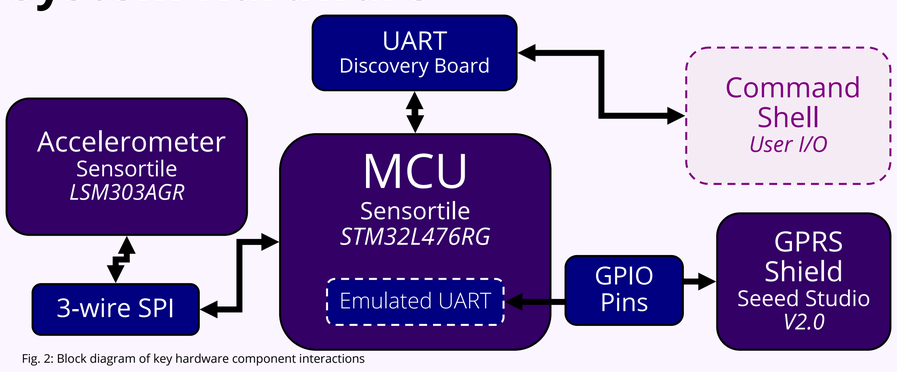

Fig. 2: Block diagram of key hardware component interactions

System Software

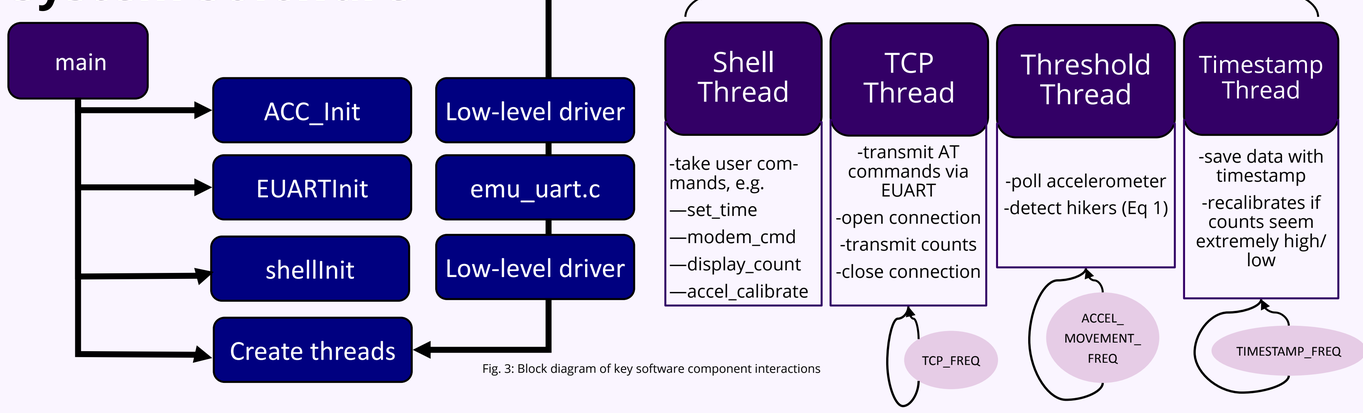

Fig. 3: Block diagram of key software component interactions

Milestone 1—Basic Thresholding

Goal: *Simplistic algorithm to count passing hiker groups*

- Recorded initial accelerometer data to adjust axes readings by the acceleration due to gravity
- Periodically read accelerometer axes data & incremented hiker group count according to Eq. 1:

$$counter \mathrel{+}= 1 \; if \; \sqrt{|x_{curr} - x_{init}| + |y_{curr} - y_{init}| + |z_{curr} - z_{init}|} \geq \omega \quad Eq.1$$

where ω is a constant threshold; x, y, and z are accelerometer axes data; *curr* indicates current axis reading; *init* indicates initial axis reading (should sum to the acceleration due to gravity)

Milestone 2—Calibration

Goal: *Store, timestamp, & test data*

- Timestamped data using an RTC
  - Allowed clock initialization from system shell
- Saved count data to RAM
- Manually dumped to .csv files
- Board re-calibrates periodically if count values seem too high or too low
- Tested counter (see Fig. 4)

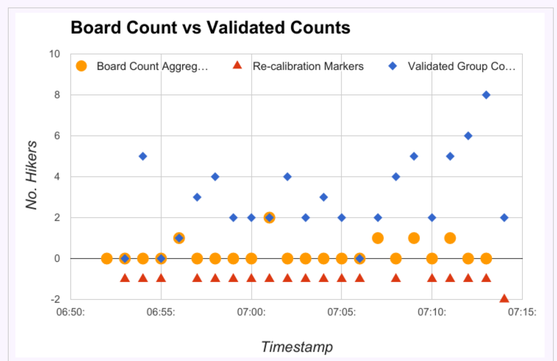

Fig 4: Here, yellow circles indicate the counts computed by our system on a bridge near Ballantine Hall. The blue diamonds indicate validated counts made by hand. The red triangles indicate times that the counter re-calibrated. Note that our threshold was set too high in this instance, as our system consistently underestimated the number of groups passing by. A smarter algorithm would dynamically adjust thresholding based on incoming signals.

Milestone 3—Emulated UART

Goal: *Communication between sensortile and cell modem.*

- The cell shield is set up to use a serial communication. As there were no hardware UARTs available, we implemented an emulated software UART[1] (EUART)
- Re-soldered sensortile cradle to allow for GPIO access
- Increased ChibiOS's clock rate to allow for accurate timing
- Implemented put_char/get_char functions that set/read a GPIO emulated UART pin at correct baud rate (833 μs/bit)
- Implemented EUARTSendCommand method to transmit commands to cell modem through the EUART

Milestone 4—Cell Data Transmission

Goal: *Dump count data to server through cell data*

- Initialize remote socket server using python
- Push sequence of AT commands to cell modem to begin TCP connection
- Transmit timestamped counts periodically
- Server buffers data & outputs a .csv

Results

We demonstrated a proof-of-concept of a trail counter that can collect basic trail usage data and transmit it to a remote server via cell data. This design has the advantage of frequent and easy access to data. Once installed and configured, it requires little user intervention.

Future Directions

Power consumption is a major concern for our current system. Immediate next steps would include implementing hardware-level interrupts to remove frequent EUART and accelerometer polling. The modem and microprocessor should be put into low-power modes when possible. A second priority would be to improve counter accuracy by using more sophisticated signal processing.

References

1. ST Microelectronics. "Implementing an emulated UART on STM32F4 microcontrollers," AN4457, DocID026046 V 2, Aug 2016.
2. Shao, Steven. "Using AT commands to control TCP/IP stack on SM5100B-D modules," Spark Fun Electronics, V 1.1, Aug 2008.

1.