PORTLAND STATE UNIVERSITY

ECE 412

CAPSTONE PROJECT PROPOSAL 2020

RISC-V Cleanflight Autopilot

Bliss Brass - brass@pdx.edu
Eric Schulte - eschulte@pdx.edu
Nikolay Nikolov - nnikolov@pdx.edu
Ruben Maldonado - mruben@pdx.edu

Advisor:
Roy Kravitz
Sponsor:
Galois

February 7, 2020

Contents

1 Overview / Executive Summary

Our objective is to port Cleanflight, a mini open-source OS for racing drones to a board running on the RISC-V architecture. We are going to achieve our aim by turning SiFive's HiFive1 Revision B RISC-V development board into the flight controller for a standard 4-propeller racing drone.

The successful completion of the project will act as a proof-of-concept for the newer RISC-V architecture, as compared to older RISC machines like ARM.

Our project sponsor, Galois Inc., has asked us to deliver a flying racing drone with the HiFive1 Rev. B board running Cleanflight by the end of the spring term in June 2020. Our sponsor, Galois, is purchasing and providing a drone kit for us to replace the native flight controller with the HiFive1 Rev. B.

2 Project Design Specification

2.1 Concept of operations / User stories

The primary purpose of this project is to demonstrate RISC-V's capabilities in a fun and exciting way. The primary users will be Galois and possibly other drone hobbyists looking to adapt future RISC-V boards for use on racing drones. It may also be used by others as a reference or base-point to launch their own RISC-V based projects.

2.2 Stakeholders

The main project stakeholder is our project sponsor, Galois Inc. We will be providing them with a functioning drone running on RISC-V architecture. The other stakeholder (though they may not know it yet) is the Cleanflight development team. We will share our RISC-V port via Github pull request at the end of the project.

2.3 Background

RISC-V is an open-source Instruction Set Architecture (ISA) developed at UC Berkeley and introduced in 2010. RISC-V is a modular ISA centered around the philosophy of 'using only the instruction registers that you need,' meaning the ISA aims to minimize wasteful use of resources.

RISC-V includes a static core, RV32I, with the option of adding one or more standard extensions that hardware can consist of or not based on the needs of an application. The modular concept enables very small and low energy implementations of RISC-V, which can be critical for embedded applications.

In recent years RISC-V has even drawn the interest of companies such as Apple, Facebook, Google, and Samsung. Interested in the modular philosophy of the ISA, these companies have begun investing resources into creating their own RISC-V based silicon. The industry currently requires more practical applications to demonstrate the capabilities of RISC-V. We aim to create a racing drone to serve as one of these demonstration platforms.

2.4 Market Analysis

All drones, racing or otherwise, are controlled using a flight control board. The flight control board is a PCB that acts as the 'brains' of the drone; receiving the user input from an RC transmitter on the ground, processing the data, and outputting control instructions to the motors. All flight control boards must include at least two components: a processing unit, and an inertial measurement unit (IMU). Additional features may consist of components such as cameras, compasses, barometers, and telemetry devices.

The popularity of drones has exploded over the last ten years. As a result, the flight controller market has grown to include a diverse set of products from hundreds of companies. It's now possible to find a quality flight controller for between $25.00-50.00$.

By porting Cleanflight to the HiFive1 Rev. B development board, our team is essentially creating a custom flight control board. Our board won't be the first RISC-V flight controller to market, but, as far as we can tell, it will be the first RISC-V board to support Cleanflight.

We'll be modeling our board's functionality around that of the Seriously Pro Racing F3 Flight Controller (SPRacingF3). The SPRacingF3 includes an accelerometer and gyroscope for determining orientation and speed, a compass for determining direction, and a barometer for determining altitude. We plan on thoroughly implementing the accelerometer and gyroscope, while the compass and barometer are stretch goals.

2.5 Requirements

Most broadly, the drone must fly with the HiFive1 Rev. B running a ported version of Cleanflight. More specifically, the drone must respond accurately and responsively to user input controlling speed, direction, and altitude.

Since Cleanflight is an open-source software project, our team must also document the porting process thoroughly enough so that someone else could recreate the project. We will share the final documentation with the Cleanflight development team via a Github and merge request at the end of the project. As such, a Cleanflight user should be able to use the Clean-Flight tool to flash their own RISC-V based flight control board.

Stretch requirements include implementing a compass and barometer to provide extra flight data.

2.6 Specifications

Our team must use Cleanflight as the flight control software, and use the HiFive1 Rev. B board as the flight control board. These are the only specifications provided by our project sponsor.

2.7 Deliverables

- Project Proposal
  - Product Design Specification
  - Project Management Plan
- Weekly Progress Report
- Final report
- ECE Capstone Poster Session poster
- Github pull request (to merge RISC-V port into Cleanflight's main-stream repository)

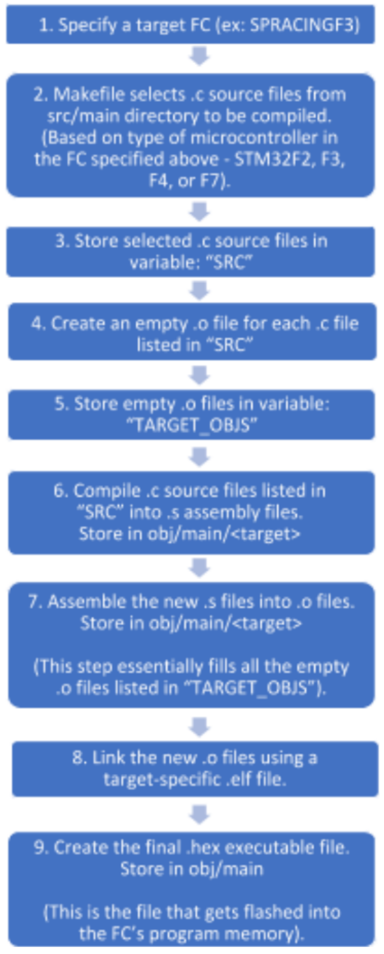
2.8 Initial Designs

2.8.1 Cleanflight's Makefile

Cleanflight's Makefile provides instructions to make utility on how to compile and build the final Cleanflight executable file. This executable file then gets flashed to the target flight control board.

To create an executable for the RISC-V environment, our team must integrate RISC-V support into the Makefile. The first step in that process is exploring how the Makefile tells the compiler to build an executable. Below is a high-level block diagram of the Cleanflight Makefile's functionality:

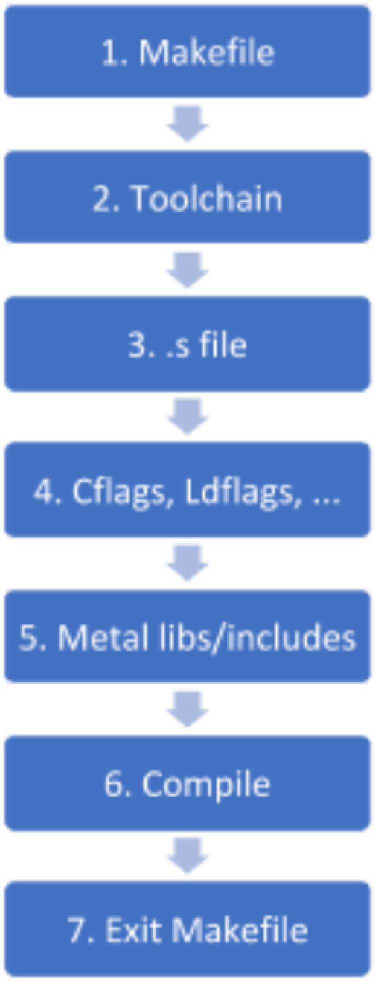
Figure 1: Makefile flowchart.


1. Specify a target FC (ex: SPRACINGF3)
2. Makefile selects .c source files from src/main directory to be compiled. (Based on type of microcontroller in the FC specified above - STM32F2, F3, F4, or F7).
3. Store selected .c source files in variable: "SRC"
4. Create an empty .o file for each .c file listed in "SRC"
5. Store empty .o files in variable: "TARGET_OBJS"
6. Compile .c source files listed in "SRC" into .s assembly files. Store in obj/main/<target>
7. Assemble the new .s files into .o files. Store in obj/main/<target> (This step essentially fills all the empty .o files listed in "TARGET_OBJS").
8. Link the new .o files using a target-specific .elf file.
9. Create the final .hex executable file. Store in obj/main (This is the file that gets flashed into the FC's program memory).

2.8.2 The Minimum Port

Before porting any specific Cleanflight drivers, we want to complete a 'dry run' of the porting process. This will involve editing Cleanflight's makefile to include RISC-V development tools and flags, and getting the makefile to produce an executable for the HiFive1 Rev. B. The idea is to learn how to build RISC-V executables using a version of Cleanflight's makefile, prior to attempting to port any specific software. We refer to this initial 'dry run' as the Minimum Port, and its steps are listed below:

Figure 2: Minimum port flowchart.



1. Start with the Cleanflight makefile.
2. Download and include a path to the prebuilt RISC-V toolchain which includes GCC, G++, GDB, etc.
3. The toolchain comes with Freedom Studio, or it can be found on SiFive's website. The path can then be hardcoded in the makefile, or the tool.mk file can be updated to find the toolchain.
4. Add paths to .s and .S files. These files can be found after compiling a sample app with HiFive1 Rev. B listed as the target.
5. Add Cflags, Ldflags, etc. These files can be found after compiling a sample app with HiFive1 Rev. B listed as the target.
6. Add metal libs/includes files. These files can be found after compiling a sample app with HiFive1 Rev. B listed as the target.
7. Build a simple example program (i.e. flashing LEDs)
8. Exit makefile

2.9 Verification Plan

After completing the minimum port, we are planning on splitting the porting work into modules.

The first module to be ported is accelerometer firmware. Once that is complete, we will unit-test the firmware with the accelerometer chip to ensure standalone functionality. After the port of the accelerometer module, the team will split into two-person subteams - each focusing on their module.

One team is responsible for porting the receiver firmware, and the other responsible for porting the ESC/motor output firmware. Similar to the accelerometer module testing, each team will unit-test their module to check for standalone functionality.

Once we confirm each module is working on its own, we will assemble the drone kit and install the HiFive1 Rev. B as the flight controller. The next stage is performing integration testing to ensure all modules function correctly together as a single package.

These are just preliminary verification plans, with more specifics to come. But as of right now, this is the general outline of our verification plan.

2.10 Risks

Therefore, we must develop an overall strategy before diving too deep into any specific code (i.e., first creating a "minimum port" discussed above).

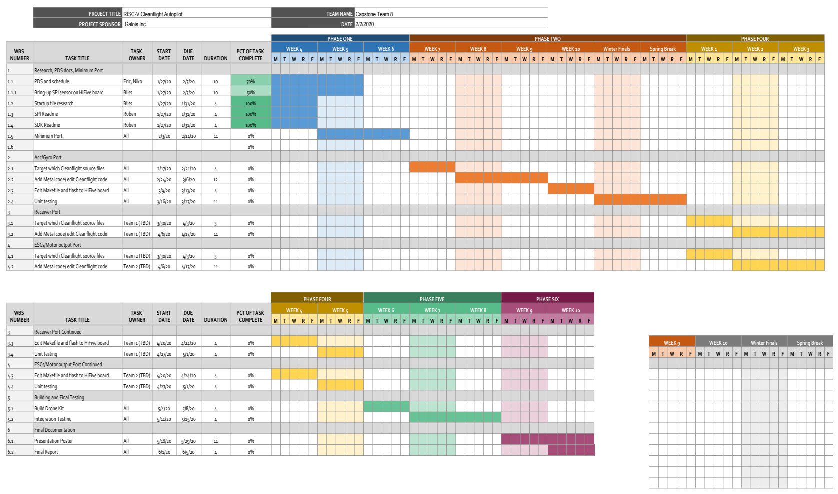
Another risk comes from the capabilities of the HiFive1 Rev. B itself. For example: are there enough GPIO pins and SPI ports to support Cleanflight? Will the program memory of the HiFive1 Rev. B be large enough for the final executable? These are the types of issues that could arise later in the porting process.

Lastly, another risk is the compatibility between the stock drone kit and the HiFive1 Rev. B. The package supports the ARM-based flight controller. While the HiFive1 Rev. B should provide all the features of an ARM microcontroller, there may be a few compatibility issues.

3 Project Management Plan

3.1 Timeline

Figure 3: Gantt Chart.



PROJECT TITLE	RISC-V Cleanflight Autopilot	TEAM NAME	Capstone Team 8
PROJECT SPONSOR	Galois Inc.	DATE	2/2/2020

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	Research, PDS docs, Minimum Port					
1.1	PDS and schedule	Eric, Niko	1/27/20	2/7/20	10	70%
1.1.1	Bring-up SPI sensor on HiFive board	Bliss	1/27/20	2/7/20	10	50%
1.2	Startup file research	Bliss	1/27/20	1/31/20	4	100%
1.3	SPI Readme	Ruben	1/27/20	1/31/20	4	100%
1.4	SDK Readme	Ruben	1/27/20	1/31/20	4	100%
1.5	Minimum Port	All	2/3/20	2/14/20	11	0%
1.6						0%
2	Acc/Gyro Port					
2.1	Target which Cleanflight source files	All	2/17/20	2/21/20	4	0%
2.2	Add Metal code/ edit Cleanflight code	All	2/24/20	3/6/20	12	0%
2.3	Edit Makefile and flash to HiFive board	All	3/9/20	3/13/20	4	0%
2.4	Unit testing	All	3/16/20	3/27/20	11	0%
3	Receiver Port					
3.1	Target which Cleanflight source files	Team 1 (TBD)	3/30/20	4/3/20	3	0%
3.2	Add Metal code/ edit Cleanflight code	Team 1 (TBD)	4/6/20	4/17/20	11	0%
4	ESCs/Motor output Port					
4.1	Target which Cleanflight source files	Team 2 (TBD)	3/30/20	4/3/20	3	0%
4.2	Add Metal code/ edit Cleanflight code	Team 2 (TBD)	4/6/20	4/17/20	11	0%

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
3	Receiver Port Continued					
3.3	Edit Makefile and flash to HiFive board	Team 1 (TBD)	4/20/20	4/24/20	4	0%
3.4	Unit testing	Team 1 (TBD)	4/27/20	5/1/20	4	0%
4	ESCs/Motor output Port Continued					
4.3	Edit Makefile and flash to HiFive board	Team 2 (TBD)	4/20/20	4/24/20	4	0%
4.4	Unit testing	Team 2 (TBD)	4/27/20	5/1/20	4	0%
5	Building and Final Testing					
5.1	Build Drone Kit	All	5/4/20	5/8/20	4	0%
5.2	Integration Testing	All	5/11/20	5/15/20	4	0%
6	Final Documentation					
6.1	Presentation Poster	All	5/18/20	5/29/20	11	0%
6.2	Final Report	All	6/1/20	6/5/20	4	0%

3.2 Milestones

1. Complete research phase
2. Complete Minimum Port (see PDS, Initial Designs section for more details)
3. Project Proposal approved by sponsor:
   (a) PDS
   (b) Project Management Plan
4. Accelerometer/Gyroscope module tested and ported to HiFive1 Rev. B
5. Receiver module tested and ported to HiFive1 Rev. B
6. ESC/motor output module tested and ported to HiFive1 Rev. B
7. Assemble drone kit and install HiFive1 Rev. B as flight control board
8. Complete final integration testing
9. Create final documentation:
   (a) RISC-V Github pull request
   (b) Final report
   (c) Presentation poster

4 Project Budget and Resources

4.1 BOM

Item	Description	Qty.	Price	Total
LHI 280 Race Quad ARF	Drone kit (including SPRacingF3 FC)	1	105.99	105.99
Flysky FS-i6X 10CH 2.4GHz AFHDS RC Transmitter w/ FS-iA6B Receiver	RC transmitter and receiver	1	45.85	45.85
Ovonic 11.1V 2200mAh Lipo Battery	Battery	1	45.85	45.85
Sparkfun MPU-9250 Breakout board	Accelerometer/gyroscope	1	14.95	14.95
HiFive1 Rev. B Development Board	RISC-V development board	2	59.00	118 .00
Total				330.64

Our total project budget comes out to roughly $315.00. This budget includes:

1. Racing drone kit
2. RC transmitter and receiver
3. LIPO battery
4. MPU-9250 accelerometer breakout board
5. HiFive1 Rev. B Development Boards

Our team hopes to receive all of these materials as soon as possible; we have already submitted a materials request to our project sponsor, and are hoping to hear back soon.

We want to start tinkering with a functioning drone kit (that includes a functioning FC), to get a better idea of the process of building Cleanflight for a specific target FC.

This will hopefully give us a better understanding of how to do the same process with the HiFive1 Rev. B, the only difference being the ported software.

Most likely, we will use the Capstone Lab and the EPL during the hardware assembly and testing phases.

5 Team members and relevant skills

Bliss Brass, CE :

- C,C++ scripting,LaTeX,Markdown
- Firmware and Linux device drivers
- Microprocessor Architecture

Ruben Maldonado, CE :

- C,C++,Python, Shell scripting
- Firmware and Linux device drivers
- Microprocessor Architecture

Nikolay Nikolov, CE :

- C,C++,Python, Shell scripting,LaTeX,Markdown
- Firmware and Linux device drivers
- Microprocessor Architecture

Eric Schulte, EE :

- Basic microprocessor system design (ECE 371)
- Hardware prototyping and testing
- Some object-oriented coding (C/C++)

6 Development Process

Note: These plans are preliminary and subject to change, but this is the overall idea of how we expect the project to proceed.

Ruben has volunteered to act as a liaison between the team and our industry sponsor and faculty advisor. We are using Slack for daily communication, and Github's Project feature for progress/task tracking.

The plan is to have everybody working on tasks in a series for most of the project. Since we're all new to software porting, we think this is the best strategy to minimize project setbacks. The first thing we're going to 'build' is what we call the 'minimum port'.

The minimum port entails creating a basic makefile, based on Cleanflight's makefile, to build an executable of a simple demo program (i.e., flash some LEDs), and then flash it to the HiFive1 Rev. B using SiFive's Freedom Studio. The simple makefile will serve as our team's entry point into the Cleanflight code. Once we can compile and build a simple demo program, we can add more complexity and start building advanced executables like ported hardware drivers.

With our custom makefile complete, our team will begin porting the necessary software for the MPU-9250 accelerometer. MPU-9250 is the device that will provide orientation and velocity/acceleration information to the flight control board. Since the accelerometer firmware will be our team's first port, we've made this phase the longest of the project. We anticipate that this phase will be quite challenging. As such, our team plans to work in series.

Once the accelerometer port is complete, we'll begin focusing on porting software for the receiver, and the Electric Speed Controllers (ESCs act as an intermediary device between the flight control board and servo motors controlling the propellers). Hopefully, soon we understand the porting process better, which is why we plan on splitting into teams of two for these ports. One group will work on porting receiver firmware, while the other ports software for the ESCs.

After the receiver and ESC ports are complete, we'll remerge as a team of four to assemble the drone kit and perform integration testing. After this milestone, we build our final ported version of Cleanflight, flash it to the HiFive1 Rev. B, and see if the drone flies.

After final integration testing is complete, we will focus on compiling our project's final documentation. The final documentation will include: assembling a Github pull request to submit our RISC-V port to the mainstream of Cleanflight, writing the final report, and creating the presentation poster.

Note: We should be completing necessary documentation (i.e., wikis, manuals) for the Github pull request along the way, not all at once at the end.