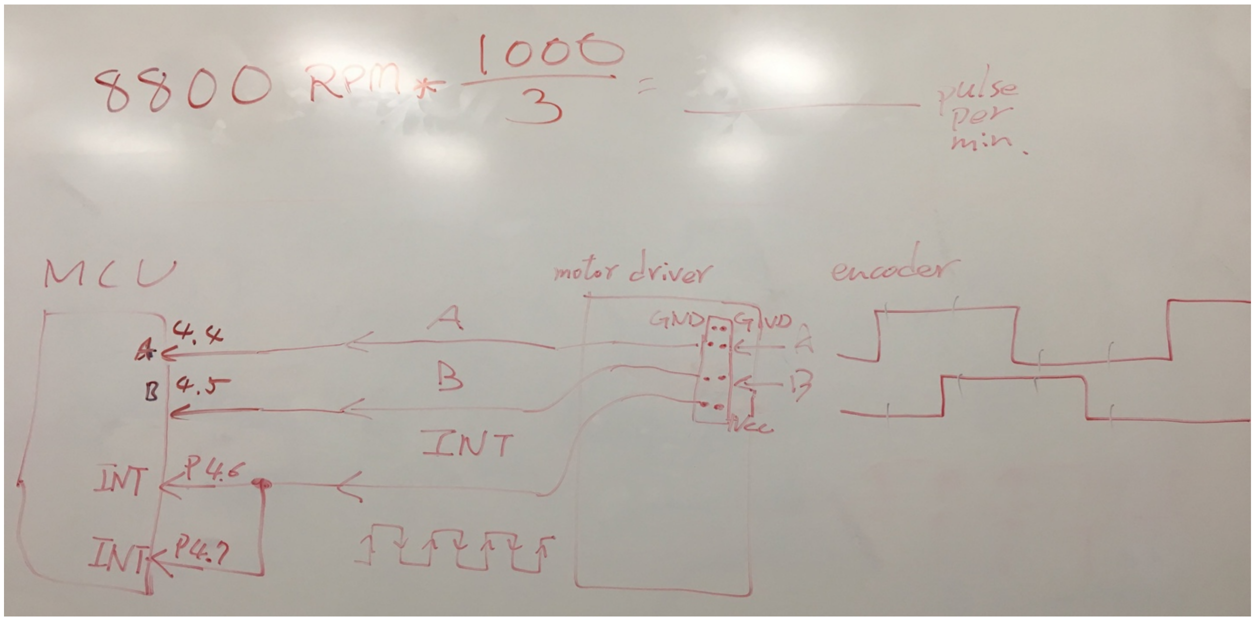
Ho-Ren Chuang prelab6 report

1. We discussed in class various ways of decoding the encoder signals on your Rover. Please (a) write code to decode the encoder signals, and (b) draw a wiring diagram showing the connections you would make between your motor controller and MSP432 in order to use your code. Turn in a main.c file with code to set up the pins to read the encoder signals and with several Interrupt Service Routines that would (1) be triggered due to edges on some signals coming from the motor controller, and (2) would increment or decrement a number based on whether the encoder is counting up or down. You may assume that the pulses coming from the encoder will always be in the correct order, i.e. it will never miss a pulse (and, as such, you do not need to do error checking). You may use any pins you want on your MSP432 to read in the signals. You only need to read the encoder from one wheel for this. For this you will need to turn in:

(a) Your code.

See bellows/the attached file.

(b) A wiring diagram to go with your code.



2. For your encoder setup, if the motor spins at 8800 rpm (the no-load speed), by how much will your encoder counter increase in a second?

8800 RPM * 1000/3 (from spec) /60 per sec = 48888.8888889 pulse per second

That's all! We wanted to make this on the shorter side. The lab will be more involved so you should start that early. Note that for your lab you will also need to compute the velocity of how quickly the wheel is moving, in case you want to start thinking about that now.

My code is also attached in main.c

```
volatile uint8_t EncodeSigA = 0xff, EncodeSigB = 0xff;
volatile bool isNotClockwise = true, oldisNotClockwise = true;
volatile bool islost = false;
volatile int32_t counter = 0;
enum StatName{AA, BB, CC, DD};
volatile static enum StatName OldCurStat = AA, NewCurStat = AA;

if((status & GPIO_PIN6) || (status & GPIO_PIN7)) {
        EncodeSigA = GPIO_getInputPinValue(GPIO_PORT_P4, GPIO_PIN4);
        EncodeSigB = GPIO_getInputPinValue(GPIO_PORT_P4, GPIO_PIN5);
        // state machine
        // 01 11 10 00 01 11 10
        // AA BB CC DD AA BB CC
        // Forward <--==Rover==---> backward
        switch (OldCurStat) {
            case DD:
                if ((EncodeSigA == 0x00) && (EncodeSigB == 0x01)) {
                    NewCurStat = AA; counter--;
                    isNotClockwise = true; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else if ((EncodeSigA == 0x01) && (EncodeSigB == 0x00)) {
                    NewCurStat = CC; counter++;
                    isNotClockwise = false; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else {
                    islost = true;
                    printf(EUSCI_A0_BASE, "lost steps\n\r");
                }
                break;
            case AA:
                // 01 11 10 00 01 11 10
                // AA BB CC DD AA BB CC
                // Forward <--==Rover==---> backward
                if ((EncodeSigA == 0x01) && (EncodeSigB == 0x01)) {
                    NewCurStat = BB; counter--;
                    isNotClockwise = true; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else if ((EncodeSigA == 0x00) && (EncodeSigB == 0x00)) {
                    NewCurStat = DD; counter++;
                    isNotClockwise = false; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else {
                    islost = true;
                    printf(EUSCI_A0_BASE, "lost steps\n\r");
                }
                break;
            case BB:
                // 01 11 10 00 01 11 10
                // AA BB CC DD AA BB CC
                // Forward <--==Rover==---> backward
                if ((EncodeSigA == 0x01) && EncodeSigB == 0x00) {
                    NewCurStat = CC; counter--;
                    isNotClockwise = true; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else if ((EncodeSigA == 0x00) && (EncodeSigB == 0x01)) {
                    NewCurStat = AA; counter++;
                    isNotClockwise = false; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else {
                    islost = true;
                    printf(EUSCI_A0_BASE, "lost steps\n\r");
                }
                break;
            case CC:
                // 01 11 10 00 01 11 10
                // AA BB CC DD AA BB CC
                // Forward <--==Rover==---> backward
                if ((EncodeSigA == 0x00) && (EncodeSigB == 0x00)) {
                    NewCurStat = DD; counter--;
                    isNotClockwise = true; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else if ((EncodeSigA == 0x01) && (EncodeSigB == 0x01)) {
                    NewCurStat = BB; counter++;
                    isNotClockwise = false; islost = false;
                    if (isNotClockwise != oldisNotClockwise)
                        printf(EUSCI_A0_BASE, "***Direction
changed***\n\r");
                } else {
                    islost = true;
                    printf(EUSCI_A0_BASE, "lost steps\n\r");
                }
                break;
        }

        if ((isNotClockwise != oldisNotClockwise) || islost) {
            cnt = 0;
        }
        else {
            cnt++;
            if(cnt >= 333) {
                cnt = 0;
                printf(EUSCI_A0_BASE, "***isForwarding (%s)
counter %l***\n\r",
                                       isNotClockwise?"NOT":"YES",
counter);
            }
        }
        OldCurStat = NewCurStat;
        oldisNotClockwise = isNotClockwise;
        //printf(EUSCI_A0_BASE, "A %x (4.6), B %x (4.7) (hex)\n\r",
        //                         EncodeSigA, EncodeSigB); //debug
    }
```