05/06/23

Time: 7.5h

Monday, June 5, 2023 8:01 AM

Start 8:00 Am

I was thinking about what I had previously looked into with the original camera

I had 2 thoughts -

1. what speed/# of images could be obtained using the 1 channel and no memory - just straight from the camera to the S-band transmitter?

2. Could the 8 channel output on the camera be connected to a parellel→SPI chip to serialize the data.

From the camera's datasheet, the output is a little strange as far as format goes - but I am thinking that rather than fixing the format before sending the the ground, it could be processed after. Resources on the ground are effectively unlimited when compared to the space section

1. According to the Camera datasheet each channel runs at 720 Mbps = 720×10^6 bps

the image size is $2592 \times 2048 = 5.31 \times 10^6$ pixels
assume higher resolution 10 bit format

$5.31 \times 10^6 \cdot 10\text{bit} = 5.31 \times 10^7$ bits/image $\quad 50.31 \times 10^6$ bit/image

$\frac{5.31 \times 10^7 \text{ bits}}{720 \times 10^6 \text{ bits/s}} = 0.07 \text{ s/image} = 13.5 \text{ fps}$

With this high speed, I would be interested to see what the S-band transmitter speed is. without using memory it would need to keep up.

The s-band data sheet

2. Since 1 channel can handle ~ 13 fps I dont really see the need for adding the complexity of the 8 channels

13 fps for 4 minutes would be a maximum of 3240 images an orbit... way more than required

I think it may be reasonable to interface this camera without the need of an MPU with Usb 3.

The main issue that I know nothing about is interfacing with RAM

looking at the selected DRAM chip, it is only 128Mb @ 50Mb/image, only about 2 images can be stored.

Is the FPGA for image compression? cause the memory connected to the FGPA denoted 'Flash (images)' is only 16Mb, which is smaller than the raw output for 1 image

I found some research papers on the Basecamp that discuss image compression on board satellites - maybe I will see some similarities between them and the design

I started reading the Papers and immediately realized my understanding of the image output is wrong

I was thinking of everything as monocromatic - for color, more data is needed than for BW

My understanding of the image sensor was 16bit output per pixel, which works for a 10 bit grey image, but a color would be 30 p.t (Assuming RGB)

The more I look into this, the more I understand the complexity concerns that Nick had.

I would be significantly simpler to use a computer with a USB3 camera, however the power is still a concern.

I have a raspberry pi 4, I could put a low level OS on it and try to run some tests to get an idea of the power. I will need to get a power meter to determine how much power the power supply is actually drawing. Im sure there is one of those at the university though

I downloaded the raspberry pi image for the TinyCore OS, and installed it to my PI. However I ran into issues connecting the device to wifi to allow it to download packages. Usually this is straight forward

Questions

- Was the FPGA intended for doing image compression? As the Flash memory connected labelled (images) is only 16Mb, while a image should be 50Mb

with Linux Distros, but this one does not seem to have the wlan files installed.
I spent about an hour trying to figure this out but did not have much success.

Looking further into the Basecamp, I found some documents discussing image compression. In addition to this, I have found that FPGA's are often used for image compression. I image that this is a core purpose for the FPGA. Earlier I had mentioned that processing could be done on the ground to reduce on board requirements, however image compression would reduce the on board requirements by doing processing before sending the data.

Updated Questions List

1. What specifically is wrong with the current computer system
2. What development has been done on the computer system/ software
3. What was the FPGA for? From what I have put together it seems like the primary function was image compression?
4. Has there been any development or research into actually interfacing the FPGA with the camera, memory and s-band?
5. I have looked into MPU such as the raspberry pi to use a USB camera and avoid a lot of the low level stuff. My main concern with this is the power. I found the spreadsheet that the power analysis was completed in and changing the primary MCU value to that of the Pi (~3W), there is a net power deficit
6. What is my scope. What things should I be prioritizing

Meeting w/ Grant

- FPGA - image capturing
- How many images hold 1-2 images minimum
- ConOps has flowcharts for software requirements
- Nothing was developed for details about onboard computer it was very high level
- Would be good to look into ESP-32 Camera

Next Steps

- ESP-32 Camera ? mid power w/ camera interface
- Look into requirements breakdown to ensure
- Look into mass requirements as well - Spreadsheet
- Camera - temp + global shutter most important

Meeting time 2:00 - 2:45

Leave 330

06/06/23 Time: 10h

June 6, 2023 8:04 AM

Start 6:04 AM

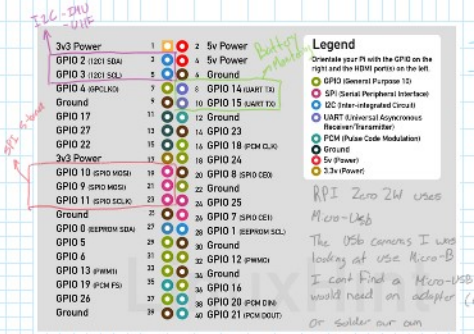
Last night I had a conversation with Nick regarding the RPI 4 and its power limitation. He agreed that USB2 should suffice, meaning I can chose a lower powered board.

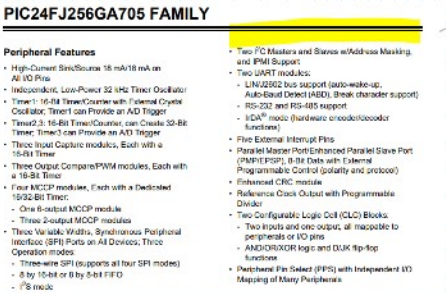
I believe the RPI Zero requires <1W in power which would be even less than the original MCU system.

I do not know if it has the same number of GPIO/Communication pins.

Questions

- What hardware is available for testing? S-band, UHF, IMU, etc? Would be nice especially when trying to test communication interfaces. Really could use any I2C SPI Uart etc sensor to do this.
- Less of a question, but I want to get a USB to header adapter and try to communicate to a PI over the UART pins. The main thing I want to look at is whether files can be communicated over the UART pins. I am getting mixed results on the feasibility of this. My thought is that this could be used to upload new control scripts to the PI
- Programming language? Python or C++. From my research both have libraries available for interacting with the hardware on the PI (GPIO, SPI, UART, I2C etc.). I expect that python will be easier, however it is not as fast as C++ as it is interpreted instead of compiled.





Pi Model	Pi State	Power Consumption
4 B	HDMI off, LEDs off	TODO
4 B	HDMI off, LEDs off, onboard WiFi	TODO
3 B+	HDMI off, LEDs off	350 mA (1.7 W)
3 B+	HDMI off, LEDs off, onboard WiFi	400 mA (2.0 W)
3 B	HDMI off, LEDs off	230 mA (1.2 W)
3 B	HDMI off, LEDs off, onboard WiFi	250 mA (1.2 W)
2 B	HDMI off, LEDs off	200 mA (1.0 W)
2 B	HDMI off, LEDs off, USB WiFi	240 mA (1.2 W)
Zero 2 W	HDMI off, LED off	100 mA (0.6 W)
Zero 2 W	HDMI off, LEDs off, onboard WiFi	120 mA (0.7 W)
Zero	HDMI off, LED off	80 mA (0.4 W)
Zero	HDMI off, LED off, USB WiFi	120 mA (0.7 W)
B+	HDMI off, LEDs off	160 mA (0.8 W)
B+	HDMI off, LEDs off, USB WiFi	320 mA (1.1 W)
A+	HDMI off, LEDs off	80 mA (0.4 W)
A+	HDMI off, LEDs off, USB WiFi	160 mA (0.8 W)

Also, as a point of reference, when you power off a Raspberry Pi (any model), it typically uses 20-30 mA (0.1W) until you physically disconnect the power.

https://www.pidramble.com/wiki/benchmarks/power-consumption

This states that the RPI Zero 2W can go as low as 0.6W

2 Power consumption

The improved performance of the Pi Zero 2 W does see it draw a little more power, which is something that you'll have to factor into your projects. With the help of Stressberry we found that the original Pi Zero W draws around 260mA at idle and 370mA when stressed. Running the same test on the new Pi Zero 2 W produced 280mA at idle and 580mA under stress.

https://topnewreview.com/raspberry-pi-zero-2-w-review/#:~:text=Power%20consumption,-The%20improved%20performance&text=Running%20the%20same%20test%20on,idle%20and%20580mA%20under%20stress.

This states around 2.9W was seen under load.

Power Analysis

This budget it based on a MATLAB simulation that determined the values for time spent in active mode. Based on 16 Orbits.

Specific Power Consumption In Active Mode			16.7		1485.2
Component	**Rate [W]**		**Time [min]**		**Draw [Wh]**
	Active	*Passive*	*Active*	*Passive*	
Structure + Payload					
Image Sensor	1.6	0.05	4.175	12.525	0.122
Communications					
UHF Transceiver	1.363	0.15	4.175	12.525	0.126
S Band Transmitter	5	0.1	4.175	12.525	0.369
Command + Control					
On-Board Computing	-	-	16.7	0	
Primary MCU	2.9	0	16.7	0	0.807
Primary MCU Flash			0	0	0
Secondary MCU	0	0	16	0	0.000
Flash for Secondary MCU			0	0	0
FPGA		0	4.175	0	0.000
FPGA RAM			0	0	0
FPGA Flash			0	0	0
Reaction Wheels	1.5	0	16.7	0	0.418
Motor		0			
Motor Driver		0			
Magnetorquers	2.1	0	16.7	0	0.585
IMU	0.0056	0.0025	4.175	12.525	0.001
PSD	0	0	0	0	0
Power					
Power System	0.2	0	16.7	0	0.056
Battery Heaters	0.8	0	16.7	0	0.223
20% Contingency					0.541
Total					**3.246**
Average Power Consumption In Active Mode (W)					11.663

Specific Power Consumption in Standby			1391.2		1485.2
Component	**Rate [W]**		**Time [min]**		**Draw [Wh]**
	Active	*Passive*	*Active*	*Passive*	
Structure + Payload					
Image Sensor	1.6	0.05	0	1391.2	1.159
Communications					
UHF Transceiver	1.4	0.15	0	1391.2	3.478
S Band Transmitter	5	0.1	0	1391.2	2.319
Command + Control					
On-Board Computing	-	-	1391.2	0	
Primary MCU	1	0	1391.2	0	23.187
Primary MCU Flash			0	0	0
Secondary MCU	0	0	0	1391.2	0
Flash for Secondary MCU			0	0	0
FPGA		0	0	0	0
FPGA RAM			0	0	0
FPGA Flash			0	0	0
Reaction Wheels	1.5	0	0	0	0
Motor		0			
Motor Driver		0			
Magnetorquers	2.1	0	0	0	0
IMU	0.0056	0.0025	0	1391.2	0.058
PSD	0	0	0	0	0
Power					
Power System	0.2	0	1391.2	0	4.637
Battery Heaters	0.8	0	1391.2	0	18.549
20% Contingency					10.677
Total					**64.065**
Average Power Consumption In Standby Mode (W)					2.763

Specific Power Consumption in Power Saving Mode			77.3		1485.2
Component	**Rate [W]**		**Time [min]**		**Draw [Wh]**
	Active	*Passive*	*Active*	*Passive*	
Structure + Payload					
Image Sensor	1.6	0.05	0	77.3	0.064
Communications					
UHF Transceiver	1.4	0.15	0	77.3	0.193
S Band Transmitter	5	0.1	0	77.3	0.129
Command + Control					
On-Board Computing	-	-	77.3	0	
Primary MCU	0.6	0	77.3	0	0.773
Primary MCU Flash			0	0	0
Secondary MCU	0	0		0	0.000
Flash for Secondary MCU			0	0	0
FPGA	1	0	0	0	0
FPGA RAM			0	0	0
FPGA Flash			0	0	0
Reaction Wheels	1.5	0	0	0	0
Motor		0			
Motor Driver		0			
Magnetorquers	2.1	0	0	0	0
IMU	0.0056	0.0025	0	77.3	0.003
PSD	0	0	0	0	0
Power					
Power System	0.2	0	77.3	0	0.258
Battery Heaters	0.8	0	77.3	0	1.031
20% Contingency					0.490
Total					**2.941**
Average Power Consumption In PSM (W)					2.283

Average Power 2.8381

According to this, if the Standby mode can use less than 1W of power, this will work. That is including the contingency. I am confident that the control system will not bring the PI close to its limit, but I am not sure if < 1W will be achieved. I am not sure how this could really be tested accurately without developing the software and testing it.

The main concern that I have with the RPI now is the uploading of new software/the ability to communicate with the OS. Previously I found that you could interface with the RPI 4 serially using the first set of UART pins. My thought was that this would allow the UHF transceiver to connect and you could essentially "Shell" into the PI, allowing for the terminal to be accessed or files to be uploaded.

The Zero only has the one UART interface which needs to connect to the battery monitering system.

I am also not sure if the UART pins can be used to interface with the OS. Something tells me when I saw that with the 4 it was exclusive to the 4.

The other concern that I have is the requirement

[SYS-COM-040] Communication shall be bi-directional, allowing for complete re-upload of system code; full transmission must be sent within the time that a connection is present on a single pass of the ground station.

For an MCU I think this is a little more clear. As the MCU just runs the script uploaded onto it. However the MPU has the control scripts as well as an OS. The OS would not be able to be changed easily. If I can get the UART interface for the UHF to work, I think uploading control scripts would be relatively straight forward.

Weight Budget

The components that I am changing from the original design are:

RPI Zero 2W	+11g
CMOSIS CMV4000	+6g
MCU	+?
USB Cable	+?
SD card	+2g
Primary MCU	-
Secondary MCU	-
FPGA	-
Python 5000	-

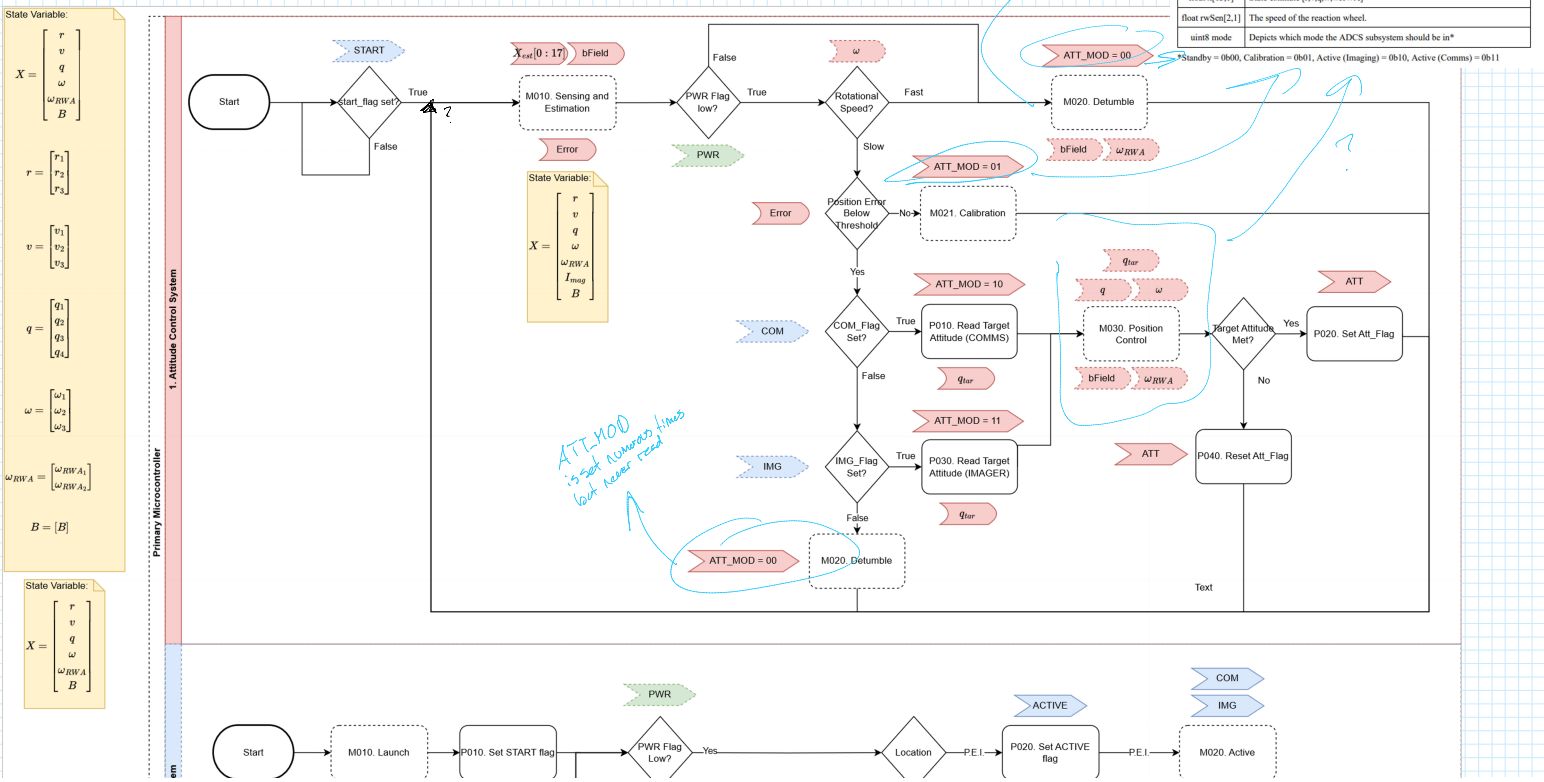
It looks like the mass budget was not filled out for the Command system before, as every part was 1g

There are 250grams allocated. I do not think that this will become an issue. The camera is not considered the Command system, so I will need to check that separately
The Python Camera and PCB were 3.1g and 3.74g respectively.

6.74g is less than that of the USB3 camera I selected, and the optical payload was previously 100g under budget so that should not be an issue.

I guess the next issue would be looking into the program structure and trying to map everything out in my head so I understand the larger flow while making the smaller functions

I am not sure what language I will use to program, that may be something to look into. I think it would be between C/C++ or Python. Python seems to be a standard on raspberry pi's but that is likely because easy to learn as people who use the PI aren't usually professionals.



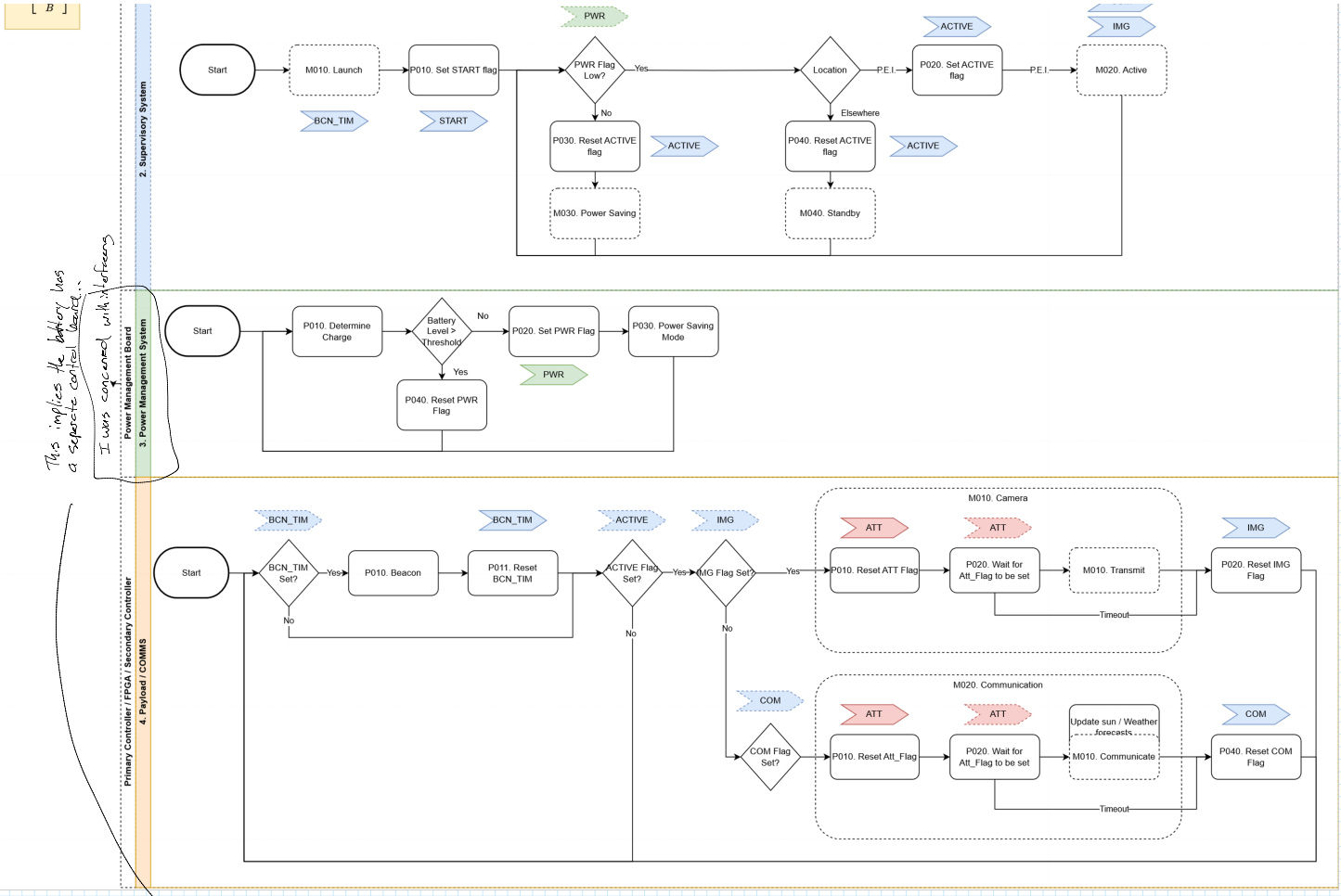
7.4.3 Control

The control function is designed to interpret the mode from the guidance function such that it can select the appropriate control logic. It then uses this logic to determine the outputs to the actuators based on the current error quaternion and the current environment.

7.4.3.1 Inputs

Input	Description
float bSen[3,1]	The local magnetic field, as measured by the magnetometer.
float error[4,1]	Error quaternion
float x[15,1]	State estimate [r,v,q,w,wRWA]
float rwSen[2,1]	The speed of the reaction wheel.
uint8 mode	Depicts which mode the ADCS subsystem should be in*

*Standby = 0b00, Calibration = 0b01, Active (Imaging) = 0b10, Active (Comms) = 0b11



Can the battery monitoring IC be programed to give a threshold? What I was seeing was that the chip would just communicate the charge to the MCU over UART and the logic/flag would have to be done in the MCU

I am not seeing anything that suggests it is programable did they have a dedicated MCU???

The Battery monitoring is not even on the Block diagram for the OBC...

looking at the mass breakdown The power management

the charging voltage for the battery pack. This voltage is received by the BQ79606A-Q1 battery charger and monitor. This IC can balance cell charging while monitoring cell voltage and temperature and protecting against under/over voltage or temperature. The BQ7960 IC also has UART interface capabilities. This will be used as a line of communication between the OBC and power management PCB, current channels for telemetry include battery voltage and battery temperature. Communication between the OBC and power management PCB will serve as the method for entering varying power modes such as active or power-saving mode (PSM). Power is

This agrees with my original assumption

Thread:
ACS
If Start Flag{
Sensing and Estimation
If Low Power {
Detumble
} else {
If High rotational speed{
Detumble
} elif Position error above thresh {
Calibration
} elif Com flag set?{
Read Target Att Comms
Position Control

If Target Att met {
Set att flag
} else {
Reset att flag
}
} elif IMG flag set {
Read target Att Imager
Position Control

If Target Att met {
Set att flag
} else {
Reste att flag
}
} else {
Detumble
}
}

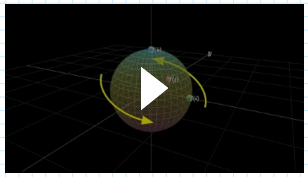
}

I see that one of the of the variables in the TRR software breakdown refers to quaternions. I don't really know what that means, so that might be something to look into before trying to understand the software plan

Quaternions

https://www.allaboutcircuits.com/technical-articles/dont-get-lost-in-deep-space-understanding-quaternions/#:~:text=Quaternions%20are%20an%20alternate%20way,not%20suffer%20from%20gimbal%20lock.

Quaternions and 3d rotation, explained interactively



Fantastic Quaternions - Numberphile

https://eater.net/quaternions

I think I have a generally understanding of what quaternians are and their format, but I would like to get a better understanding in this context. What is the axis? What is the point being rotated

State Variable:

$$X = \begin{bmatrix} r \\ v \\ q \\ \omega \\ \omega_{RWA} \\ B \end{bmatrix}$$

$$r = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$$

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

$$\omega_{RWA} = \begin{bmatrix} \omega_{RW,A_1} \\ \omega_{RW,A_2} \end{bmatrix}$$

$$B = [B]$$

→ Presumably q refers to the quaternion as it is 4D

I am struggling to make sense of all there flow charts. I can understand the larger processes, but know what the variables are and the math/calculations that need to take place under a given function are not clear.

Part of me wants to start trying to program the ones that make the most sense just so that the whole thing becomes less overwhelming, however that could just lead into wasted time down the line.

I still have not determined what language is the best.

C/C++ Vs Python

Python appears to be easier to use and will likely be much faster to get something working

On the other hand, C/C++ would be compiled which means that they will run significantly faster. One thing would be looking into the complexity of threading in each language as that will likely be important.

https://realpython.com/intro-to-python-threading/

Python

```python
import logging
import threading
import time

def thread_function(name):
    logging.info("Thread %s: starting", name)
    time.sleep(2)
    logging.info("Thread %s: finishing", name)

if __name__ == "__main__":
    format = "%(asctime)s: %(message)s"
    logging.basicConfig(format=format, level=logging.INFO,
                        datefmt="%H:%M:%S")

    logging.info("Main    : before creating thread")
    x = threading.Thread(target=thread_function, args=(1,))
    logging.info("Main    : before running thread")
    x.start()
    logging.info("Main    : wait for the thread to finish")
    # x.join()
    logging.info("Main    : all done")
```

→ I like the idea of having logs might be helpful debugging

← Thread targets function

https://www.geeksforgeeks.org/multithreading-in-cpp/

It looks like threading is native in both languages. Additionally it appears that there is some similar syntax between the languages. In both the thread references a function and both feature the same .join() function to bring threads back to the main thread.

The other thing is interfacing with the pins in python and C++

I know that python has a library for this, however I have not seen how it works in C++
https://www.digikey.ca/en/maker/blogs/2019/how-to-use-gpio-on-the-raspberry-pi-with-c

It appears that the Raspbian OS includes a library for C++ to interface with the IO pins. I was considering not using the Raspbian OS and finding a lighter linux distro, however I imagine I could manually install the library if the OS doesn't contain it natively.

Our Simple Example – A Blinking Light with Disable Switch Code

Copy Code

```
#include <iostream>             // Include all needed libraries here
#include <wiringPi.h>

using namespace std;            // No need to keep using "std"

int main()
{
wiringPiSetup();                        // Setup the library
pinMode(0, OUTPUT);             // Configure GPIO0 as an output
pinMode(1, INPUT);              // Configure GPIO1 as an input

// Main program loop
while(1)
{
        // Button is pressed if digitalRead returns 0
        if(digitalRead(1) == 1)
{
        // Toggle the LED
        digitalWrite(0, !digitalRead(0));
delay(500);     // Delay 500ms
}
}
        return 0;
}
```

The code actually appears to be extremely similar to the Arduino Language which I am familiar with.

Another concern would be interfacing with the various communication protocols (UART SPI I2C)
https://www.electronicwings.com/raspberry-pi/raspberry-pi-uart-communication-using-python-and-c
This link describes UART for both Python and C

https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial/all
This link has a tutorial for both SPI and I2C on the PI in both Python and C++

Regarding image acquisition, I believe OpenCV is designed for C++ and is also compiled for python, meaning that it could be used in each of the languages. Additionally, it may have some image compression methods built in that could be useful for data transfer.

OpenCV can compress images in PNG format
https://vovkos.github.io/doxyrest-showcase/opencv/sphinxdoc/enum_cv_ImwriteFlags.html

From what I am reading, the higher compression the longer it takes to compress. Depending on what this time frame is, it is possible that the software would compress the images while not in position to take images as there is lots of down time. Images would then to transferred faster as the images would be much smaller.

I think that covers all aspects of the software...

At this point, I do not think that the language will make too much of a difference regarding capability. The C++ language would be faster as it is compiled as opposed to interpreted.

I created a GitHub Repo and started laying out the software. My plan for formatting the software is a main.cpp file that will run all the control systems. This file will contain the threads that will be executed for each of the subsystems in the Flowchart above. The functions for each of the threads will be separated into a separate .cpp and .h file to allow for better organization.

I laid this out for the ACS thread and tried to compile to ensure I formatted it right. I ran into compilation issues but from looking into the errors it appears that the compiler is not referencing the external .cpp and .h files for the ACS functions that are referenced in the main.cpp file.

Leave 4:00pm

Start 6:15
I want to try to figure out what the compilation issue is cause that is going to hold me back and I am excited to get into the programming part.

I got the code to compile using the build code button by changing the vscode tasks.json file to:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: g++.exe build active file",
            "command": "C:\\MinGW\\bin\\g++.exe",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${fileDirname}\\threads\\*.cpp",
                "${fileDirname}\\threads\\*.h",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe"

            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
```

It still does not compile through the 'Run Code' button in the upper left, but at least I know its just a settings issue, not an issue with the software I am writing.

I tried looking for some logging libraries. I feel like it will be very nice for development and use to have a logging system to track what processes are happening when and if there are any errors.

There is a library from Google called glog however I have having trouble installing it.

I am used to Arduino where I can just drop a .zip in the lib folder and everything works. Unfortunately it does not seem that simple. I feel like it would be good to have the logging system involved from the beginning.

```
/**
 * @file main.cpp
 * @author Damien Doucette (dldoucette@upei.ca)
 * @brief Main control software for the UPEI Spudnik-1 CubeSat
 * @version 0.1
 * @date 2023-06-06
 *
 * @copyright Copyright (c) 2023
 *
 */
//Include External libraries
#include <iostream>
#include <thread>
//Include Local libraries
#include "threads/acs.h"
using namespace std;
//Declare global variables
bool startFlag = 1;
bool pwrFlag = 0;
bool comFlag = 0;
bool imgFlag = 0;
void acs(){
    /*
    This function will contain the thread for the ACS software for the Satellite
    Sub-functions for this thread are stored in a seperate file under the 'threads' folder
    The logic for this thread was based off the '0.Main.drawio' file in the 'ConOps' GDrive
    */
    //Delcare local vairables
    double w = 0;
    double posErr = 0;
    double bField[3];
    double w_RWA[2];
    uint8_t ATT_MOD = 0x00; //Will have to check what default value is supposed to be
    //Logic defined in FlowChart
    printf("ACS Function started\n");
    if(startFlag){
        printf("Start flag true\n");
        sensingAndEstimation();
        if(pwrFlag){
            if(w < 1){
                if(posErr < 1){
                    if(comFlag){
                        ATT_MOD = 0x10;
                        readTargetAttitudeCOMMS();
                        positionControl();
                        if(1){ //Target attitude met
                            //Set attFlag
                        } else {
                            //Reset attFlag
                        }
                    } else {
                        if(imgFlag){
                            ATT_MOD = 0x11;
                            readTargetAttitudeIMG();
                            positionControl();
                            if(1){ //Target attitude met
                            //Set attFlag
                            } else {
                                //Reset attFlag
                            }
                        } else {
                            ATT_MOD = 0x00;
                            detumble();
                        }
                    }
                } else {
                    ATT_MOD = 0x01;
                    calibration();
                }
            } else {
                ATT_MOD = 0x00;
                detumble(); //Might need arguements? one call uses them and one doesnt in flowchart
            }
        }
    }
```

```
}
void setup(){
    printf("Beginning setup loop...\n");
    printf("Setup Complete.\n");
}
void loop(){

}
int main()
{
    setup();
    thread acsThread(acs);
    printf("ACS Thread started...\n");
    // while(1){
    //     loop();
    // }
}
```

I created the thread object based off the of the documentation I found earlier. It is not throwing any errors, however the thread does not appear to be running. The script also terminates so it is not getting stuck anywhere…

I found my problem. If I don't include a .join() function at the end the thread will not run

Leave 8:15

07/06/23 Time:7.25h

June 7, 2023 9:11 AM

Start 9:15

I was considering whether I should also move the logic/flowchart part of the code for each thread into the separate file or include it in the main file. One way to reference a thread is to make a class. I could organize each section into a class with the logic being the initializer function and the sub functions being defined.

Another thing that would be good is to try to figure out what all of the variables represent. I tried to do this before but the ConOps and TRR have different names for the same variables which makes it confusing.

I feel like trying to start in the ACS software is not a good idea. I did this because it seemed to be the most developed thread. The issue is that it the higher level controlling function. I feel like developing the smaller sections would be less overwhelming cause each section would be smaller and in doing them I would get a better understanding of the variables and data handled in the ACS.

https://3.basecamp.com/4059304/buckets/9061579/uploads/3425065398
I found this document by Josh O that has a detailed description of the parameters that match the flowchart.
It looks like this document is a work in progress one... I wonder if there is a polished version that would be better to reference.

5.2 Simulated System

5.2.1 Sensors

WRITE DESCRIPTION HERE

5.2.1.1 Inputs

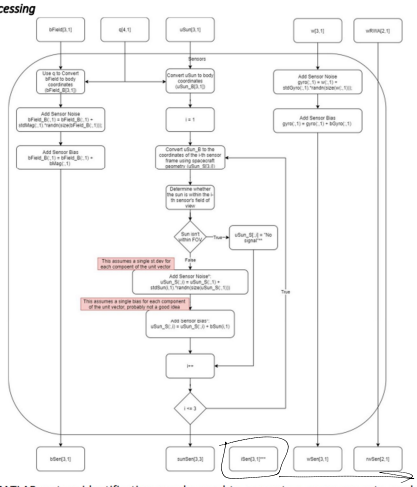
Input	Description
float q[4,1]	The quaternion from the inertial frame to the body frame
float w[3,1]	The angular velocity of the satellite, expressed in body frame coordinates.
float bField[3,1]	The local magnetic field vector, expressed in inertial frame coordinates.
float uSun[3,1]	The unit sun vector, expressed in inertial frame coordinates.
float wRWA[2,1]	The speed of the reaction wheels

5.2.1.2 Parameters

Parameter	Description
float stdSun[3,1]	Standard deviation for each sun sensor*
float bSun[3,1]	Bias for each sun sensor*
float stdGyro[3,1]	Standard deviation for each gyro
float bGyro[3,1]	Bias for each gyro
float stdMag[3,1]	Standard deviation for each magnetometer.
float bMag[3,1]	Bias for each magnetometer

*Note that MATLAB system identification may be used to generate a more accurate model of the sun sensor, rather than simply using a sensor bias and standard deviation.

5.2.1.3 Processing



Does this block correspond to a block in the Main Flowchart
MO30 has nearly the same inputs
I dont understand how the sun unit vector is both an input and an output

iSens is an output that is not connected to anything

*Note that MATLAB system identification may be used to generate a more accurate model of the sun sensor, rather than simply using a sensor bias and standard deviation.

5.2.1.4 Outputs

Output	Description
float sunSen[3,3]	The unit sun vector as measured by each of the three sun sensors
float wSen[3,1]	The satellite's angular speed, as measured by the gyroscope.
float bSen[3,1]	The local magnetic field, as measured by the magnetometer.
float rwSen[2,1]	The speed of the reaction wheel.
float iSen[3,1]	The current through each of the magnetorquers.

6.1.2 Navigation

Description?

6.1.2.1 Inputs

Input	Description
float rwSen[2,1]	The speed of the reaction wheels.
struct satrec	Previous orbital data structure
float wSen[3,1]	The satellite's angular speed, as measured by the gyroscope.
float pPrev[7,7]	Covariance of the previous attitude estimate
float sunSen[3,3]	The unit sun vector as measured by each of the three sun sensors
float bSen[3,1]	The local magnetic field, as measured by the magnetometer.
float xPrev[15,1]	Previous state estimate [r,v,q,w,wRWA]
float tOB	On-board time
char TLE[2,69]	Two-line element data

6.1.2.2 Parameters

Parameter	Description
float MFE	Minutes from epoch
float lat	Satellite latitude
float lon	Satellite longitude
float alt	Satellite altitude
float B[3,1]	Modeled magnetic field vector
float U[3,1]	Modeled sun vector
Q[7,7]	Attitude model covariance matrix
R[7,7]	Attitude measurement covariance matrix

6.1.2.3 Outputs

Output	Description
float p[7,7]	Covariance of the current attitude estimate
float x[15,1]	Current state estimate [r,v,q,w,wRWA]
struct satrec	Current orbital data structure

Commented [JO4]: Expand on the reference frames of r (ECI),v(ECI),q(body-ECI), and w (body)

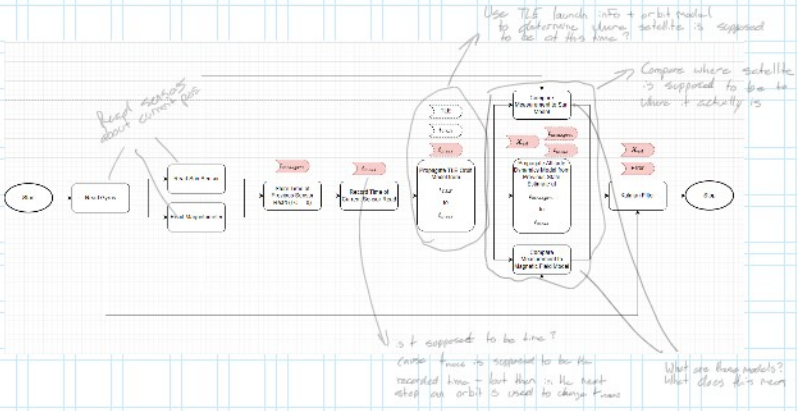
Im realizing this is organized the same as the TRR document, however for some reason

Use TLE launch info + orbit model

float x[15,1]	Current state estimate [r,v,q,w,wRWA]
struct satrec	Current orbital data structure

Commented [JO4]: Expand on the reference frames of r [ECI],v[ECI],q[body-ECI], and w (body)

Im realizing this is organized the same as the TRR document, however for some reason



Break 12:00
Back 12:30

Updated GitHub Repo

I want to try to find someone documentation listing the limits of some of these things — especially the special reference frames and the timescale of doing the conversion/calculations etc.

Looking further into the Document from Josh above, I got a lot more information. The first thing that I noticed is that the plan is to use variable current in the magnetorquers like I had originally assumed.

$$\vec{m} = |m|\left[\frac{\vec{m}}{|\vec{m}|}\times\frac{\dot{\vec{B}}}{|\dot{\vec{B}}|}\right] - \frac{\dot{\vec{m}}}{|\dot{\vec{B}}|}(\vec{B}\times\vec{r}) - \frac{\dot{\vec{r}}}{|\dot{\vec{B}}|}(\vec{B}\times\vec{r})$$

Note how $|\vec{m}|$ decreases as $\vec{B}$ and $\dot{\vec{B}}$. This ensures that the magnetorquers are only used when they are most effective (ie. when small moments are required to produce large torques, rather than when large moments are required to produce small torques) and prevents the undesired torque components that were previously mentioned from getting too significant. In knowing the magnitude and direction of the magnetic moment required to produce the desired torque, it is then possible to calculate the current required through the magnetorquers such that this can torque can be achieved. The required current can be determined as follows:

$$I = \frac{|\vec{m}|}{nA} = \begin{bmatrix} \frac{1}{n_x A} & 0 & 0 \\ 0 & \frac{1}{n_y A} & 0 \\ 0 & 0 & \frac{1}{n_z A} \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}$$

Where n is the number of turns, I is the current, and A is the cross-sectional area if of the magnetorquer.

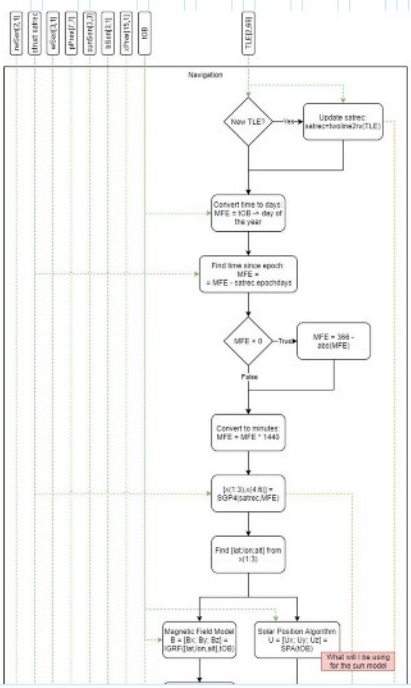
This means that more PWM pins are going to be required...

6.2.2.4 *Processing*

The navigation function (Depicted in **Figure _below_**) first determines the satellite's orbital state, to do so, it checks the TLE data. If the data has been updated since the last iteration, then "satrec" will be updated using the "twoline2rv" function provided in the SGP4 software package. Otherwise, the "satrec" structure from the previous iteration will be used. Once the observation time has been determined, its "epochdays" field will be used in the conversion of the OBC time to minutes from epoch (MFE) as required by the SGP4 algorithm. Both the "satrec" structure and the MFE are then used as inputs to the "SGP4" function, also provided in the SGP4 software package; this function then propagates the state of the satellite through time to estimate the orbital position and orbital velocity vectors in terms of ECI coordinates.

Once the orbital state has been determined, it is then possible to estimate the state of attitude. To do so, the navigation function first uses the IGRF model and an SPA to estimate both the local magnetic field vector in terms of NED coordinates, and the unit sun vector in terms of ECI coordinates. After converting the magnetic field vector into ECI coordinates, these two values are then fed into the TRIAD method which, as described in **Section _Triad Algorithm_**, will compare these values to the body-frame measurements from the 3-axis magnetometer and the sun-sensors such that a measurement-based estimate of the current ECI-body quaternion can be obtained. In terms of angular velocity, this can be measured directly about the body frame using the IMU's 3-axis gyroscope. In addition to the measurement-based estimation, the magnetic field and unit sun vectors are also used as inputs to an attitude model which pairs the basic attitude dynamics (shown in **Section _Attitude Dynamics_**) with a fourth-order Runge-Kutta solver for a theoretical estimate of both the current quaternion and the angular velocity. The measurement- and model-based estimations are then compared using a Kalman filter, which will output the most probable state of attitude.

Upon completing execution, the navigation function will then output the estimated state vector, consisting of the orbital position and orbital velocity vectors obtained from the SGP4 propagator, the quaternion and angular velocity from the Kalman filter, and the speed of the reaction wheels as read from the motor drivers. To supplement this, it will also output the covariance matrix of the quaternion and angular position, allowing decisions to be made based on the uncertainty of the estimate.



=> look into SGP4 software package

SGP4

https://github.com/aholinch/sgp4

SGP4 test file

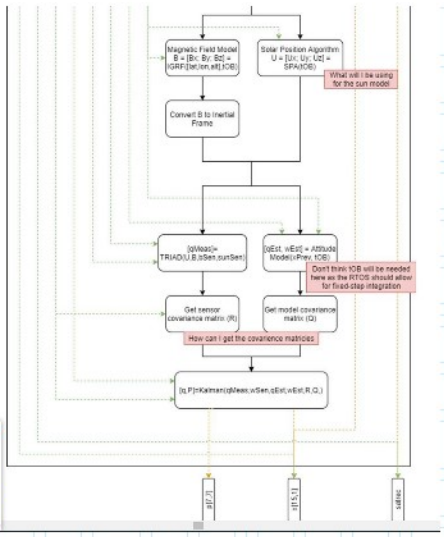
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "tle.h"
typedef struct VERIN {
    char line1[70];
    char line2[70];
    double startmin;
    double stepmin;
    double stopmin;
} VERIN;
/**
 * returns the count of verins read and sets the point to an array created with malloc.
 */
int readVERINs(VERIN **listptr)
{
    char line[256];
    char *str = NULL;
    FILE *in_file = NULL;
    VERIN *verins = NULL;
    int cnt = 0;

    in_file = fopen("../../data/SGP4-VER.TLE","r");

    while(fgets(line,255,in_file) != NULL)
    {
        if(line[0]=='1')cnt++;
    }
    if(in_file)
    {
        fclose(in_file);
    }
    verins = (VERIN*)malloc(cnt*sizeof(VERIN));
    *listptr = verins;
    cnt = 0;
    in_file = fopen("../../data/SGP4-VER.TLE","r");

    while(fgets(line,255,in_file) != NULL)
    {
        if(line[0]=='1')
        {
            strncpy(verins[cnt].line1,line,69);
            verins[cnt].line1[69]=0;
            fgets(line,255,in_file);
            strncpy(verins[cnt].line2,line,69);
            verins[cnt].line2[69]=0;
            str = &line[70];
            sscanf(str,"%lf %lf %lf",&verins[cnt].startmin,&verins[cnt].stopmin,&verins[cnt].stepmin);
            cnt++;
        }
    }
    if(in_file)
    {
        fclose(in_file);
    }
    return cnt;
}
/**
 * 2-norm distance for two three vectors
 */
double dist(double *v1, double *v2)
{
    double dist = 0;
    double tmp = 0;
    // just unroll the loop
    tmp = v1[0]-v2[0];
    dist += tmp*tmp;
    tmp = v1[1]-v2[1];
    dist += tmp*tmp;
    tmp = v1[2]-v2[2];
    dist += tmp*tmp;
    return sqrt(dist);
}
void runVER(VERIN *verins, int cnt)
{
    FILE *in_file = NULL;
    TLE tle;
    double r[3];
    double v[3];
    double rv[3];
    double vv[3];
    double mins = 0;
    char line[256];
    int i=-1;
    char *ind = NULL;
    double rdist = 0;
    double vdist = 0;
    double rmrr = 0;
    double vmrr = 0;
    int cnt2 = 0;
    in_file = fopen("../../data/tcppver.out","r");
    while(fgets(line,255,in_file) != NULL && i <cnt)
    {
        ind = strstr(line,"##");
        if(ind != NULL)
        {
            i++;
            tle.parseLines(verins[i].line1,verins[i].line2);
        }
        else
        {
            sscanf(line,"%lf %lf %lf %lf %lf %lf %lf\n",&mins,rv,&rv[1],&rv[2],vv,&vv[1],&vv[2]);
            tle.getRV(mins,r,v);
            rdist = dist(rv,
```

Verin seems to refer to the TLE



6.1.4.4 Processing

The control function, seen in Figure _Below_ first looks to the mode input for the selection of a control logic. In the case where mode = 0b00 (detumbling), the rate of change of the magnetic field vector will first be used to approximate the angular speed vector (as mentioned in section _B Dot_). To get this value, the difference between the current and previous magnetometer measurements will be taken and divided by the time differential between measurements; note that these values will be updated at a consistent update rate in real-time, so the time differential will be a constant. After approximating the angular speed, the magnetic moments will then be defined such that the resultant torque vector will have a direction which opposes that of the angular velocity vector as much as possible and a magnitude which is proportional to the perpendicularity between the magnetic field and angular velocity vectors; note that the only case where the direction will fully oppose that of the angular velocity would be when the angular velocity vector is orthogonal with the magnetic field vector. After finding the magnetic moment the time derivative of the the magnetic field vector will be compared with a limit to determine whether the satellite is currently tumbling. If the value is less than the limit, then the satellite is not tumbling and the reaction wheels will be slowed to prevent future saturation. Otherwise, the wheels will be kept at a constant rate to prevent any further tumbling.

How would this be done in real time? On their board maybe ISR's were being used. I dont have access to ISR's on the pi.

This many course problems

Comm Interfaces

I2C

https://raspberry-projects.com/pi/programming-in-c/i2c/using-the-i2c-interface

This website gives a simple implementation of the I2C protocol on RPi. I feel like this is understandable enough to implement.

It would be nice to get some sensor that interfaces with I2C so I can try to get this working. Knowing that I can interface will all of the hardware will be comforting when tackling the task of all the control software.

I added this into the sensingAndEstimation function. I am running into issues where the libraries aren't available, however this is supposed to be interfacing with the OS on the PI and therefor I am not going to be able to get this section running on my Windows PC.

For this reason, I at some point I am going to need to switch to testing software on a PI or maybe any Linux distro will work.

For . = 0:2

sunSenSi3[0]

i	4i+0	4i+1	4i+2	4i+3
ss[0] = I[0]	[I1]	[I2]	[I3]	
1	4	5	6	7
2	8	9	10	11

I added a loop to convert the 12 analog signals for the sun sensors into the x-y position for each sun sensor

Stop 5:00pm

```
    $ind = strstr(line,"xx");
    $if($ind != NULL)
    {
        i++;
        tle.parseLines(versns[1].line1,versns[1].line2);
    }
    else
    {
        sscanf(line,"%lf %lf %lf %lf %lf %lf %lf\n",&mins,rv,&rv[1],&rv[2],vv,&vv[1],&vv[2]);
        tle.getRV(mins,r,v);
        rdist = dist(r,rv);
        vdist = dist(v,vv);
        rerr += rdist;
        verr += vdist;
        cnt2++;
        if(rdist > 1e-7 || vdist > 1e-8)
        {
            printf("%s %lf %e %e\n",tle.objectID,mins,rdist,vdist);
        }
    }
}
rerr = rerr/cnt2;
verr = verr/cnt2;
printf("Typical errors r=%e km, v=%e km/s\n",1e6*rerr,1e6*verr);
if(in_file)
{
    fclose(in_file);
}
}
int main(void)
{
    int cnt = 0;
    VERIN *versns = NULL;
    cnt = readVERIN(&versns);
    printf("read %d versns\n",cnt);
    runVER(versns,cnt);
    free(versns);
    return 0;
}
```

I don't understand why all of this is in a test file. I seems like a lot of this stuff for reading the files and such should be a pre-defined function somewhere

Also the main thing referenced in the Document was a twoline2rv command which does not appear to be present.

I found another source for a SGP4 library and this one has a twoline2rv function in the .h file

However it is not being accessed when

08/06/23

June 8, 2023 8:29 AM

Time: 7h

Start 8:30

I am going to need to find a adapter to begin testing with the serial interfacing of the PI.
https://www.amazon.ca/dp/B08G1JTN4N/ref=sspa_dk_detail_7?pd_rd_i=B08G1JTN4N&pd_rd_w=Pv7nK&content-id=amzn1.sym.d8c43617-c625-45bd-a63f-ad8715c2c055&pf_rd_p=d8c43617-c625-45bd-a63f-ad8715c2c055&pf_rd_r=57TD0R0VWC644SMHWKZJ&pd_rd_wg=ve61z&pd_rd_r=043066fe-1ffa-4aae-b5bf-faa9403d4200&s=electronics&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWw&smid=A2VPXM5R5067ZO&th=1
Something like that seems like what I need. I am sure that there is something like this on Campus, but I do not know who to talk to for it -- Megan would be my best guess.

Additionally I want to find some hardware to interface with. Ideally this would be the hardware that will be used on the cubesat like the IMU, UHF, S-band etc, but I am not sure what is actually available in regards to that. Any sensor or device that uses the same protocols should work for a proof of concept when developing the communication software.

Task List:

- Test serial communication to PI → Need adapter
- Get communication interface working for the various protocols (I2C, SPI, UART) → need hardware
- Better understand the control system. What each variable actually represents, etc.
- If the actual hardware is available, write the software to parse the data from the hardware to the appropriate variables
- I know there are motors/motor drivers available, so getting them working would also be good → could do this later this afternoon - or at least start
- Next week, I would like to have a meeting with Nick and just go over everything that I have been working on. Additionally he would likely be able to explain some of the variables and processes in the ADCS system.

I am going to install Raspbian on my PI and try and install VSCode. If I can do my development in the environment that the software will be deployed on it may be easier to do testing and ensure the relevant libraries are installed.

On second thought, I don't even really need to do the development on the RPI, I could just use the github repo to transfer the software onto the pi and then compile and run for testing. This could be done by just shelling to the PI. I will need to be posting everything to the github but it may work. I can also send the files directly to the PI without the need for uploading them to github first. I could make a small executable that just sends the current version to the pi whenever I want.

That was getting hard so Im going to go back to using git for now

I tried just running the compiler in the raspberry pi terminal to compile the software, however I am running into the issue that I first had where the g++ compiler is not finding the external files when compiling. I had fixed it before because vscode generated a file that the compiler would reference to tell it where to look. I no longer have that when I am just doing things in the command line. It may be easiest just to use a raspberry pi to develop and test on entirely.

I installed VSCode on my raspberry pi and cloned the github repo. My RPI4 is the base version and it is struggling to run the IDE. The PI that Nick said he had would be much better. I am also struggling to get the G++ compiler working on the raspberry pi. Specifically within VSCode. I am going to leave that for now because there is not much point trying to debug it when I have to keep waiting for the program to catch up.

I read through the GitHub linked to the main basecamp.

I found that not a whole lot was developed as far as original code goes, however there were libraries for the reaction wheel motors and the IMU that would be helpful for me to use when I go to interface with those devices.

I read through the TRR again just to see if anything about the ADCS became clearer after my other reading

I may try the SCADA report again. The first time I tried to read it it was hard to follow

Break 12:30

Return 2:00

I went to the University so I can attempt to find some sensors that I can try to interface with. Unfortunately I forgot my RPI so I will not be able to test directly, but if I can find some parts I can start trying to test tomorrow.

Earlier I had mentioned that I found a library in the GitHub to use the motor controllers for the reactions wheels.

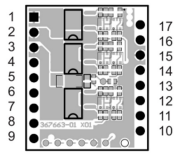
```
// Class constructor
Motor::Motor(PinName encoder_pin, PinName pwm_pin) : encoder{encoder_pin}, pwm(pwm_pin){
    encoder.rise(callback(this, &Motor::Pulse_Count));
    pwm.period_us(DRIVER_PWMPERIOD);
}
```

It appears that the library only takes two arguments when initializing the motor object, the encoder pin

and the pwm pin. I know I had previously looked into the data sheet on these motors, and something tells me that I thought there were more GPIO pins than this,

3 Pin assignment DEC Module 24/2

Top view



1 PWM
4 Digital pins
- Motor
- MCU
- Other

3.1 Pin assignment

Pin	Signal	Description
1	W1	Motor winding 1
2	W2	Motor winding 2
3	W3	Motor winding 3
4	+V_{DC}	Supply voltage 8...24 VDC
5	Gnd	Ground
6	V_{DC} Hall	+5 VDC output voltage
7	H1	Hall sensor 1
8	H2	Hall sensor 2
9	H3	Hall sensor 3

Pin	Signal	Description
17	Set value speed	Set value speed input (PWM)
16	Set current limit	Set current limit input
15	Gnd	Ground
14	Direction	Direction input (Digital)
13	Enable	Enable input (Digital)
12	DigIN2	Digital input 2 (Digital)
11	DigIN1	Digital input 1 (Digital)
10	Ready	Status indication output

Previously I had found that 4 GPIO pins would be required. These are for the direction, and enable pin (Which probably does not actually need to be GPIO but if the pins are available it may be useful)
And then two digital input pins. I do not remember what these digital input pins are, I may need to review the datasheet.

5 Functional Description of Inputs and Outputs

5.1 Inputs

5.1.1 Speed range and mode selection with «DigIN1» und «DigIN2»

The digital inputs «DigIN1» [11] and «DigIN2» [12] determine both, the operation mode (digital speed controller or digital speed actuator) and the speed range in speed set value mode.

		Motor type		
DigIN1	DigIN2	1 pole pair	4 pole pair	8 pole pair
0	0	Open loop speed control, 0...100 % PWM depending on the «Set value speed» input voltage		
1	0	500...5 000 rpm	125...1 250 rpm	62...625 rpm
0	1	500...20 000 rpm	125... 5 000 rpm	62...2 500 rpm
1	1	500...80 000 rpm	125...20 000 rpm	62...10 000 rpm

Please note

⇨ If the signal level of the digital inputs DigIN1 [11] and DigIN2 [12] are changed, the new levels are adopted by a disable-enable procedure.

Looks like the digitial input pins are used to set the speed range of the motor. Similar to the enable pin, this could likely be hard wired and not controlled in software, but if the pins are available it they could be used.

In addition to this, it appears that the motor driver interprets the hall sensor encoders and calibrates to the speed internally, there is no output on the driver for an encoder pin, so I do not know why the function would include a variable for the encoder pin…

I can put together my own library for controlling this motor driver, my main concern is that this is not the motor driver that is being used currently.

Here at the university, there are some motor drivers but they do not look exactly likes the ones in the attached data sheet.

There are no relevant numbers on these drivers to try to figure out what they are. The pins also have no labels. I cant find any other information about the motor drivers on basecamp…

I reached out to Jakob Bryant as I know he had worked with the motors previously. He may have some insight as to what is going on if he remembers.

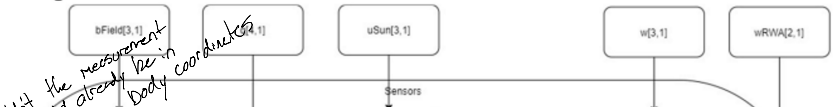
Reading the datasheet again, I do not believe that the dig pins need to be changed. I think the pwm, direction, and maybe enable are the only relevant parameters.

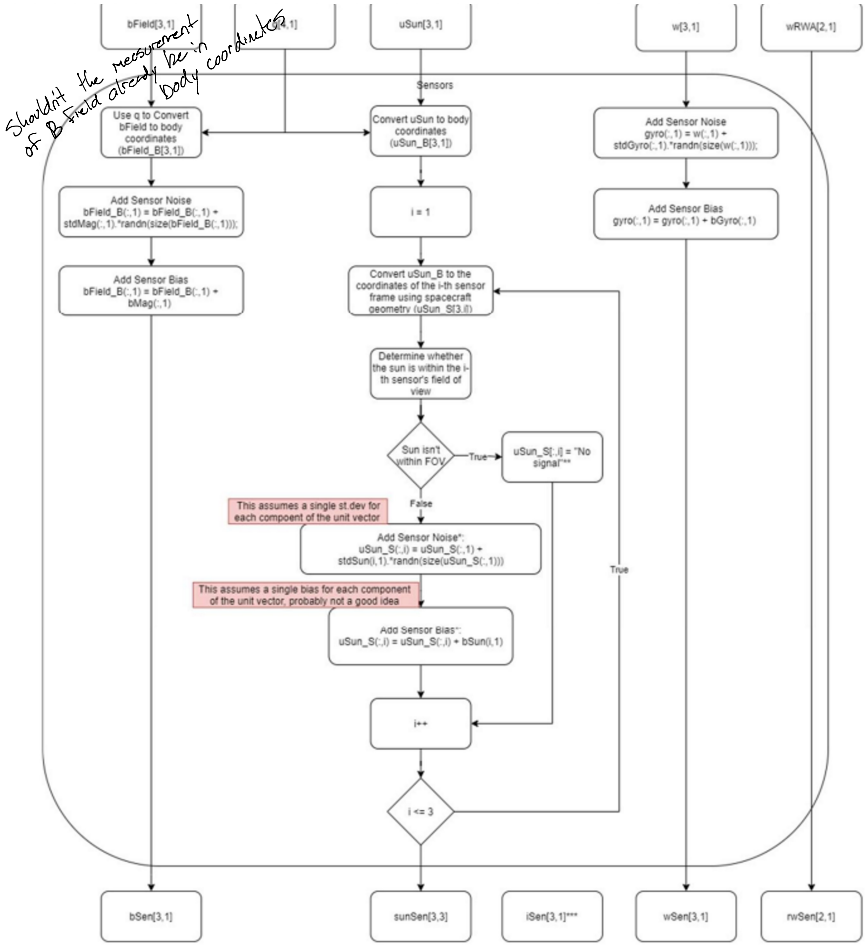
I created a library for the motor driver linked on the Basecamp and attempted to move the I2C code I had found into a library with a class. This way I can reuse all the code by creating a new instance of the obj for each slave

I have not tested either of there as I cant on my pc so they likely will not work.

I went back to the ADCS document cause I think something is starting to make more sense.

5.2.1.3 Processing





Should/if the measurement of B field already be in body coordinates

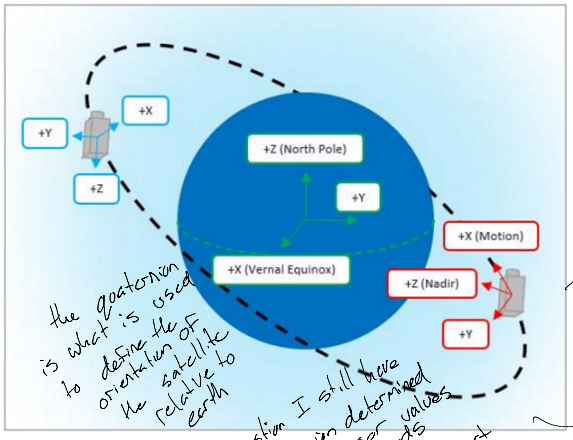


Figure 7-1: Earth centered inertial frame (green), spacecraft body frame (blue), and orbital reference frame (red).

is the orbital reference frame used? That seems relevant

the quaternion is what is used to define the orientation of the satellite relative to earth

Maybe that is how the orientation of the sat is determined —

the magnetic field of earth can be measured in body coordinates, and the direction of the field relative to earth is known — so the direction of the satellite relative to earth can be known

One question I still have is the quaternion determined by comparing sensor values to body frame records of is it used to convert the two reference frames

End 5:00pm

09/06/23

Time: 6h

June 9, 2023 8:17 AM

Start 8:15

I remembered to bring my RPI today. I am thinking that I could start testing some of the functions like the I2c and the motor. The motor I still do not know the pinout of the drivers that are here, but I should be able to see if I am interfacing with the pins correctly by using an LED.

The school wifi is being difficult. The UPEI_WiFI_Setup has an expired security tag so it just get warnings and cant actually redirect to anywhere.

The eduroam network is just greyed out and I cant connect to it.

I am trying to connect it to ethernet through my PC, but that isnt working great either.

https://raspberrypi.stackexchange.com/questions/58014/pi-3-cannot-connect-to-enterprise-wifi-using-gui

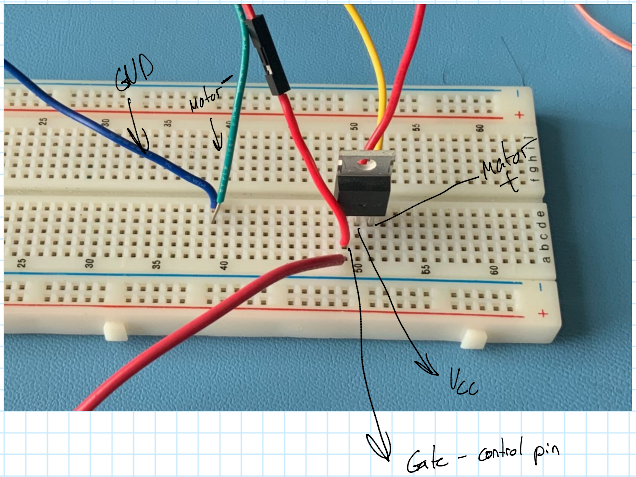
I got the EDUROAM installer downlaoded and ran the setup .py file. It said that it was sucessful but I still cant connect.

It asked permission to generate a wpa_supplicant.conf file for me but the file is just blank, it never actually filled in the network information

I finally got it connected. Idk who made this installer, but it is supposed to fill in the wpa_supplicant.conf file which holds all the credentials and connection information, but it cant. I thought it was a permissions issues cause you need sudo to edit the network file so I ran the installer script with sudo and it just crashed. Since it was just a .py file I found the section where it is supposed to write the information to the file and just printed it out to the terminal. I manually copy and pasted the information into the wpa_supplicant.conf file and the network connected.

That should not have been a 1.5h process…

I put together a circuit to drive a basic 2 wire motor I found lying around. I think I wired it correctly and when I check it with a multimeter the voltage going to the pi should not exceed 5V but when I change the transistor pin value, the voltage becomes negative for a short duration and I am concerend with damaging my PI. I might just use some LED's after all cause I am much less concerned about shorting something with them.



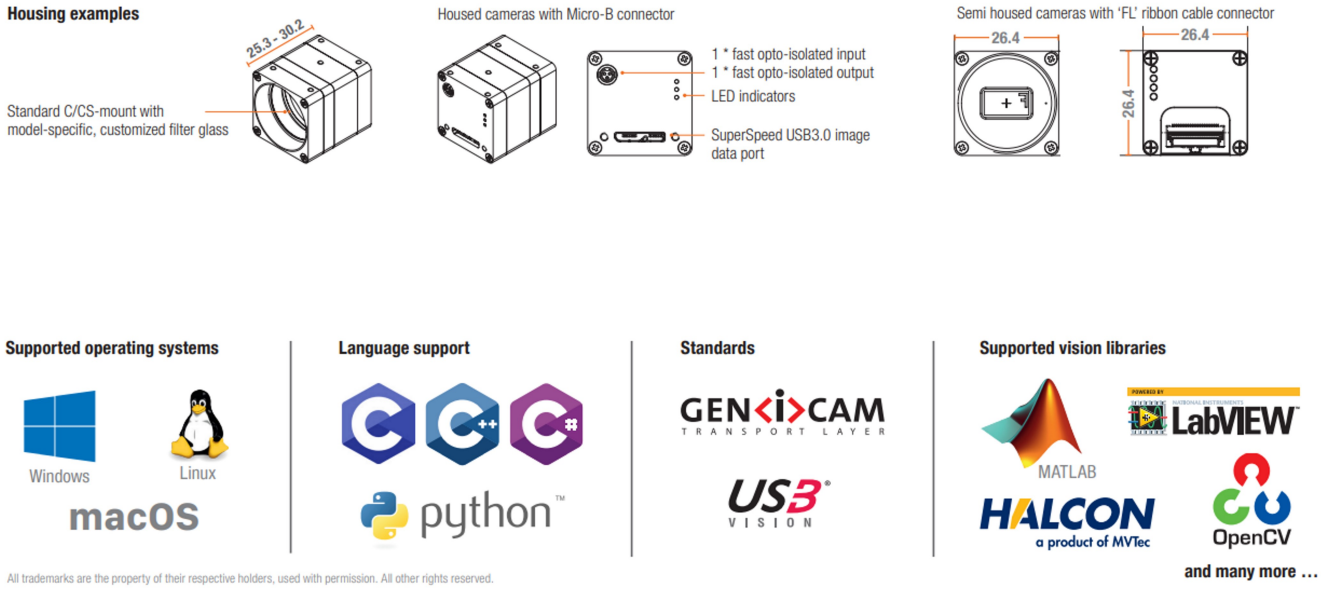
I switched to just using LED indication to display if the pinout is working. I had to install the wiringPi library even though it is supposed to come pre-installed

I installed the library and included a -lwiringPi command in the compiler I am using on the RPI however I am running into errors of the functions being undefined. The linker does not appear to be working properly.

I am able to interface with the GPIO pins using the terminal commands, it is just not finding the libraries when compiling my cpp files

45min lunch break

Grant stopped in and mentioned that a big thing that they want is to be able to start collecting images. I think for the rest of the day I am going to work on looking into interfacing the camera with the PI -- I may get someone to let me into the robotics lab to see if the camera Nick had me using during my MV project is still there.



This is from the camera that I was primarily looking at. It is supported by the linux OS and c++ which means I should be able to interface with it.

Right now, I do not have this camera, so maybe I should focus on the Lucam one that I was using previously just to get something working. Im sure some of the details will be transferable.

I downloaded the SDK for the Lucam cameras for ARMv8 (64bit) systems, which is what my Pi should be running. I opened the readme however the installation does not. It is giving me a command I am supposed to copy into the terminal and replace the placeholders for <revision> and <arch> with the ones that match my version and architecture. When I do that, I get an error that the file specific can not be found. There is no files in the lucam directory that share a similar format, so I think it is trying to pull something from online. It is using a tar command.

I changed to just using the file name that was in the download that didn't match the format at all and it seems to have worked. Now the next command does not work...

It keeps giving me an error that a version file is missing.

There is a version file in the folder, however it is not the version.h that the software is looking for.
There is a function in the folder that creates a "support package" that can be sent to lumenera. Im assuming that compiles all my files so they can see exactly what is running on my end and find the error.

I went back to the wiringPi setup and found I made a stupid mistake. I use -I (capital I) as the argument not -l (lower L) when including the wiringPi library in the compiler.

The script not compiles and runs however the GPIO pins are not actually being changed. There are issues being raised when I try to run the software it just isnt working

I can check that I am connected to the proper pins because I can control the IO pins in the terminal and the pin numbers are working for the LED's

Leave 3;00