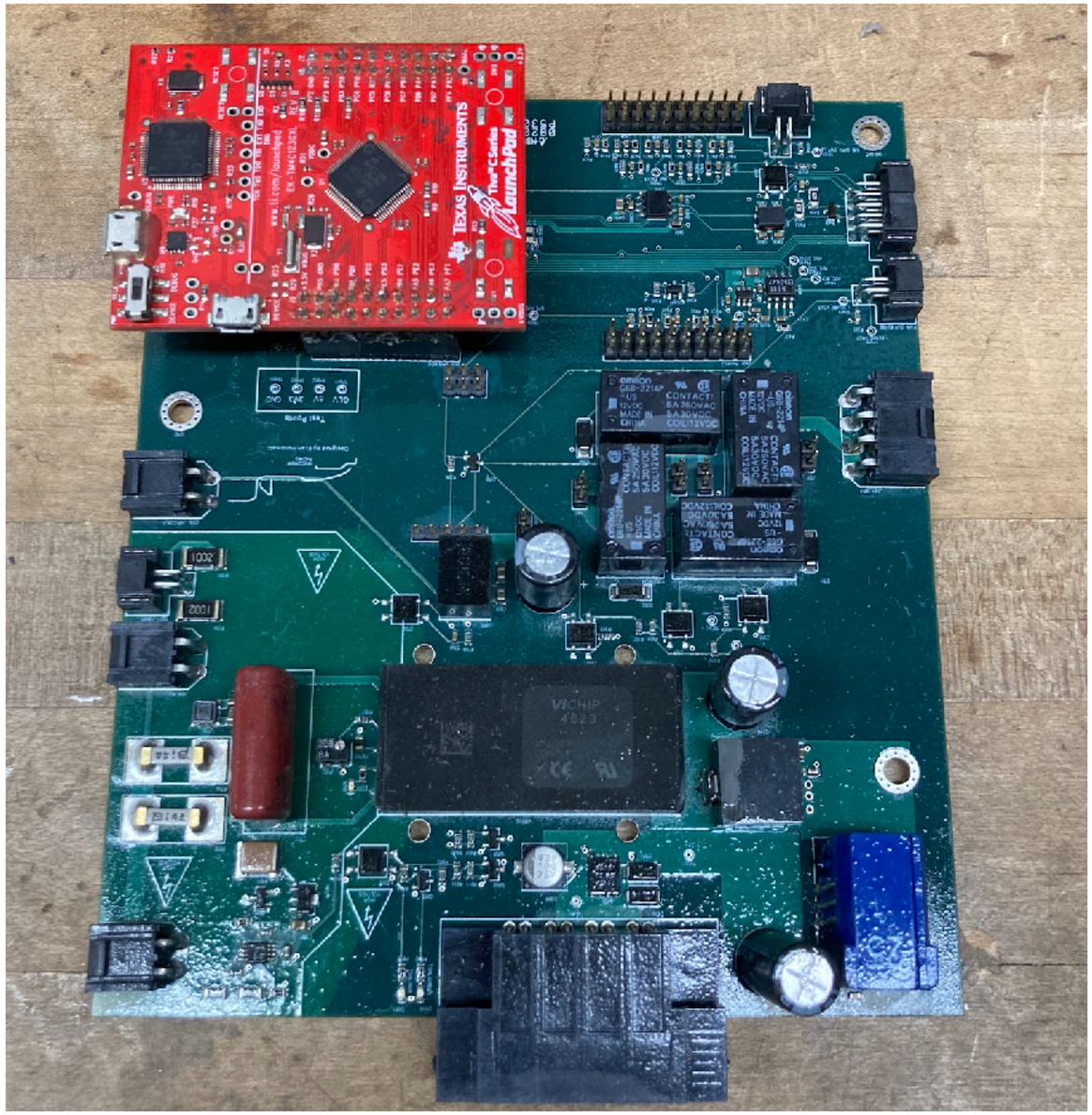
BMS Main Components

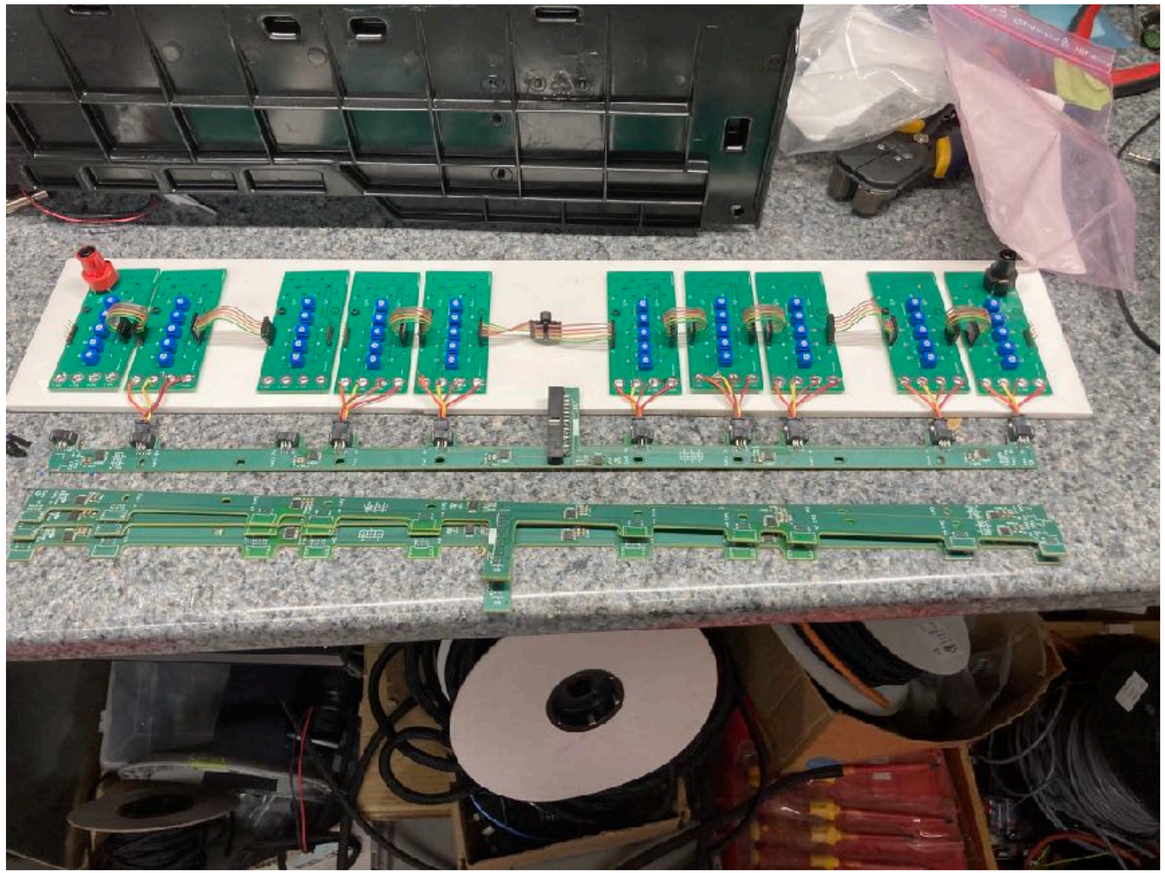
Battery workflow

Main board + MCU

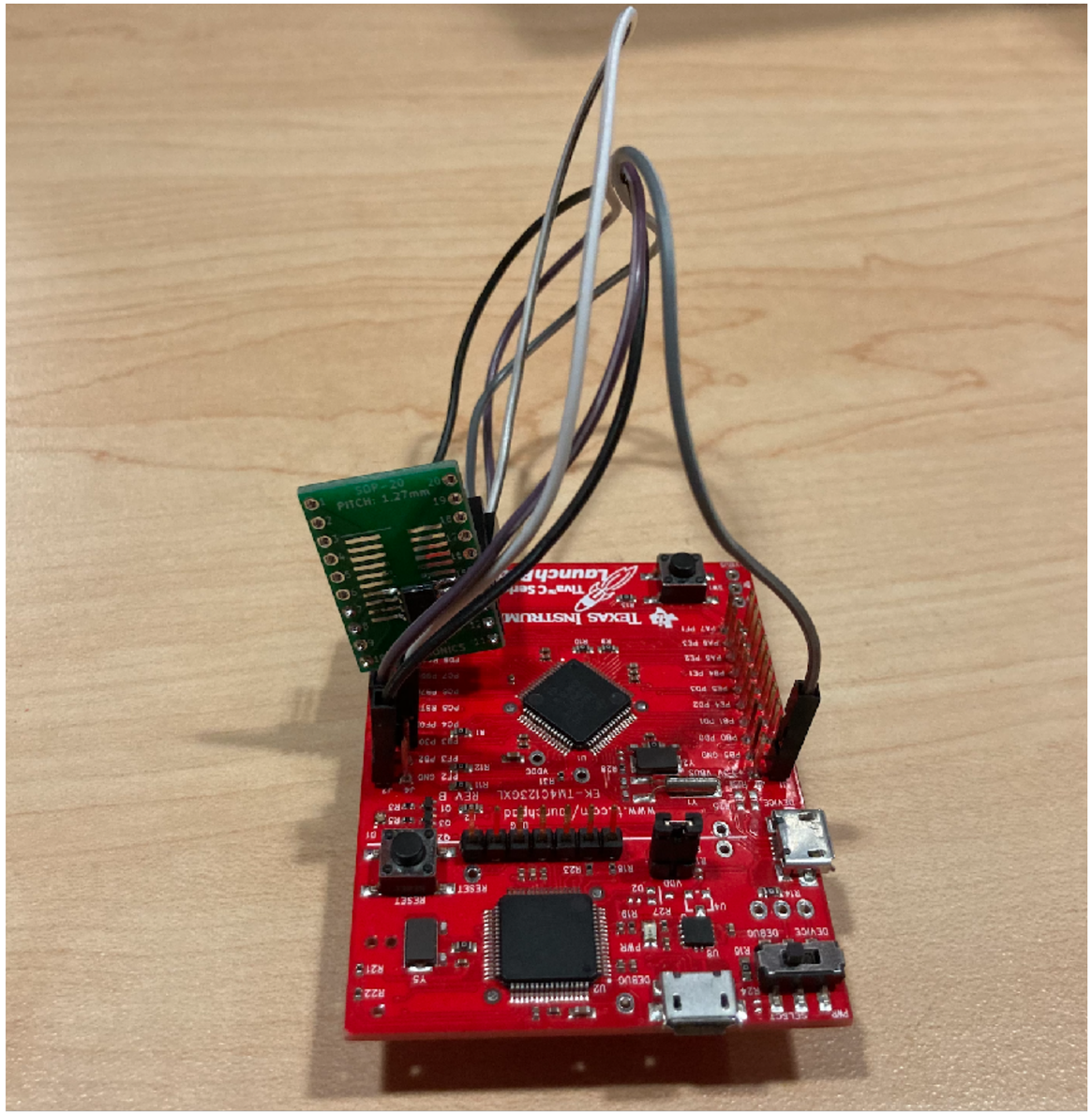


Battery secondary Board (There were none In our inventory)

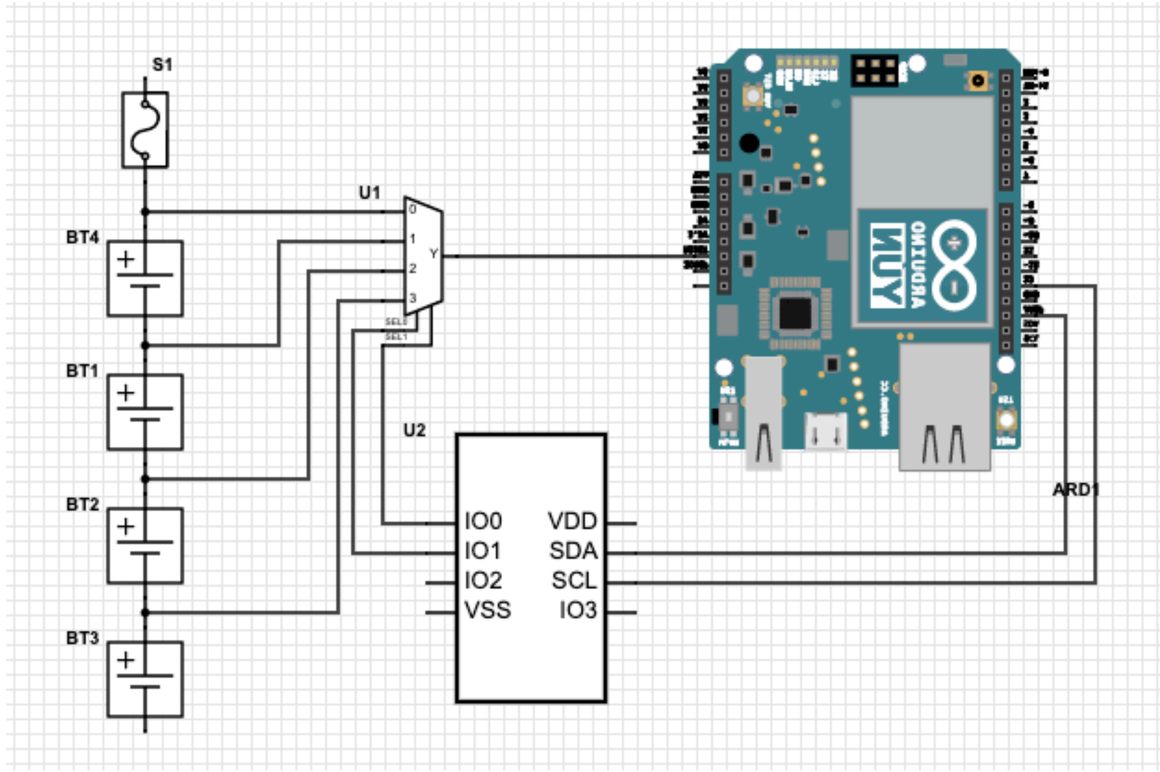
Battery trace Board



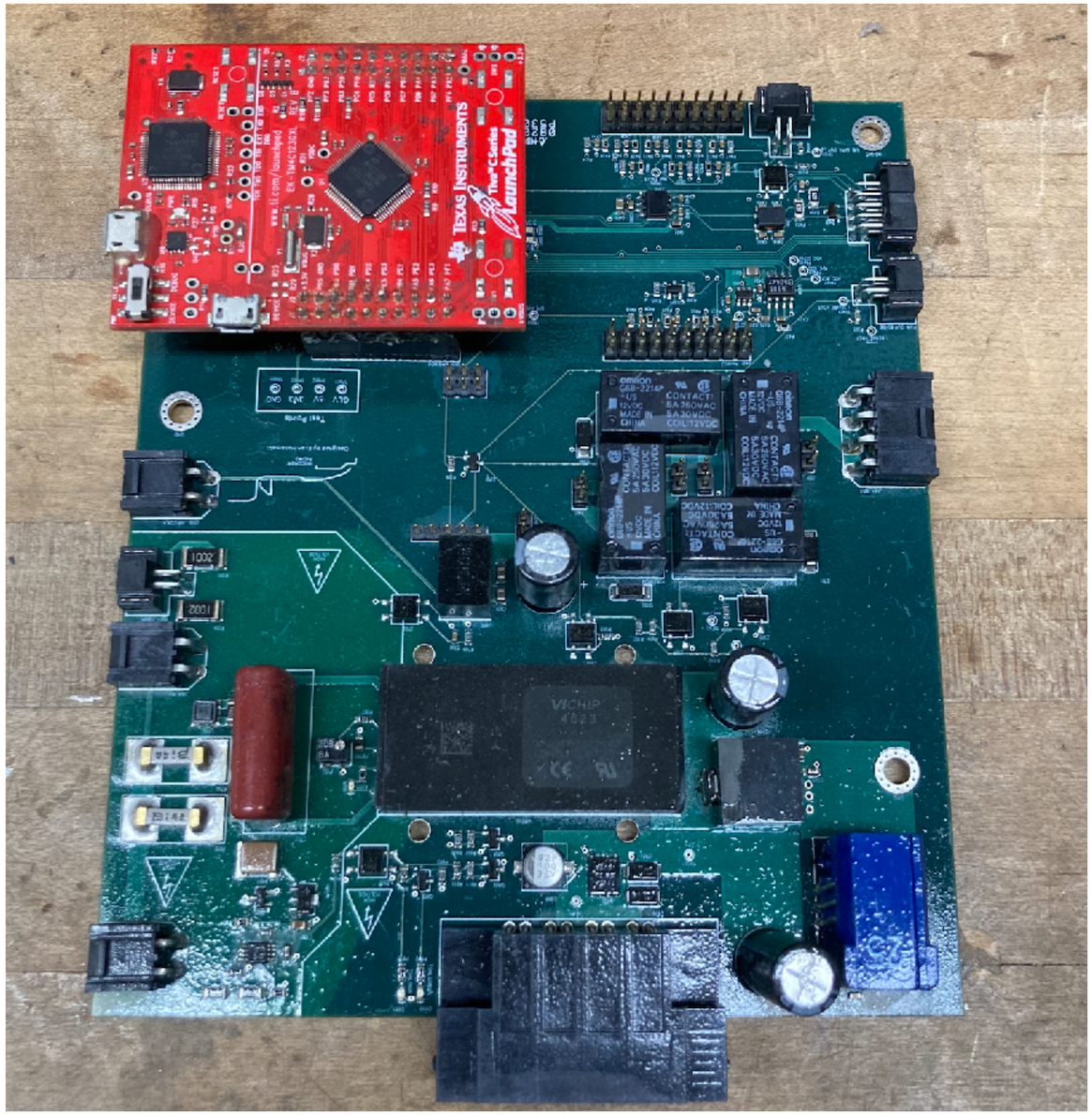
I/O Expander



Multiplexer + Batteries



Voltage reading Goes back to Battery secondary And main board



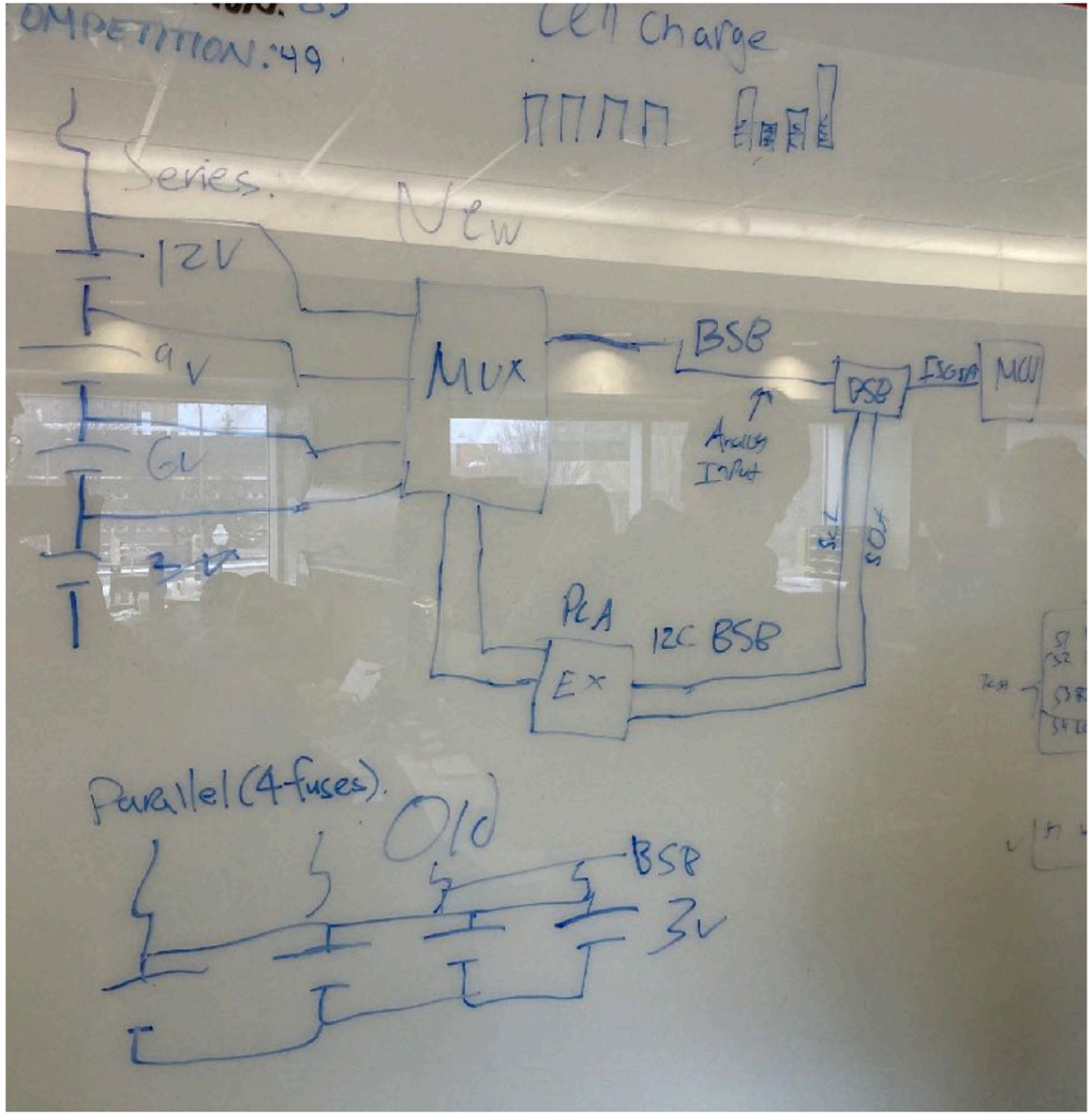
Rinse & Repeat

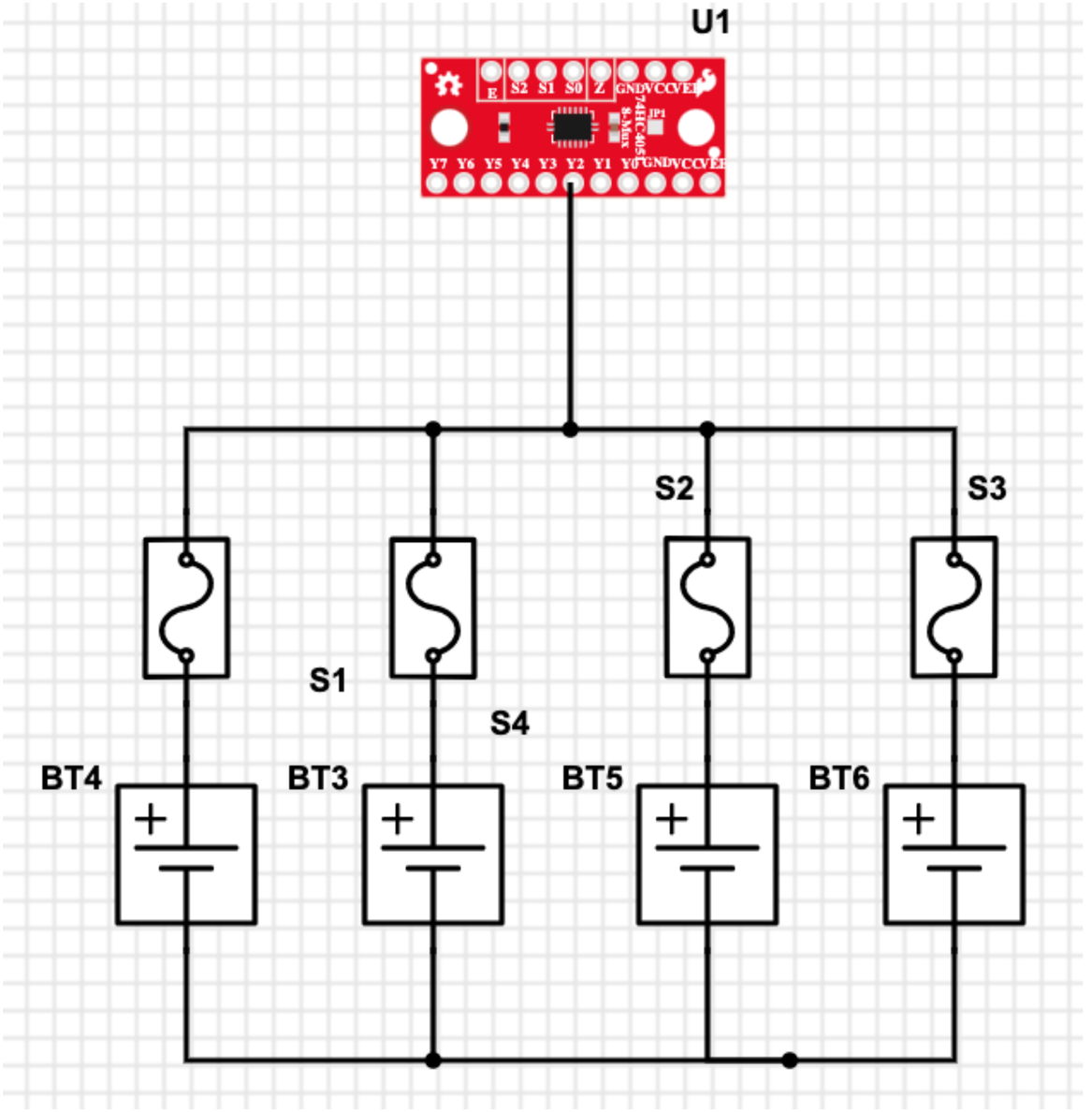
WR-222e Battery Multiplexing

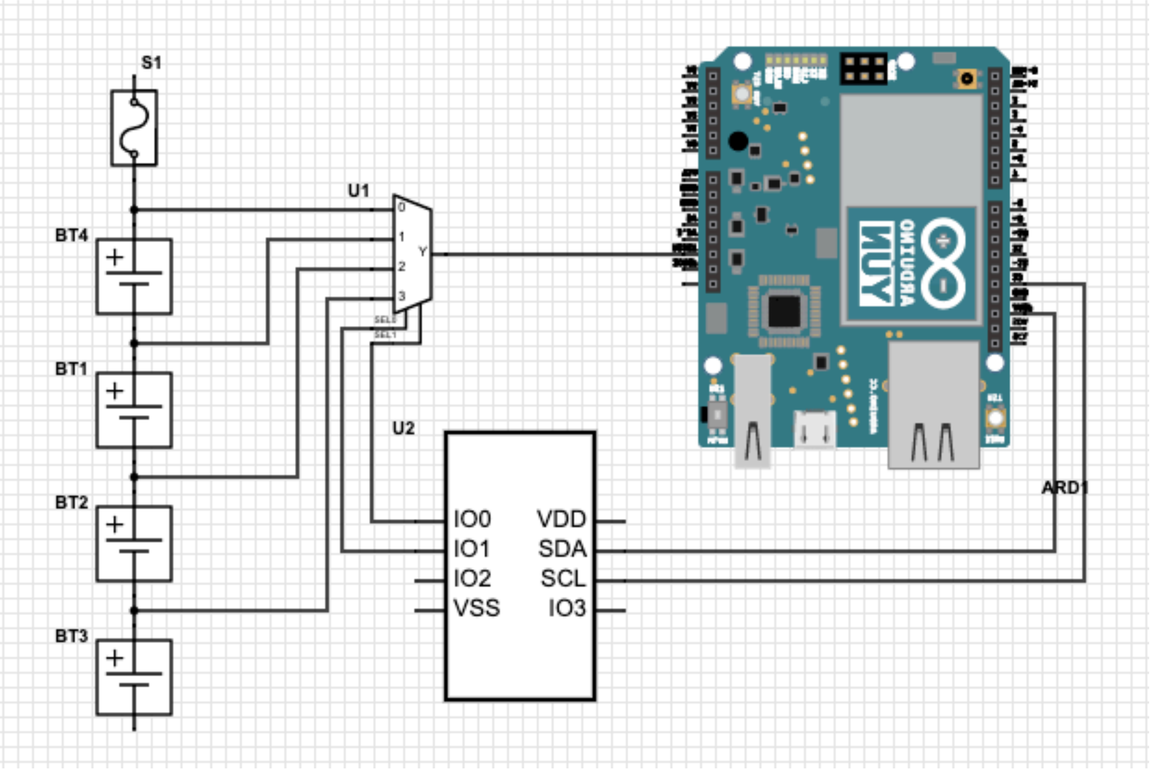
Project Introduction

WR222-e Battery Layout

- From <u>parallel</u> to <u>series</u> connection
- Uses a <u>quarter</u> of fuses from previous battery layout
- Makes 4x more voltage detections
- Purpose: Cell balancing and charging easier, batteries will not share voltages




U1
E
S2
S1
S0
Z
GND
VCC
VEE
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
GND
VCC
VEE
S1
S2
S3
S4
BT4
BT3
BT5
BT6

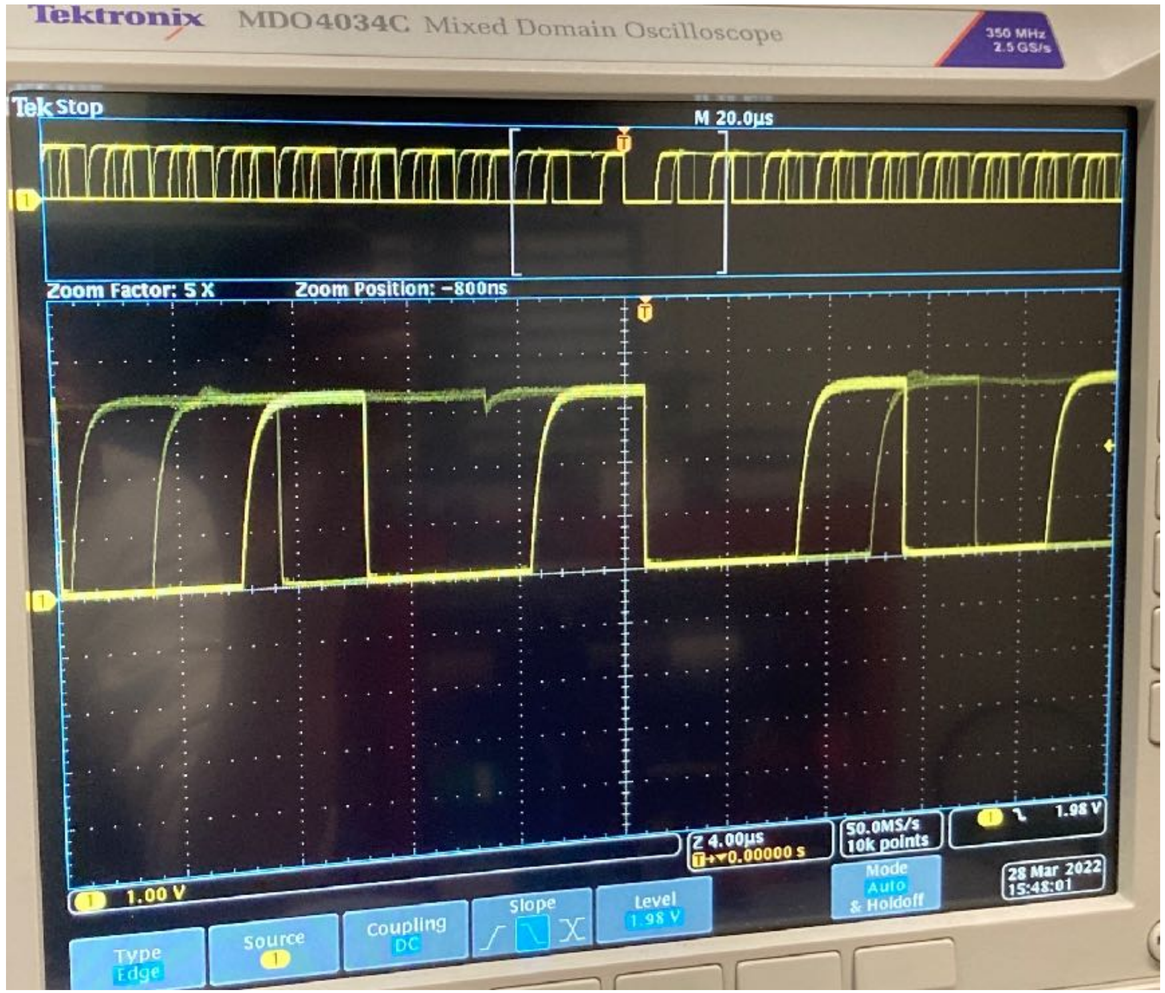

S1
U1
BT4
BT1
BT2
BT3
U2
IO0
IO1
IO2
VSS
VDD
SDA
SCL
IO3

Doing work

- Start with master-slave loopback to send I2C messages on Launchpad (dev kit)
- Proceed to transmit commands to I/O expander (PCA9536)
- Tester code to create and validate square waves over oscilloscope
- Integrating code onto existing Batter Management System

```
while (1)
{
    SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);

    // volatile uint32_t ui32Loop;

    init();
    InitI2C0();

    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3); // Initialize Launchpad
    GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_5); // Set B5 pin to take input

    // Wait until master module is done transferring
    while (I2CMasterBusy(I2C0_BASE))
    {
    }

    int counter = 0; // Counter to keep track of port num
    char output_command[2]; // [0] = Command byte, [1] = Data to register byte
    output_command[0] = PCA9536_PIN_OUTPUT;

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, GPIO_PIN_3); // Flash LED to green

    I2CSendString(PCA9536_ADDRESS, PCA9536_PIN_CONFIG_ALL); // Turn Port 0123 into output port

    while (1)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, GPIO_PIN_3);

        // Muxing through the ports
        switch (counter)
        {
            case 0:
                output_command[1] = PCA9536_PORT1;
                break;

            case 1:
                output_command[1] = PCA9536_PORT2;
                break;
```

Square wave on Oscilloscope!



Integrating onto Existing code base

bms.c	working temperatures and voltages	4 months ago
can.c	one big massive commit	1 year ago
can_pack.c	ADD: finish readding auto charging	2 years ago
charger.c	Updated charging voltages, will need to verify once pack is assembled....	4 months ago
current.c	ADD: finish readding auto charging	2 years ago
eeprom.c	Release 1.4.0	3 years ago
fault_manager.c	ADD: finish readding auto charging	2 years ago
faults.c	UPDATE: faults, make temperature faults more clear, and update all fau...	2 years ago
lookup.c	Fixed 2d luts.	4 years ago
ltc1380.c	deletes for 6811	1 week ago
ltc1864.c	See CHANGELOG 1.1.3	3 years ago
ltc6811.c	Update pca9536.c according to btb testing	1 day ago
main.c	Updated temp framework, still working on small bug with mux swap	5 months ago
params.c	added in temperature OOR comments	3 weeks ago
pca9536.c	Update pca9536.c according to btb testing	1 day ago
scheduler.c	fixed fault matrix message mux, removed hp scheduler, changed sche...	3 years ago

https://gitlab.com/wisconsinracing/BMS

Format transmit Commands

```c
/**
 * @brief Format message to send from LTC to GPIO
 * @param channel number of battery port, range 0-3
 */
void pca9536_format_transmit_i2c(uint8_t channel) {
    uint8_t transmit[6];

    // first 8 bit is ouptut command
    // lower 8 bit is data to port (where reg number changes)
    uint16_t msg = pca9536_set_channel(channel);
    uint8_t cmd = (msg & 0xff00) >> 8;
    uint8_t data = (msg & 0x00ff);

    transmit[0] = ( ((LTC6811_START & 0xf) << 4) | ((PCA9536_ADDRESS >> 4) & 0xff) );
    transmit[1] = ( ((PCA9536_ADDRESS & 0xf) << 4) | (LTC6811_ACK & 0xf) );
    transmit[2] = ( ((LTC6811_BLANK & 0xf) << 4) | ((PCA9536_PIN_OUTPUT & 0xf)) );
    transmit[3] = ( ((PCA9536_PIN_OUTPUT & 0xf) << 4) | (LTC6811_ACK & 0xf) );
    transmit[4] = ( ((LTC6811_BLANK & 0xf) << 4) | ((data >> 4) & 0xff) );
    transmit[5] = ( ((data & 0xf) << 4) | (LTC6811_NACKSTOP & 0xf) );

    ltc6811_transmit_i2c(transmit, PCA9536_NUM_I2C_DATA);
    // Transmit:  Slave Address | Command Byte | Data to Port
}
```

WR_I2C Library

- https://github.com/libochengdi/WR_I2C
- Wisconsin Racing's first I2C library for hardware communications
- Simple, straight forward function calls to perform I2C send and receive on device addresses

```
/**
 * Instructions:
 *      1. Set SysCtlClockSet
 *      2. Call init() and InitI2C0()
 *      3. I2CReceive(SLAVE_ADDR, reg num) to read from slave's port
 *      4. I2CSendString(SLAVE_ADDR, command) to write command to device
 */

//*****************************************************************************
//
// Function Declarations
//
//*****************************************************************************

void init_gpio_periph(void);

void InitI2C0(void);

uint32_t I2CReceive(uint32_t slave_addr, uint8_t reg);

void I2CSendString(uint32_t slave_addr, char array[]);

void InitConsole(void);

void init(void);

void FlashLED(int portnum);

#endif /* WR_I2C_H_ */
```


Tektronix
Milwaukee
ACCURATE PATTERN
Snap-on
FCA
FIAT CHRYSLER AUTOMOBILES
MERCURY
YAMAHA
MICRO-EPSILON
MODINE
NSK
MONSTER
THERMTECH
BOSCH
OSHKOSH
BAUM
Weasler
J D LASER
Aluma-Tec
QMI
NELSON
TE AIRTECH
SHORAI
TDK·Lambda
RMSTATOR
Miller
WISCONSIN
8
UWGP
CIRRUS
ANSYS
ISTHMUS
alphagraphics
Carbotech

Fin