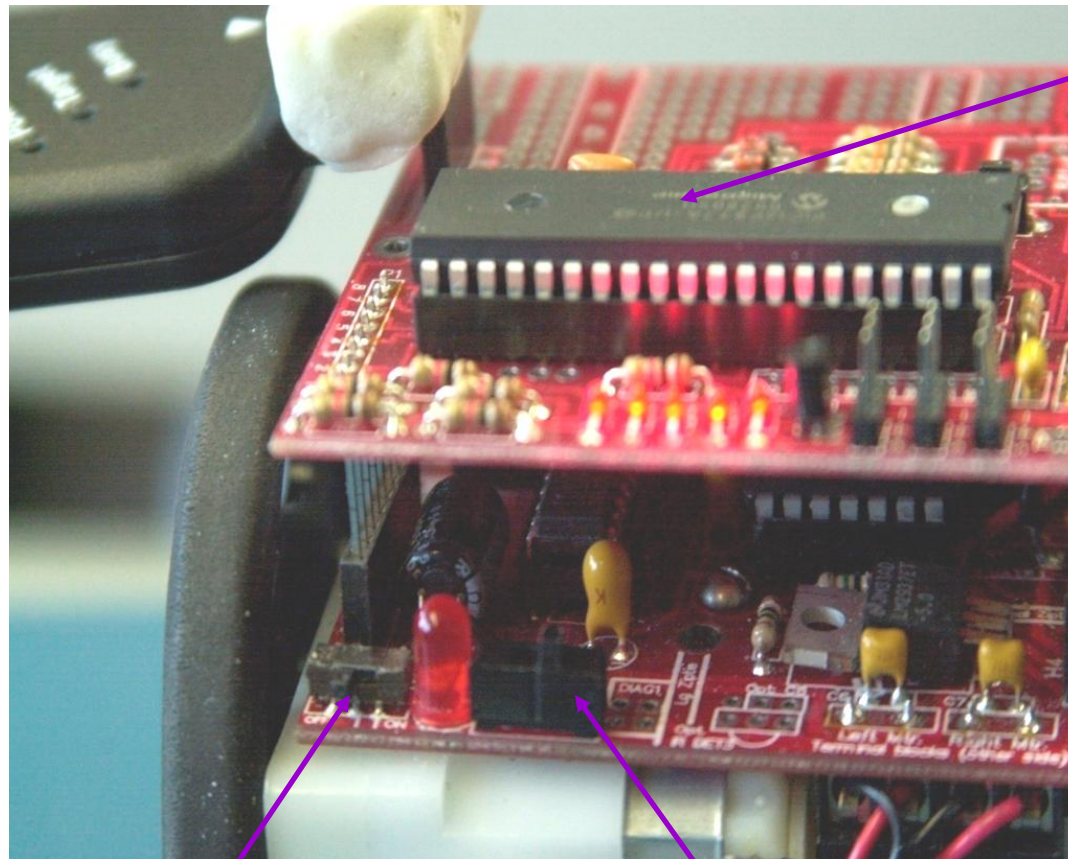


Sumovore Robot

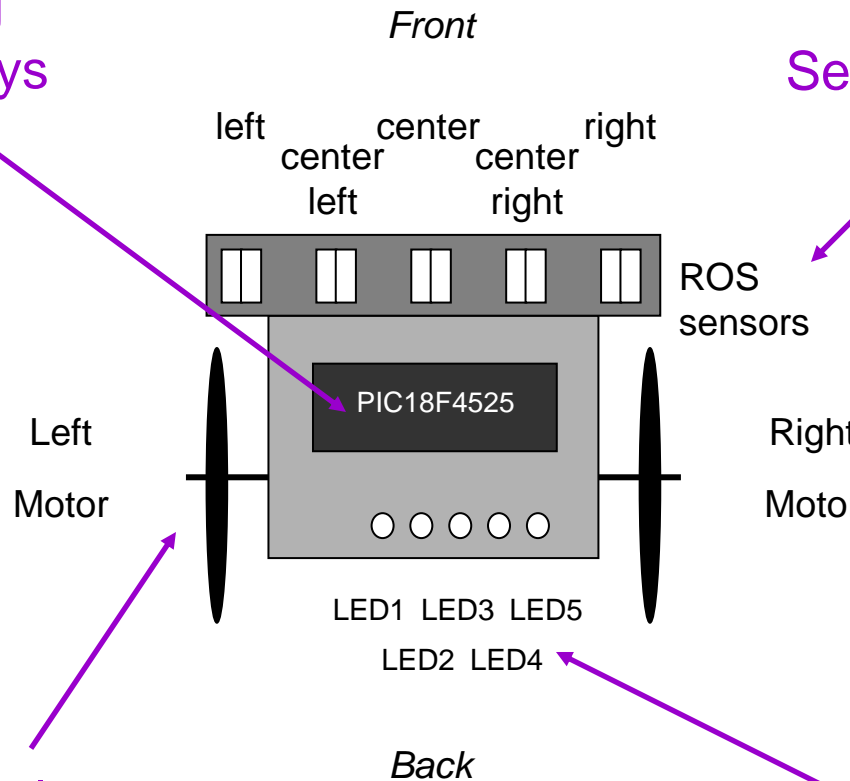


The MCU

Switch #2: Power
switch for motors.

Switch #1: Power switch
for brain board

Program
Logic and timing
Timers and delays

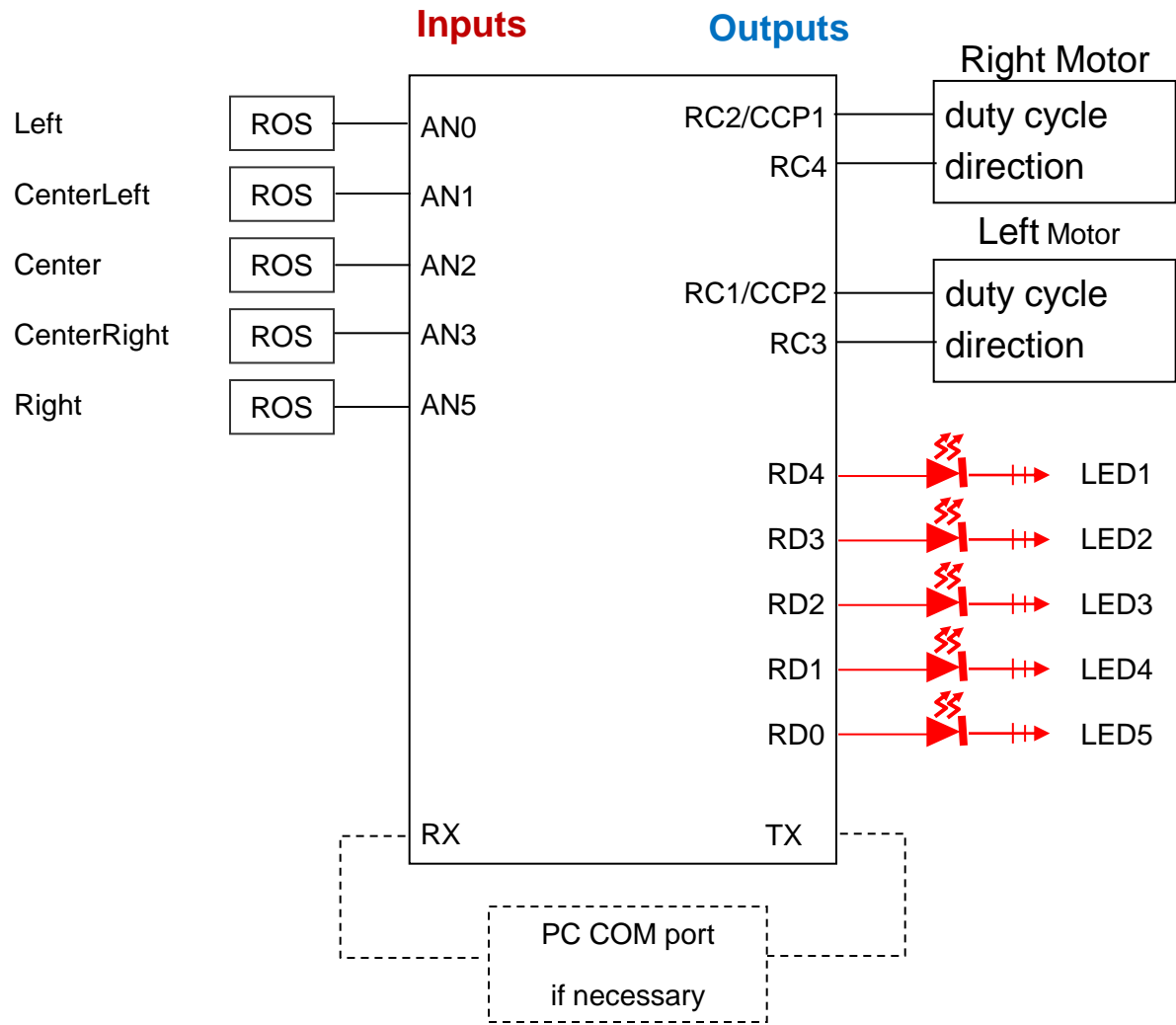


Sensors - ADC

Motor control:
Speed and Direction
PWM

LEDs: Indicators
DIO

Robot from above.



Robot pin connections.

Example Code

- Library
 - ❖ sumovore.h (many defines & some functions)
 - ❖ sumovore.c
- Example
 - ❖ main.c
 - ❖ motor_control.c
 - ❖ motor_control.h

```
// main.c

#include "sumovore.h"
#include "motor_control.h"

void main(void)
{
    initialization();

    // threshold = 400u; // if you wish to change

    while(1)
    {
        check_sensors();
        set_leds();
        motor_control(); // in motor_control.c
    }
}
```

Basic Logic

Do once

Configure PIC for ADC,
PWM, etc

Loop

- Check where the robot is on the line
- Flash LEDs appropriately
- Control wheels

motor_control.c (Part A)

```
#include "sumovore.h"
#include "motor_control.h"
void spin_left(void);
void turn_left(void);
void straight_fwd(void);
void turn_right(void);
void spin_right(void);

void motor_control(void)
{
    // very simple motor control
    if ( SeeLine.b.Center ) straight_fwd();
    else if (SeeLine.b.Left) spin_left();
    else if (SeeLine.b.CntLeft) turn_left();
    else if (SeeLine.b.CntRight) turn_right();
    else if (SeeLine.b.Right) spin_right();

    f ( (SeeLine.B ) == 0b00000u) motors_brake_all();
}
```

motor_control.c (Part B)

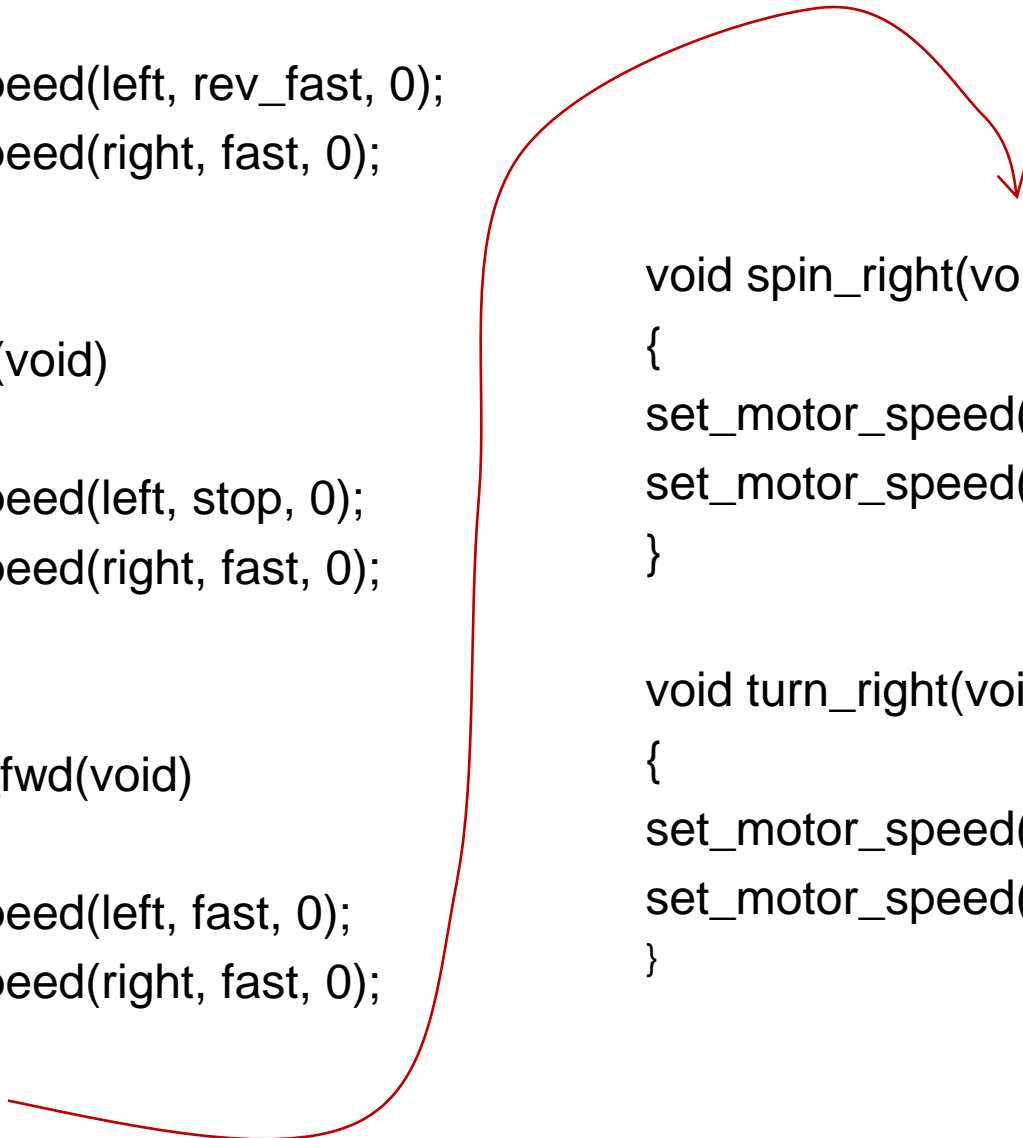
```
void spin_left(void)
{
    set_motor_speed(left, rev_fast, 0);
    set_motor_speed(right, fast, 0);
}
```

```
void turn_left(void)
{
    set_motor_speed(left, stop, 0);
    set_motor_speed(right, fast, 0);
}
```

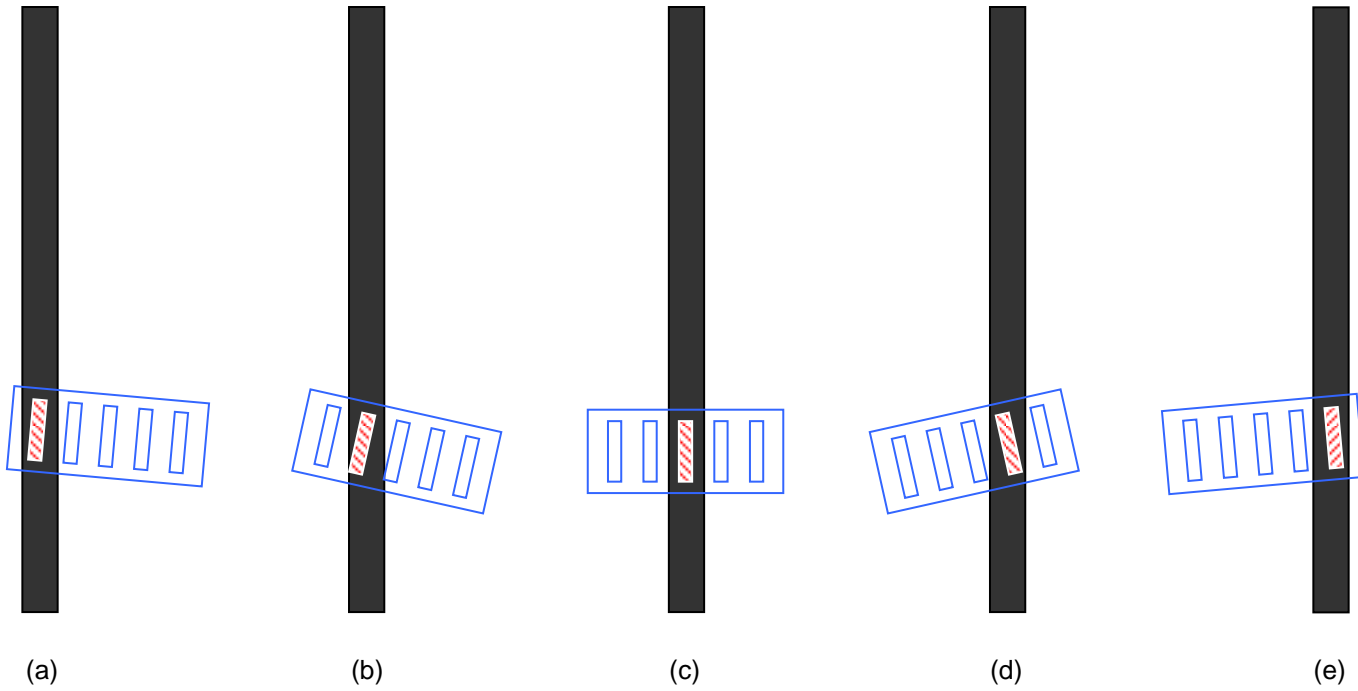
```
void straight_fwd(void)
{
    set_motor_speed(left, fast, 0);
    set_motor_speed(right, fast, 0);
}
```

```
void spin_right(void)
{
    set_motor_speed(left, fast, 0);
    set_motor_speed(right, rev_fast, 0);
}
```

```
void turn_right(void)
{
    set_motor_speed(left, fast, 0);
    set_motor_speed(right, stop, 0);
}
```



motor_control.c



Your Job

Modify `motor_control.c` to handle the complex path the robot will encounter.

sumovore.h

```
#define setLED1(a) PORTDbits.RD0=~a // When a = ON or OFF,  
#define setLED2(a) PORTDbits.RD1=~a // setLEDn(ON) turns on LEDn  
#define setLED3(a) PORTDbits.RD2=~a // setLEDn(OFF) turns off LEDn  
#define setLED4(a) PORTDbits.RD3=~a // a could also be any char or integer  
#define setLED5(a) PORTDbits.RD4=~a // but only the least significant  
                                // bit will be used.  
  
#define set_all_LEDs(a) PORTD=(((~a)&0b00011111)|(PORTD&0b11100000))  
    // in set_all_LEDs(a) a can be any 5 bit value or variable  
    // if a has more than 5 bit the most significant bits will  
    // be ignored  
  
#define LeftIR !PORTDbits.RD5 // RD5 is pin 28  
#define RightIR !PORTDbits.RD6 // RD6 is pin 29  
    // Note that the IR detectors output low when an object  
    // is detected so the not "!" of the pin is used  
    // in the define to maintain positive logic
```

sumovore.h

```
#define EnableRmotor PORTCbits.RC2 // pin_c2 -- Enable Right motor
#define EnableLmotor PORTCbits.RC1 // pin_c1 -- Enable Left motor
#define RmotorGoFwd PORTCbits.RC5 // rev for bb2 -- Right motor forward
#define RmotorGoFwdCmp PORTEbits.RE0 // rev for bb2 -- Right motor forward complement
#define LmotorGoFwd PORTCbits.RC0 // rev for bb2 -- Left motor forward
#define LmotorGoFwdCmp PORTEbits.RE1 // rev for bb2 -- Left motor forward complement

#define RLS_LeftCH0 ADC_CH0 // AN0 (left reflective line sensor)
#define RLS_CntLeftCH1 ADC_CH1 // AN1 (center left reflective line sensor)
#define RLS_CenterCH2 ADC_CH2 // AN2 (center reflective line sensor)
#define RLS_CntRightCH3 ADC_CH3 // AN3 (center right reflective line sensor)
// Pin RA4 No connection
#define RLS_RightCH4 ADC_CH4 // AN4 (right reflective line sensor)
// note PortA is only 6 bits!

#define YES 0b1 // used to turn on an individual bit
#define NO 0b0 // used to turn off an individual bit
#define ON 0b1
#define OFF 0b0
```

sumovore.h

```
#define AN0_AN4 0B1010 // used for configuration of PCFG3:PCFG0
    // A/D port Configuration Control Bits
    // these determine which pins are analog inputs
    // see page 224 of PIC18F4525 datasheet
```

```
#define THRESHOLD_DEFAULT 512u
```

```
struct sensors
```

```
{
    unsigned Left:1;
    unsigned CntLeft:1;
    unsigned Center:1;
    unsigned CntRight:1;
    unsigned Right:1;
    unsigned :3;
};
```

```
union sensor_union
```

```
{
    unsigned char B;
    struct sensors b;
};
```

sumovore.h

```
void initialization(void); // defined in sumovore.c
```

```
unsigned int adc(unsigned char channel); // defined in sumovore.c
```

```
enum motor_speed_setting { rev_fast, rev_medium, rev_slow, stop, slow, medium,  
    fast };
```

```
enum motor_selection { left, right };
```

```
void set_motor_speed(enum motor_selection the_motor, enum  
    motor_speed_setting motor_speed, int speed_modifier);  
    // defined in sumovore.c
```

```
void motors_brake_all( void );
```

```
void set_leds(void);
```

```
void check_sensors(void);
```

```
extern union sensor_union SeeLine;
```

sumovore.c

```
#pragma config WDT = OFF
#pragma config OSC = INTIO67 // allows osc1 (pin 13) and osc2 (pin 14) to be
    used as inputs
    // note there is a crystal attached to these pins on the
    // brainboard
#pragma config MCLRE = OFF
#pragma config LVP = OFF
// #pragma config lines must come before #include "sumovore.h" as sumovore.h
    redefines OFF!!!

#include <p18F4525.h>
#include <usart.h>
#include <stdio.h>
#include <adc.h>
#include <pwm.h>
#include <timers.h>
#include "..\Functions\osc.h"
#include "sumovore.h"
```

sumovore.c

```
void openPORTCforPWM(void);  
void openPORTCforUSART(void);  
void openPORTA(void);  
void openPORTB(void);  
void openPORTD(void);  
void openPORTE(void);
```

```
union sensor_union SeeLine = 0;  
unsigned int threshold; // value compared to adc result
```

sumovore.c

```
void initialization(void)
{
    set_osc_32MHz(); // to change the internal oscillator frequency (see osc.h osc.c)
    openPORTCforUSART();
    OpenUSART( USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE & USART_EIGHT_BIT &
        USART_CONT_RX & USART_BRGH_HIGH,
        16); // for 19200 bit per second
        // (32000000/19200/16)-1 = 103.17
        // actual buad rate is 32000000/(16*(103+1)) = 19230.8 baud
    openPORTCforPWM();
    openPORTA();
    openPORTB();
    openPORTD();
    openPORTE();
    PORTD = 0; // TURN ALL LED'S OFF
    OpenADC(ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_6_TAD , ADC_CH1 & ADC_INT_OFF & ADC_VREFPLUS_VDD &
        ADC_VREFMINUS_VSS, AN0_AN4);
// AN0-AN4 is defined in sumovore.h the others are defined in adc.h (C18 library)
    RmotorGoFwd = NO; // NO is defined as 0b0 in sumovore.h
    RmotorGoFwdCmp = NO;
    LmotorGoFwd = NO;
    LmotorGoFwdCmp = NO;
// PWMperiod = [(period)+1]x 4 x Tosc x TMR2
// period      Tosc      TMR2Pre      pwm_period      freq
// 255  3.13E-08      16      5.12E-04      1.95E+03
    OpenTimer2(TIMER_INT_OFF & T2_PS_1_16 & T2_POST_1_1); // TMR2 prescale is 16
    OpenPWM1(199); // TPWM = (199+1)*4*(31.25 ns)*16
                    // = 0.400 ms or 2500 Hz

    OpenPWM2(199);
    SetDCPWM1(0); // TDC = 64*(31.25 ns)*16
                  // = 0.032 ms
                  // = 0% * TPWM (800 will give 100%)

    SetDCPWM2(0);
    threshold = THRESHOLD_DEFAULT;
}
}
```

sumovore.c

```
void openPORTCforUSART(void)
{
    TRISCbits.TRISC6 = 0; // set TX (RC6) as output
    TRISCbits.TRISC7 = 1; // and RX (RC7) as input
}
void openPORTCforPWM(void)
{
    TRISCbits.TRISC0 = 0; // Direction Left M
    TRISCbits.TRISC1 = 0; // Enable Left M
    TRISCbits.TRISC2 = 0; // Enable Right M
    TRISCbits.TRISC3 = 0; // I2C SCL
    TRISCbits.TRISC4 = 0; // I2C SDA
    TRISCbits.TRISC5 = 0; // Direction Right M
    // TRISC6 and TRISC 7 initialized in openPORTCforUSART()
}
void openPORTA(void)
{
    TRISA = 0B11101111; // RA0/AN0, RA1/AN1, RA2/AN2, RA3/AN3, RA5/AN4
    SET AS INPUTS
        // RA4 not used set as output
        // bits RA6 and RA7 are left as inputs (crystal still attached
        // on sumovore)
}
```

sumovore.c

```
void openPORTB(void)
{
    TRISB = 0B11000000; // PORTB mostly not used
                        // reserve pins 39 (RB6/PGC) and 40 (RB7/PGD)
                        // as inputs to avoid conflict if ISP and PICKit2
}

void openPORTD(void)
{
    TRISD = 0b01100000; // RD7 not connected
    // RD6 is IR Right, RD5 is IR Left, RD4 is LED5, RD3 is LED4
    // RD2 is LED3, RD1 is LED2 and RD0 is LED1
}

void openPORTE(void)
{
    TRISE = 0b000; // all outputs
                // E0 and E1 are now used for motor direction and
                // dynamic braking
                // E2 is not used
}
```

sumovore.c

```
void set_motor_speed(enum motor_selection the_motor, enum motor_speed_setting motor_speed, int
    speed_modifier)
{
    const static int motor_speeds[] = { -800, -600, -400, 0, 400, 600, 800};
    int duty_cycle;
    enum e_direction {reverse,forward} dir_modifier= forward;

    duty_cycle = motor_speeds[ motor_speed ] + speed_modifier;
    if ( duty_cycle < 0 )
    {
        dir_modifier = reverse;
        duty_cycle = -1 * duty_cycle;
    }
    if ( duty_cycle > 800 ) duty_cycle = 800;

    if (the_motor == left)
    {
        SetDCPWM2((unsigned int) duty_cycle );
        if ( dir_modifier == reverse ) LmotorGoFwd = NO;
        else LmotorGoFwd = YES;
        LmotorGoFwdCmp = !LmotorGoFwd;
    }
    else
    {
        SetDCPWM1((unsigned int) duty_cycle );
        if ( dir_modifier == reverse ) RmotorGoFwd = NO;
        else RmotorGoFwd = YES;
        RmotorGoFwdCmp = !RmotorGoFwd;
    }
}
```

sumovore.c

```
void motors_brake_all( void ) // created june 26, 2009
{
    SetDCPWM1(800u ); // enable motors 100% for braking
    SetDCPWM2(800u ); //
    LmotorGoFwdCmp = NO; // ground all direction lines
    LmotorGoFwd = NO; // motor terminals will have dead short
    RmotorGoFwdCmp = NO;
    RmotorGoFwd = NO;

}

unsigned int adc(unsigned char channel)
{
    SetChanADC( channel );
    ConvertADC();
    while( BusyADC() );

    return ReadADC();
}
```

sumovore.c

```
void check_sensors(void)
```

```
{  
    SeeLine.b.Left = ( adc(RLS_LeftCH0) > threshold );  
    SeeLine.b.CntLeft = ( adc(RLS_CntLeftCH1) > threshold );  
    SeeLine.b.Center = ( adc(RLS_CenterCH2) > threshold );  
    SeeLine.b.CntRight = ( adc(RLS_CntRightCH3) > threshold );  
    SeeLine.b.Right = ( adc(RLS_RightCH4) > threshold );  
}
```

```
void set_leds(void)
```

```
{  
    setLED1(SeeLine.b.Left);  
    setLED2(SeeLine.b.CntLeft);  
    setLED3(SeeLine.b.Center);  
    setLED4 (SeeLine.b.CntRight);  
    setLED5(SeeLine.b.Right);  
}
```