07/08/23

August 7, 2023 8:12 AM

Start 8:15

On Friday, Nick was going to order my PCB. I also sent him the digikey list of components I would need. I should be able to move on from that for now.

To Do:

- Test all 5 PWM outputs programmed over I2C
  - Wait for dev board to be made
- Test all 12 ADC channels -- see if noise of channel 7 is removed with new PIC with no other wires attached
- Setup SPI on Raspberry Pi
  - Wait for dev board to have a test device (DAC)
- Figure out file transfer to the Pi over UART -- Or the SSH over UART that I saw last week
- Figure out how to use camera
  - Wait for camera to arrive

SSH over UART

I am seeing that there is a way to configure the RPI to use network protocols such as SSH over the UART pins

I have a lot of experience shelling into PC's and raspberry PI's and that interface is what I was hoping to get out of the UART. I was hoping that I would be able to just use tools like nano and vi in the command line to edit code files. But that does not really work. I know it works over SSH, so if I can get that setup, then I will be able to do what I had hoped.

https://gist.github.com/rudylacrete/a8c474aa6e8b221530a654d1f363f7c8

https://www.waveshare.com/wiki/Raspberry_Pi_PPP_dial-up

PPPD seems to be what I need. This website gives a breakdown of pretty much exactly how to do it, so hopefully that works. I want to make sure that I don't actually need internet to do this, so when I run this, I should disconnect the PI from the internet. I don't want this to be going through the local network.

How do I shell into this from my computer? Do I use Putty to open a serial port or do I use putty to SSH? And what would the IP be cause the devices arent on the same local network....

https://serverfault.com/questions/1075855/ssh-over-serial-on-windows

https://www.hanselman.com/blog/connect-to-a-device-over-serial-com-port-on-windows-10-with-wsl1-tty-devices-with-windows-terminal-and-minicom

It looks like it would be easier to do this on a UNIX system so that I have the same tools on each end and I can follow the same setup.

I am removing "console = serial0, 9600" from the cmdline.txt file on the RPI
\

I stole apc that had linux installed and tried to establish the pppd connection

I do not get any output from the comands on the pi, asside from an output listing the parameters.

on the PC I get an output where clearly it is trying to connect to the pi — I can also see that there is data on the UART wires comming from the pc side

The two devices do not establish a connection, and I am not sure why

I think it must be on the PI end, cause the pc is giving me an output and is attempting to do things over UART — the RPI is not

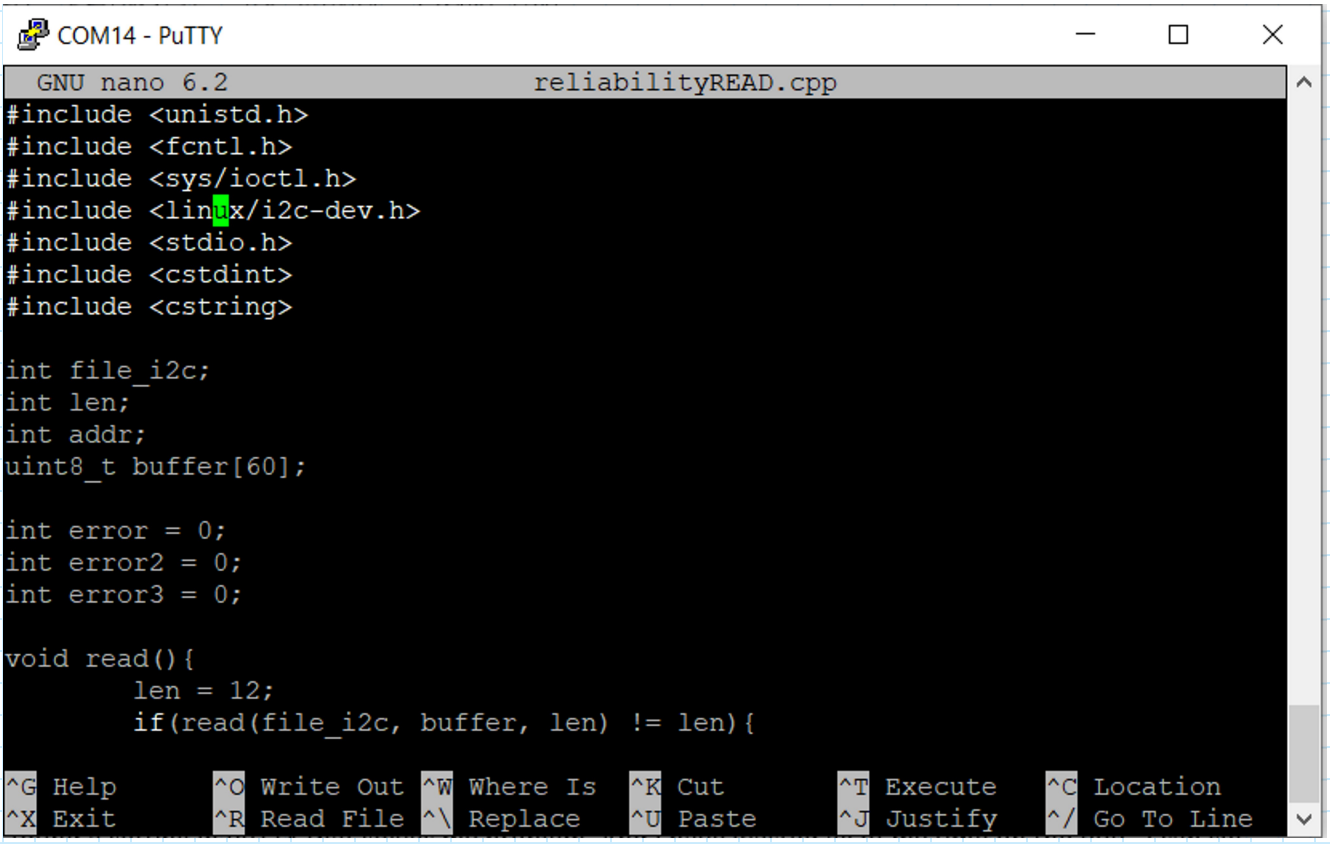
I found that the RPI does not appear to be connecting the serial port porperly with the command. I can run the command ps -ef and it tells be the tasks and what TTY they are holding. There is a ? For the TTY on the thread running the pppd command.

On the PC side, when I can the RPI command, it tied the tty port to that command and could be seen using the command

Every time I run the command it creates a new task…

I found a setting in PuTTY that solves my problem. And I have wasted all of my time on the PPP. I can set it to give Linux functions so the arrow keys go through as the arrow command as opposed to [[A.

Now I am able to log into the PI and edit the files with nano



And I can use the Ctrl+X commands to save and leave the file.

So currently the RPI is connected using the USB-UART converter, but I don't think it should make any difference when that is swapped to the UHF transceiver. As long as the RPI is receiving UART, I don't really see what would need to change, only the BAUD rate, which can be done by modifying the /boot/firmware/cmdline.txt file

https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/el.2020.1417

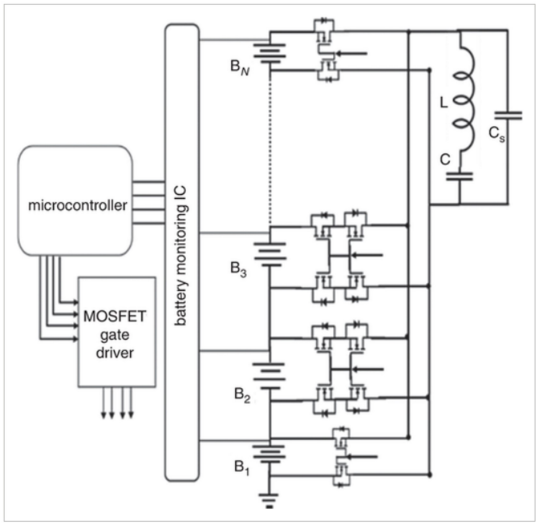
Leave: 3:30

08/08/23

Tuesday, August 8, 2023 8:06 AM

Start 8:00

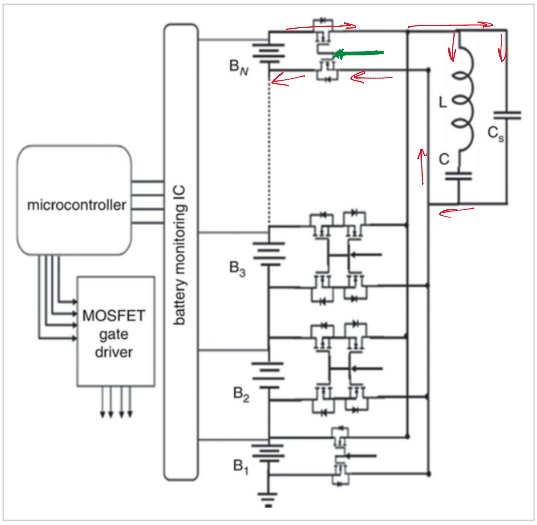
https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/el.2020.1417



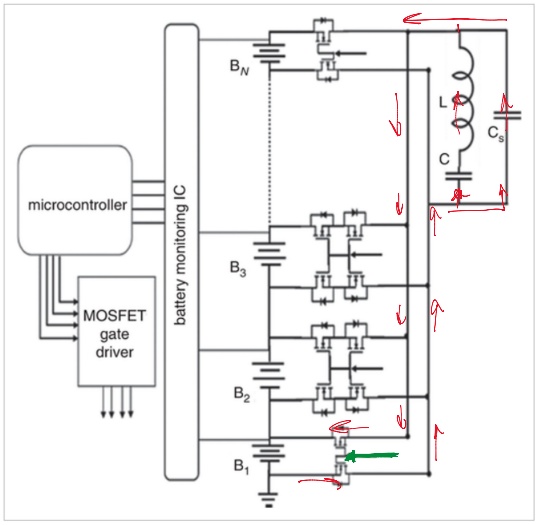
This seems like the easiest battery balancing method, the high cell is connected to the LC circuit and discharges into it then the low cell is connected to the LC circuit and the cell is charged from the LC

So if B1>B2>B3>B4

Charging



Discharge



Again I am confused how the PWM is used. I get why you want to cycle a MOSFET with PWM. but how do you control which one? is it just changing the output pin of the Microcontroller? that could actually work

Assume B1 – P1 (Pin 1)
B2 – P2
B3 – P3
B4 - P4

Pseudo code

generate complimentary pwm
Repeat:
- measure all 4 cell voltages
- detect highest and lowest
- connect PWM to pin controlling highest cells MOSFETS
- connect complimentary PWM to low cells MOSFETs

This should be easy to impliment, and should be robust.

I am not sure how to determine what L C and Cs to use.

Two-cell voltage balancing process analysis

Based on the circuit analysis proposed in [4], the resonant analysis is obtained below. All associate MOSFET switches in a cell are considered a loop and the loop resistance is R_{loop} and voltage drop in the MOSFET switch is V_{MS_D}. The capacitor voltage across in the energy carrier ($C + C_S$) is $V_{Capacitor}$. When the highest capacitive cell (cell voltage is $V_{C\_highest}$) associate switch is turn on the energy starts to transfer and the voltage across the L will be equal

$$L\frac{\mathrm{d}i}{\mathrm{d}t} = V_{C\_highest} - V_{Capacitor} \tag{1}$$

$V_{Capacitor}$ starts to charge and store the energy and the amplitude of the resonant current is

$$I_m = \frac{V_{C\_highest} - V_{Capacitor}}{\omega L} = V_{C\_highest} - V_{Capacitor}\sqrt{\frac{C+C_S}{L}} \tag{2}$$

After the fast phase gate pulse, $V_{Capacitor}$ reaches the maximum voltage

$$V_{Capacitor\_Max} = V_{C\_highest} - V_{Capacitor} \tag{3}$$

In the discharging time, when the lowest capacitive cell (cell voltage is V_{C_lowest}) associate switch is turn on the amplitude of the resonant current is

$$I_m = \frac{V_{Capacitor} - V_{C\_lowest}}{\omega L} = V_{Capacitor\_Max} - V_{C\_lowest}\sqrt{\frac{C+C_S}{L}} \tag{4}$$

In the second phase-gate pulse, $V_{Capacitor}$ reaches the minimum voltage

$$V_{Capacitor\_Min} = V_{Capacitor\_Max} - V_{C\_lowest} \tag{5}$$

During the balancing time, energy carrier charging and discharging current is

$$i_{charge} = \frac{V_{C\_highest} - (V_{MS\_D} + V_{Capacitor})}{\omega L} e^{-\sigma t}\sin\omega t \tag{6}$$

$$i_{discharge} = \frac{V_{Capacitor} - (V_{MS\_D} + V_{C\_lowest})}{\omega L} e^{-\sigma t}\sin\omega t \tag{7}$$

Where damped oscillation, $\sigma = R_{loop}/l$ and angular frequency, $\omega = \sqrt{1/(L(C+C_s))}$

The flowing current of the balancing circuit is estimated by the equation

$$i = \frac{V_{C\_highest} - 2V_{MS_D} - V_{C\_lowest}}{\omega L} e^{-\sigma t}\sin\omega t \tag{8}$$

Energy transfer efficiency can be calculated by this equation

$$\eta = \frac{V_{C\_lowest} \times i}{V_{C_{highest}} \times i} \times 100\% \tag{9}$$

This circuit worked like a damped oscillation circuit and energy transfer depends on the amplitude oscillation time. As the parasitic resistance remains low then maximum efficiency is possible.

How do I determine what the L, C and Cs should be. I guess I could define the current I want based on the max in the charging curve.

1625mA is the rated max current in the charging cycle. The maximum voltage should be 4.2V, and the minimum would techincally be 0V. But I the cell should never reach 0V. I believe that we wanted to keep them above 80%, or at least the low power mode is actives at 80% capacity.

So the minimum would be 3.6V (nominal voltage) * 0.8
3.6*0.8=2.88V

Vc_high = 4.2V
Vc_low = 2.8V
Vmsd = I * 4.9 https://www.infineon.com/dgdl/irfb7545pbf.pdf?fileId=5546d462533600a401535619e1ca1e66

I don't know what omega is cause the equation is cut off
Also I don't know how I would really calculate the loop resistance.

I am not sure how I can calculate this

The damping value is critical to determining the max current and the time. Which are the two things that I care about, so I cant solve this at all without the rest of that equation.

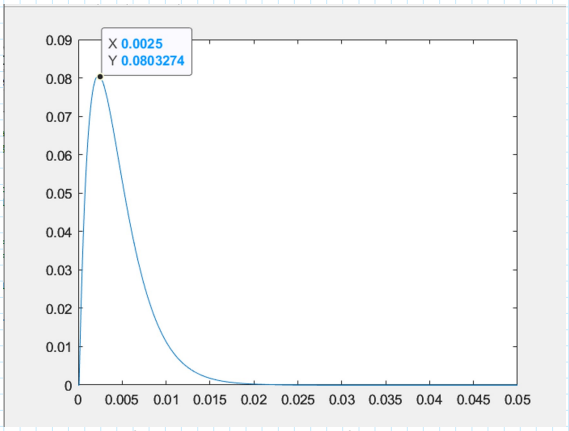
Where damped oscillation, $\sigma = R_{loop}/l$ and angular frequency, $\omega = \sqrt{1/(L(C+C_S))}$

It looks like its just a lowercase L, but that is not defined anywhere...

If I assume that is supposed to be inductance, and I use the values from the paper, I get something like this.

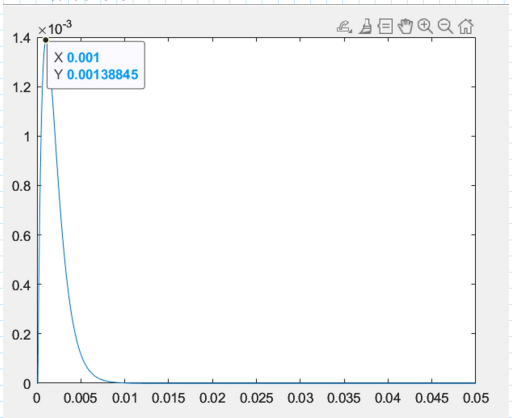
Table 1: List of the components

Parts		Value
switches	single MOSFET	IRF450A nMOSFET
	bidirectional MOSFET	2N7000 nMOSFET
resonant tank	inductor	100 µH
	capacitor	220 µH
	single capacitor (C_S)	2200 µH
gate driver	optocoupler	817c
	logic gate	SN7404
monitoring IC	different amplifier	LM 350 Opamp
	resistance	100 Ω
microcontroller	Arduino Uno	atmega328p
battery	lead-acid	12 V 4.5 Ah



I get a peak current of only ~80mA and it takes ~0.02s to fully charge the capacitor.

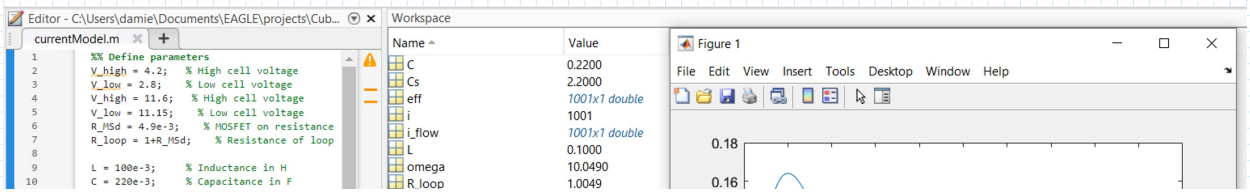
NVM I found an error.

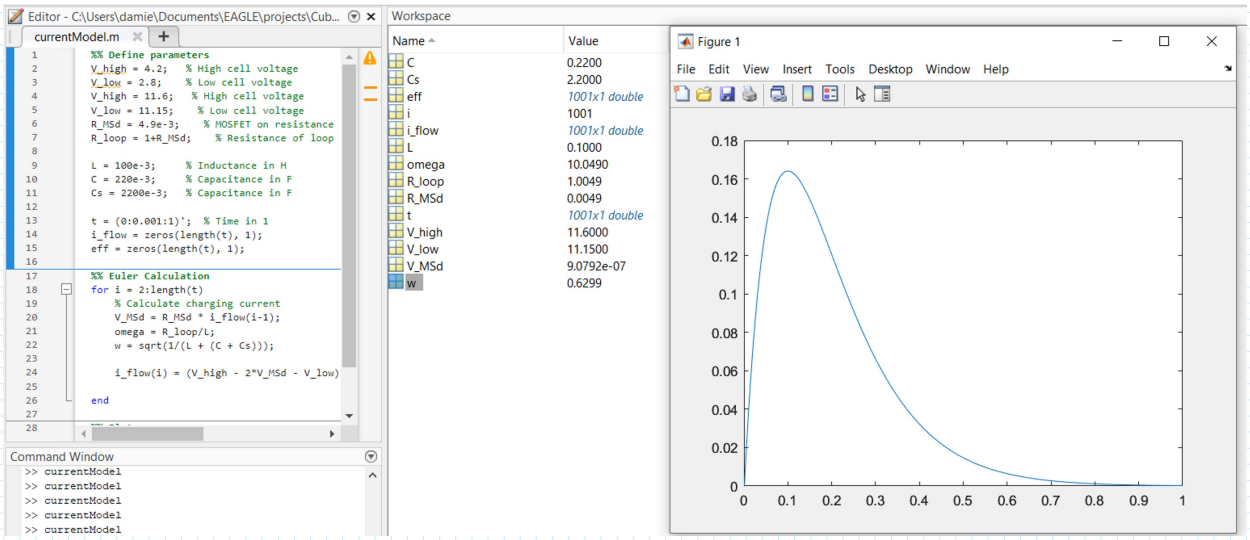


The current is way less. This seems like nothing. But it does also take hours for this happen.

If we say the voltage changed from 11.6 to 11.35V, lets say that that is a
11.6/12=0.9667
11.35/12=0.9458
0.9667-0.9458=0.0209% change in capacity
That should not take hours...
I think that my assumption of capacity is wrong, I think the voltage is only related linearly to capacity in the operating range, then there is a sudden drop below a certain voltage.

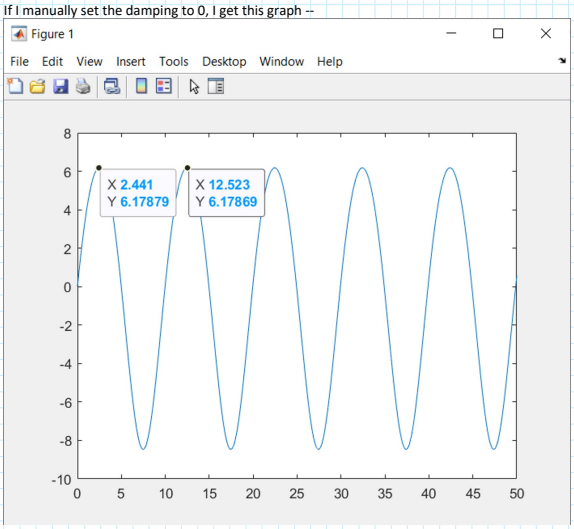
The anti-series bidirectional MOSFET switches are used to protect the negative voltage in the MOSFET intrinsic body diode because the low-voltage MOSFET cannot protect the body diode conduction. All associate switches of each cell are operated at the same gate pulse and turned on or off by near-zero current switches for minimising the switching loss. In the proposed balancing circuit, the switching gate pulse is 50% of the duty cycle and the switching frequency is equal to the resonant frequency so that all MOSFETs switched are achieved soft switching. Also, this soft-switching minimised the impedance and allow flowing the maximum current and reduced the voltage balancing time. When the cell voltage variation occurred in the EESS string between two cells then associate switches of the highest energy capacitive cell are turn on in the fast phase MOSFET gate pulse and energy temporally store in single switches-capacitor and series LC tank. In the second phase MOSFET gate pulse, all associate switches of the lowest energy capacitive cell are turned on and the stored energy is transferred. This process is repeated until the voltage balancing achieves between two cells.





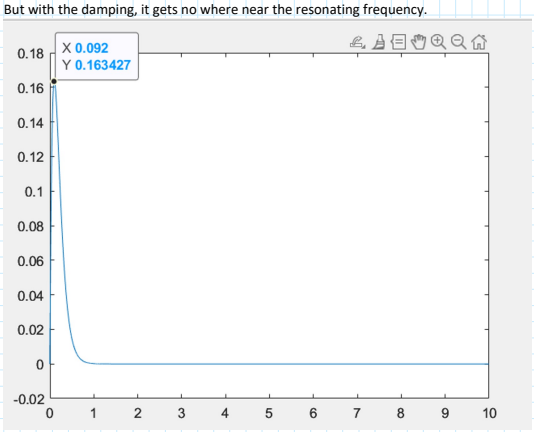
From what I can tell, their circuit had a resonance frequency of 0.6299Hz, which would be a period of 1/0.6299=1.5876, this doesn't really match the graph.

If I manually set the damping to 0, I get this graph --



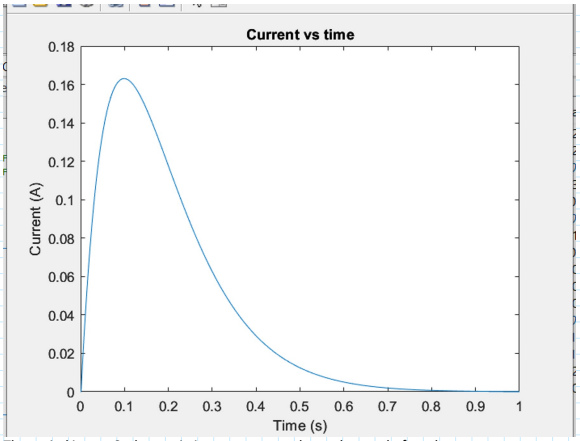
The frequency is the same, as it is not dependent on omega. I get a period of ~10s which would be a frequency of 0.1, which is not close to 0.63
Angular frequency is 2pi*freq, not just freq.
But that means that freq < angular freq...
This adds up.
Freq is now 0.1 which is equal to a period of 10s.

But with the damping, it gets no where near the resonating frequency.



The current is effectively 0 by 1s, as opposed to 10. This should mean that the voltage between the cell and the capacitor is ~0V, which is what they want for the soft switching.
So they could really use a frequency

NOPE
Found another error.



The period is now 3s, but again it appears to reach steady state before that.

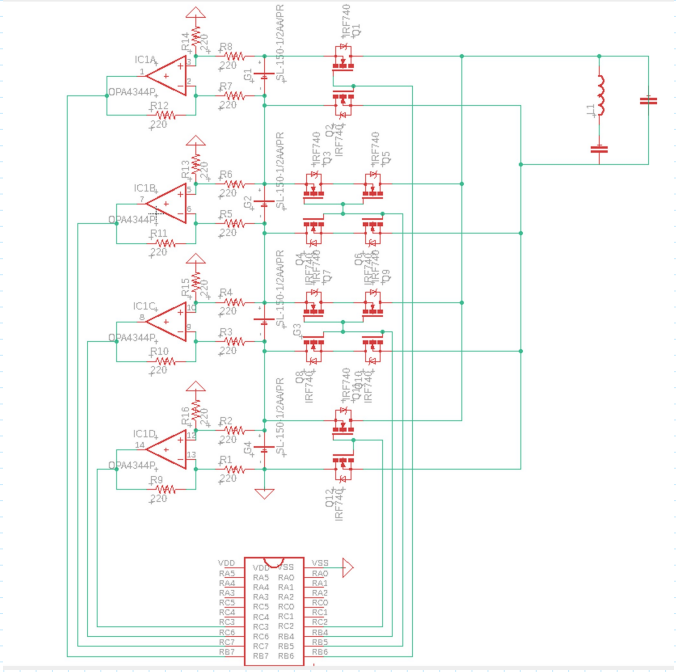
This appears to be really slow switching, I know the other paper used a frequency of like 28kHz, this is around 1 Hz.

For the PCB, I think I would just need to use the previous design but swap the management chip for a PIC connected to this circuit

It sounds like Nick might want me to create a test PCB for the monitoring / balancing circuit

For that I would need
- PIC
- 4 Differential amps
- The cell balancing circuit
- Current measurement circuit

https://www.analog.com/media/en/technical-documentation/data-sheets/OP400.pdf



This should be the minimum circuit I need for testing the cell balancing.

Leave 3:30

09/08/23

Wednesday, August 9, 2023 8:38 AM

Start 8:30

To-Do	In Progress	On Hold	Complete
Find library and set up code to use SPI on the Raspberry pi	Create PIC code for testing the cell balancing circuit	Test cell balancing circuit and algorithm	UART on RPI to allow terminal access
Document progress I have made this summer		Perform reliability testing on ADC PIC	Program PIC to receive commands from RPI and generate PWM signals
		Perform reliability testing on PWM PIC	Create RPI library to send PWM commands to PIC
		Build peripheral development board	Program PIC to read 12 analog channels and send the data back the the RPI over I2C
		Take images in C++ code using the Ximea camera on RPI	Create RPI library to request ADC readings from PIC and read them from the I2C bus
			Create library to read the IMU
			Program PIC to set up UART
			Design development board for easier and more reliable testing

This is all I can think of for things to do. Most of the 'On Hold' Items are due to the fact that I do not have the hardware required to complete the task.

Cell Balancing Software

C:/Users/damie/Documents/GitHub/SpudNik-1_2023/PIC18/CellBalancingTest.X/main.c

```
/*
 * File:   main.c
 * Author: damie
 *
 * Created on August 8, 2023, 3:08 PM
 */


#include <xc.h>

uint8_t channel_map[4] = {
    0b010011,   //RC3
    0b010110,   //RC6
    0b010111,   //RC7
    0b001111,   //RB7
};

void ADCsetup(){
    /*Configure Pin*/
    TRISB = TRISB | 0b10000000; //Set RB7 to inputs
    TRISC = TRISC | 0b01001000; //Set RC3, RC6 to inputs
    ANSELB = ANSELB | 0b10000000; //Set RB7 to analog inputs
    ANSELC = ANSELC | 0b01001000; //Set RC3, RC6 to analog inputs

    /*Configure ADC*/
    ADREFbits.NREF = 0;     /*Set negative reference
                             * 0: Vref- is connected to AVss
                             * 1: Vref- is connected to external Vref-*/

    ADREFbits.PREF = 0b11;    /*Set positive reference
                             * 11: Vref+ is connected to internal FVR module
                             * 10: Vref+ is connected to external Vref+
                             * 01: Reserved
                             * 00: Vref+ is connected to VDD*/

    /*IF 0b11 was selected above, configure the FVR module*/
    FVRCONbits.ADFVR = 0b11;    /*Configure the output voltage of the FVR
                                 * 11: 4x = 4.096V
                                 * 10: 2x = 2.048V
                                 * 01: 1x = 1.024V
                                 * 00: OFF*/
    FVRCONbits.EN = 1;  //Enable FVR module

    ADCON0bits.FM = 1;      //Right justify
    ADCON0bits.CS = 1;      //Set clock to ADCRC Clock (~600kHz)
    ADACQ = 32;             //Set acquisition time
}

uint16_t ADCread(int channel){
```

1.1 of 4 2023.08.09 09:11:56

ADC pins for the 4 cells

Forgot C7

the charging voltage is 4.2V while the nominal is 3.6V — will I need to measure >4.096V?

C:/Users/damie/Documents/GitHub/SpudNik-1_2023/PIC18/CellBalancingTest.X/main.c

```
    /*Set the ADC channel to ground and read to clear charge*/
    ADPCH = 0b111011;
    ADCON0bits.GO = 1;  //Start acquisition
    while(ADCON0bits.GO); //Wait until done

    /*Set channel to selected pin and measure ADC*/
    ADPCH = channel_map[channel];   //Set ADC to pin
    ADCON0bits.ON = 1;      //Turn on ADC
    ADCON0bits.GO = 1;  //Start acquisition
    while(ADCON0bits.GO); //Wait until done
    uint8_t resultHigh = ADRESH;    //Read high register
    uint8_t resultLow = ADRESL;     //Read low register

    /*Convert 2 8bit to 1 16bit*/
    uint16_t result = resultHigh;
    result = result << 8;
    result = result | resultLow;

    return result;
}

void PWMsetup(){
    //Set pins to output
    TRISCbits.TRISC2 = 0;
    TRISBbits.TRISB4 = 0;
    TRISBbits.TRISB5 = 0;
    TRISBbits.TRISB6 = 0;

    RB6PPS = 0x0A;  //Configure to PWM1S1P1_OUT
    RC7PPS = 0x0B;  //Configure to PWM1S1P2_OUT

    PWM1ERS = 0b0000;   //Sets the external reset source to disabled
    PWM1CLK = 0b0010;   //Sets the clock source for PWM - Set to Fosc

    /*Set PWM Period*/
    PWM1PR = 0xFFFF;    //Number of clock periods in PWM period (Effectively the resolution)
    PWM1CPRE = 0x00;    //Clock pre-scaler (n+1)
    PWM1GIE = 0x00;     //Interrupt register -- Disable/Enable interrupts
    PWM1CONbits.LD = 1; //Reload the PR, P1, and P2 registers on the next cycle


    /*Configure Slices (Slice is one of the dual outputs*/
    PWM1S1CFGbits.POL1 = 0; //Set polarity of output 1, 0 = high true
    PWM1S1CFGbits.POL2 = 0; //Set polarity of output 2, 0 = high true

    PWM1S1CFGbits.PPEN = 1; //Disable Push-Pull to alternate outputs
    PWM1S1CFGbits.MODE = 0b0001; //Right align

    PWM1S1P1 = 0xFFFF;  //Set duty cycle
    PWM1S1P2 = 0xFFFF;  //Set duty cycle
```

Remove

Might want to change depending on the desired frequency defined by the RLC circuit resonance

Enable push pull

100% duty cycle w/ push pull should give me ~50% duty on each with a small delay to ensure I don't short

C:/Users/damie/Documents/GitHub/SpudNik-1_2023/PIC18/CellBalancingTest.X/main.c

```
    PWM1CONbits.EN = 1; //Enable PWM module
}

void main(void) {
    uint16_t Vcell[4];
    uint16_t max = 0;
    uint16_t min = 0xFFFF;
    uint8_t index_max = 0;
    uint8_t index_min = 0;

    for(int i = 0; i < 4; i++){
        uint16_t reading = ADCread(i);
        Vcell[i] = reading;

        if(reading > max){
            max = reading;
            index_max = i;
        }

        if(reading < min){
            min = reading;
            index_min = i;
        }
    }

    switch(index_max){
        case 0:
            RC2PPS = 0x0A;  //Configure to PWM1S1P1_OUT
            break;
        case 1:
            RB4PPS = 0x0A;  //Configure to PWM1S1P1_OUT
            break;
        case 2:
            RB5PPS = 0x0A;  //Configure to PWM1S1P1_OUT
            break;
        case 3:
            RB6PPS = 0x0A;  //Configure to PWM1S1P1_OUT
            break;
    }
    switch(index_min){
        case 0:
            RC2PPS = 0x0B;  //Configure to PWM1S1P2_OUT
            break;
        case 1:
            RB4PPS = 0x0B;  //Configure to PWM1S1P2_OUT
            break;
        case 2:
            RB5PPS = 0x0B;  //Configure to PWM1S1P2_OUT
            break;
        case 3:
            RB6PPS = 0x0B;  //Configure to PWM1S1P2_OUT
```

I forgot the PWMsetup and ADCsetup functions

=> store voltages

define variables to determine high/low cell

read 4 cells, track which is high and which is low

set PWM1 to high cell MOSFETs

set PWM1 compliment to low cell MOSFETs

C:/Users/damie/Documents/GitHub/SpudNik-1_2023/PIC18/CellBalancingTest.X/main.c

```
            break;
    }

    float Vt = (Vcell[0] + Vcell[1] + Vcell[2] + Vcell[3]);   -> Battery Voltage

    return;
}
```

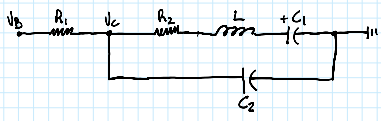

I tested and I can just switch the output channel of the PWM without disabling it. I also found that I can attach multiple pins to the output, so I need to clear the output of the PWM.

```
RB5PPS = 0x0A;
RB6PPS = 0x29;
__delay_ms(500);
RB6PPS = 0x0A;
RB5PPS = 0x29;
__delay_ms(500);
```

I can clear by setting the PPS to >0x28 as that is the last defined output, the rest don't correspond to a real PPS output.

I also tested the complimentary signals, I can see that there is an extremely small delay between switching to make sure that there is no possibility of shorting.

Checking Nick's Calculation



Generalized Impedance

Generalized Impedance

$V = IR$

$Z_{R_2} = R_2$

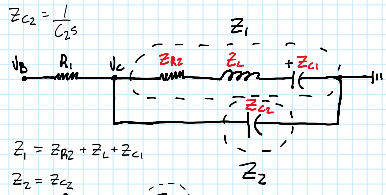
$V = L\frac{dI}{dt} = L \cdot I \cdot \frac{d}{dt} = L \cdot I \cdot s = Z_L \cdot I \quad Z_L = Ls$

$Z_L = Ls$

$I = C\frac{dV}{dt} = C \cdot V \cdot \frac{d}{dt} = C \cdot V \cdot s \qquad \frac{I}{V} = \frac{Cs}{I} \qquad V = I \cdot \frac{1}{Cs}$

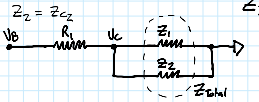
$Z_{C_1} = \frac{1}{C_1 s}$

$Z_{C_2} = \frac{1}{C_2 s}$



$Z_1 = Z_{R_2} + Z_L + Z_{C_1}$

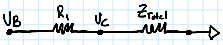
$Z_2 = Z_{C_2}$



$$Z_{Total} = \frac{Z_1 Z_2}{Z_1 + Z_2} = \frac{Z_1 \cdot \frac{1}{C_2 s}}{Z_1 + \frac{1}{C_2 s}} \cdot \frac{C_2 s}{C_2 s} = \frac{Z_1}{Z_1 C_2 s + 1}$$

$$Z_{Total} = \frac{Z_1}{Z_1 C_2 s + 1} = \frac{R_2 + Ls + \frac{1}{C_1 s}}{\left(R_2 + Ls + \frac{1}{C_1 s}\right) C_2 s + 1} = \frac{\left(\frac{R_2 C_1 s + LC_1 s^2 + 1}{C_1 s}\right)}{\left(\frac{R_2 C_1 s + LC_1 s^2 + 1}{C_1 s}\right) C_2 s + 1} \cdot \frac{C_1 s}{C_1 s} = \frac{LC_1 s^2 + R_2 C_1 s + 1}{\left(LC_1 s^2 + R_2 C_1 s + 1\right) C_2 s + C_1 s}$$

$$Z_{Total} = \frac{LC_1 s^2 + R_2 C_1 s + 1}{LC_1 C_2 s^3 + R_2 C_1 C_2 s^2 + C_2 s + C_1 s}$$



$V_C = x V_B \quad x = ?$

$$V_C = \frac{Z_{Total}}{R_1 + Z_{Total}} V_B$$

Transfer Function

$$\frac{V_C}{V_B} = \frac{Z_{Total}}{R_1 + Z_{Total}}$$

$$\frac{V_C}{V_B} = \frac{\frac{LC_1 s^2 + R_2 C_1 s + 1}{LC_1 C_2 s^3 + R_2 C_1 C_2 s^2 + C_2 s + C_1 s}}{\frac{LC_1 s^2 + R_2 C_1 s + 1}{LC_1 C_2 s^3 + R_2 C_1 C_2 s^2 + C_2 s + C_1 s} + R_1} \cdot \frac{LC_1 C_2 s^3 + R_1 C_1 C_2 s^2 + C_2 s + C_1 s}{LC_1 C_2 s^3 + R_1 C_1 C_2 s^2 + C_2 s + C_1 s}$$

$$\frac{V_C}{V_B} = \frac{LC_1 s^2 + R_2 C_1 s + 1}{LC_1 s^2 + R_2 C_1 s + 1 + R_1 LC_1 C_2 s^3 + R_1 R_2 C_1 C_2 s^2 + R_1 C_2 s + R_1 C_1 s}$$

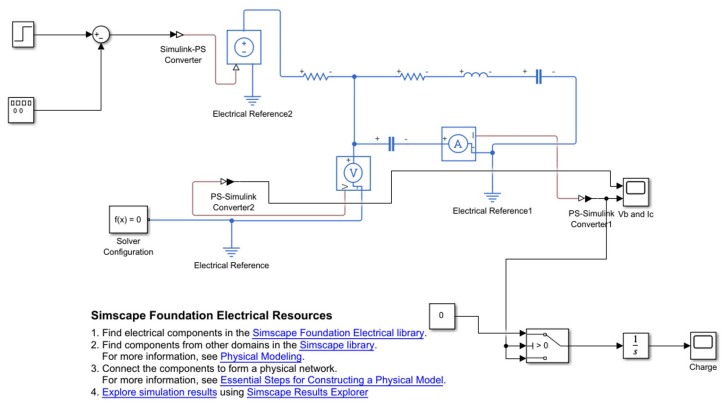
$$\frac{V_C}{V_B} = \frac{LC_1 s^2 + R_2 C_1 s + 1}{R_1 LC_1 C_2 s^3 + (LC_1 + R_1 R_2 C_1 C_2) s^2 + (R_2 C_1 + R_1 C_2 + R_1 C_1) s + 1} \checkmark$$

SimScape

I created a Simscape model of the circuit to see if I would get the same results as Nick Transfer function based Simulink model

I created the model so that all the voltage and current was pulled directly from the circuit model, instead of calculating it in Simulink blocks

I got the exact same results as the transfer function model, meaning both should be correct.



Now I need to change parameters and see how the C_1, C_2, L, R_1 and R_2 effect the circuit

The main parameters I care about are:

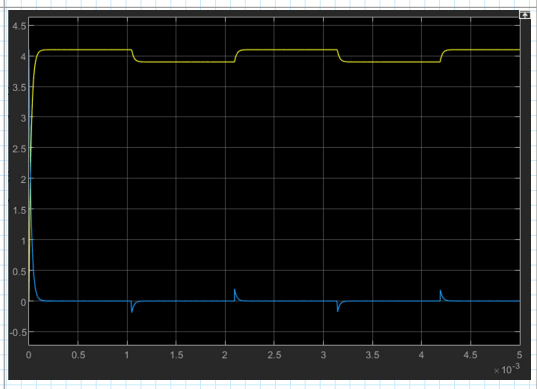
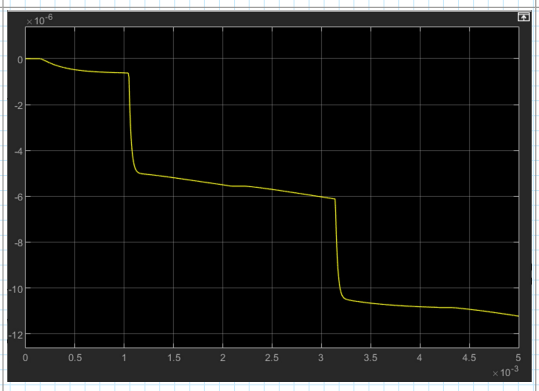
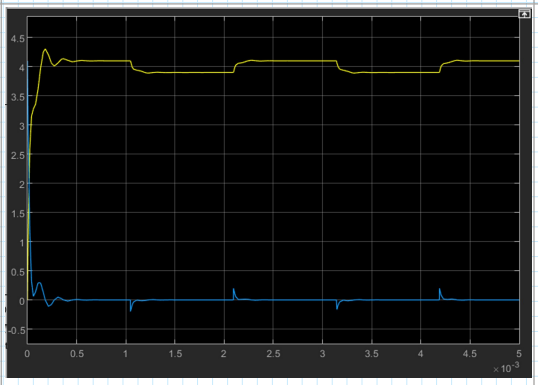
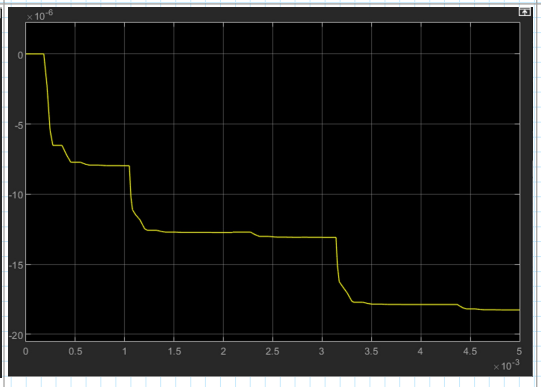
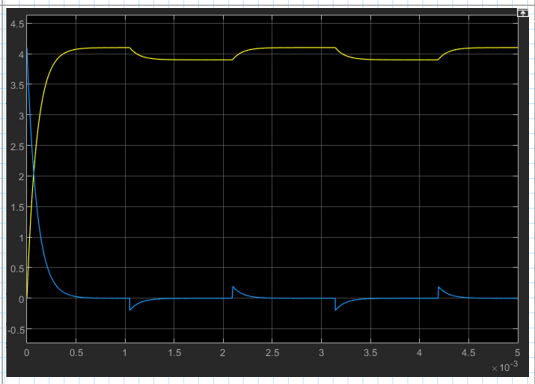
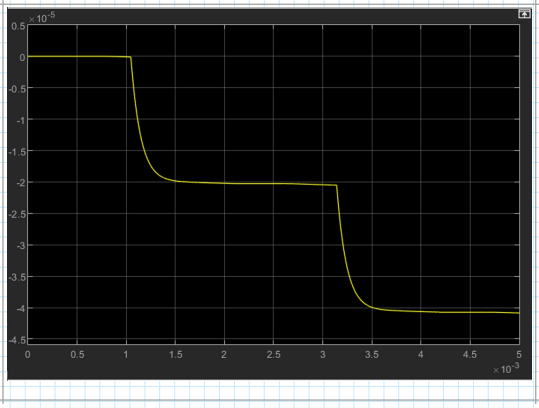
Now I need to change parameters and see how the C_1, C_2, L, R_1 and R_2 affect the circuit

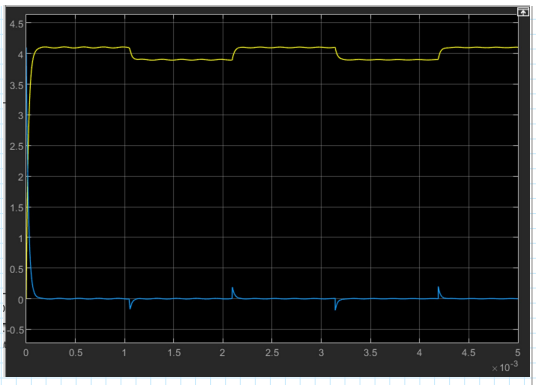
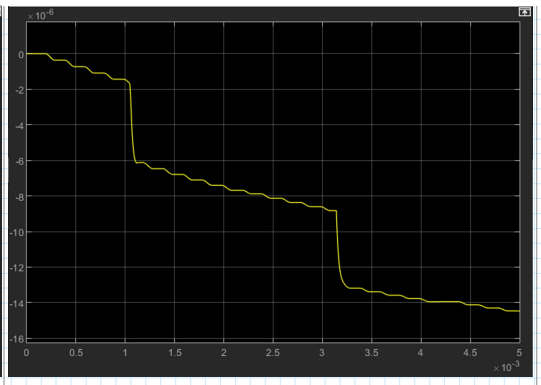
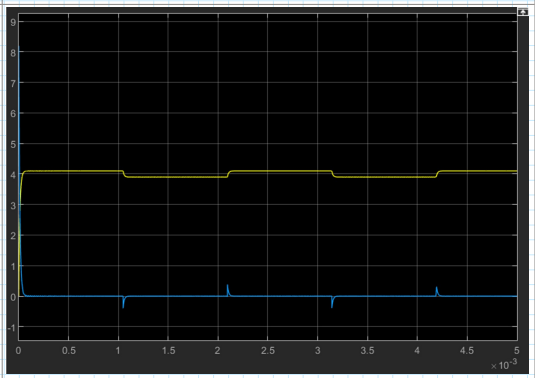
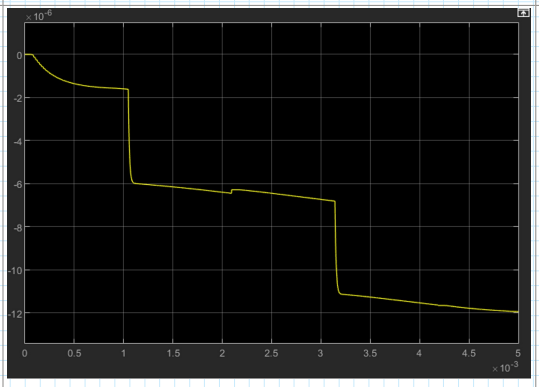
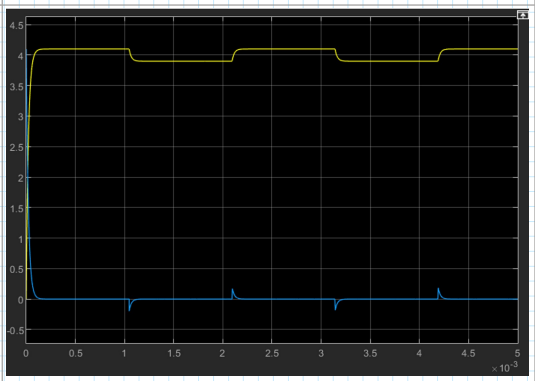
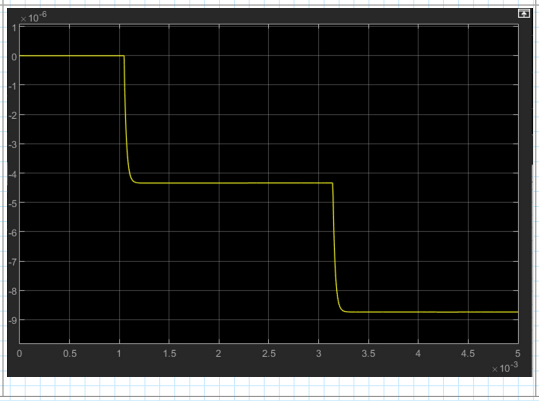
The main parameters I care about are:
- Current
- Frequency
- Rate of charge transfer.

Once I select component(s), I need to determine the switching frequency to maximize charge transfer. I can do this visually from the simscape model. I just want the circuit to switch right as steady state is reached

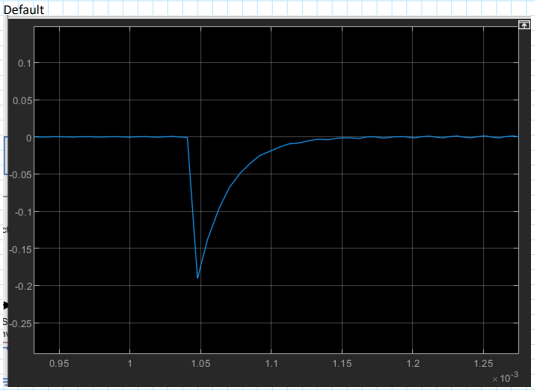
Default Parameters:

C1 = 1e-7
C2 = 2.2e-5
L = 1e-4
R1 = 1
R2 = 1

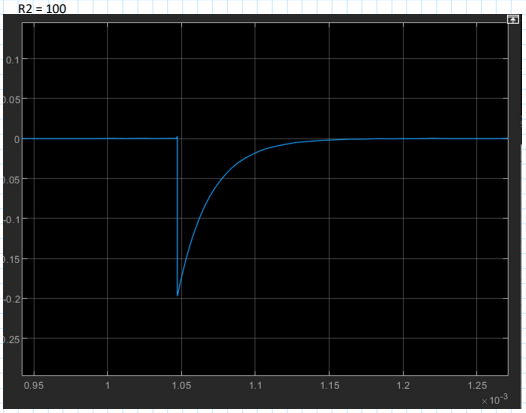
Parameter	Vc and Ic	Charge	Effects
Default			N/A
C1 = 1e-5			Increase C1 causes oscillations and increases the rate of charge transfer slightly
C2 = 1e-4			Increasing C2 increases the charging time for the capacitor and increases the charge transfer rate significantly

L = 1e-2			Increasing the inductance causes high frequency oscillations and slightly increases charge transfer rate
R1 = 0.5			Decreasing R1 causes the starting current to be significantly higher and reducing the charging time of the capacitor voltage
R2 = 100			Increasing R2 causes much faster response on the Charge transfer plots

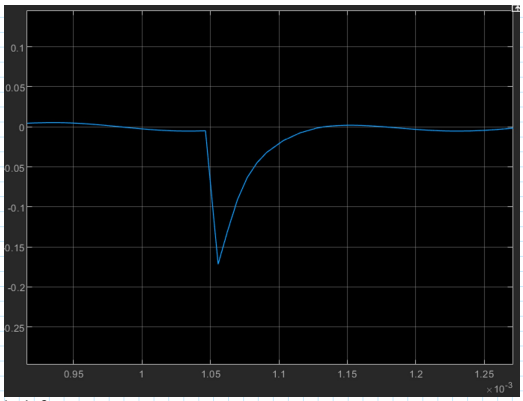
Default



R2 = 100



The current change when the capacitor is connected is linearly proportional to the R2 value it appears



L = 1e-2

Increasing inductance causes lower current and slows the rate of current change

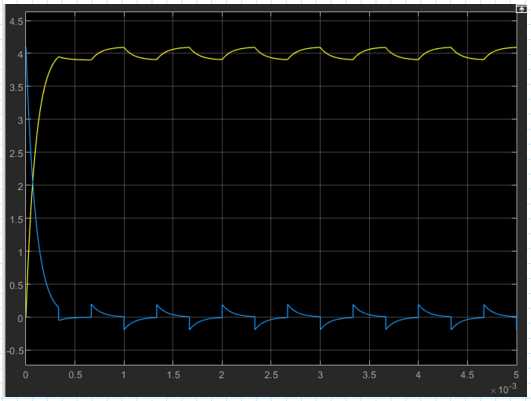
I think we should use the 100uF capacitor me and Nick were looking at. The inductor and other capacitor are not too significant.

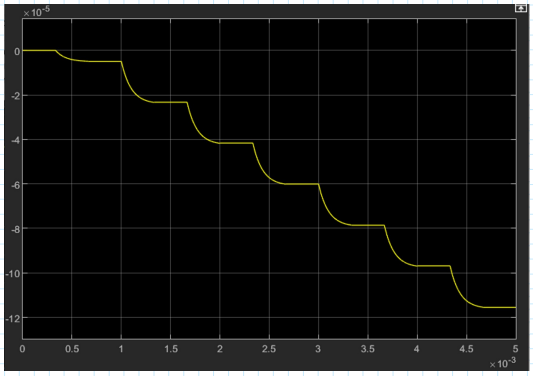
With the 100uF capacitor, the charge graph appears to only take 0.5ms to transition. So I want a frequency of 1/0.5e-3 in order to maximize the charge rate

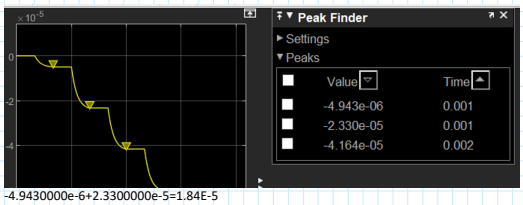
Which is a frequency of 2kHz

Angular frequency is the matlab parameter, which is 2pi*f

4*pi is the angular frequency I need to input
Nick Recommended using a slightly lower frequency to allow MOSFET switching time. I found that 1.5kHz seems to be a good looking value.







-4.9430000e-6+2.3300000e-5=1.84E-5
-2.3300000e-5+4.1640000e-5=1.83E-5

So assuming 1.835e-5 C/cycle, at 1.5kHz, I get

1.8350000e-5*1500=0.027525

I get 0.02C/s

Which is 0.02*3600=72.0
72C/h

There are bigger tantelum capacitors that are reasonable cost,
220uF, 330uF

How does the energy transfer change when I use a 330 instead of 100
Most of the charging is in 2ms, so frequency would be 500Hz

I want to redo the first one again using the 2000Hz so that I can have an even comparison, cause I cant allow the same tolerance for switching tolerance on this one.

-6.0346000000e-5+1.2056600000e-4=6.022E-5

6.022E-5 C/cycle

500*6.022000000e-5=0.03011
0.03011 C/s

0.03011*3600=108.396

108.394C/h

BACK TO 100uF
-1.640535500000e-5+3.3360203e-5=1.6954848E-5
2000*1.6954848E-5=0.0339097

0.0339*3600=122.04

I am actually getting more energy transfer out of a smaller capacitor cause the frequency can be higher.

Leave 4:00

10/08/23

August 10, 2023 8:37 AM

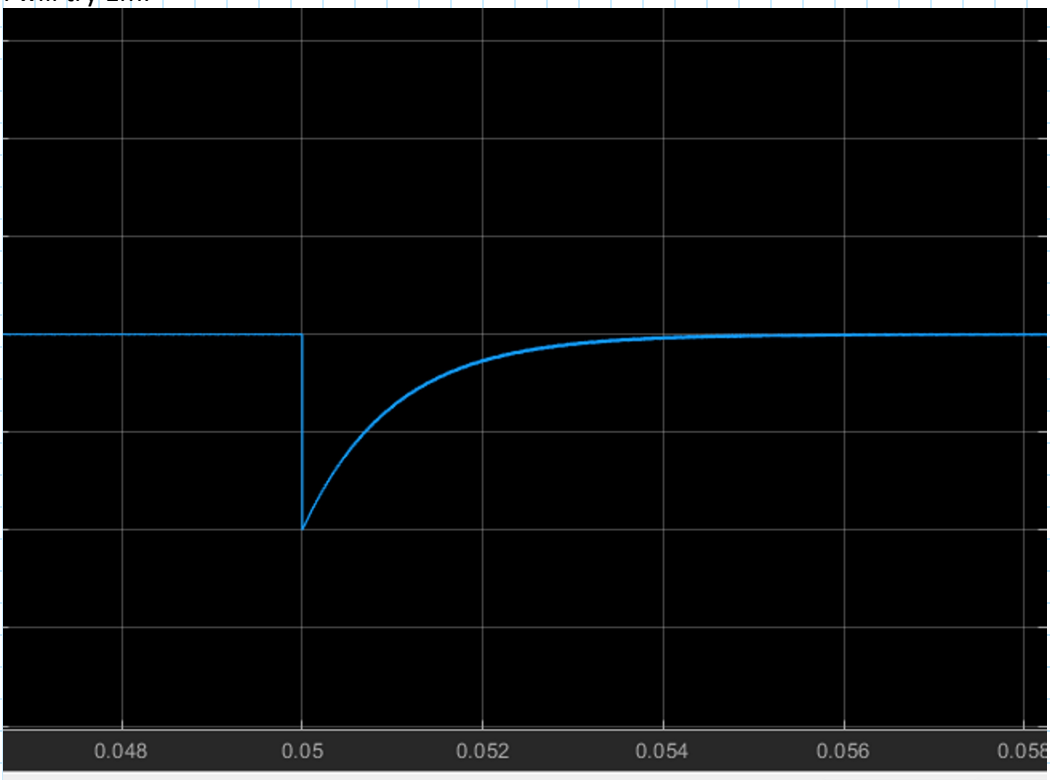
Start 8:30

Yesterday I was trying different capacitor values. The theory was that increasing the size of the capacitor would increase the rate that we were able to transfer charge. I tried two different capacitances and found that the rate of charge was nearly identical. This was because the charge/cycle would increase but the cycles/s would decrease. It appears that these factors are both linear.

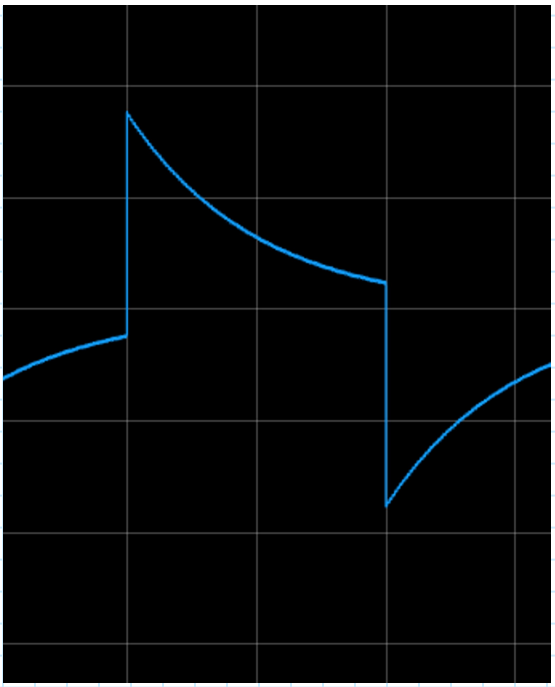
I am going to dramatically increase the capacitance to see if I still get roughly the same C/s

Before I used 100uF and 330uF.

I will try 1mF



Well use a settling time of 4ms, which would be a frequency of 250Hz



It needs to be a bit slower
Lets do 200Hz

Peaks

	Value	Time
	-1.675e-04	0.010
	-3.370e-04	0.015
	-5.065e-04	0.020

-1.675000000000e-4+3.37000000000e-4=0.0001695
-3.3700000000e-4+5.065000000000e-4=0.0001695
0.0001695 C/cycle
0.0001695 * 200=0.0339
0.0339 C/s
0.0339*3600=122.04

Again, we hit the exact same value for the rate of charge transfer.

ERROR: I found that I used 2X the frequency I should have up until this point. This is because I counted one cycle as a charge cycle. But it should also include the discharge cycle.

I will switch to the proper frequency and redo the calculations:

R1 = 3, C2 =100uF, Freq = 500Hz
(6.09200000000e-5-2.3700000000000e-5)/2=1.861E-5
1.861E-5 C/cycle
500*1.861E-5=0.0093

0.0093*3600=33.48 C/h

R1 = 1, C2 = 100uF, Freq = 1000Hz
3.623000000e-5-1.68600000e-5=1.937E-5
1.937E-5 C/cycle
1000*1.937E-5=0.0194

0.0194*3600=69.84 C/h

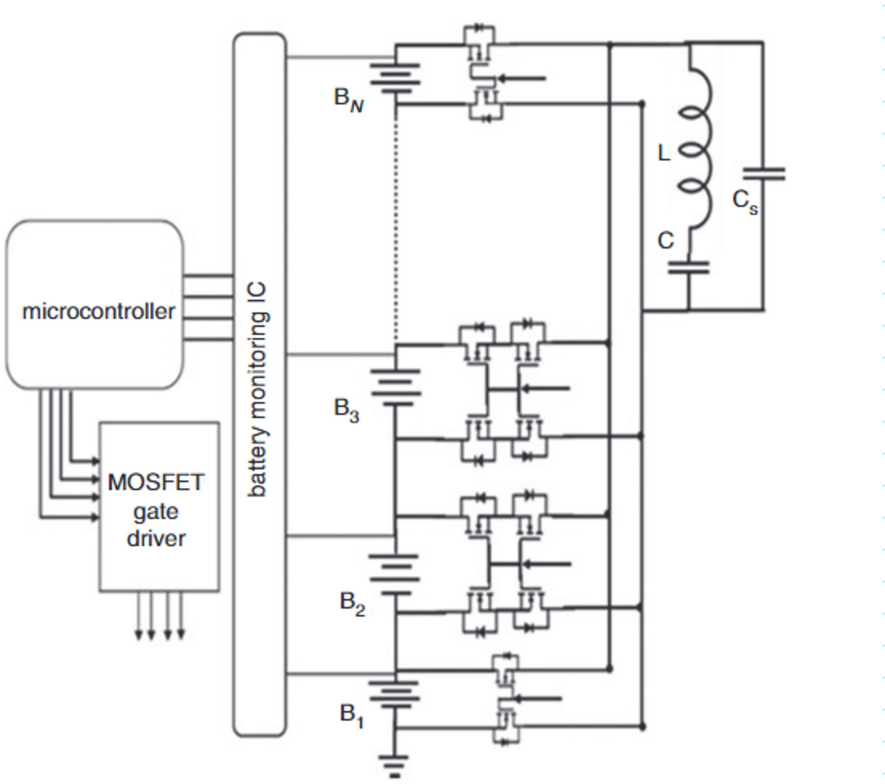
R1 = 1, C2 = 1mF, Freq = 100Hz
5.656000000000e-4-3.680000000e-4=0.0001976
0.0001976 C/cycle
100*0.0001976=0.01976

0.01976*3600=71.136 C/h

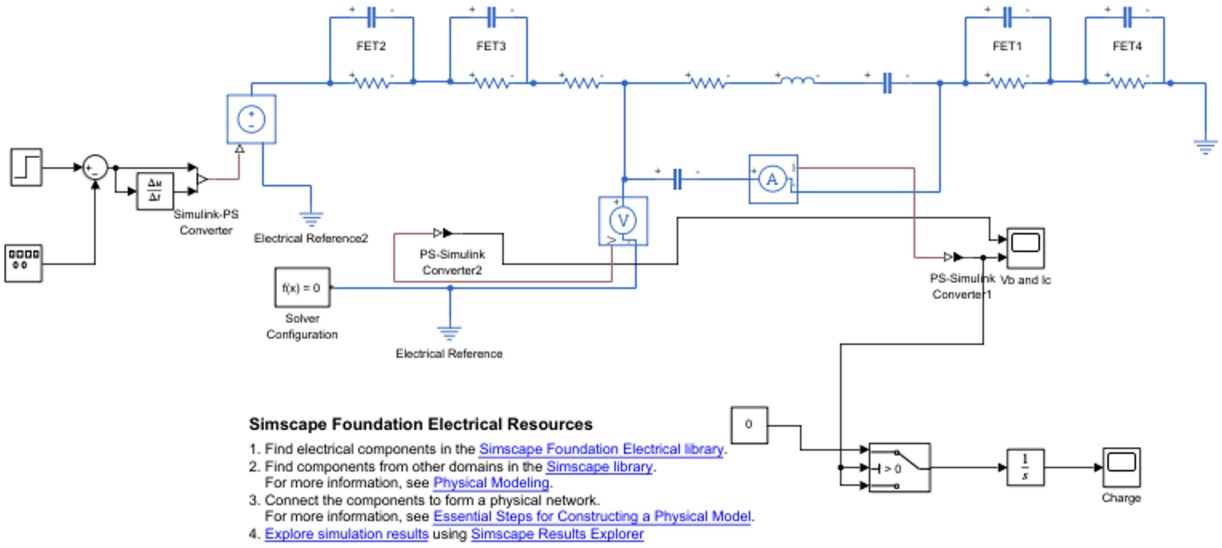
It appears that the most significant factor is the loop resistance, R1. This is going to be largely determined by the MOSFET on resistance. Looking at the simple surface mount ones that Nick had suggested, they have a resistance of 1.5Ohm, which is why I tested an R1 value of 3Ohm (2 FETs).

Nick sent a datasheet for similar MOSFETS that have 0.2Ohm resistance, so I will look into those instead.

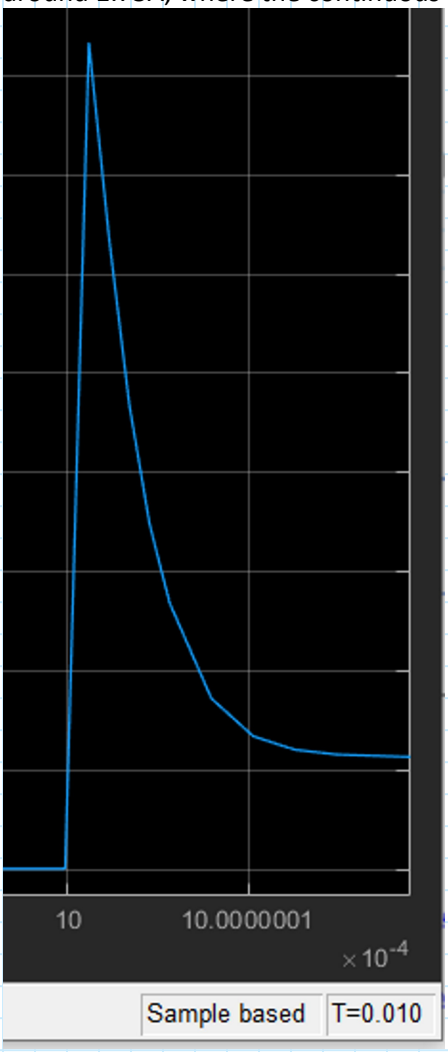
Also Nick pointed out there will be more than 2 Fets in the loop. There will be a max of 4



The datasheet also specifies capacitance in the FETs. This may be good to include just to make sure we don't reach a frequency where that matters.



I tried to add in the FET capacitance, but I am not sure that this is right?
I put them in parallel because the Cds is between drain and source, and the resistance is Rds, drain source, so I assume parallel. This causes much higher current when the FETs are first switched. However it appears to peak around 1.75A, where the continuous current for these FETs is rated for 1.7A.



The peak is in 0.0000001e-4s = 1e-11s = 10e-12s

So 10ps, and the 1.75A peak is way shorter than that.

3.87600000000e-5-1.84800000000000e-5=2.028E-5
2.028E-5 C/cycle
1000*2.028E-5=0.0203

0.0203*3600=73.08 C/h

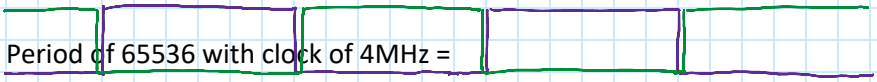
C2 = 1uF, Freq = 100kHz
1.9747000000e-4 C in 10e-3s

1.9747000000e-4/10.000000000e-3=0.019747

0.019747*3600=71.0892

This frequency is includes the charging and discharging cycle.

On the PIC it is going to generate the push-pull PWM frequencies, which means that each one should be double the frequency found in the model I believe



Period of 65536 with clock of 4MHz =

1/4.000000e6=4.0E-6 cycles/s
4.000000e-6*65536=0.262144 = 26ms

The scope is showing an on period of 16.4ms…
1/4.000000000000e6=2.5E-7 cycles/s
2.500000000000e-7*65536=0.016384
= 16.38ms
I am dumb, that is right.

So if my model says I need a frequency of 1000Hz, the period actually needs to be double on the PIC, so each should be 500Hz.

500Hz/4.0e6Hz=
500/4.0e6=0.0001

65536*0.0001=6.5536, so my counter could need to be 65536-6.55 to get 500Hz

That doesn't seem right.

65536-6.5536=65529.4464
65529.4464 * 2.5e-7=0.0164
Definitely wrong

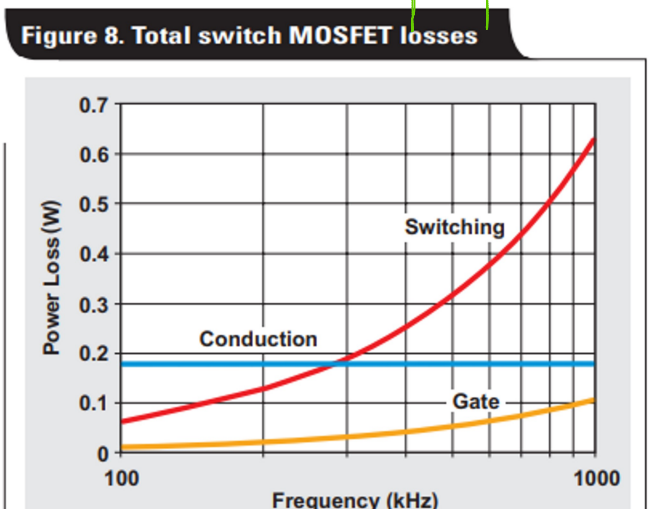
500Hz = 1/500=0.002s

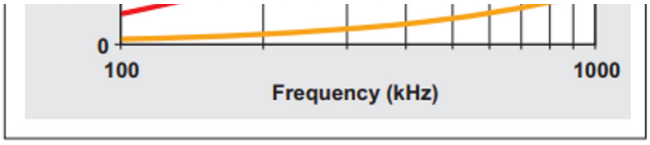
2ms period
0.002/2.500000e-7=8,000.0

So counting to 8000 should give me the right delay. To get a the overall period to relate to 1kHz (1ms)… that is wrong, I need double the frequency on the PIC, so I have 0.5ms each. Shit

2000Hz = 1/2000=0.0005
0.0005000/2.5000000e-7=2,000.0
Count to 2000 to get a overall period of 1ms

https://www.ti.com/lit/an/slyt664/slyt664.pdf?ts=1691606175708&ref_url=https%253A%252F%252Fwww.google.com%252F





MOSFETs have a few ways that they can waste power, however two of them can be minimized by minimizing the switching frequency. As I found before, increasing the size of the capacitor does not change the rate that charge can be transferred. But it does change the frequency.

For this reason, we should opt for a larger capacitor at lower frequency to improve the efficiency of the circuit.

We can get
100uF for ~$2.21
220uF for ~$3.07
330uF for ~$4.62

After that, they get to ~$100 to go any bigger.

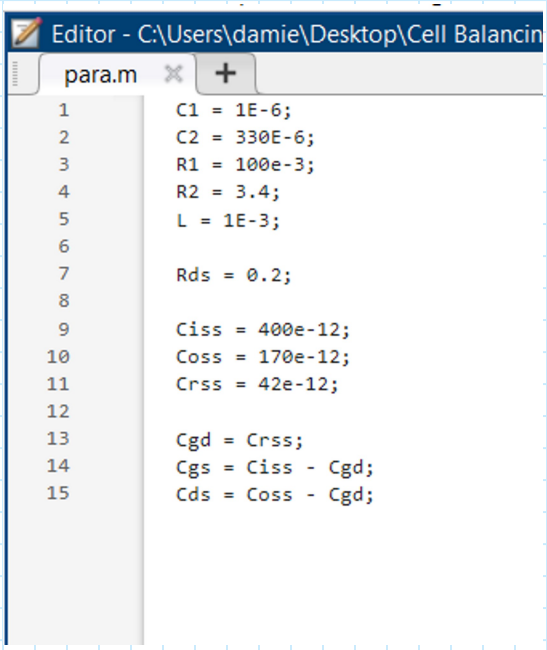
Earlier I found that 1uF required 100kHz, and I found that 100uF required 1kHz. Clearly the two are a 1:1 relatationship.

330/100=3.3 X increase in capacitance

1000/3.3=303.0303 Hz frequency would be required.

Using those values:
1.2140000000e-4-5.84000000000e-5=6.3E-5
6.3E-5*303=0.0191
0.0191*3600=68.76 C/hr

Slightly lower than the higher frequencies



These are the values for the selected components,

With these, we get
-6.017096808e-5+1.2574895e-4=6.5578E-5
6.5578E-5 C/cycle
6.5578E-5*303=0.0199
0.0199 *3600=71.64 C/hr

This should be good.

11/08/23

August 11, 2023 8:07 AM

Start 8:00

I think the only thing that I have left right now, without waiting on parts, it to set up the SPI on the RPI.

I was looking quickly at the end of the day yesterday to see what options I had for this. It seems that there is a linux/spi/spidev library that can be used similar to the linux/i2c-dev library I used for the i2c

However this looked really complicated. I also have struggled to find proper documentation for it.

https://github.com/sckulkarni246/ke-rpi-samples/tree/main/spi-c-ioctl
https://kickstartembedded.com/2021/09/13/using-spi-in-linux-using-c-spidev/

I found a mistake I made in the PCB… I connected the chip select pin to a GPIO port, however it appears that there is a designated GPIO port that the SPI is supposed to use which will allow it to be handled in the SPI software….

This should not be too big of a deal, I can either use the GPIO pin I wired, or jump the correct one over to the DAC with a wire. Its not great but oh well, it will work.

The loopback test that this person had posted online works properly with no issues on the RPI.

I am not sure what it is doing however, there are a bunch of lines where I am not sure what they are configuring. And I can not find any decent documentation on what the functions and parameters do.

I removed a bunch of the commands in the middle that did not make sense.

The S-band transmitter is MAX 1.6Mbps. I currently have it running at 1Mbps, I am not sure if I can read the 1.6 or if it needs to be a power of 2. I manually set the speed to 1.6M and the timing on the scope did change, one bit is now 620ns

1/620e-9=1.6129E6

It looks like it is going slightly faster than the max speed. Maybe I will drop it down just to make sure it does not exceed.
1.4Mbps
720ns = 1/720e-9=1.3889E6

Now I am below the maximum speed.

The loopback test is still working, so it seems that the RPI is capable of running at this speed.

As far as transferring an image goes, the Ximea software will need to store the image in some format in memory. I hope that I will be able to get the address of the image and the size. This way I can just run the SPI to send back the image my reading from the start of the memory location for how ever many bytes it is.

```
#include "..\include\xiApi.h"
#include <memory.h>
#define HandleResult(res,place) if (res!=XI_OK) {printf("Error after %s (%d)",place,res);goto finish;}

HANDLE xiH = NULL;
XI_RETURN stat = XI_OK;

// image buffer
XI_IMG image;
memset(&image,0,sizeof(image));
image.size = sizeof(XI_IMG);
```

Clear memory

&image will give reference to image location and sizeof(image) will give the number of bytes reserved for the image

```
// Image buffer
XI_IMG image;
memset(&image,0,sizeof(image));
image.size = sizeof(XI_IMG);

// Retrieving a handle to the camera device
stat = xiOpenDevice(0, &xiH);
HandleResult(stat,"xiOpenDevice");

// Setting Exposure Time parameter (10ms)
stat = xiSetParamInt(xiH, XI_PRM_EXPOSURE, 10000); // microseconds
HandleResult(stat,"xiSetParam (exposure time set)");

// Note:
// The default parameters of each camera might be different
// in different API versions, please set all parameters
// expected by your application to required value.

// Start acquisition
stat = xiStartAcquisition(xiH);
HandleResult(stat,"xiStartAcquisition");

// getting image from camera
stat = xiGetImage(xiH, 1000, &image);
HandleResult(stat,"xiGetImage");

printf("OK - Got one image (%dx%d) from camera\n", image.width, image.height);
```

memory

&image will give reference to image location

and sizeof(image) will give the number of bytes reserved for the image

↳ Pass these two arguments to SPI and it can just send data

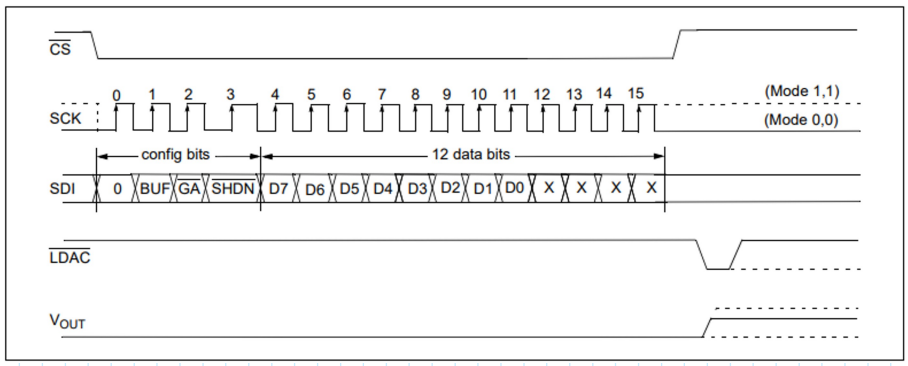
↳ If they want a burst of images over PEI, just store the references and lengths in an array that the SPI function can access then pass the SPI function the 2 arrays and the number of images

```
Uint16_t imageRef[32]
Uint16_t imageLen[32]
int count = 0
while(over PEI){
    image = take image
    imageRef[count] = &image
    imageLen[count] = sizeof(image)
    count ++
}
SendImage(&imageRef, &imageLen, count);
```

For the DAC

command. The write command consists of 16 bits and is used to configure the DAC's control and data latches. Register 5-1 through Register 5-3 detail the input register that is used to configure and load the DAC register for each device. Figure 5-1 through Figure 5-3 show the write command for each device.

FIGURE 5-2: *Write Command for MCP4911 (10-bit DAC). Note: X are don't care bits.*



REGISTER 5-1: WRITE COMMAND REGISTER FOR MCP4921 (12-BIT DAC)

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
0	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
bit 15															bit 0

REGISTER 5-2: WRITE COMMAND REGISTER FOR MCP4911 (10-BIT DAC)

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
0	BUF	$\overline{GA}$	$\overline{SHDN}$	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	x	x
bit 15															bit 0

REGISTER 5-3: WRITE COMMAND REGISTER FOR MCP4901 (8-BIT DAC)

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
0	BUF	$\overline{GA}$	$\overline{SHDN}$	D7	D6	D5	D4	D3	D2	D1	D0	x	x	x	x
bit 15															bit 0

Where:

bit 15 0 = Write to DAC register
1 = Ignore this command

bit 14 **BUF:** V_{REF} Input Buffer Control bit
1 = Buffered
0 = Unbuffered

bit 13 $\overline{\textbf{GA}}$**:** Output Gain Selection bit
1 = 1x ($V_{OUT} = V_{REF} * D/4096$)
0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)

bit 12 $\overline{\textbf{SHDN}}$**:** Output Shutdown Control bit
1 = Active mode operation. VOUT is available.
0 = Shutdown the device. Analog output is not available. VOUT pin is connected to 500 kΩ (typical).

bit 11-0 **D11:D0:** DAC Input Data bits. Bit x is ignored.

The SPI can go up to 20MHz so I can leave the 1.4MHz I have set now.

Since the first two and last

The loop back is working, but the scope seems weird.

I am reading 0…11100101011…0

The command should be 39b0 which is 0011100110110000

00111001 10110000

It looks like there is 1 zero in the middle but I think that is cause it transmits 1 byte at a time.
So if I split up the scope value where that middle 0 is, I get
111001 1011

Then I just need to add 0's to the two sides (it is active high so I cant see start and end 0's on the scope)
00111001 10110000

Now they match.
However the digital logic graph in the datasheet does not show space for a 0 in the middle… Can I change the buffers to be larger and avoid this?

Im being dumb, I never showed the Clock line. There is a pause in the clock line, therefor that 0 is not going to be read by the DAC. Everything is fine.

This script should work for testing the DAC.

Regarding the image, I should be able to make a script that will work under the principle I define before
Where you give it a list of starting points, and the length of each image, and the number of images.

I wonder how large the references are?

Actually I have the API for the camera downloaded, so I can probably use it

12:00 lunch

Back 1:00

I was thinking that I could also have an array that says when a slot in the array is available.

So it would be a table like this:

Address	length	available

0x8932	32	0
0x4823	0	1

And the SPI method could change the available value to let the image taking method know which slots can be used again.

Additionally, I just clear the length after finishing transmitting the image. That way the image taking script can just itterate through the array until it finds one with length 0.

If this gets into too many images, it may be good to figure out how to store them on the RPI as a file, and then transfer that over SPI.

That is actually probably better cause a JPEG or other image format is probably compressed some compared to the raw image data.

https://stackoverflow.com/questions/32436857/how-to-store-images-in-c-with-ximea
Looks like the best way to do this would be OpenCV, which means I need to install that too.

Then im thinking I can use OpenCV to save the image into some compressed type, and then just read the file using basic c++ functions and transmit that data over the SPI.

So the images would be coming back as PNG or JPEG.

The install for openCV seems fairly simple.
https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html

So to recap
I want to capture an image using the Ximea API, then use OpenCV library to convert that Raw image into some compressed format (Like JPEG) and save that as a file somewhere on the PI.

Then I want another thread that is running the SPI code, where it reads the JPEG file using the standard C++ file libraries and sends the JPEG image byte by byte to the S-band.

Then once it is done, the file can be deleted, and it can move on to the next image

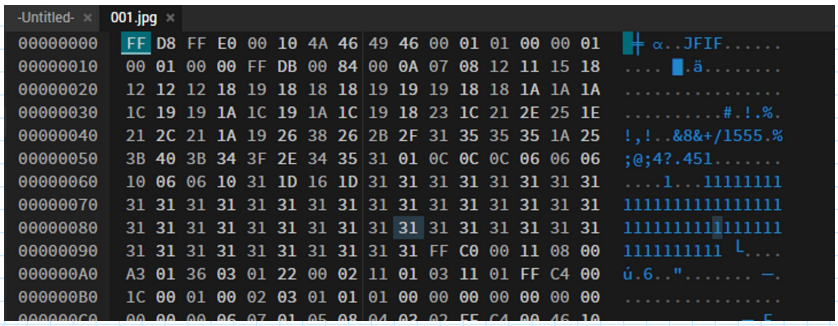
SpudNik-1_2023 > image > transmitImages.cpp > main()

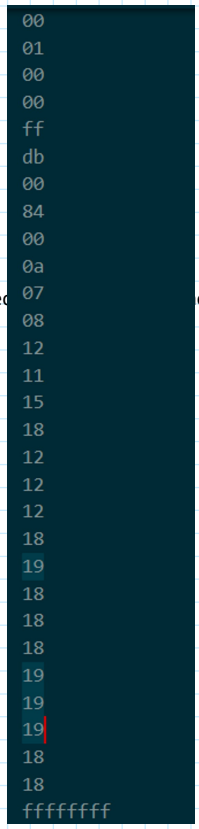
```cpp
#include <stdio.h>

using namespace std;

int main(){
    FILE* image;
    //char* filename = "images_BACKUP/test.bin";
    char* filename = "images_BACKUP/001.jpg";
    image = fopen(filename, "r");

    int c;
    if(!image){
        printf("File not found.\n");
    } else {
        while(1){
            c = fgetc(image);
            printf("%02x\n", c);
            if(c < 0){
                break;
            }
        }
    }

    fclose(image);
    return 0;
}
```

I got a script to start reading a jpg file as binary, however it is not getting through the whole file.
I opened the image in an online Hex viewer and this is what I get:

-Untitled- × 001.jpg ×

```
00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01  ╪ α..JFIF......
00000010  00 01 00 00 FF DB 00 84 00 0A 07 08 12 11 15 18  .... ▌.ä........
```





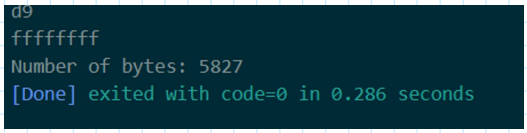
My script is not getting very far into it. It appears that it makes it almost to the end of line 3.

I notice that the value at this point is 1A. That sounds like a special character… But since this is an image file those spec ean anything.
But im thinking that something to do with the file reader is interpreting it different.

Maybe it has to do with the fgetc function.

I found you can add a b to the "r" argument in fopen to open it as a binary file

This is allowing me to get WAY more bytes when reading. I will add a counter to make sure that I get all the bytes.



Looks like I am getting the whole thing properly.
I should be able to write this to another file byte by byte to simulate sending it over the S-band

SpudNik-1_2023 > image > transmitImages.cpp > main()

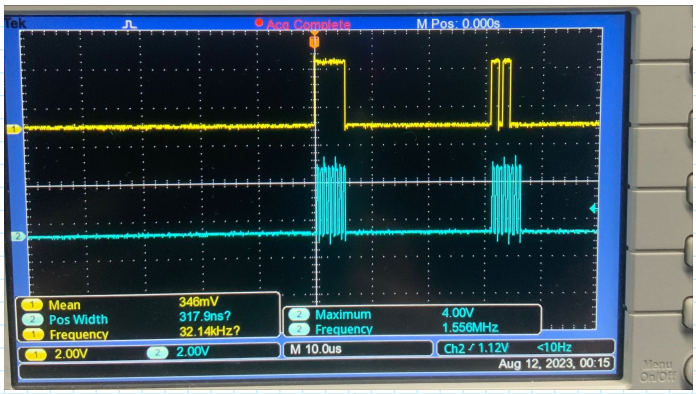
```cpp
#include <stdio.h>

using namespace std;

int main(){
    FILE* image;
    FILE* out;
    //char* filename = "images_BACKUP/test.bin";
    char* filename = "images_BACKUP/001.jpg";
    image = fopen(filename, "rb");
    out = fopen("out.jpg", "wb");

    int c;
    int count = 0;
    if(!image){
        printf("File not found.\n");
    } else {
        while(1){
            c = fgetc(image);
            printf("%02x\n", c);
            if(c < 0){
                break;
            }
            fwrite(&c, 1, 1, out);
            count++;
        }
    }
    printf("Number of bytes: %d", count);
    fclose(image);
    return 0;
}
```

transmitImages.cpp U | out.jpg U | test.bin U

SpudNik-1_2023 > image > out.jpg

I was able to successfully read the image byte by byte, and write it into a new file. Since I was able to see the image, I think that means it works fine.

I'm going to add the SPI to this script and try and run it on the PI to see how much delay there is between bytes.



It is really slow. I am running at maybe 1/5 of the speed possible by the SPI. I wonder if I can increase this by not using the fgetc function and doing something more basic.

From that, I can see that there is a delay of ~25us/byte

I can read the file and send the SPI in larger chunks.

Doing 8 bytes at a time, I get 55us between transmissions.
55/8=6.875us delay/byte
This is on average much faster.

How big of chunks can I do?
I tried 32, and that appears to work, with a delay of 40us in between.
40/32=1.25us delay/byte

I did a test in the original SPI script and I found the delay is from the SPI, not the file read. I definitely can not redo the SPI by myself in a simpler way, so im going to have to compromise.

I can get the image to transfer on the PI like I did before, but it only works with the fgetc command to do one byte at a time.
I can't use the fgets. I found that it is getting stuck with the 0x0a value as that is the new line character, and it stops at a new line character. I found another command fread that is similar

I was able to successfully take a JPG file from the raspberry pi, read it with my code, send the data over the SPI line, read the SPI line, and reconstruct the image.