3UCubed: Command and Data Handling

Presented by: Haley Joerger[1], Erika Diaz Ramirez[1], Laura Peticolas[1]

Contributed by: Jared King[2], Shane Woods[2], Sanjeev Mehta[2], Jonathan Perez[1], Douglas Clarke[1], Marcus Alfred[3], Jeffery Reedy[1], and the 3UCubed Team

Sonoma State University[1], University of New Hampshire[2], and Howard University[3]

https://imap.princeton.edu/engagement/student-collaboration

This work was supported by a grant to University of New Hampshire as part of NASA's IMAP Student Collaboration project, grant number 80NSSC20K1110.

3UCubed: 3 Universities; 3U CubeSats; Upwelling, Uplifting Undergraduates

OVERVIEW

- NASA IMAP Student Collaboration and the 3UCubed CubeSat
- Software Requirements
- CD&H Overview
- Instrument Software and Testing
- Student Work on F’
- F’ VS SDK

NASA’s IMAP Student Collaboration: 3UCubed

Program Goals

1 *We aim to compliment NASA’s IMAP mission science research,*

2
- ***develop a hands-on research experience for students,***

3
- ***and contribute to diversifying space science*****.**

Project Overview

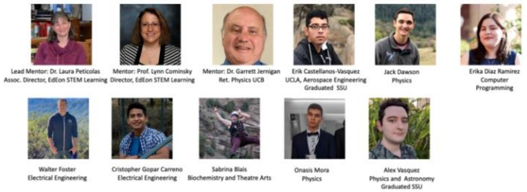
- Partnership between 3 Universities with diverse student enrollment:

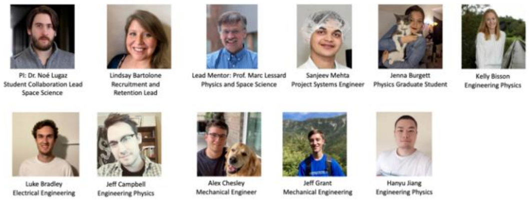
Howard University,

Sonoma State University and

University of New Hampshire

- Collaborate to **design, build, test and launch 3U CubeSats**





3UCubed: 3 <u>U</u>niversities; 3<u>U</u> CubeSats; <u>U</u>pwelling; <u>U</u>plifting <u>U</u>ndergraduates

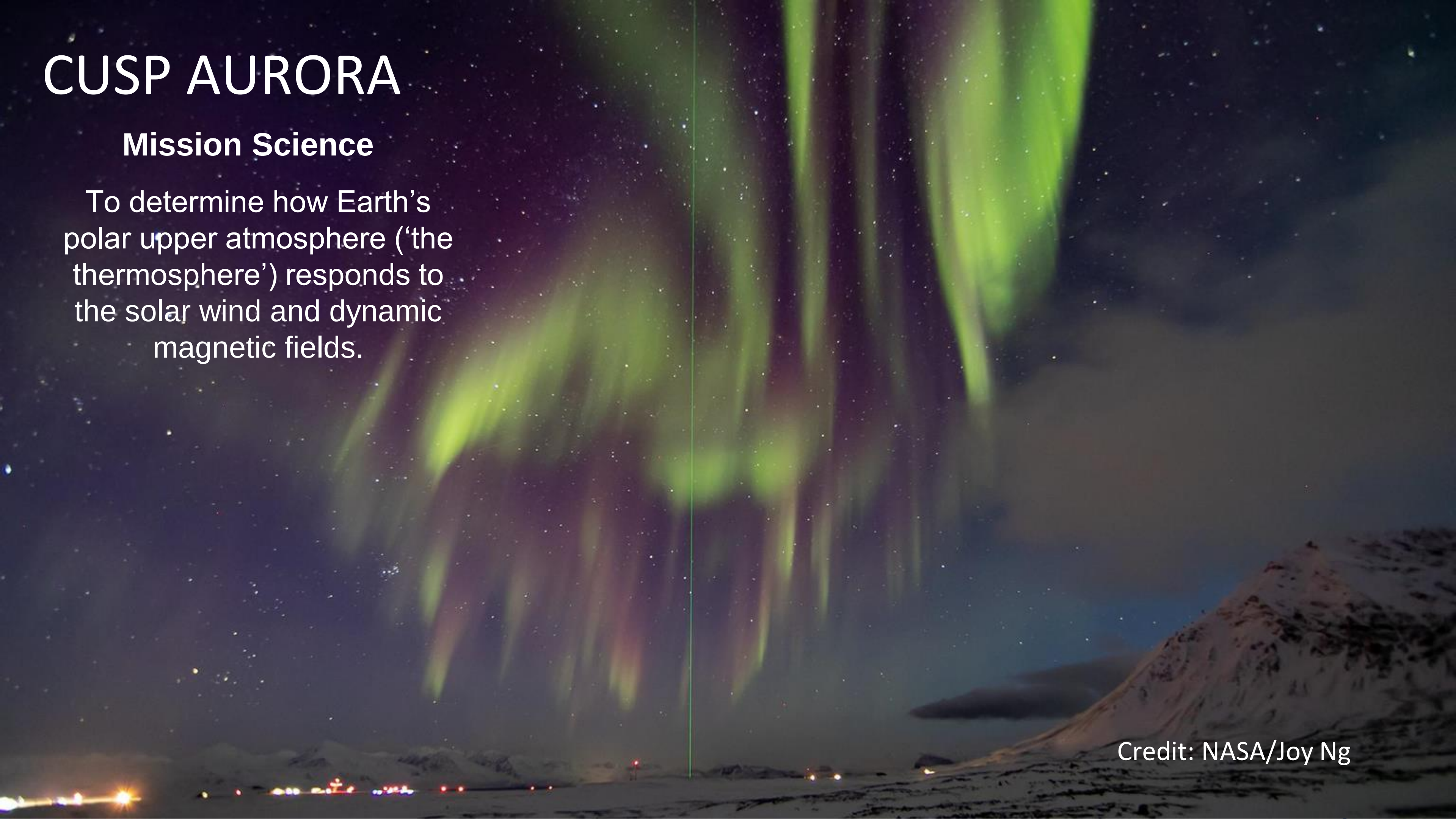
SSU

1867

CUSP AURORA

Mission Science

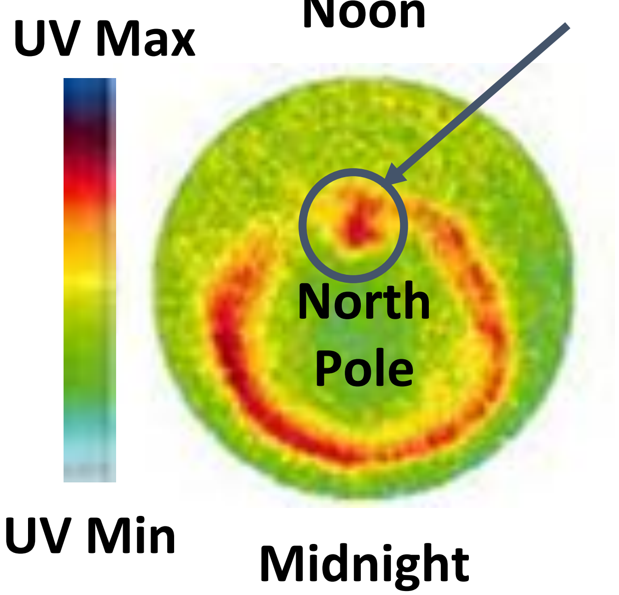
To determine how Earth's polar upper atmosphere ('the thermosphere') responds to the solar wind and dynamic magnetic fields.



Credit: NASA/Joy Ng

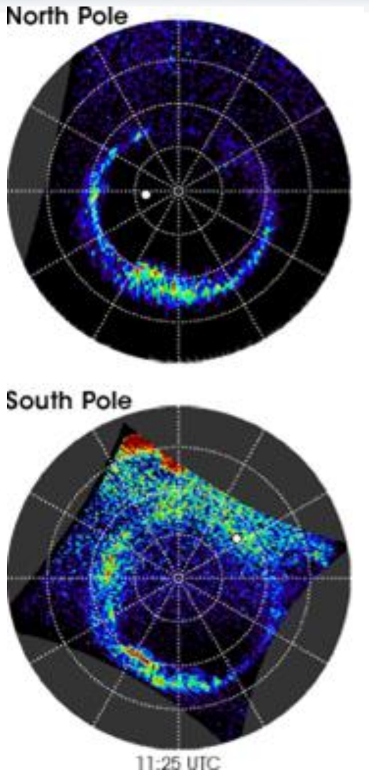
INTERACTIVE VIRTUAL PRESENTATION WITH STUDENTS

Cusp Aurora in the Ultraviolet (UV)



You went to the Jamboard

(Slide 7)and circled the cusp area.



C&DH REQUIREMENTS

SAT-007

- capable of interfacing all subsystems and components as necessary with each other based on their respective subsystem or component ICDs

SAT-008

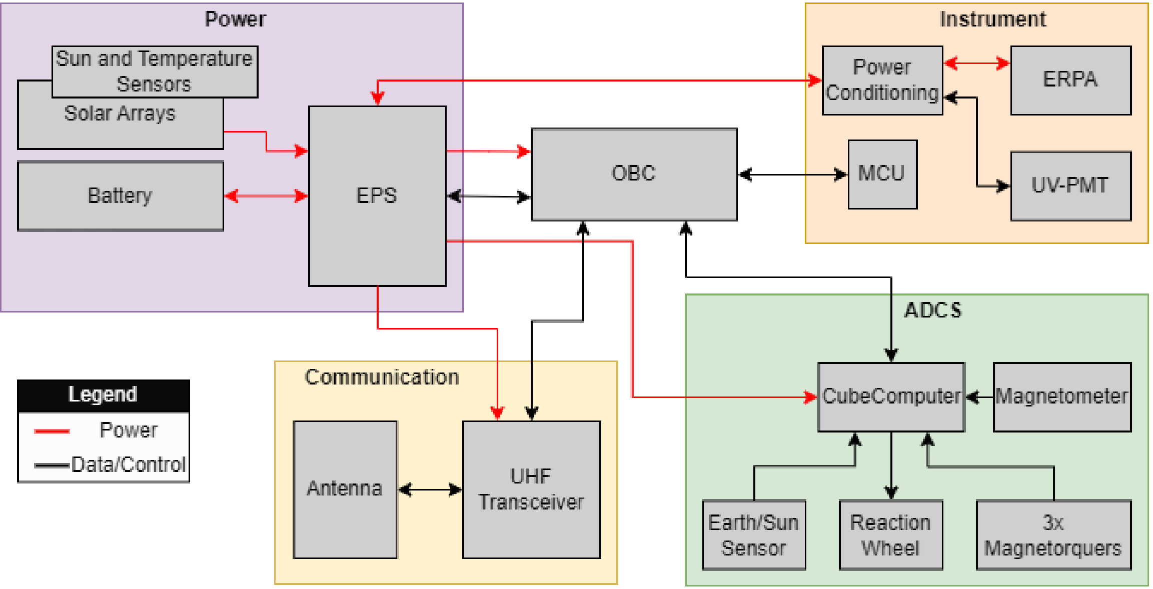
- capable of downlinking data and receiving and complying with ground commands

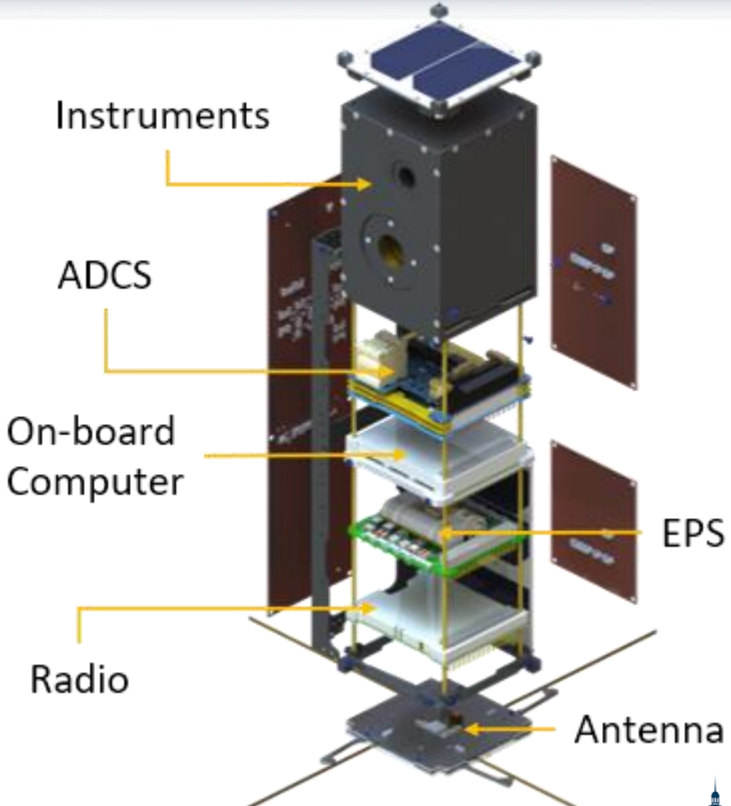
SAT-009

- capable of interfacing with Ground Station Equipment hardware for testing

- Software shall interface with the onboard hardware, including commands to the OBC and telemetry downloads
- Software shall interface with the Ground Station Equipment hardware.
- The subsystem or component ICDs shall include software ICDs.

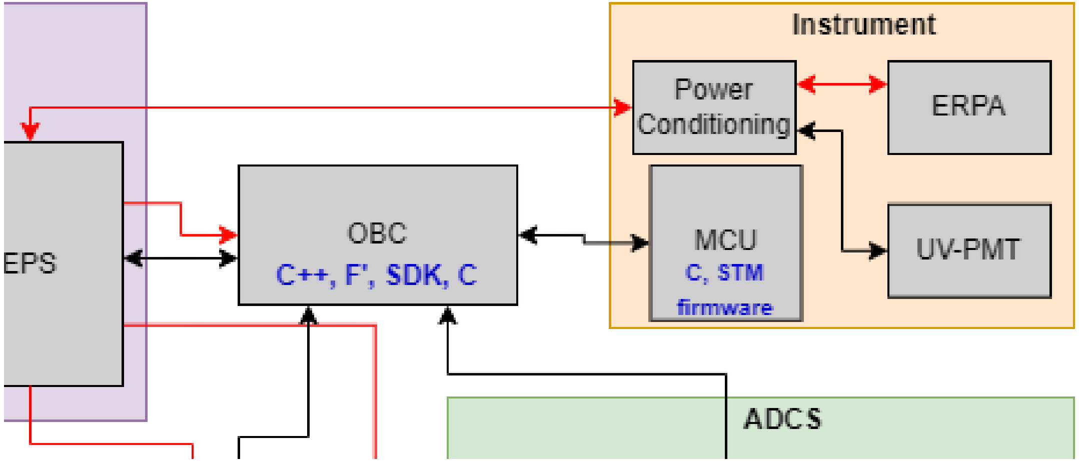
C&DH HARDWARE BLOCK DIAGRAM





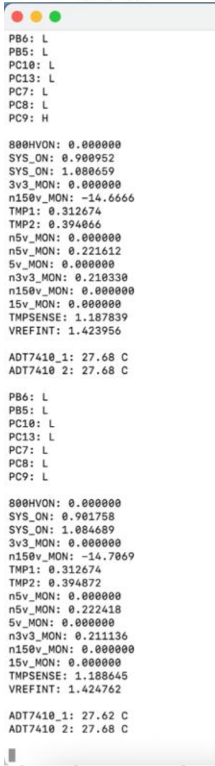
3UCubed: 3 Universities; 3U CubeSats; Upwelling, Uplifting Undergraduates

C&DH SOFTWARE BLOCK DIAGRAM



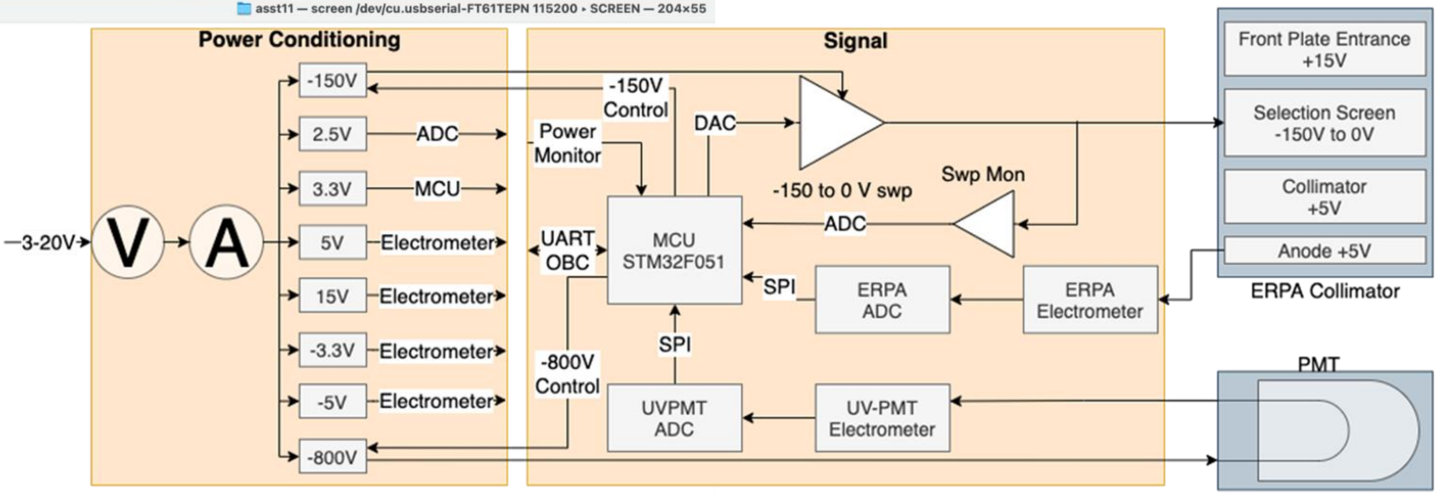
3UCubed: 3 Universities; 3U CubeSats; Upwelling, Uplifting Undergraduates

TESTING INSTRUMENTS



asst11 — screen /dev/cu.usbserial-FT61TEPN 115200 ▸ SCREEN — 204×55

```
PB6: L
PB5: L
PC10: L
PC13: L
PC7: L
PC8: L
PC9: H

800HVON: 0.000000
SYS_ON: 0.900952
SYS_ON: 1.080659
3v3_MON: 0.000000
n150v_MON: -14.6666
TMP1: 0.312674
TMP2: 0.394066
n5v_MON: 0.000000
n5v_MON: 0.221612
5v_MON: 0.000000
n3v3_MON: 0.210330
n150v_MON: 0.000000
15v_MON: 0.000000
TMPSENSE: 1.187839
VREFINT: 1.423956

ADT7410_1: 27.68 C
ADT7410 2: 27.68 C

PB6: L
PB5: L
PC10: L
PC13: L
PC7: L
PC8: L
PC9: L

800HVON: 0.000000
SYS_ON: 0.901758
SYS_ON: 1.084689
3v3_MON: 0.000000
n150v_MON: -14.7069
TMP1: 0.312674
TMP2: 0.394872
n5v_MON: 0.000000
n5v_MON: 0.222418
5v_MON: 0.000000
n3v3_MON: 0.211136
n150v_MON: 0.000000
15v_MON: 0.000000
TMPSENSE: 1.188645
VREFINT: 1.424762

ADT7410_1: 27.62 C
ADT7410 2: 27.68 C
```



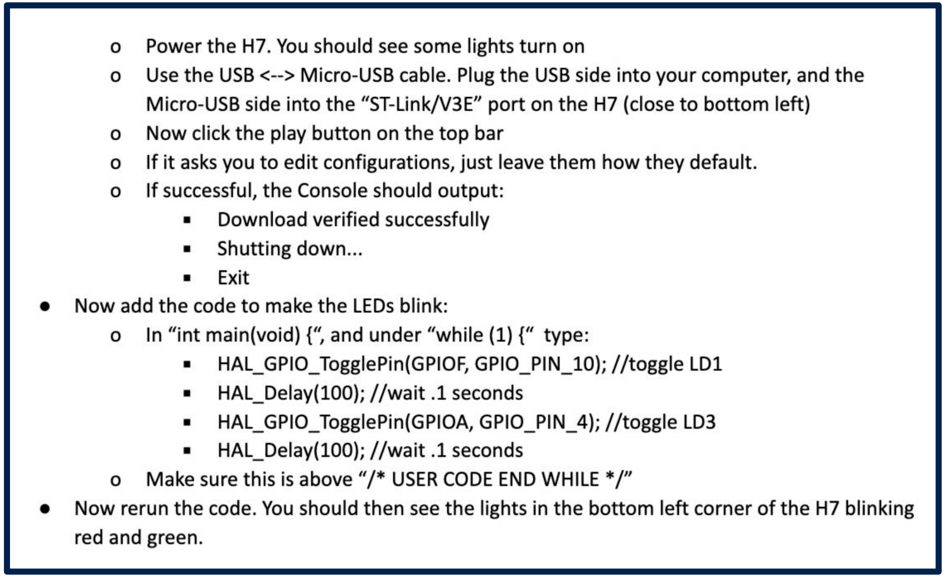
- Quick Tests
- Functional Tests
- (Future) Operational Tests

F’ STUDENT WORK

- 3 page implementation
- Current tasks
- Working with JPL

TUTORIAL 1

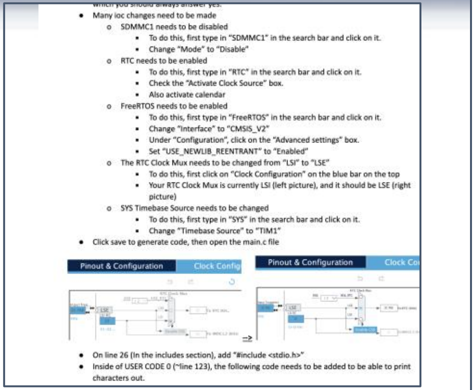
- Getting started with hardware: STM32 microcontroller
- Develop and flash firmware
- Learning how to configure interfaces



  - Power the H7. You should see some lights turn on
  - Use the USB <--> Micro-USB cable. Plug the USB side into your computer, and the Micro-USB side into the “ST-Link/V3E” port on the H7 (close to bottom left)
  - Now click the play button on the top bar
  - If it asks you to edit configurations, just leave them how they default.
  - If successful, the Console should output:
    - Download verified successfully
    - Shutting down...
    - Exit
- Now add the code to make the LEDs blink:
  - In “int main(void) {“, and under “while (1) {“ type:
    - HAL_GPIO_TogglePin(GPIOF, GPIO_PIN_10); //toggle LD1
    - HAL_Delay(100); //wait .1 seconds
    - HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4); //toggle LD3
    - HAL_Delay(100); //wait .1 seconds
  - Make sure this is above “/* USER CODE END WHILE */”
- Now rerun the code. You should then see the lights in the bottom left corner of the H7 blinking red and green.

TUTORIAL 2

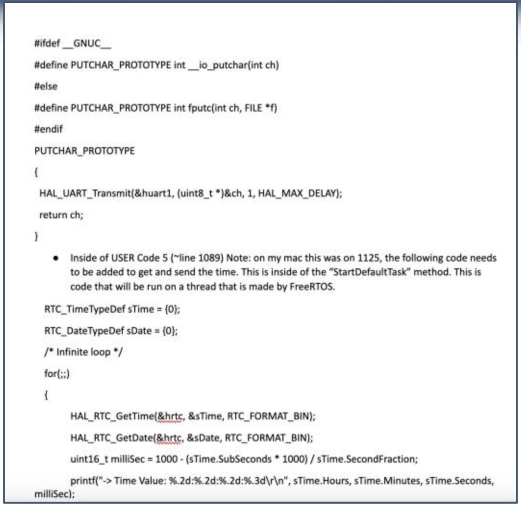
- Program Real Time Clock (RTC) on FreeRTOS
- Configure oscillators and power supply



which you should always answer yes.

- Many ioc changes need to be made
  - SDMMC1 needs to be disabled
    - To do this, first type in "SDMMC1" in the search bar and click on it.
    - Change "Mode" to "Disable"
  - RTC needs to be enabled
    - To do this, first type in "RTC" in the search bar and click on it.
    - Check the "Activate Clock Source" box.
    - Also activate calendar
  - FreeRTOS needs to be enabled
    - To do this, first type in "FreeRTOS" in the search bar and click on it.
    - Change "Interface" to "CMSIS_V2"
    - Under "Configuration", click on the "Advanced settings" box.
    - Set "USE_NEWLIB_REENTRANT" to "Enabled"
  - The RTC Clock Mux needs to be changed from "LSI" to "LSE"
    - To do this, first click on "Clock Configuration" on the blue bar on the top
    - Your RTC Clock Mux is currently LSI (left picture), and it should be LSE (right picture)
  - SYS Timebase Source needs to be changed
    - To do this, first type in "SYS" in the search bar and click on it.
    - Change "Timebase Source" to "TIM1"
- Click save to generate code, then open the main.c file

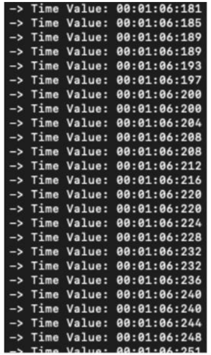
- On line 26 (In the includes section), add "#include <stdio.h>"
- Inside of USER CODE 0 (~line 123), the following code needs to be added to be able to print characters out.



```
#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif
PUTCHAR_PROTOTYPE
{
 HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, HAL_MAX_DELAY);
 return ch;
}
```

- Inside of USER Code 5 (~line 1089) Note: on my mac this was on 1125, the following code needs to be added to get and send the time. This is inside of the "StartDefaultTask" method. This is code that will be run on a thread that is made by FreeRTOS.

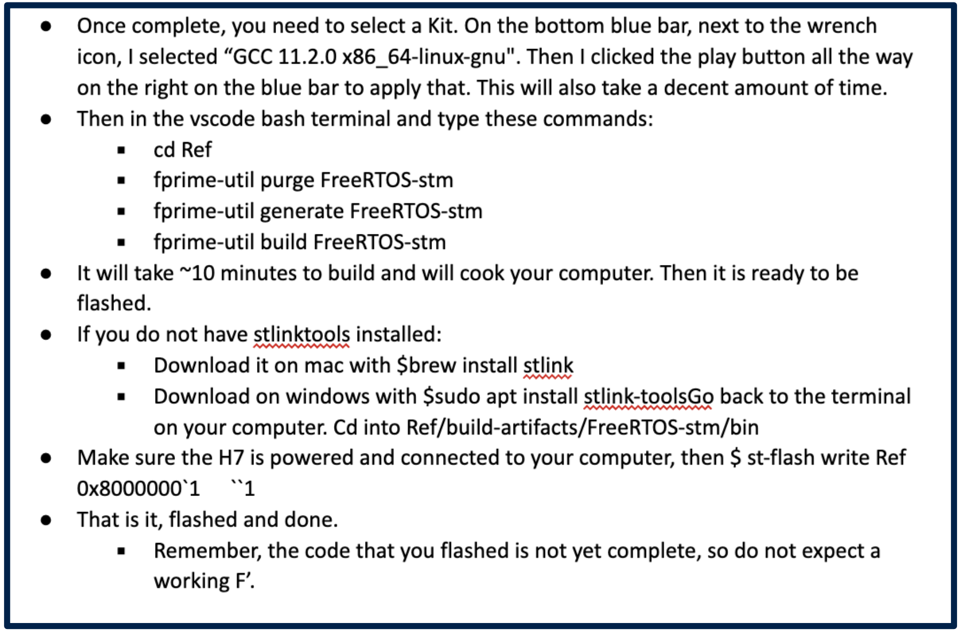
```
  RTC_TimeTypeDef sTime = {0};
  RTC_DateTypeDef sDate = {0};
  /* Infinite loop */
  for(;;)
  {
        HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN);
        HAL_RTC_GetDate(&hrtc, &sDate, RTC_FORMAT_BIN);
        uint16_t milliSec = 1000 - (sTime.SubSeconds * 1000) / sTime.SecondFraction;
        printf("-> Time Value: %.2d:%.2d:%.2d:%.3d\r\n", sTime.Hours, sTime.Minutes, sTime.Seconds,
milliSec);
```



```
-> Time Value: 00:01:06:181
-> Time Value: 00:01:06:185
-> Time Value: 00:01:06:189
-> Time Value: 00:01:06:189
-> Time Value: 00:01:06:193
-> Time Value: 00:01:06:197
-> Time Value: 00:01:06:200
-> Time Value: 00:01:06:200
-> Time Value: 00:01:06:204
-> Time Value: 00:01:06:208
-> Time Value: 00:01:06:208
-> Time Value: 00:01:06:212
-> Time Value: 00:01:06:216
-> Time Value: 00:01:06:220
-> Time Value: 00:01:06:220
-> Time Value: 00:01:06:224
-> Time Value: 00:01:06:228
-> Time Value: 00:01:06:232
-> Time Value: 00:01:06:232
-> Time Value: 00:01:06:236
-> Time Value: 00:01:06:240
-> Time Value: 00:01:06:240
-> Time Value: 00:01:06:244
-> Time Value: 00:01:06:248
-> Time Value: 00:01:06:251
```

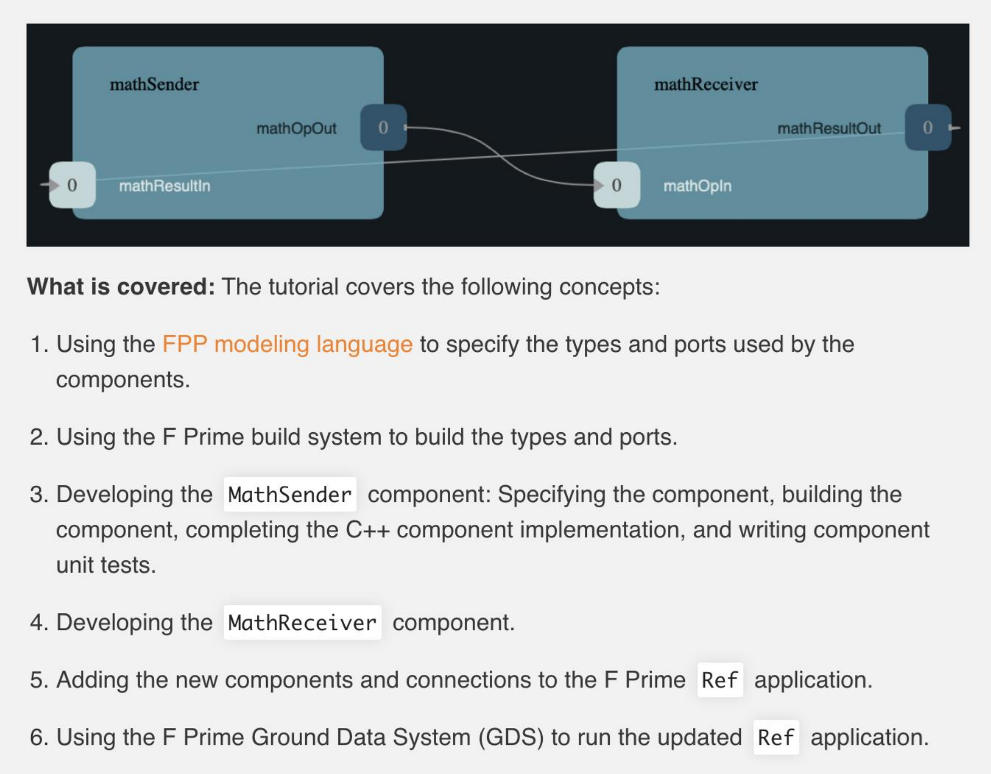
TUTORIAL 3

- Setup F’ for STM32
- Flashing F’ onto STM32
- Firmware development and revision control during collaboration



- Once complete, you need to select a Kit. On the bottom blue bar, next to the wrench icon, I selected “GCC 11.2.0 x86_64-linux-gnu". Then I clicked the play button all the way on the right on the blue bar to apply that. This will also take a decent amount of time.
- Then in the vscode bash terminal and type these commands:
  - cd Ref
  - fprime-util purge FreeRTOS-stm
  - fprime-util generate FreeRTOS-stm
  - fprime-util build FreeRTOS-stm
- It will take ~10 minutes to build and will cook your computer. Then it is ready to be flashed.
- If you do not have stlinktools installed:
  - Download it on mac with $brew install stlink
  - Download on windows with $sudo apt install stlink-toolsGo back to the terminal on your computer. Cd into Ref/build-artifacts/FreeRTOS-stm/bin
- Make sure the H7 is powered and connected to your computer, then $ st-flash write Ref 0x8000000`1 ``1
- That is it, flashed and done.
  - Remember, the code that you flashed is not yet complete, so do not expect a working F’.

CURRENT: MATH TUTORIAL

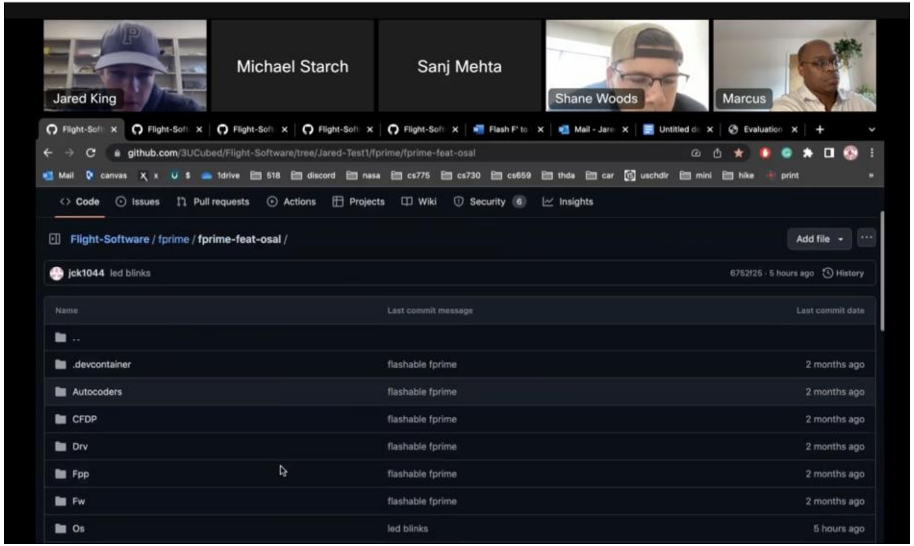


What is covered: The tutorial covers the following concepts:

1. Using the FPP modeling language to specify the types and ports used by the components.
2. Using the F Prime build system to build the types and ports.
3. Developing the `MathSender` component: Specifying the component, building the component, completing the C++ component implementation, and writing component unit tests.
4. Developing the `MathReceiver` component.
5. Adding the new components and connections to the F Prime `Ref` application.
6. Using the F Prime Ground Data System (GDS) to run the updated `Ref` application.

JPL

- F’ is currently not supported on STM32 MCUs
- While we are trying to achieve this, JPL is providing direct support



github.com/3UCubed/Flight-Software/tree/Jared-Test1/fprime/fprime-feat-osal

Flight-Software / fprime / fprime-feat-osal /

jck1044 led blinks 6752f25 · 5 hours ago History

Name	Last commit message	Last commit date
..		
.devcontainer	flashable fprime	2 months ago
Autocoders	flashable fprime	2 months ago
CFDP	flashable fprime	2 months ago
Drv	flashable fprime	2 months ago
Fpp	flashable fprime	2 months ago
Fw	flashable fprime	2 months ago
Os	led blinks	5 hours ago

FLIGHT SOFTWARE COMPARISON

FPrime (F')

- From NASA's Jet Propulsion Lab
- Use STM as OBC emulator
- Capable of having
  - queues,
  - threads, and
  - testing tools, including component architectures and libraries
- Requires laborious implementation
- Open Source

Software Development Kit (SDK)

- From EnduroSat
- Comes with an OBC
- Includes
  - interface drivers,
  - software drivers for sensors,
  - task management,
  - file storage, and
  - application software for sub-systems interface and diagnostics
- EnduroSat Intellectual property

THANK YOU!

Questions?

Come find us if you have more questions.