microSWIFT Wave Buoy

EJ Rainville, Jake Davis, Chris Anderson, Ping-Chun Lin

About microSWIFT

MicroSWIFT is an expendable version of the **Surface Wave Instrument Float with Tracking** (SWIFT) platform developed at the U.W. Applied Physics Laboratory (APL). The small, low-cost instruments measure ocean waves and telemeter their data back home via satellite. MicroSWIFTs have been used to study waves at the coast, in hurricanes, in sea ice, and more.

microSWIFTs measuring a breaking wave in the surf zone

Deploying a microSWIFT into sea ice in the Alaskan Arctic

Project Background and Motivation

The microSWIFT is built around a Raspberry Pi and runs operational code written in Python. This combination of developer-friendly hardware and easy-to-read code has made it effective tool for ocean wave research and student learning.

Objectives: The existing codebase lacks a maintainable structure and offline testing capabilities. To improve, the milestones we will try to reach are:

- ➔ Clean-up and document the existing codebase
- ➔ Refactor the existing code to improve maintainability
- ➔ Develop a **checkout** tool to test hardware and software prior to a deployment.

Use Cases and User Stories

Researchers: High background knowledge and low to medium level technical competency

Goal: deploy microSWIFT and configure settings.

- Install microSWIFT software on a new buoy
- Run diagnostic test on sensors and transmission unit
- Based on diagnostic tests, know if a sensor is working or not
- Configure sensor and transmission settings

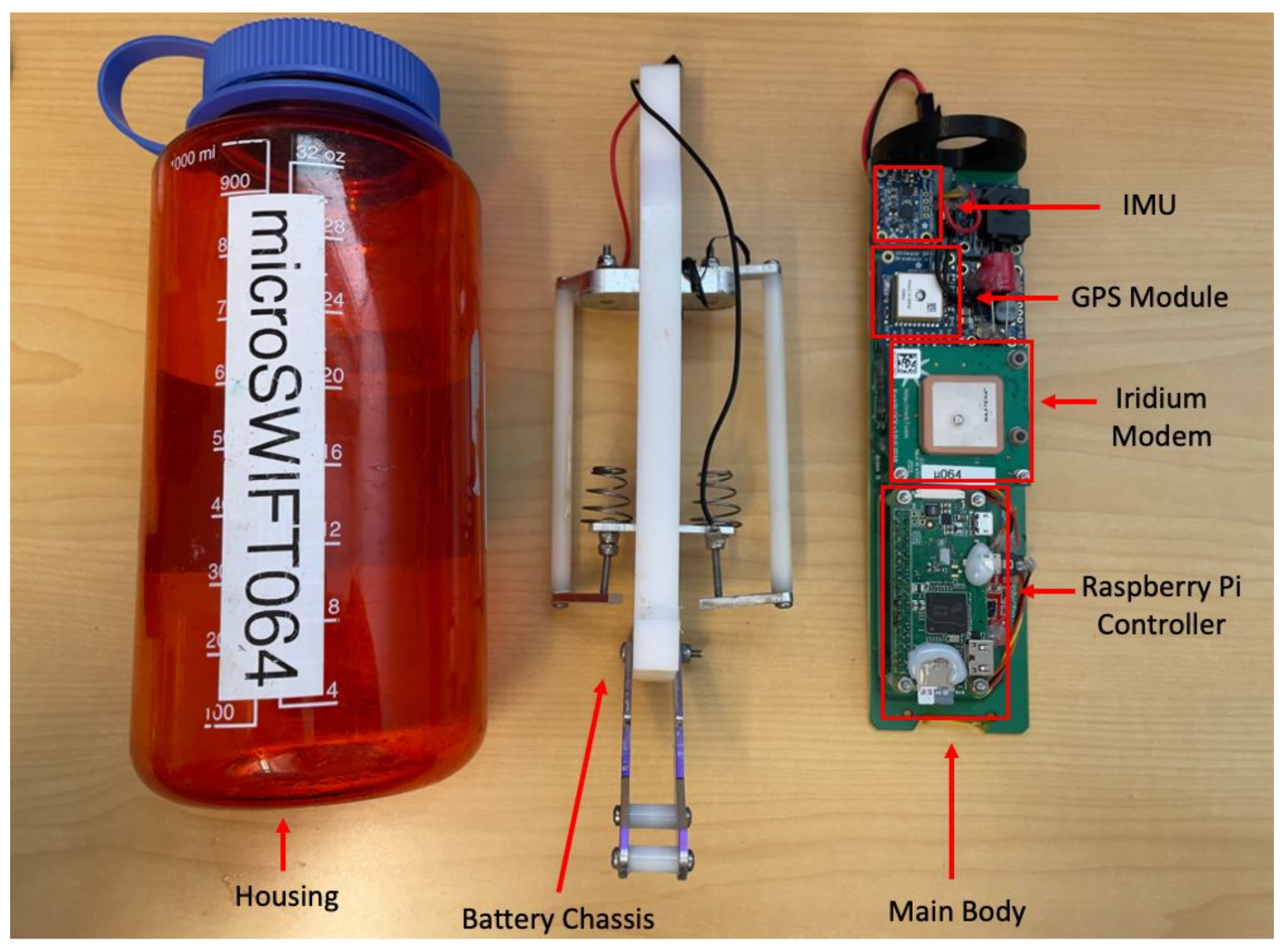
Technicians & Developers: Low to medium background knowledge and high level technical competency

Goal: build microSWIFT and add components

- Individual sensor testing
- Full system diagnostics
- Fully configure system
- Read documentation and add features

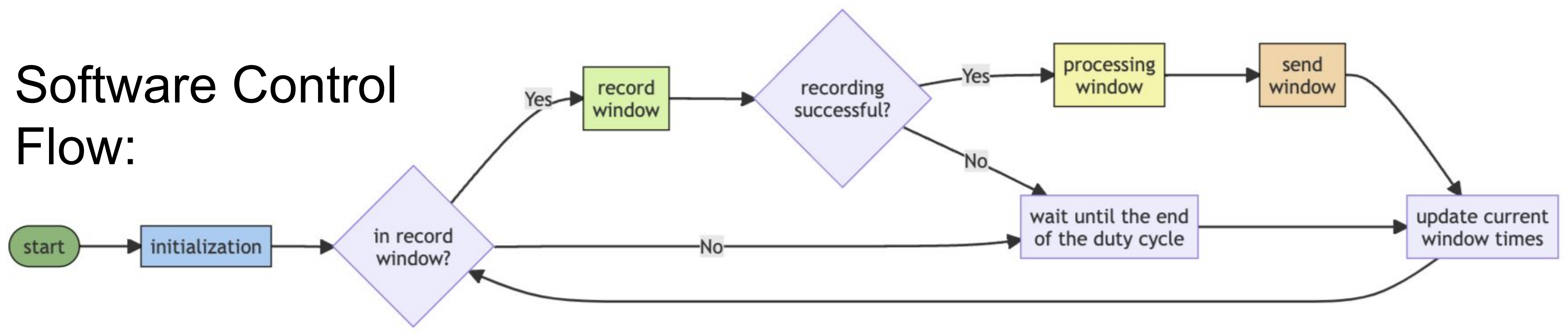
Hardware and Software Design

Hardware:



Link to microSWIFT Video:

Software Control Flow:



Use of mocks:

- ➔ Mock hardware framework to help both the technician and developer interact more effectively with microSWIFT.
- ➔ Replicate the behavior of driver and hardware functionalities
- ➔ Enable the entire microSWIFT codebase to be run on a PC
- ➔ Conditional module imports (see below) automate the use of these mocks when the code detects that it is not being run on the buoy.

Mocked Import Statements:

```python
try:
    import busio
    import board
    import RPi.GPIO as GPIO
    from . import adafruit_fxos8700, adafruit_fxas21002c
except ImportError as e:
    from ..mocks import mock_busio as busio
    from ..mocks import mock_board as board
    from ..mocks import mock_rpi_gpio as GPIO
    from ..mocks import mock_adafruit_fxos8700 as adafruit_fxos8700
    from ..mocks import mock_adafruit_fxas21002c as adafruit_fxas21002c
    print(e, "Using mock hardware")
```

Milestones Reached and Future Improvements

Milestones Reached:

- ➔ Cleaned-up and documented (most of) the codebase
- ➔ Refactored the existing code to improve maintainability
- ➔ Developed hardware mocks for offline testing

Future Work:

- ➔ Finish development of a checkout tool to test hardware and software prior to deployment

Acknowledgements

We would like to acknowledge all the help from Alex de Klerk, Emily Iseley, Nate Clemmett, Joe Talbert, and Jim Thomson for making this project possible. We have many others to thank as well.

PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING