Software Development Process(SDP)

Authors: Jake Goodwin, Aidan Agee, Blake Babb, Patrick Iacob

DATE: 2023-11-16

Principles

- We will respond to asynchronous communication within 24 hours
- We will be at meetings on time and pay attention
- All changes need to be isolated to their own git branch
- Each work item need a corresponding GitHub issue
- Pull Requests have to be reviewed by at least one team member
- We will use a kanban board to continuously work on the backlog
- Once a work item is complete, a pull request is created
- Blocks need to be discussed as soon as possible

Process

Task Selection: * Kanban style * Select highest priority item within your role's domain

To Solve an Issue and Meet Acceptance Criteria: 1. Write failing tests.
2. Write code to pass tests. 3. Repeat. 4. Open a pull-request and merge

Steps in software development

1. Create a github issue/milestone.
2. Assign github task/issue.
3. Create fork of repo.
4. Write tests using the test framework.
5. Push the tests to the fork (optional).
6. Write Code to pass the tests.
7. Test the code.
8. Push to the fork repo.
9. Create a pull request with description.
10. Get approval for the PR(pull request).

Goals & Objectives

Develop a personal data acquisition system that records all the data a user might want, and is cheap and easy to set up and use. * Record data on acceleration, force, position, etc. * Minimal setup * Can be hooked up to bike, go-kart, etc.

Project Scope

- Design of simple UI to display data.
- Design of hardware/schematics for system.
- Firmware for sensor modules in rust.
- SBC with rust software to store/log sensor data over CAN.

Roles

ROLE	PERSON	RESPONSIBILITIES
UI	Blake	Develops the web page front end
SBC/SW	Aidian	Develop logic to relay sensor data to UI
FIRMWARE	Patrick	Develop firmware for microcontrollers
HARDWARE	Jake	Design schematics, wiring diagrams & PCB files

These are the general outlines for the four different roles in the project. We have a verbal agreement at the moment that we will help out with parts of the project outside our roles as needed.

Tooling

Purpose	Name
Version Control	Git
Project Management	GitHub Projects
Documentation	Rustdocs & MD
Test framework	Rust & Cmocka
Editor	ANY
Schematics & PCB	KiCAD
Communication	Discord/Teams/Email

Version Control

Git will allow our team to track changes in the projects files over time. Also prevents the loss of work from hardware failures.

Project Management

GitHub projects is integrated into github organizations as well as git. The project management software makes the collaboration between developers easy and will make tracking milestones and issues for the entire project across multiple repositories a possibility.

Documentation

Documentation will primaily be done through the built-in rust-docs feature. This is accesiable via the CLI(command line interface) tooling. This will encapsulate how the code itself and any interfaces are documented.

Because the documentation is genreated as part of the code this will ensure that up to date and accurate documentatio is always availble.

Secondary documentation meant for non-developers will be done using a combination of markdown and LaTex where needed. This will be availble usally in a PDF format.

Schematics & PCB

The KiCAD program gives access to the schematics and PCB designs to all team members due to the software begin free and open-source.

It will allow us to comment, label and design the needed circuits for the physical hardware of the system; providing a good troubleshooting resource as well.

Communication

Discord: * Used to coordinate team meetings. * To share ideas/brainstorm * give updates on project.

Teams: * TA meetings.

GITHUB: * To discuss project issues. * share documentation.

Definition of Done(DOD)

- Acceptance criteria all satisfied by code changes
- Changes have been merged to master after completing the Pull Request Process
- A completed Pull Request has at least one approval and no marks for “Needs Work”
- All tests pass with changes implemented and no reversion is required
- Relevant documentation for the feature has been updated
- Discussion points are prepared for next meeting

TESTING

Rust:

The testing for all code repositories will be done using a testing harness or framework. For rust this takes the form of the `cargo test` command, which is part of the package managment system(tool-chain).

These tests will be used as one of acceptance critera for a branch to be pulled into the main branch.

C:

Some libraries or areas where the use of C code is needed we plan to use cmocka as the unit testing framework. This combined with Cmake as the build system will give us a host agnostic development cycle.

Quality Assurance

Quality assurance will mostly be handled by adhearance to style standards enforced by the lanuages LSP(language server protocol) servers. The two that will see extensive use in this project being:

1. Rust-analyzer
2. clangd

Feedback

Feedback on the work done will take place in the github projects. The issues and discussion boards are the main locations for this, with the weekly meetings and discord being a secondary and informal medium for minor feedback.

Release Cycle

For the moment we will used semantic versioning with the standard Major.minor.patch format. This will help when it comes to dealing with any major changes that break APIs.

Contingency Plans

Feedback can be shared during weekly standup with the TA, or over Discord if they are more time-sensitive, after which it should be reviewed by the whole team, and then incorporated. Changes to the whole process will require more comprehensive feedback and approval from the team before going into effect, after which related documents should be modified as soon as possible.

In the event of unexpected challenges, the team should be notified immediately, and if serious enough should be brought up with the TA, project partner or instructor, otherwise they should be brought up during regular meetings.

Timeline

- 12/15/2023: Version 0 complete with breadboard organized hardware and visual UI elements
- 03/22/2023: Version 1 complete with functionality between firmware, SBC, and UI

Environments

Environment	Infrastructure	Deployment	What is it for?	Monitoring
Production	Github releases	Release	Packaging install files.	N/A
Staging	Github actions			Github Pull requests
Development	Local	Github commits	Development and unit tests of microcontroller-based sensors	Manual