FSW Onboarding and Training

[This is a template DO NOT modify]
Make a copy of this document, and replace the *<NAME>* with your name. Then share this in #fsw-training

Guidelines

- Go through all the material carefully. It is meant to make your time programming in FSW easier.
- Be aware that the goal is for you to try and do these by yourself, or in pairs with other new members. Try finding resources or playing around with tools yourself before consulting #fsw-training for help. But make sure to share your ideas and process as much as possible to get it reviewed by someone else from FSW
- If you ever wish to have something changed, or if you used another resource that you feel should be included in our guides, reach out to the FSW leadership

Formatting this document

- Since this document belongs to you and only you, feel free to leave notes or comments on certain sections. Treat this as something you can refer back to when trying to work with the IntelliSat and the Orbital Platform in the future.
- Removing any sections, or altering the task descriptions, should only be done after consulting with FSW Leadership. This document will be tracked, so make sure to consult FSW Leadership if you'd like any changes in what you are working on.

Information to Know Before Starting Onboarding

High-Level Goals

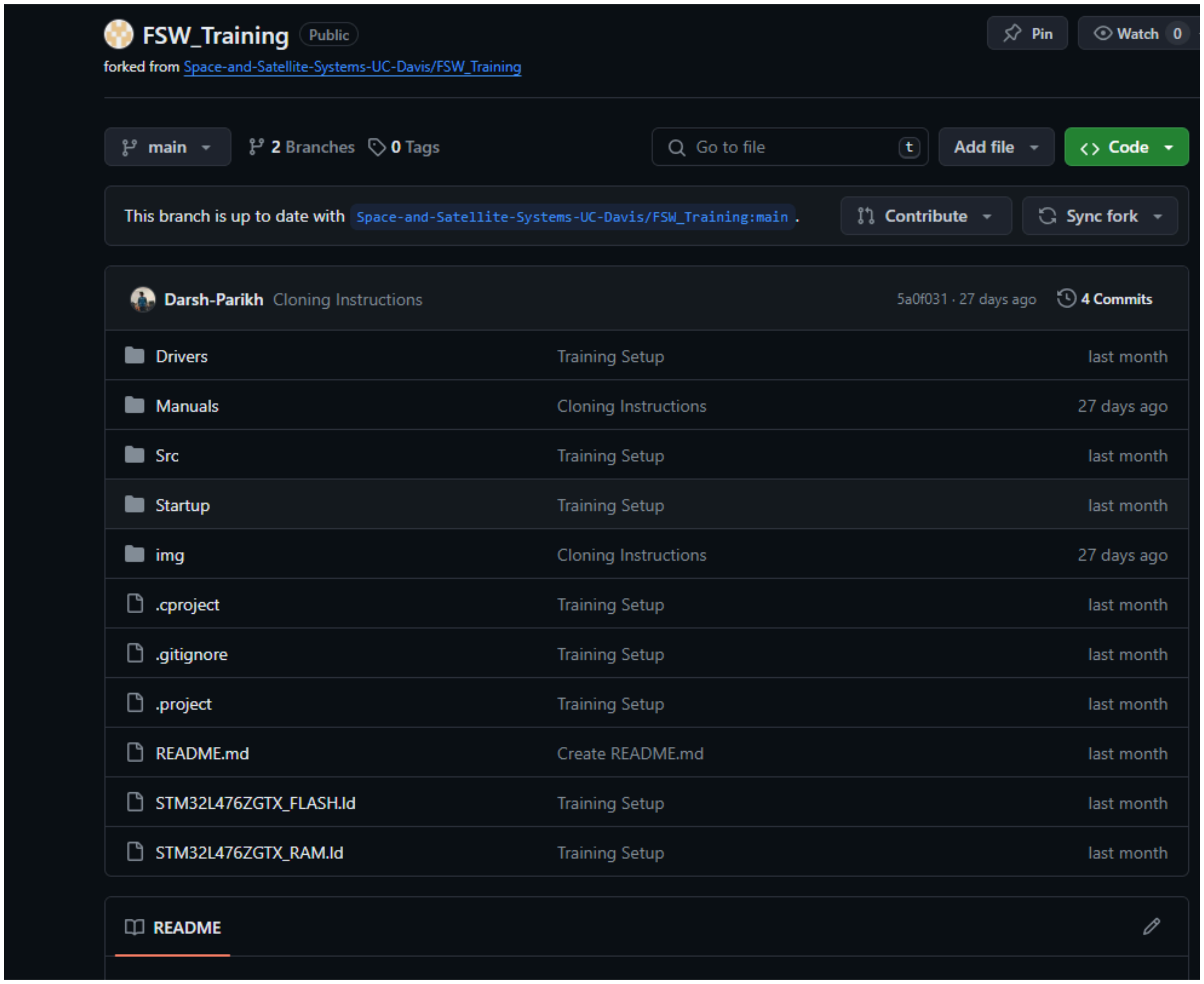
- Get the IDE
- Get some testing code, and learn what IntelliSat looks like

Order of Operations

- Complete the Tasks below at your own pace.
  - You can do some out of order (some are readings or videos). Use your judgment to guess which ones to do out of order, and which must be done in order.
  - Make sure to complete them by the end of the week
- Once done, let FSW Leadership know. This is so you can test on the Orbital Platform

Tasks & their requirements

- [ ] Make a Github account, ideally with your UC Davis email
  Skip if you already have an account
- [ ] Install CubeIDE
  - [ ] Install the EGit plugin
- [ ] Fork the FSW_Training repository on GitHub. Make sure to uncheck the *Copy Main Branch Only* checkbox when creating a fork
  Share the screenshot of the forked repository, and also confirm that your fork has a *gpio_testing* branch



FSW_Training Public
forked from Space-and-Satellite-Systems-UC-Davis/FSW_Training

This branch is up to date with Space-and-Satellite-Systems-UC-Davis/FSW_Training:main .

Darsh-Parikh Cloning Instructions 5a0f031 · 27 days ago 4 Commits

Drivers	Training Setup	last month
Manuals	Cloning Instructions	27 days ago
Src	Training Setup	last month
Startup	Training Setup	last month
img	Cloning Instructions	27 days ago
.cproject	Training Setup	last month
.gitignore	Training Setup	last month
.project	Training Setup	last month
README.md	Create README.md	last month
STM32L476ZGTX_FLASH.ld	Training Setup	last month
STM32L476ZGTX_RAM.ld	Training Setup	last month

- ☐ Clone the forked FSW_Training repository, and set it up as a Project in CubeIDE
  All of this must be done in CubeIDE. Some of the provided reads might be helpful here
- ☐ Read through for Intro to Training skipping, the "The Exercise" portion (do not worry about the exercise, you don't need to do it).
- ☐ Confirm that your code builds with no errors
  You can test the code only physically on the Orbital Platform but still check for compilation errors in your own time.
- ☐ Videos / Readings to go through
  - ☐ Basics of Git & Common Git Commands
    Even though we use EGit, it's important to know what happens behind the scenes
  - ☐ Difference between Git and GitHub
  - ☐ Basics of STM32CubeIDE
    Fun Fact: CubeIDE is just a modified version of Eclipse IDE
  - ☐ Basics of EGit
    - Additional Reference Guide
  - ☐ Intro to IntelliSat

Things to try on the Orbital Platform

- How to flash your code onto the Orbital Platform
    - NOTE: There may be multiple revisions of the Orbital Platform. Make sure you are building and flashing code for the right one. Check the *OP_REV* macro in *Src/inc/globals.h*, and change it depending on the revision you are working on.
- Explore debugging
    - See you print statements via the serial console
    - Observe the CubeIDE GDB debugger pausing the board when a button is pressed
    - Observe variables changing via the CubeIDE GDB debugger

Onboarding

High-Level Goals

- A deep dive into our brand of Embedded Systems programming, with GPIO programming

Order of Operations

- Complete the Tasks below at your own pace.
    - You can do some out of order (some are readings or videos). Use your judgment to guess which ones to do out of order, and which must be done in order.
    - Make sure to complete them by the end of the week
- Once done, let FSW Leadership know. Then, unless told otherwise, you will get to

Pre-Requisites

- You have completed read through the important information
- You have committed all your code to the master branch. Optionally, you could push the code back to the GitHub repository

Tasks & their requirements

- ☐ Switch to the gpio_testing branch of the FSW_Training repository
  You should be doing this in CubeIDE, via EGit
- ☐ Within the *Playground/* folder, you'll find a *gpio.c* file.
  Share the screenshot of where this file is in the Project Viewer in CubeIDE
- ☐ Follow the instructions in the GPIO.md file
    - ☐ Part 1: Blinky Code

        - [ ] Part 2: _gpio_configure_mode()
- [ ] Confirm that you’re code builds with no errors
- [ ] Videos / Readings
    - [ ] The STM32l476 Datasheet
    - [ ] The STM32l476 Reference Manual
      We probably refer to this document the most when working on drivers
    - [ ] Bare Metal Programming Video
      There’s a lot of info here, but this is also to understand that we use Bare Metal rather than HAL
    - [ ] Bare Metal GPIO on the STM32
      Keep in mind this is for STM32F. Ours is STM32L, so make sure to refer to its reference manual for the GPIO registers on our board (Section 8.5)
    - [ ] Bare Metal Programming Guide
      Focus on Introduction & Human Readable Programming in particular. The later sections are good to read to understand FSW Drivers in particular
    - [ ] Bitwise operations in C

Things to try on the Orbital Platform

- Debugging via the Logic Analyzer. Check if the gpio pin is alternating states every 10 ms.
- Switch the gpio pin to an LED pin, and the delay to 1 sec. Reflash the new code and check whether the LED is blinking every second.
    - Find which pins are used for LEDs on the Orbital Platform
        - Refer to the sheet called *GPIO Assignments*

End of FSW Onboarding

Congratulations on completing the general SSS FSW team onboarding. You should now be somewhat familiar with:

- Bare Metal C Programming for Embedded Systems
- How to use CubeIDE and integrate it with git and GitHub
- The STM32L4 reference manual
- The general structure of the FSW codebase, and where key functions and macros are located
- How to build and flash code for the Orbital Platform

By now you should have been added to a specific FSW subteam. Participate in their meetings and engage in any tasks/readings given by them
Additionally, you are invited to contribute to FSW’s IntelliSat repository. Get it on your computer by following the instructions here:

https://github.com/Space-and-Satellite-Systems-UC-Davis/IntelliSat/blob/main/Manuals/Getting_Started.md