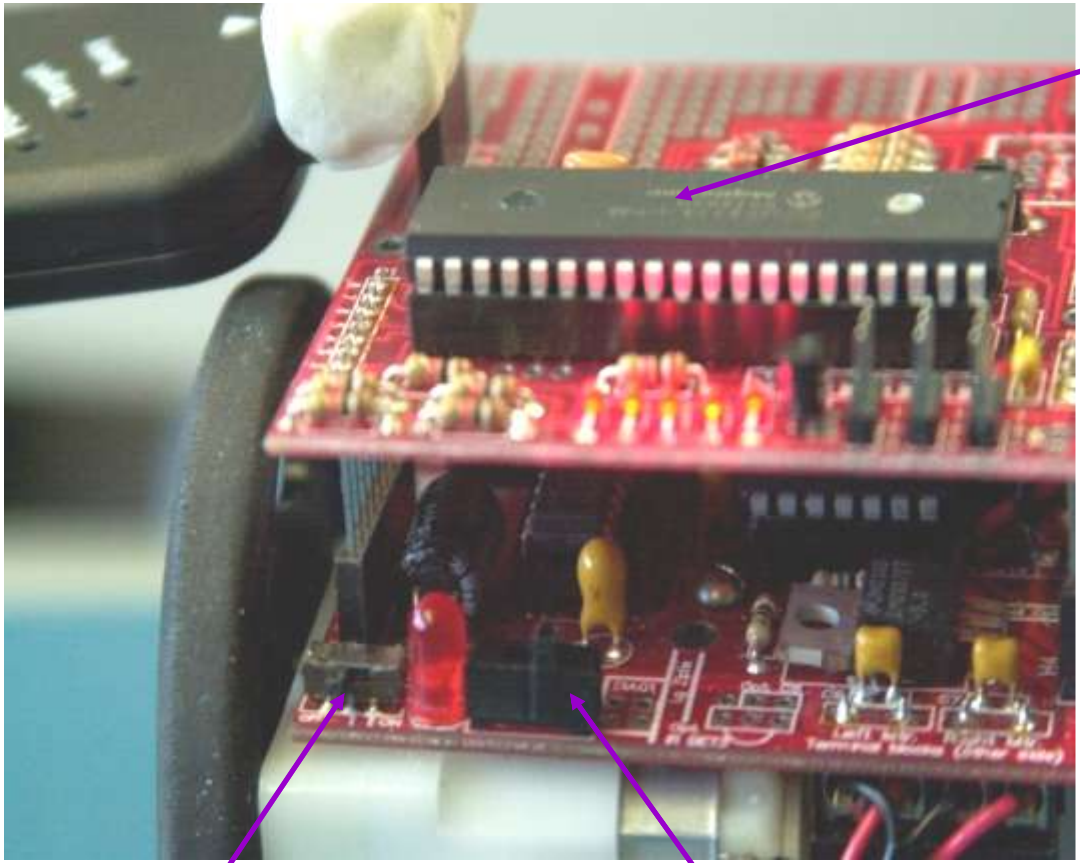
Sumovore Robot



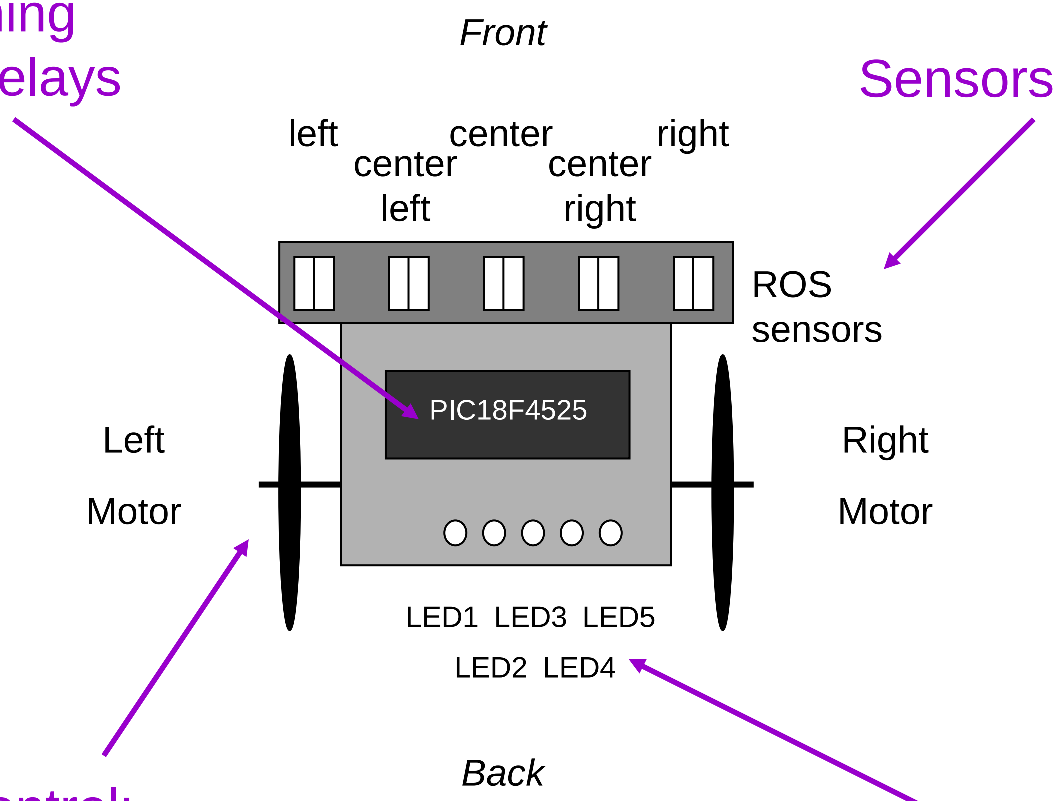
The MCU

Switch #2: Power switch for motors.

Switch #1: Power switch for brain board

Program
Logic and timing
Timers and delays

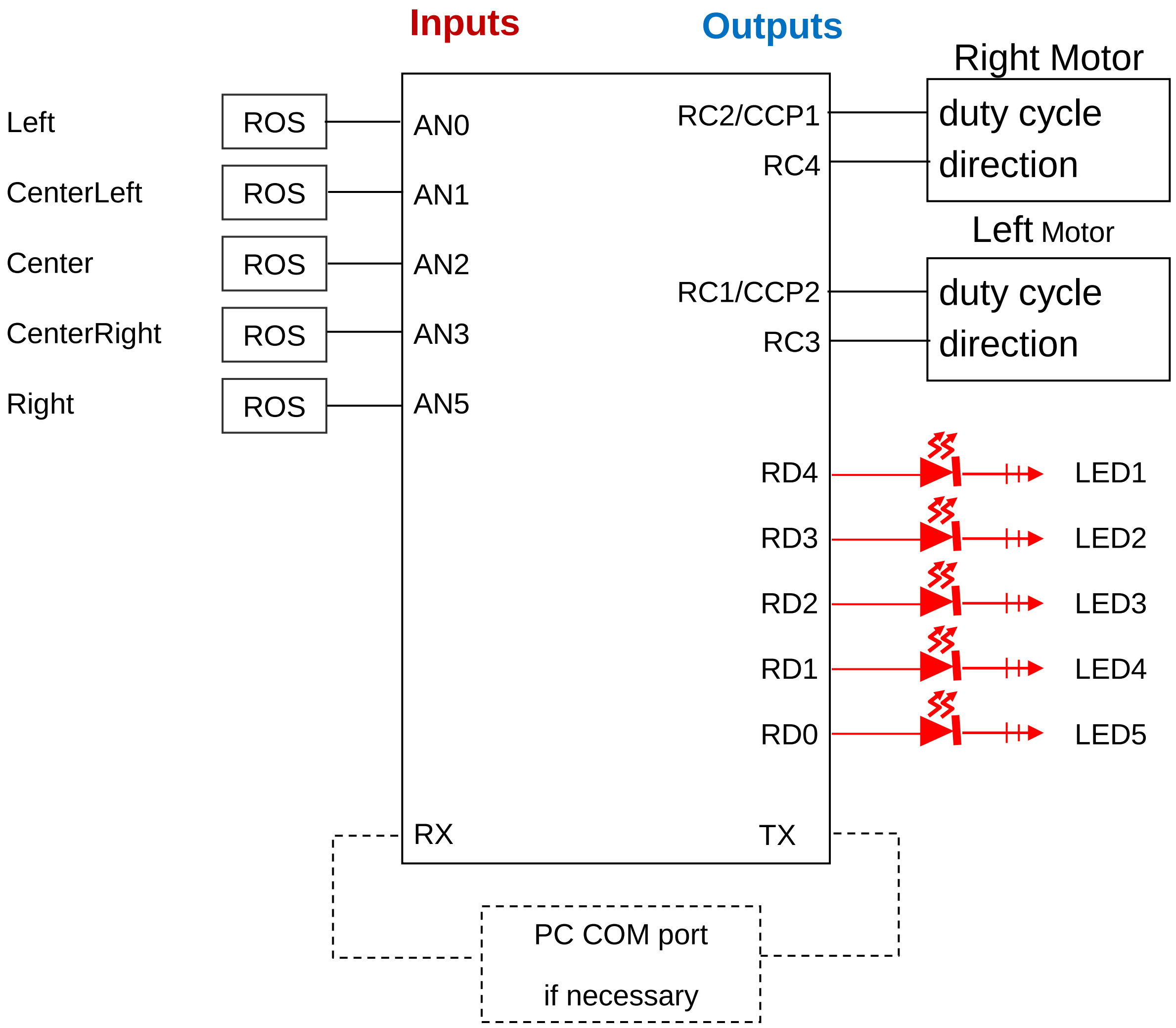
Sensors - ADC



Motor control:
Speed and Direction
PWM

LEDs: Indicators
DIO

Robot from above.



Robot pin connections.

Code

- Library (should not have to modify)
    - sumovore.h (many defines & some functions)
    - sumovore.c
    - interrupts.c & interrupt.h (don’t touch!)
- Example
    - main.c
    - motor_control.c (you work here)
    - motor_control.h

```
// main.c

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

int 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

check_sensors();

- Defined in sumovore.c
- ADC on the 5 IR sensors
- Compare value to *threshold*
- May need to tweak *threshold*
- Results stored as a group of 5 values in SeeLine.B (or indiviually SeeLine.b.position)

```
// from sumovore.h

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

union sensor_union
{ // so that B is restricted to 5 valid bits
  struct
     { unsigned B:5;
       unsigned :3; };
  struct sensors b; };

extern union sensor_union SeeLine;
```

```
//From 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 );
}
```

You may need to tweak `threshold` depending on track and lighting conditions.

set_leds();

From 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 inte
#define setLED5(a) PORTDbits.RD4=~a // but only the least significant
                                    // bit will be used.
```

From sumovore.c

```
void set_leds(void)  // SeeLine is a global variable
{ setLED1(SeeLine.b.Left);
  setLED2(SeeLine.b.CntLeft);
  setLED3(SeeLine.b.Center);
  setLED4 (SeeLine.b.CntRight);
  setLED5(SeeLine.b.Right);     }
```

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(); I

  if ( (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);
}
```

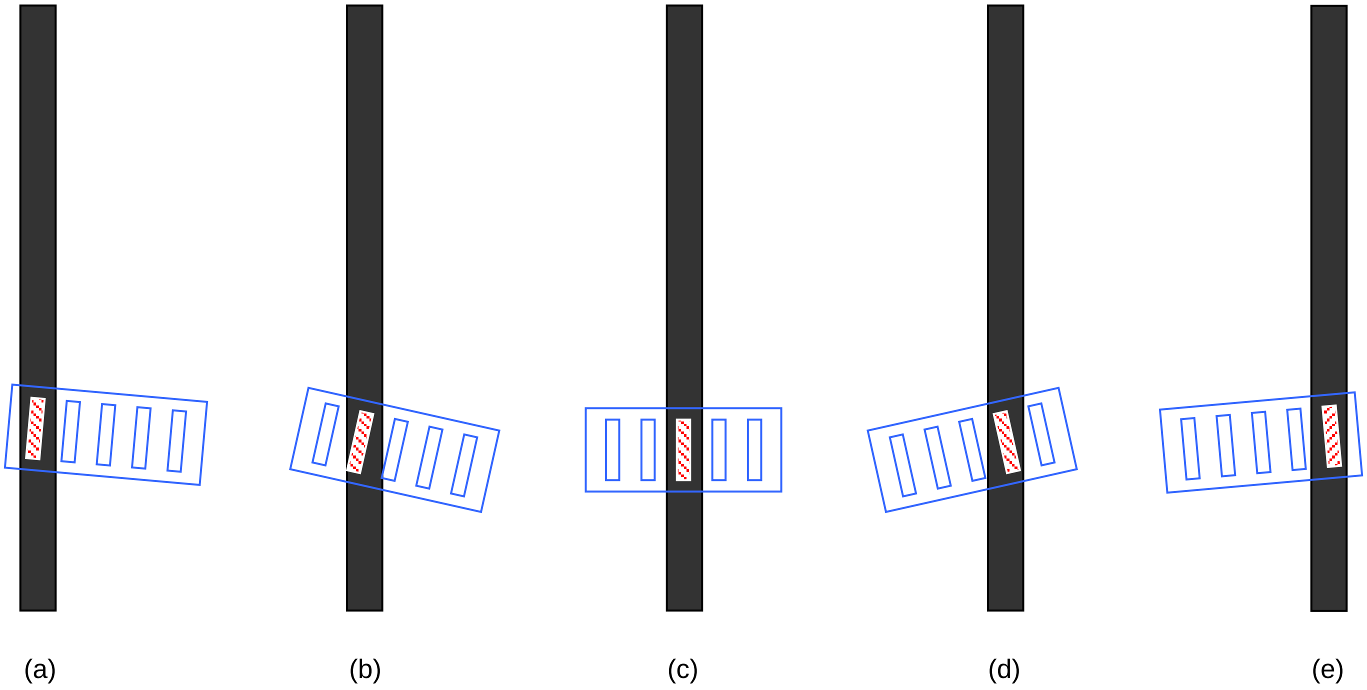
```
void spin_right(void)
{
set_motor_speed(left, fast, 0);
set_motor_speed(right, rev_fast, 0);
}

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

set_motor_speed(dir^{n},value,mod)

```
enum motor_selection { left, right };

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

void set_motor_speed(enum motor_selection
  the_motor, enum motor_speed_setting
  motor_speed, int speed_modifier);
```

```
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}; //DC 0 to 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; }

}  // end set_motor_speed
```

motor_control.c



Your Job

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