UNT Robocamp

SumoBot Instruction Manual

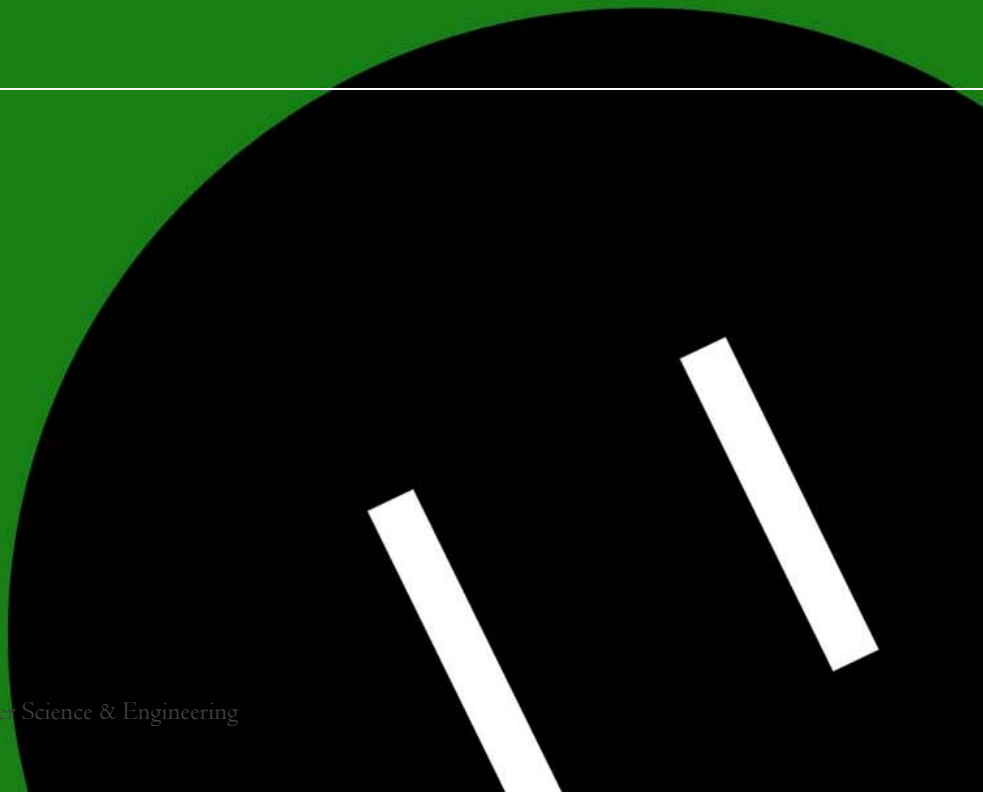
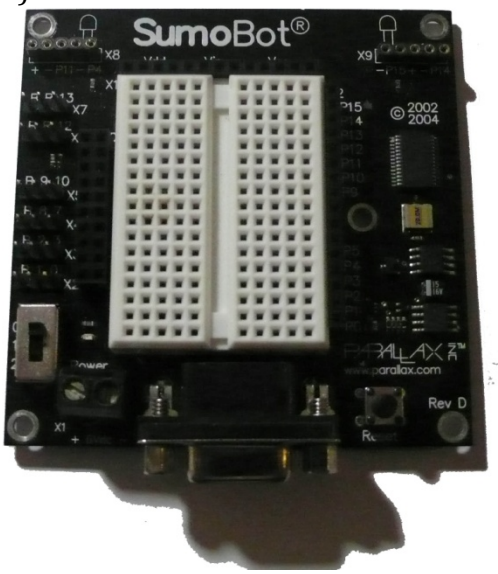


Table of Contents

Chapter 1
SumoBot Parts

Here is an overview of all the parts included in each SumoBot kit. Please take a moment to make sure you have all the right parts to build your SumoBot. You can also refer to this chapter when you are not sure which part the assembly instructions are talking about.

a) Sumo circuit board



b) (2) Servos

c) (2) Servo extension cables

d) (2) QTI circuit boards

e) (2) Wheels

f) (2) Rubber bands

g) Battery holder

h) Sumo chassis

i) Sumo front scoop

j) (12) 3/8” 4/40 pan head machine screws

k) (2) 3/8” 4/40 flathead screws

l) (12) 4/40 nut

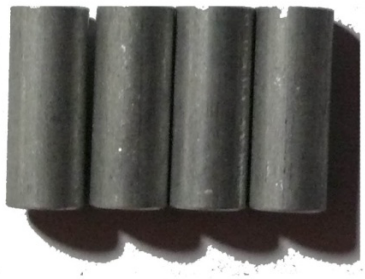
m) (2) 1” 4/40 pan head machine screws

n) (4) 1/4” 4/40 pan head machine screws

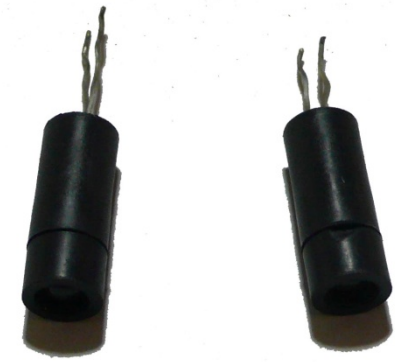
o) (2) Nylon washers

p) (2) 1.25” 4/40 standoffs

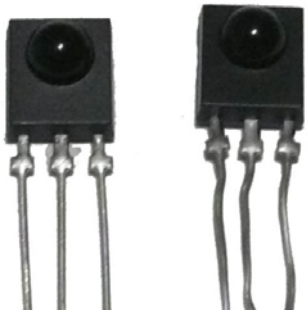
q) (4) 1/4" round 5/8” 4/40 standoffs



r) (2) Infrared LEDs with shield covers



s) (2) Infrared receivers



t) 470 ohm resistor

u) Red LED

v) (2) Jumper wires

w) Speaker attachment

Chapter 2
SumoBot Assembly

First things first, let's build your SumoBot. Remember that robotics, even on a small scale, is a serious endeavor and shouldn't be taken lightly. Patience is a virtue. Take the time to follow the construction steps carefully and you'll have your SumoBot running in no time.

Tools Required

All you will need for your robot is a screwdriver that a counselor will give you with your robot parts. All the parts needed for each step are listed below.

Step #1 – Install the Battery Box

- Battery box
- (2) 4/40 3/8" flat-head countersunk machine screws
- (2) 4/40 nuts
- SumoBot frame

Step #2 – Install the Servo Motors

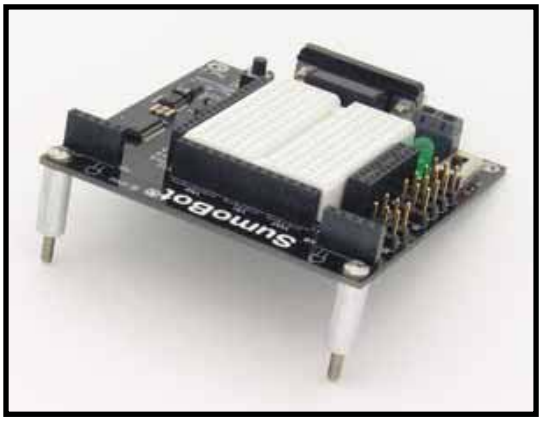
- (2) Parallax servos
- (8) 4/40 3/8" pan-head machine screws
- (8) 4/40 nuts
- SumoBot frame

Step #3 – Intall the Rear SumoBot PCB Standoffs

- (2) 5/8” round standoffs
- (2) 4/40 3/8” pan-head machine screws
- SumoBot frame

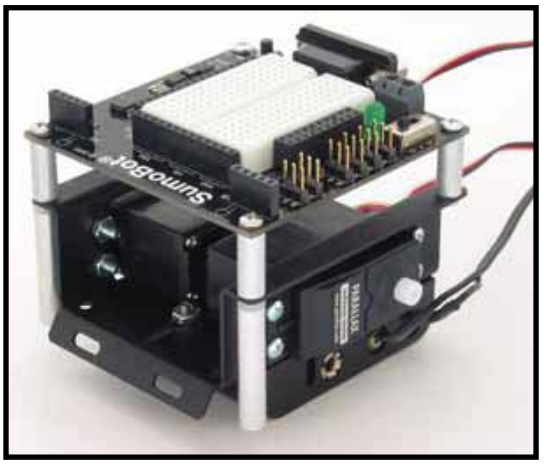
Step #4 – Install the Front SumoBot PCB Standoffs

- (2) 5/8” round standoffs
- (2) 4/40 1” pan-head screws
- SumoBot PCB



Step #5 – Mount the PCB

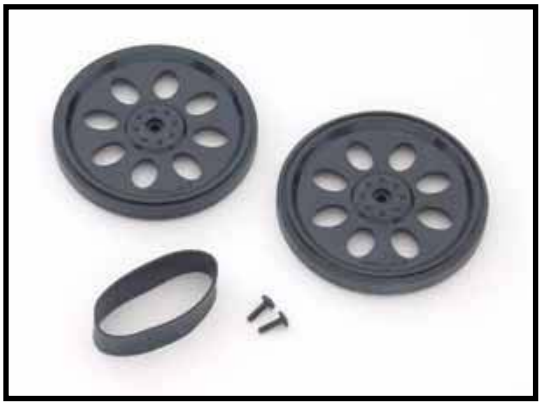
- SumoBot PCB
- (2) 4/40 3/8” pan-head machine screws
- (2) 1-1/4” round stand-offs
- (2) Nylon washers
- SumoBot frame



Step #6 – Prepare the Wheels

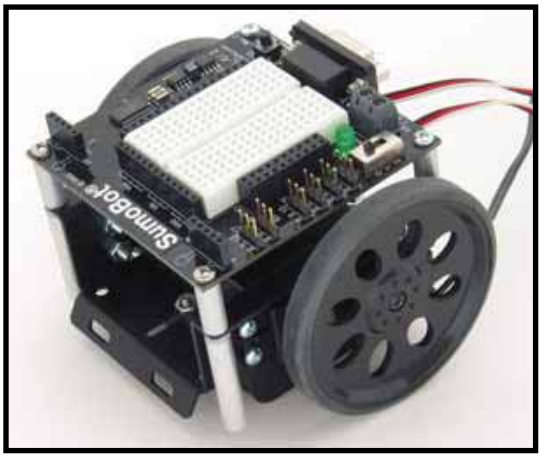
- (2) SumoBot wheels
- (2) SumoBot rubber tires

NOTE: If you can't get the rubber bands on the tires, feel free to ask a camp counselor.



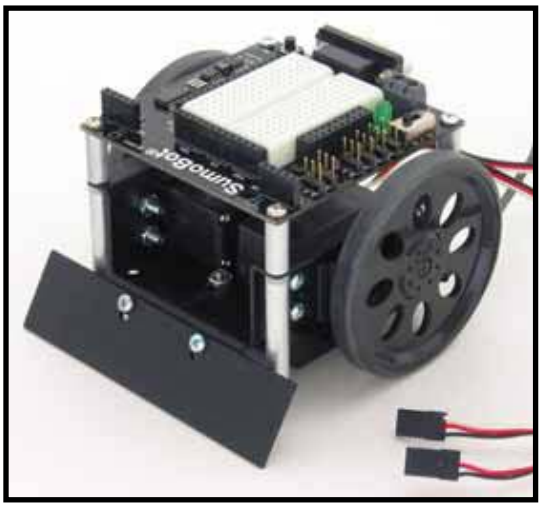
Step #7 – Mount the Wheels

- (2) Prepared wheels
- (2) Black servo-horn screws
- SumoBot frame



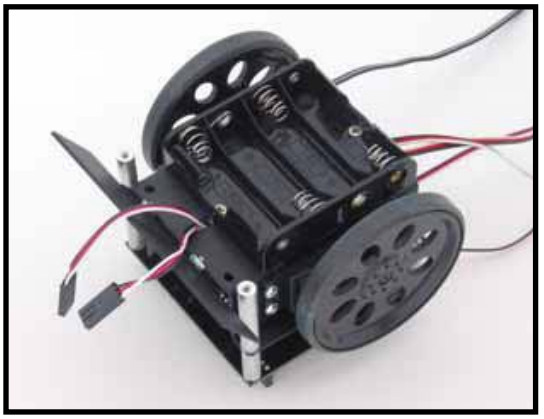
Step #8 – Mount the Scoop

- SumoBot scoop
- (2) 4/40 1/4" pan-head machine screws
- (2) 4/40 nuts
- SumoBot frame



Step #9 – Install Line Sensor Wires

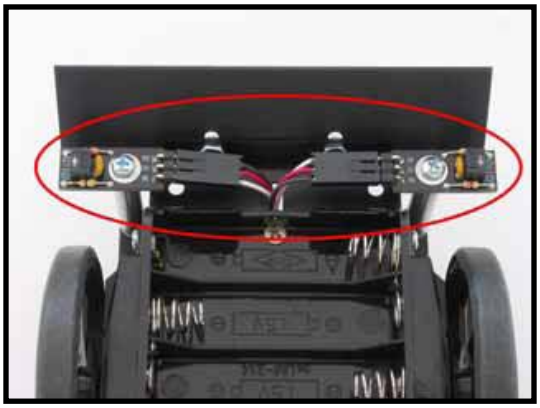
- (2) 10” 3-pin extension cables
- SumoBot frame



Step #10 – Install the QTI Line Sensors

- (2) QTI line sensors
- (2) 4/40 1/4” pan-head machine screws
- SumoBot frame

NOTE: When you connect the wires, make sure the pins are oriented so that the Black wire goes into the pin labeled “B”. If not done right, your line sensors will not work correctly.



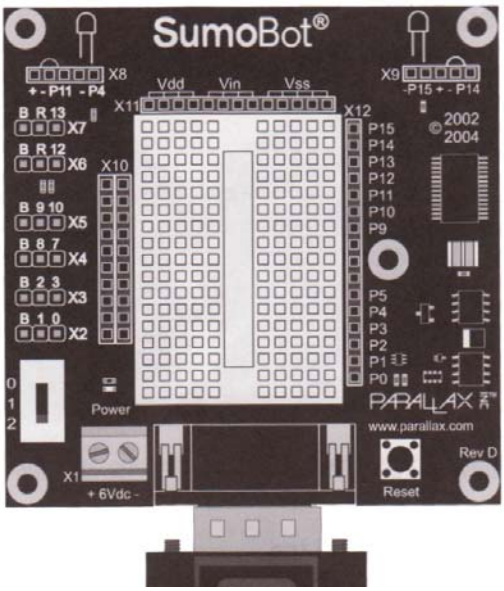
Step #11 – Make the Connections

Plug into the servo motors and QTI sensors into the SumoBot PCB connectors as indicated below:

X7 = Left Servo Motor
X6= Right Servo Motor
X5 – Left QTI Line Sensor
X4 = Right QTI Line Sensor

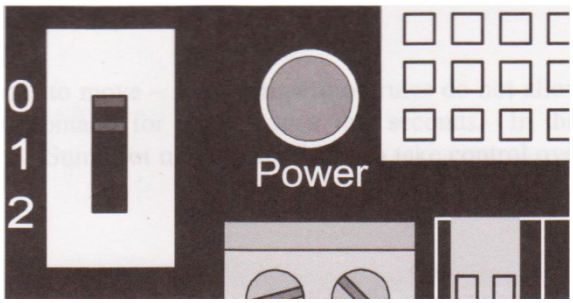
Connect the battery pack wires to the SumoBot PCB connector X1. The battery pack’s white-striped lead connects to the + terminal

NOTE: When plugging in your wires, make sure that you plug in the black wire to the pin that has a “B” above it; your robot won’t work right if you don’t do this correctly.



Step #12 – Power the SumoBot

The SumoBot PCB has a three-position power switch. The state of each position is shown below. The three-position switch has a middle position that powers the entire circuit except the servos.



Position 0 – No Power
Position 1 – Power PCB
Position 2 – Power PCB and Servos

Chapter 3
Intro to Coding the SumoBot

When you get to writing a program for your SumoBot, you will find that things are a lot easier if you follow the proper rules and syntax to make your program run. This section will go over some simple rules and terms on programming your SumoBot correctly.

Common Coding Terms

DEBUG – DEBUG is a method of writing output to the debug console in the BASIC Stamp Editor. It can output variables to show their values, or text inside sets of quotation marks "like this". You will see plenty of examples of how to use this command in the following programs.

Definitions - These are usually put at the beginning of your program to define names to parts of your SumoBot, or variables to perform other talks in your program, such as counting. If one of these is forgotten or mistyped, the program will create an error. In the chapters ahead, you will see the definitions split up into three sections for easy recognition: I/O Definitions, Constants, and Variables. **I/O Definitions** define a PIN number location of a part of your SumoBot to a name. **Constants** are assigned numbers that can be used instead of writing a number out in the program. It may seem like a waste of time, but it makes your code easier to read when you are trying to debug something. **Variables** are like constants but their number can be changed in the program dynamically, such as a counter.

END - You always need this at the end of your programs, but before the subroutines. The function of this is quite obvious, END tells the program when it is done.

Equals (=) - Equal signs can be used in two very different ways: as an assignment or as a conditional check. An equal sign is used as a conditional check only when it is used inside of an if/then block statement. You can check if a variable is equal to a constant value or another variable (ex: if(bob = 16) then... or if(bob = driving_age) then…). When the equal sign is used not used in an if/then block, it is used as an assignment. This means that you are storing a value into a variable (ex: life = 42).

GOTO – GOTO jumps to a part of the code defined by a name followed by a colon (*like_so:)*. These are often used for subroutines.

PAUSE – PAUSE is a command to stop your program for an specific amount of time assigned in milliseconds. For example, PAUSE 1000 would pause for one second.

PULSOUT – PULSOUT is a command to run the servo motors. This command is followed by a pin number, then a pulse amount for it to use. These amounts will be further elaborated in the following chapters.

Stamp Definition – These two lines at the top of every program indicate which coding terms and the program will use. These definitions go at the beginning of your program

before anything else. Without these, your code won't be able to get interpreted by the Stamp editor.

Punctuation is Key

When programming in any programming language, it is very important to write your code exactly as it is shown, comments not included, or else you will get errors and your program will not run. So when you write your code, pay attention to every capitalization, every space, every comma, don't leave anything out that isn't a comment. Please see the following examples:

'LMotor PIN 13' is not the same as *'lmotor pin 13'*
'Reset: ' is not the same as *'Reset; '*

Remember the Stamps

If you do not have these two statements at the beginning of your program, your code will not run:

' {$STAMP BS2}
' {$PBASIC 2.5}

If you have any questions or issues with your programs, feel free to ask one of the counselors for help.

Loops

DO Loops constantly repeat whatever is inside the loop without stopping; however, you could stop these by making a condition to jump into another part of your code. DO Loops begin by writing DO, and end by writing LOOP.
Example:

```
DO
     DEBUG "Stop hitting yourself!", CLREOL
     PAUSE 1000
LOOP
```

FOR Loops are like DO Loops, except that they repeat the inside of their loops according to a set amount of times that you assign to an integer range. FOR Loops begin by starting with FOR, and ending the loop with NEXT.
Example:

```
DEBUG "I can count this high!", CLREOL
FOR count = 1 TO 25
     DEBUG VAR count, CLREOL
     PAUSE 1000
NEXT
```

Chapter 4
SumoBot Locomotion

The first thing we want our SumoBots to do is to move. We will be using two Parallax Continuous Rotation servo motors.

Servos

When both motors are moving in the same direction, the SumoBot will move in that direction. When the SumoBot servo motors turn in different directions, the SumoBot will rotate. The rate of movement is determined by each motor's speed. The BASIC Stamp 2's PULSOUT command sends a pulse to the servo motor instructing it to move. The length of the pulse sent determines which direction and how fast the motor moves. We will be using three different PULSOUT numbers for all of our motor speeds:

- 650 makes the assigned servo go in reverse
- 750 puts the assigned servo in neutral
- 850 makes the assigned servo go forward

After sending the pulse, the servo expects a pause of about 20 milliseconds before the next pulse is sent, hence the PAUSE 20 command you will see in the short wheel alignment program below.

Wheel Alignment Program

To make sure your SumoBot moves correctly, we need to make sure the servo motors are centered. Open up the BASIC Stamp Editor and write the following program that will be used to align the SumoBot motors:

```
' SumoBot_2.1_Motor_Align.BS2
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----{I/O Definitions}--------------------------------------
LMotor          PIN 13              ' left servo motor
RMotor          PIN 12              ' right servo motor

' -----{Constants}-------------------------------------------
LStop           CON 750             ' left motor stop
RStop           CON 750             ' right motor stop

' -----{Initialization}--------------------------------------
Reset:                              ' initialize motor outputs
     LOW LMotor
     LOW RMotor

' -----{Program Code}-----------------------------------------
Main:
     DO
          PULSOUT LMotor, LStop     ' stop left
```

```
          PULSOUT RMotor, RStop    ' stop right
          PAUSE 20
     LOOP
END
```

Move the SumoBot power switch to position 1, and then download the code using the Run command from the Run menu, or by pressing the ▹ button on the toolbar. As soon as the program is downloaded, disconnect the SumoBot from the computer and switch it into position 2. If either motor turns, use your small screwdriver and adjust the centering potentiometer (Figure 2.1) until the motor stops.

Don't worry how the program works right now; that will become clear in due time.

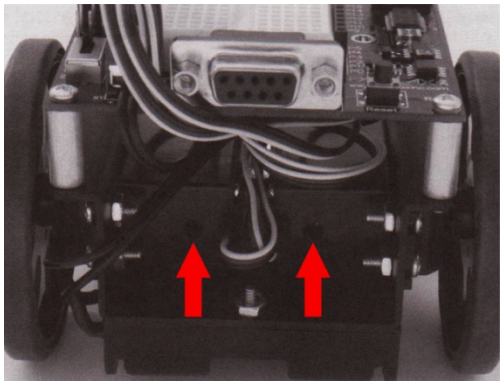


Figure 2.1

SumoBot Motion Test

With the motors aligned, we can now make a program to get your SumoBot moving and show you how the servos work when properly aligned. Write the following program into your BASIC Stamp Editor and download it to your SumoBot:

```
' SumoBot_2.2_Motor_Test.BS2
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----{I/O Definitions}-------------------------------------
LMotor          PIN 13              ' left servo motor
RMotor          PIN 12              ' right servo motor
' -----{Constants}-----------------------------------------
LFwd            CON 850             ' left motor fwd
LStop           CON 750             ' left motor stop
LRev            CON 650             ' left motor reverse
RFwd            CON 650             ' right motor fwd
RStop           CON 750             ' right motor stop
RRev            CON 850             ' right motor reverse
' -----{Variables}-----------------------------------------
pulses          VAR Byte            ' servo pulses counter
' -----{Initialization}--------------------------------------
Reset:                              ' initialize motor outputs
     LOW LMotor
     LOW RMotor
     PAUSE 2000                     ' pauses for 2 seconds

' -----{Program Code}----------------------------------------
Main:
     FOR pulses = 1 TO 65           ' move fwd
```

```
     PULSOUT LMotor, LFwd
     PULSOUT RMotor, RFwd
     PAUSE 20
     NEXT

     FOR pulses = 1 TO 30          ‘ pivot on left wheel
     PULSOUT LMotor, LStop
     PULSOUT RMotor, RFwd
     PAUSE 20
     NEXT

     FOR pulses = 1 TO 60          ‘ pivot 180 on right wheel
     PULSOUT LMotor, LFwd
     PULSOUT RMotor, RStop
     PAUSE 20
     NEXT

     FOR pulses = 1 TO 55          ‘ spin clockwise
     PULSOUT LMotor, LFwd
     PULSOUT RMotor, RRev
     PAUSE 20
     NEXT

     FOR pulses = 1 TO 55          ‘ spin counterclockwise
     PULSOUT LMotor, LRev
     PULSOUT RMotor, RFwd
     PAUSE 20
     NEXT

     FOR pulses = 1 TO 65          ‘ move reverse
     PULSOUT LMotor, LRev
     PULSOUT RMotor, RRev
     PAUSE 20
     NEXT

Hold_Position:
     DO
     PULSOUT LMotor, LStop
     PULSOUT RMotor, RStop
     PAUSE 20
     LOOP

END
```

After the program is downloaded, remove the cable from the SumoBot and flip the switch to position 2 on your SumoBot. This program will run the robot though all of the key motions it can perform and then stops. Be sure to switch it to position 0 (off) when it is

finished. If it doesn't move at all, you may have set the power to position 1 or it didn't have the program uploaded onto it properly. If you are having trouble, flag down a counselor.

Chapter 5
Line Sensors and Border Detection

Now that we have the SumoBot moving, the next task is to program the line sensors, called QTIs, to scan the playing surface so that it doesn't drive itself out of the ring. The QTI uses a reflective infrared sensor to allow the SumoBot to scan for the ring's border.

Line Sensor Test

We need to make a program to test our line sensors. This program shows output from the SumoBot that will display the changes in the values for the infrared sensors. These values will determine if the SumoBot is about to be out of the ring, and will attempt to fix this if it happens in your competition program. Write and run this program to test and evaluate the QTI sensors:

```
' SumoBot_3.1_Line_Sensor_Test.BS2
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----{I/O Definitions}-------------------------------------
LLinePwr        PIN  10             ' left line sensor power
LLineIn         PIN  9              ' left line sensor input
RLinePwr        PIN 7               ' right line sensor power
RLineIn         PIN 8               ' right line sensor input

 ' -----{Variables}-----------------------------------------
lLine           VAR  Word           ' left sensor raw reading
rLine           VAR  Word           ' right sensor raw reading

 ' -----{Program Code}---------------------------------------
Main:
     DO
     GOSUB Read_Left
     GOSUB Read_Right

     DEBUG HOME,
          "Left ", TAB, "Right", CR,
          "-----", TAB, "-----", CR,
          DEC lLine, CLREOL, TAB, DEC rLine, CLREOL
     PAUSE 100
     LOOP
     END
' -----{Subroutines}----------------------------------------
Read_Left:
     HIGH LLinePwr                ' activate sensor
     HIGH LLineIn                 ' discharge QTI cap
     PAUSE 1
     RCTIME LLineIn, 1, lLine     ' read sensor value
```

```
     LOW LLinePwr                   ‘ deactivate sensor
     RETURN

Read_Right:
     HIGH RLinePwr                  ‘ activate sensor
     HIGH RLineIn                   ‘ discharge QTI cap
     PAUSE 1
     RCTIME RLineIn, 1, rLine       ‘ read sensor value
     LOW RLinePwr                   ‘ deactivate sensor
     RETURN
```

This program includes the first occurrence of a subroutine. Subroutines are easy ways to repeat pieces of code many times without having to write it again each time. Subroutines are written after the end of the code and start with the name of the subroutine followed by a colon (:) and ended with a RETURN statement.

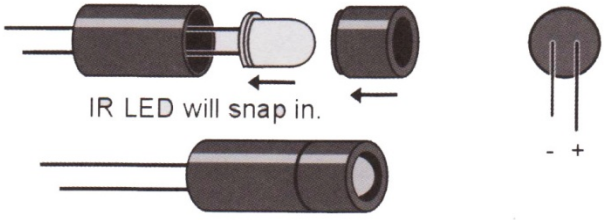
After uploading your code to your SumoBot, keep the cable in and run the code. You will see an output window pop up. The window will display your “Left” and “Right” sensors along with a value ranging between high and low numbers. These numbers are the frequency ranges for the infrared sensors. Move the SumoBot between a white surface and a black surface, the numbers for each sensor should move up and down respectfully. If they don’t both function properly, make sure your cables for the sensors are in the correct pins. If you are still having trouble, ask one of the counselors for help.

Chapter 6
Infrared Headlights and Object Detection

The BASIC Stamp can use infrared LEDs and detectors to detect objects to the front and side of your SumoBot. This is accomplished by the SumoBot shining an invisible path of infrared light ahead of it and determining when the light reflects off an object. These will be like your SumoBot's headlights, letting it see and react accordingly to its surroundings.

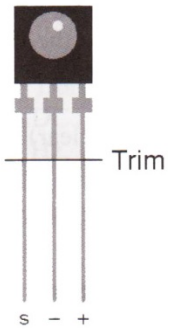
Figure 4.1 shows the assembly for the IR LEDs into their protective shells. The shells are important because they prevent stray IR light from falling directly onto the detector and causing false input.

Figure 4.1: IR LED, Standoff, and Shield Assembly

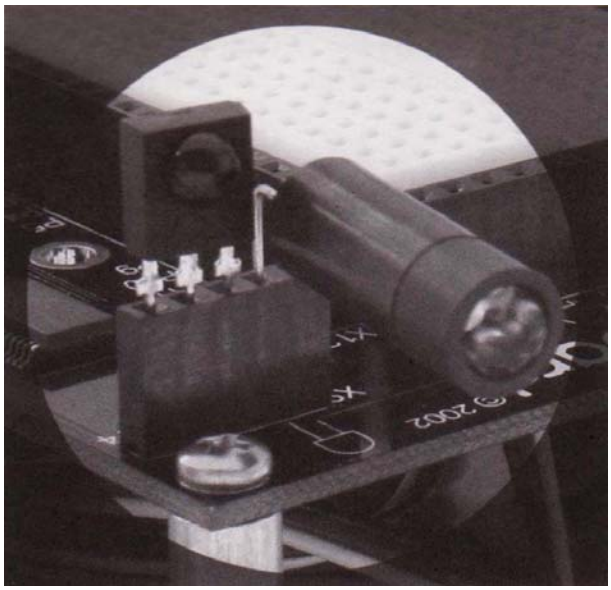


After assembly, bend the wires downward at a 90 degree angle so that when looking at the back side of the shield, the positive (longer) wire is on the right. Figure 4.2 shows the IR detector; make sure the ball on the IR detector is pointing the same way as the IR LED or else it will not gather the proper input.

Figure 4.2: IR Detector Trimming



Your IR assembly should look something like this when done:



Infared Test Program

The purpose of this program is to use your IR assembly to detect and react to objects that you put in front of it. Write this and download it into your SumoBot:

```
' SumoBot_4.1_Infrared_Test_Program.BS2
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----{I/O Definitions}-------------------------------------
LfIrOut             PIN  4          ' left IR LED output
LfIrIn              PIN  11         ' left IR sensor input
RtIrOut             PIN  15         ' right IR LED output
RtIrIn              PIN  14         ' right IR sensor input

' -----{Variables}-----------------------------------------
irBits              VAR  Nib        ' storage for IR target data
irLeft              VAR  irBits.BIT1
irRight             VAR  irBits.BIT0

' -----{Program Code}----------------------------------------

Main:
  DO
     FREQOUT LfIrOut, 1, 38500
     irLeft = ~LfIrIn
```

```
    FREQOUT RtIrOut, 1, 38500
    irRight = ~RtIrIn

    DEBUG HOME,
         "L   R", CR,
         "-----", CR,
         BIN1 irLeft, "   ", BIN1 irRight
    PAUSE 20
  LOOP
END
```

Keep the cable in your robot and place your hand in front of each sensor to check if the numbers change. If either of them is not reacting, check your IR assembly to make sure the LED is correctly installed. If you need any further help, ask a counselor for assistance. Now that we have made sure all the parts of your SumoBot are working properly, we can finally get your robot competing against other robots!

Chapter 7
LEDs and Speakers

There are two accessories you can add to your SumoBot light emitting diodes, LEDs, and speakers. LEDs are mostly just for fun, but you can play notes with a speaker.

LEDs

LEDs can be added onto your SumoBot for a little extra zazz. They really don't have too much application but to look pretty, but you can add them as a visual countdown to starting your robot, or to flash when they react to something. Installation is pretty easy, here is how:

Installing an LED

- LED
- 470 ohm resistor
- Jumper wire

After you install the LED, you can test it by adding the LED's pin number location (P0 in this case, according to the picture) to your code and writing an on/off command for it. Here is the definition you will need to add, and a small example for a five second delay counter.

```
' -----{I/O Definitions}-------------------------------------
LED       PIN 0
' -----{Variables}-------------------------------------
pulses    VAR  Byte

Main:

Start_Delay:
     FOR pulses = 1 TO 5
          HIGH LED
          PAUSE 500
          LOW LED
          PAUSE 500
     NEXT
END
```

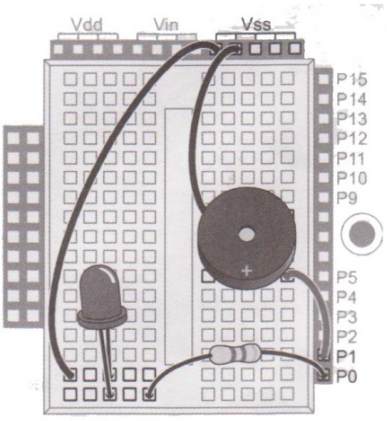
Some of the programs we will do later use a second LED. Try to install a second one by yourself. Use the example above to connect up the first LED on the left and a second one on the right. Connect the second LED to pin P2.

Speakers

The speaker outputs an assigned frequency for a set amount of time. Unfortunately, you can't make two speakers play at the same time. However, you can make simple songs or warning noises for your SumoBot.

Installing a Speaker

- Speaker
- (2) Jumper wires



After you install the speaker, you can test it by adding the speaker's pin number location (P1 in this case, according to the picture) to your code and writing a FREQOUT command for it. The format for a FREQOUT command is as follows:

```
' -----{I/O Definitions}-------------------------------------
Speaker   PIN 1

' FREQOUT (definition name), (length in millisecs), (Frequency)

FREQOUT Speaker, 500, 344
```

Chapter 8
Competition Code

It's time to get your robot ready to rumble! The program in this chapter brings all the parts together, and adds some intelligence for fighting against your opponents. This program is technically correct, but the behavior of your robot will be slightly "off". There are some logic problems that give your robot a low IQ, but everyone else's robots will have the same program.

HINT: Once you are done typing, look through the code and find where to change the robot's behavior. Think of what you would do as a sumo wrestler in certain situations and change the code accordingly to make a more appropriate behavior. This will dramatically boost your robot's IQ – after all, you are much smarter than it.

Sorry, but there's plenty of typing ahead.

```
' SumoBot_5.1_Competition_Code_1.0.BS2
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----{I/O Definitions}-------------------------------------

LMotor    PIN 13
RMotor    PIN 12
LLinePwr  PIN 10
LLineIn   PIN 9
RLinePwr  PIN 7
RLineIn   PIN 8
LfIrOut   PIN 4
LfIrIn    PIN 11
RtIrOut   PIN 15
RtIrIn    PIN 14
StartLED  PIN 0
SpeakerPin PIN 1

' -----{Constants}------------------------------------
SlowCon   CON 30
FastCon   CON 250
StopCon   CON 750
LFwdFast  CON StopCon + FastCon
LFwdSlow  CON StopCon + SlowCon
LStop     CON StopCon
LRevSlow  CON StopCon - SlowCon
LRevFast  CON StopCon - FastCon
RFwdFast  CON StopCon - FastCon
RFwdSlow  CON StopCon - SlowCon
RStop     CON StopCon
RRevSlow  CON StopCon + SlowCon
RRevFast  CON StopCon + FastCon
```

```
' -----{Variables}--------------------------------------
lLine     VAR   Word
rLine     VAR   Word
lineBits  VAR   Nib
uturncounter   VAR   Nib
lbLeft    VAR   lineBits.BIT1
lbRight   VAR   lineBits.BIT0
pulses    VAR   Byte
temp      VAR   Byte
counter   VAR   Byte
irBits    VAR   Nib
irLeft    VAR   irBits.BIT1
irRight   VAR   irBits.BIT0

' -----{EEPROM Data}-------------------------------------
RunStatus       DATA  $00 ' run status

' -----{Initializations}---------------------------------
Reset:      ' Wait for you to press the RESET button
  READ RunStatus, temp
  temp = ~temp
  WRITE RunStatus, temp
  IF (temp > 0) THEN GOTO Done
  LOW LMotor
  LOW RMotor

Start_Delay:
FOR pulses = 1 TO 5
  HIGH StartLED
  PAUSE 950
  LOW StartLED
  FREQOUT SpeakerPin, 50, 3000
NEXT

' -----{Main}-------------------------------------
Start:      ' Any starting behavior goes here and runs once
  GOSUB Lunge
  GOSUB Lunge

  GOTO Main

Main:
  GOSUB Read_Line_Sensors  ' Results stored in lineBits

  IF lineBits = %00 THEN ' I dont see anything below, go ahead
    GOTO look_ahead
```

```
  ELSEIF lineBits = %01 THEN ' I see the line with my right
sensor!
    GOSUB Spin_Right
    GOSUB Backup
    GOSUB Backup
  ELSEIF lineBits = %10 THEN ' I see the line with my left
sensor!
    GOSUB Lunge
    GOSUB Lunge
  ELSE ' lineBits = %11   OMGZ, I see the line with both
sensors, EVADE!

    GOSUB Backup
    GOSUB Backup
    GOSUB Backup
    GOSUB Rotate180
  ENDIF

Look_Ahead:
  GOSUB Read_IR_Sensors       ' Results stored in irBits

  IF irBits = %11 THEN        ' I see something ahead!
      GOSUB lunge
      'DEBUG "lunge", CR
    ELSEIF irBits = %10 THEN  ' I see something to the left!
      GOSUB spin_left
      'DEBUG "spinning left, ir", CR
    ELSEIF irBits = %01 THEN  ' I see something to the right!
      GOSUB spin_right
      'DEBUG "spinning right, ir", CR
    ELSE        ' I don't see anything at all, walk forward
      GOSUB creep_forward
    ENDIF
    GOTO Main
END ' end of Main

' -----{Subroutines}-------------------------------------
Read_Line_Sensors:
  HIGH  LLinePwr
  HIGH  RLinePwr
  HIGH  LLineIn
  HIGH  RLineIn
  PAUSE 1
  RCTIME LLineIn, 1, lLine
  RCTIME RLineIn, 1, rLine
  LOW LLinePwr
  LOW RLinePwr
```

```
  'convert readings to bits
  LOOKDOWN lLine, >=[500,0], lbLeft
  LOOKDOWN rLine, >=[500,0], lbRight
  'DEBUG HOME, "L  R", CR, "-----", CR, BIN1 lbleft, "  ", BIN1
  'lbright
  RETURN

Read_IR_Sensors:
  FREQOUT LfIrOut, 1, 38500
  irLeft = ~LfIrIn
  FREQOUT RtIrOut, 1, 38500
  irRight = ~RtIrIn
  'DEBUG HOME, "L  R", CR, "----", CR, BIN1 irLeft, "  ", BIN1
  'irRight

  RETURN

Look_Right:
  FOR pulses = 1 TO 5
    PULSOUT  LMotor, LFwdSlow
    PAUSE 20
  NEXT
  RETURN

Look_Left:
  FOR pulses = 1 TO 5
    PULSOUT  RMotor, RFwdSlow
    PAUSE 20
  NEXT
  RETURN

Creep_Forward:
  FOR pulses = 1 TO 5
    PULSOUT LMotor, LFwdSlow
    PULSOUT RMotor, RFwdSlow
    PAUSE 20
  NEXT
  RETURN

Lunge:
  FOR pulses = 1 TO 10
    PULSOUT  LMotor, LFwdFast
    PULSOUT  RMotor, RFwdFast
    PAUSE 20
  NEXT
```

```
  RETURN

Backup:
  FOR pulses = 1 TO 10
    PULSOUT  LMotor, LRevFast
    PULSOUT  RMotor, RRevFast
    PAUSE 20
  NEXT
  RETURN

Spin_Left:
  FOR pulses = 1 TO 10
    PULSOUT  LMotor, LRevFast
    PULSOUT  RMotor, RFwdFast
    PAUSE 20
  NEXT
  RETURN

Spin_Right:
  FOR pulses = 1 TO 10
    PULSOUT  LMotor, LFwdFast
    PULSOUT  RMotor, RRevFast
    PAUSE 20
  NEXT
  RETURN

Rotate180:
  FOR pulses = 1 TO 30
    PULSOUT  LMotor, LFwdFast
    PULSOUT  RMotor, RRevFast
    PAUSE 20
  NEXT
  RETURN

Dosomething:
  ' program whatever you want here.

  RETURN

Done:

END
```

There you have it, after you upload your code onto your SumoBot; your robot will be ready to fight in the ring! Just no cheating of any kind, and try not to kick them around or drop them – keep those battle rings on the floor! You have learned everything necessary to fight;

however, there are additional activities in the next few chapters, such as adding lights onto your bot, if you are interested in making your robot cooler.

Chapter 9
Robo Art

There are plenty of artistic things you can do with your SumoBot. If you want to win one of our fabulous prizes though, you may want to think outside of the box and be a little bit more creative. Instead of drawing some simple spirals, try spelling a word. Or if you want to draw something a bit more complex, try drawing a character or maybe some kind of scene, like the kind you used to draw in kindergarten to get hung up on the refrigerator door. Unfortunately for the songs, you cannot play more than one note at a time, but maybe you can combine both a song and drawing? Or if we have the resources, you could use two SumoBots and do a song in harmony! The only limit is you.

Drawing

There are plenty of artistic things you can do with your SumoBot. Remember though, if you want to win, be creative. Drawing with your SumoBot is very simple; tape a marker to a side of the bot so that the tip is touching the ground. Make sure that the marker isn't pressing too hard on the ground, because this can make your SumoBot's movements slower or impaired. When you want to program what your bot will draw, consider making subroutines for each kind of movement you would need so that you can easily map out the pattern your SumoBot will draw. Subroutines could be things like make a circle, rotate left 180 degrees, move forward one second, or hold position for 5 seconds. Refer to the competition code in Chapter 5 for subroutine examples. Here's a simple Spirograph example to try out:

```
' SumoBot_7.1_Drawing.BS2
' {$STAMP BS2}
' {$PBASIC 2.5}

' -----{I/O Definitions}-------------------------------------
LMotor          PIN 13              ' left servo motor
RMotor          PIN 12              ' right servo motor

' -----{Constants}-----------------------------------------
LFwd            CON 850             ' left motor fwd
LStop           CON 750             ' left motor stop
LRev            CON 650             ' left motor reverse
RFwd            CON 650             ' right motor fwd
RStop           CON 750             ' right motor stop
RRev            CON 850             ' right motor reverse

' -----{Variables}-----------------------------------------
pulses          VAR Byte            ' servo pulses counter

' -----{Initialization}--------------------------------------
Reset:                              ' initialize motor outputs
     LOW LMotor
     LOW RMotor
```

```
     PAUSE 2000                    ‘ pauses for 2 seconds

‘ -----{Program Code}----------------------------------------
Main:
DO
     GOSUB Spin360
     GOSUB ForwardaBit
LOOP
END

ForwardaBit:
     FOR pulses = 1 TO 15          ‘ move fwd slightly
     PULSOUT LMotor, LFwd
     PULSOUT RMotor, RFwd
     PAUSE 20
     NEXT
     RETURN

Spin360:
     FOR pulses = 1 TO 130         ‘ pivot ~360 on right wheel
     PULSOUT LMotor, LFwd
     PULSOUT RMotor, RStop
     PAUSE 20
     NEXT
     RETURN
```

Songs

Songs are quite amusing to program on SumoBots. To do this, you will need to install a speaker, as explained in Chapter 6. These programs can be quite tedious, but very fun. It is very easy to think up a song to imitate, but programming it can be a little bit more difficult. You will have to use the FREQOUT command, again, described in Chapter 6, to program every single note. Remember that the speakers can only play one note at a time, so harmony is very hard to do. Songs take a lot of patience but are very rewarding. You could consider writing subroutines for repeating parts of songs to slim the contents of your program down a bit. Here is an example of a two note song:

```
‘ SumoBot_7.2_Two_Note_Song.BS2

‘ {$STAMP BS2}
‘ {$PBASIC 2.5}

‘ -----{I/O Definitions}-------------------------------------
Speaker   PIN 1
```

```
Main:
     DO
          FREQOUT Speaker, 2000, 1047   'Play C in the 6th octave
          PAUSE 20
          FREQOUT Speaker, 500, 3520    'Play A in the 7th octave
          PAUSE 1000
     LOOP
END
```

Note	Frequency
C_6	1047
$C^{\#}_6/D^{b}_6$	1109
D_6	1175
$D^{\#}_6/E^{b}_6$	1245
E_6	1319
F_6	1397
$F^{\#}_6/G^{b}_6$	1480
G_6	1568
$G^{\#}_6/A^{b}_6$	1661
A_6	1760
$A^{\#}_6/B^{b}_6$	1865
B_6	1976

Note	Frequency
C_7	2093
$C^{\#}_7/D^{b}_7$	2217
D_7	2349
$D^{\#}_7/E^{b}_7$	2489
E_7	2637
F_7	2794
$F^{\#}_7/G^{b}_7$	2960
G_7	3136
$G^{\#}_7/A^{b}_7$	3322
A_7	3520
$A^{\#}_7/B^{b}_7$	3729
B_7	3951

The tables above are a conversion chart that shows the frequency that corresponds to a note in a particular octave. The higher the frequency, the higher pitch the note will be. For more note conversions or more on the science and math of how we got these numbers, visit the following website:

http://www.phy.mtu.edu/~suits/notefreqs.html

Chapter 10
Additional Exercises

Here are some additional exercises to try out if you have extra time on your hands. The The Line Follow exercise helps you understand how to program the line sensors and the Object Seeker exercise can help you with some more advanced strategies for the SumoBot Competition.

Line Follow

Now that you have a little programming experience under your belt, this program has some subroutine calls left out. It is up to you to decide how your robot should move to properly follow a line. This exercise is for using the QTI sensors on the bottom of your SumoBot to detect the colors of the SumoBot ring and move within its borders. Subroutines that move your robot are already defined, you just need to call them properly to get your robot to follow that line.

```
‘ SumoBot_Line_Follow.BS2
' {$STAMP BS2}
' {$PBASIC 2.5}

LMotor  PIN 13
RMotor  PIN 12
LLinePwr  PIN 10
LLineIn   PIN 9
RLinePwr  PIN 7
RLineIn   PIN 8

lLine     VAR Word
rLine     VAR Word
lineBits  VAR Nib
lbLeft    VAR lineBits.BIT1
lbRight   VAR lineBits.BIT0

LFwdFast  CON 1000
LStop     CON 750
LRevFast  CON 500
RFwdFast  CON 500
RStop     CON 750
RRevFast  CON 1000

counter VAR Byte

Reset:
  LOW LMotor
  LOW RMotor

Main:
```

```
  DO
    GOSUB Read_Line_Sensors    'left sensor---------right sensor
                               '    0                   0
    IF lbLeft = 0 AND lbRight = 0 THEN
     ‘ gosub to a subroutine here
                               '    1                   0
    ELSEIF lbLeft = 1 AND lbRight = 0 THEN
     ‘ gosub to a subroutine here

                               '    0                   1
    ELSEIF lbLeft = 0 AND lbRight = 1 THEN
      ‘ gosub to a subroutine here

                               '    1                   1
    ELSEIF lbLeft = 1 AND lbRight = 1 THEN
     ‘ gosub to a subroutine here
    ENDIF
  LOOP
END

' [ Subroutines ]---------------------------------

Halt:
  FOR counter = 0 TO 5
    PULSOUT LMotor, LStop
    PULSOUT RMotor, RStop
    PAUSE 20
  NEXT
  RETURN

Spin_Left:
  FOR counter = 0 TO 5
    PULSOUT LMotor, LFwdFast
    PULSOUT RMotor, RRevFast
    PAUSE 20
  NEXT
  RETURN

Spin_Right:
  FOR counter = 0 TO 5
    PULSOUT LMotor, LRevFast
    PULSOUT RMotor, RFwdFast
    PAUSE 20
  NEXT
  RETURN

backup:
```

```
  FOR counter = 0 TO 5
    PULSOUT LMotor, LRevFast
    PULSOUT RMotor, RRevFast
    PAUSE 20
  NEXT
  RETURN

Forward:
  FOR counter = 0 TO 5
    PULSOUT LMotor, LFwdFast
    PULSOUT RMotor, RFwdFast
    PAUSE 20
  NEXT
  RETURN

Read_Line_Sensors:
  HIGH LLinePwr
  HIGH RLinePwr
  HIGH LLineIn
  HIGH RLineIn
  PAUSE 1
  RCTIME LLineIn, 1, lLine
  RCTIME RLineIn, 1, rLine
  LOW LLinePwr
  LOW RLinePwr
  LOOKDOWN lLine, >=[1000, 0], lbLeft
  LOOKDOWN rLine, >=[1000, 0], lbRight
  RETURN
```

Object Seeker

This program will use the infrared LEDs on the front of the robot to detect things ahead of it and try to ram into them. If it detects an object to the left, it will move left. To the right, it will move right. It will be on a never-ending search till the end of time, or until you turn it off. This program is similar to that of the line following exercise. I have removed the subroutine calls to get the robot to move towards an object. It is up to you to decide how your robot should move when it sees something in front of it with the front IR sensors.

```
‘ SumoBot_Object_Seeker.BS2
‘ {$STAMP BS2}
‘ {$PBASIC 2.5}

‘ -----{I/O Definitions}-------------------------------------

LMotor    PIN 13
RMotor    PIN 12
LLinePwr  PIN 10
```

```
LLineIn   PIN 9
RLinePwr  PIN 7
RLineIn   PIN 8
LfIrOut   PIN 4
LfIrIn    PIN 11
RtIrOut   PIN 15
RtIrIn    PIN 14
LeftLED   PIN 0
RightLED  PIN 2


‘ -----{Constants}-------------------------------------
SlowCon   CON 30
FastCon   CON 250
StopCon   CON 750
LFwdFast  CON StopCon + FastCon
LFwdSlow  CON StopCon + SlowCon
LStop     CON StopCon
LRevSlow  CON StopCon - SlowCon
LRevFast  CON StopCon - FastCon
RFwdFast  CON StopCon - FastCon
RFwdSlow  CON StopCon - SlowCon
RStop     CON StopCon
RRevSlow  CON StopCOn + SlowCon
RRevFast  CON StopCon + FastCon

‘ -----{Variables}-------------------------------------
lLine     VAR   Word
rLine     VAR   Word
lineBits  VAR   Nib
pulses    VAR   Byte
irBits    VAR   Nib
irLeft    VAR   irBits.BIT1
irRight   VAR   irBits.BIT0

‘ -----{Initializations}-------------------------------------
Reset:
  LOW LMotor
  LOW RMotor

‘ -----{Main}-------------------------------------
Main:

DO
     GOSUB Read_IR_Sensors    ‘get information from sensors
     GOSUB look_ahead         ‘decide what to do with that info
LOOP
```

```
END

‘ -----{Subroutines}-------------------------------------

  look_ahead:
    IF irBits = %11 THEN        ' I see something ahead!
     ‘ gosub to a subroutine here

    ELSEIF irBits = %10 THEN  ' I see something to the left!
     ‘ gosub to a subroutine here

    ELSEIF irBits = %01 THEN  ' I see something to the right!
     ‘ gosub to a subroutine here

    ELSE        ' I don’t see anything at all
     ‘ gosub to a subroutine here

    ENDIF
  RETURN

Read_IR_Sensors:
  FREQOUT LfIrOut, 1, 38500
  irLeft = ~LfIrIn
  FREQOUT RtIrOut, 1, 38500
  irRight = ~RtIrIn
  'DEBUG HOME, "L  R", CR, "----", CR, BIN1 irLeft, "  ", BIN1
irRight

‘ -----{Display IR Status}-------------------------------------
  IF irLeft = 1 THEN
    HIGH LeftLED
  ELSE
    LOW LeftLED
  ENDIF

  IF irRight = 1 THEN
    HIGH RightLED
  ELSE
    LOW RightLED
  ENDIF
  RETURN


creep_forward:
  FOR pulses = 1 TO 5
    PULSOUT LMotor, LFwdSlow
    PULSOUT RMotor, RFwdSlow
```

```
    PAUSE 20
  NEXT
  RETURN

lunge:
  FOR pulses = 1 TO 2
    PULSOUT  LMotor, LFwdFast
    PULSOUT  RMotor, RFwdFast
    PAUSE 20
  NEXT
  RETURN

Spin_Left:
  FOR pulses = 1 TO 10
    PULSOUT  LMotor, LRevFast
    PULSOUT  RMotor, RFwdFast
    PAUSE 20
  NEXT
  RETURN

Spin_Right:
  FOR pulses = 1 TO 10
    PULSOUT  LMotor, LFwdFast
    PULSOUT  RMotor, RRevFast
    PAUSE 20
  NEXT
  RETURN
```