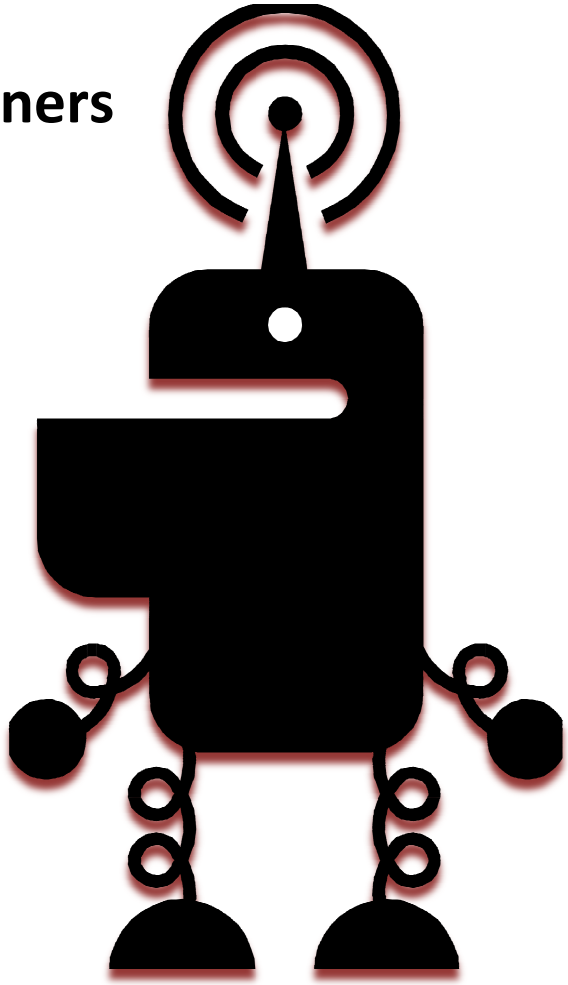
Workshop – **Robots for Beginners**

Version 1.0

Summer, 2011



Based on the Parallax S2 Robot

Neil@vectorr.com 828-484-4444
Weaverville, NC

This workshop is based on the Scribbler 2 robot and programming tools which are the intellectual property of Parallax, Inc. to whom I give great thanks. No special rights to their materials is expressed nor implied by use or dissemination of this workshop.

When in doubt, contact the author:

Neil Rosenberg
106 Thistle Tree Lane
Weaverville, NC 28787
828-484-4444
Neil@vectorr.com

Thanks, and ENJOY!

Workshop – Robots for Beginners

Beep!

Welcome to the "**Robots for Beginners**" workshop, a fun hands-on learning experience. You decide how fast to proceed, and you'll have plenty of chances along the way to take side-trips if something looks interesting to you.

- **This will make you think** -- Be prepared for some real challenges; believe it or not, failure is a GOOD THING! It's your chance to figure out a better way.
- **Get ready to program your own robot** – You will be using the powerful Parallax "Scribbler 2" robot with its Graphical User Interface (GUI) programming language. You are in command -- for better or worse the robot will do exactly what you tell it to do.
- **Step-by-step is the best way** -- Skills you develop in one project are used in the projects that follow. So it's really best to do all of the projects in order (they each have a number). That way you won't get lost wondering how to do something that looks impossible.
- **Mini-Competitions to test your skills** -- Depending on the size of the group in your workshop, you may also get to go head-to-head (actually robot-to-robot) as part of a team in one or more mini-competitions. Even if you're working alone, you can still challenge yourself in time trials.

Each project also has a difficulty rating, ranging from one to five "Einsteins". Look for the little icons in the upper right corner of the project sheet:

Easy Peasy.

A bit more challenging.

Put on your thinkin' cap.

Get ready for Brain Strain.

Let's get going!

What’s needed

- Scribbler 2 robot with 6 fresh AA batteries.
- S2 Robot Start-Up Guide
- PC running Windows XP, 2K, Vista or 7
- Serial to USB adapter (assuming your PC doesn’t have a serial port)
- Scribbler Program Maker (S2 v1.3 or later)
- Other supplies per project, such as markers and paper

Step 1:

Carefully read through the S2 Robot Start-Up Guide

This great little booklet came with your S2 robot, let’s go through it to learn about the many features of your Scribbler.

1. Check out the top and bottom view, get familiar with the major components, such as wheels, power switch, indicators and sensors/detectors. This really is a powerful machine!

2. Follow the instructions to insert the batteries and power up your robot.

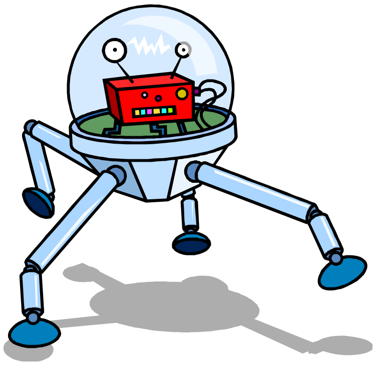
3. Run and test each of the 8 demo programs.

4. Download and install (if necessary) the S2 GUI. Once you’re convinced it’s running and you can communicate with your robot, you’re ready to get going.

If you have any problems with your robot, computer, adapter and etc. ask your workshop leader to help. It’s usually something really simple.

Step 2 and beyond:

Get the activity sheet for Project 1 and … ***start your engines***!



The Projects

First Session

Project Number	Title	Description	Difficulty
1	Light those LEDs	Your first program, get going with the GUI and your S2	☺
2	Sound OFF (and ON)	Learn about steps in a program, make some music	☺
3	Straight on till Dawn	Baby's first steps out into the real world	☺
4	A"maze"ing Turns	Sometimes the only way to get ahead is by going sideways	☺ ☺ ☺
5	Subroutines	How to keep your programs small and simple	☺ ☺
6	Simple Figures, Loops	Using your growing collection of subroutines	☺ ☺
7	Write your initials	More, bigger, better	☺ ☺
C1	Drawing Competition	Pit your skills	

Second Session

Project Number	Title	Description	Difficulty
8	The Monitor (no Merrimack?)	Let's see what we can "SEE".	☺
9	Stop at the Line	First "exposure" to using line sensors and flags in your programs	☺ ☺
10	Running Ragged (some call it "suicide")	A bit more structure to your program as you navigate from line to line	☺ ☺ ☺
11	Follow the Black Brick Road (sorry Dorothy)	Intro to line following	☺ ☺ ☺
12	Get Better at Following	How to bullet-proof your algorithm (my "algo" what?)	☺ ☺ ☺
13	Feeling your way	First experience with front-mounted sensors	☺ ☺ ☺
14	Obstacle avoidance	Roomba's got nothing on us, baby.	☺ ☺ ☺ ☺
C2	The Candy Run	First one home…YUM!	

Whatever we don't finish in the workshop, do at home!

Robots for Beginners

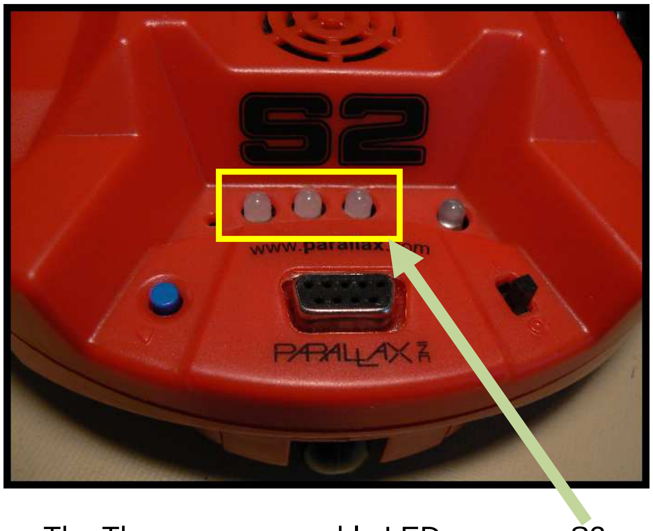
Project 1:Light those LEDs

I see the LIGHT!

Since this is probably your first time programming the S2, we will take it fairly slow for now. Later projects will assume you know your way around the GUI and are stronger at troubleshooting programs.

As you already know (you read the S2 Robot Startup Guide, right??) there are three bi-color LEDs (Light Emitting Diodes) on the top of your S2. Well, actually there are four, but the one nearest the power switch is not controllable by your programs.

Here is a photo of the S2 (with the programming adapter and cable unplugged) with the LEDs outlined:



The Three programmable LEDs on your S2

For purposes of this project, let's call the LEDs "Left", "Middle" and "Right". When it says "bi-color" it means that there are two LED emitters in each one, red and green. In fact when both are on we see amber, so it's really tri-color. (But wait, when it's off it's dark, so isn't that a fourth color? Now we're really confused.)

Regardless, in the GUI we can only set the LEDs to green or off. If and when you learn the more capable "Spin" language for the S2 then you will be able to see the other colors.

So, let's start controlling some lights. You'll be blinking with the best of them in no time.

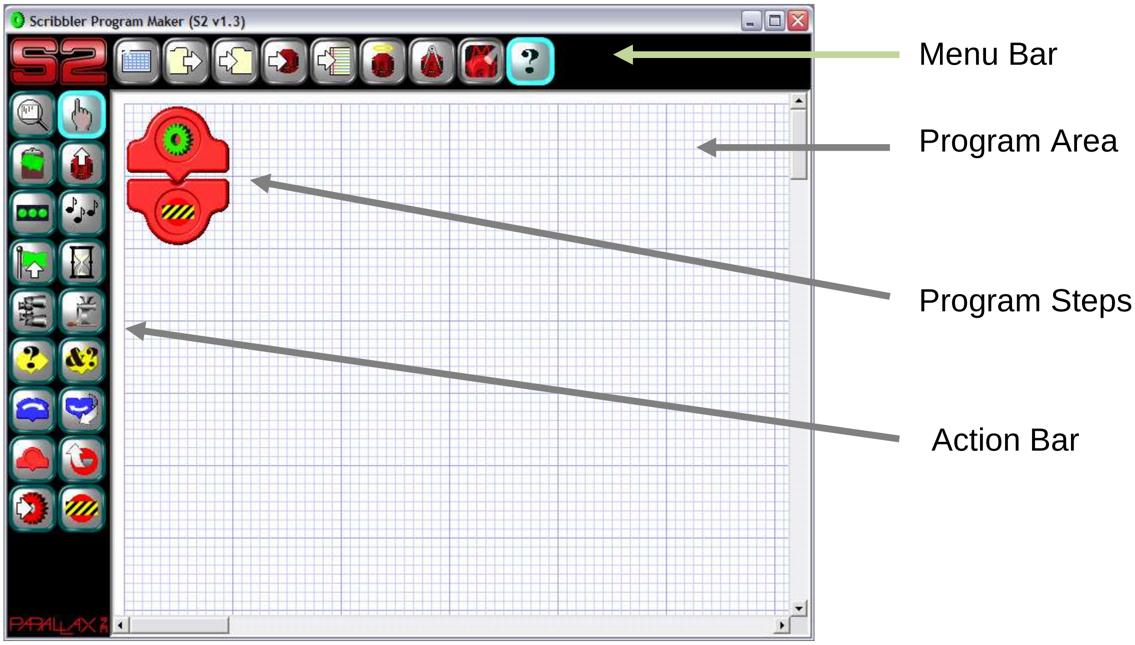
Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

If you get stuck, try to work through it. If you get REALLY stuck ask for help.

1. Your S2 needs fresh batteries. There is a little door on the bottom where you can look to see the batteries. However a dead battery looks like a new one, so let's check. Turn on your S2 (lifted so the wheels are OFF the table). After a second or two the blue light above the power switch should turn on. It's pretty bright, so don't look at it for extended periods. If the light does <u>not</u> come on, it probably needs new batteries. When you successfully see the blue light, TURN THE S2 OFF.

2. Plug in your USB to Serial adapter and cable. If you're not sure how to do this check with your neighbor or ask the instructor. Make sure all connections are fully seated but do not force anything.

3. Locate and run the S2 Program on your PC. A window will appear that looks like the following:



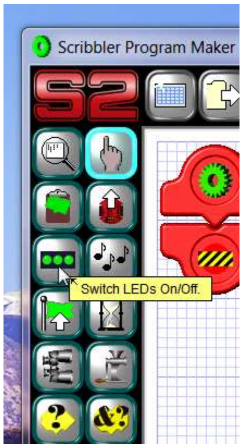
The big open area (the graph paper) in the lower right is your Program Area. Notice that right now there are only two items (Program Steps), the upper one signifies the Start of your program and the lower one the End. Since there's nothing between Start and End, this is an empty program; it will not do anything.

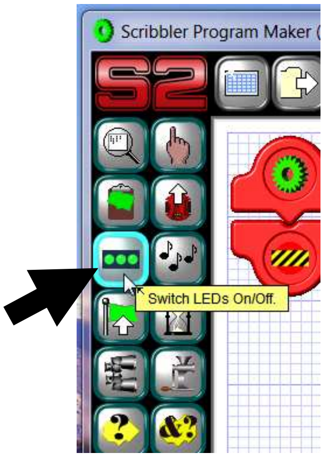
Along the left side of the window is the Action Bar containing blocks which, except for three at the top (Zoom, Select and Paste) can be selected and placed into your program. Each block has a different meaning and effect; you will learn about most of them in future projects. Think of them as "verbs", these are the commands that tell your S2 what to do and when.

Along the top of the window are several command buttons (the "Menu Bar"), such as "Clear Worksheet", "Load Worksheet" and so on. As you hover your mouse above each one you will see a brief description (a "tooltip") that tells you what that button does. Perhaps the most helpful button is the "?" one on the far right, it will bring up the help system with plenty of good and useful information.

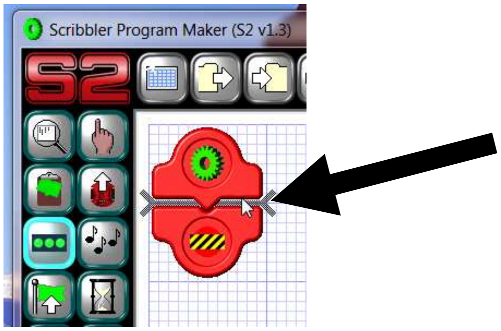
Hint -- Try pressing the "?" button now, YOU'LL LIKE IT.

4. Move your mouse over the Action Bar as shown, informational text is displayed as shown in the left picture below. In this instance it is over the LED block (something we're about to use). Click once (left button) and it now appears highlighted (encircled in light blue/green) as shown in the right picture below. You are now ready to insert this block into your program.

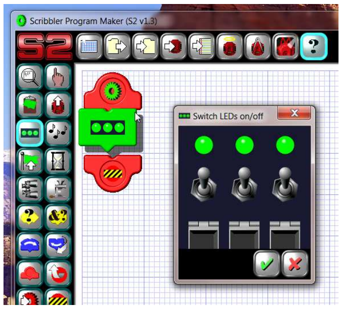




5. Move your mouse cursor over to the Program area to the right as shown, in-between Start and End. As you move you will see some "crosshatching" indicating it's ready to place an Action Block there:

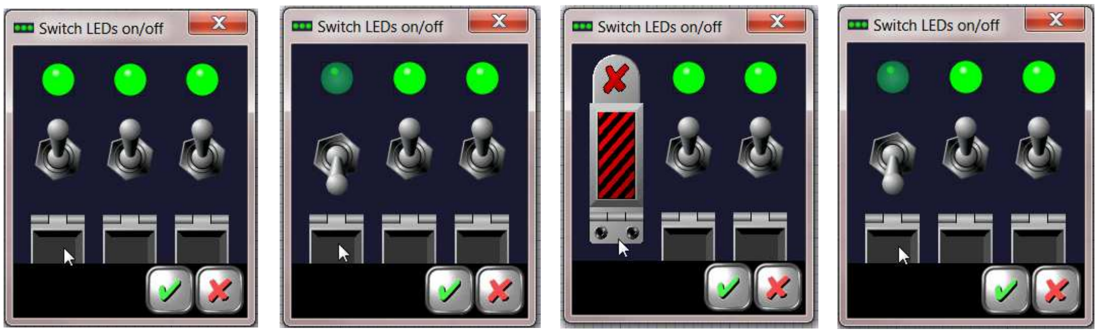


6. Without changing position, click your left mouse button once and you will see an LED Action Block has been inserted between the Start and End Program Blocks as shown. You will also see an option window depicting the three LEDs, Left, Right and Middle:



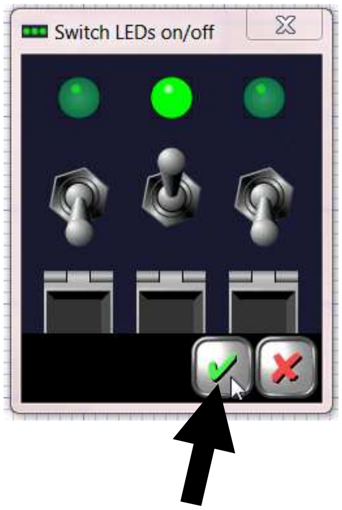
Hint – As you add Action Blocks, what you're really doing is creating a step-by-step procedure for your S2 to follow. Later when this program is "uploaded" to your robot, it will perform each of the steps that you select, and in the order you placed them from top to bottom.

7. Each LED can be instructed to do one of three things – Turn ON (light up), Turn OFF (go dark), or stay the same as it was. As shown, click repeatedly on the graphic below one of the LED “toggle switches”, you will see a sequence as shown below:



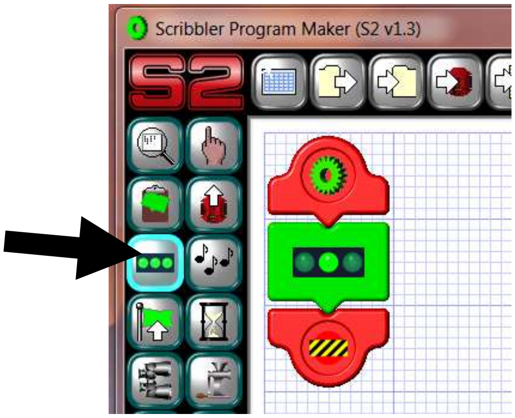
If the respective LED is shown as lighted, that’s what it will do after this Action Block has been “executed”. Similarly if it’s dark, that’s what you can expect. The odd looking one in the third picture is the one to leave this LED as-is. This is a programming trick that allows you to modify the appearance of some lights while leaving the others unchanged. For example if the Left LED was already set by an earlier program step and you want to turn on the Middle LED and leave the Left LED alone, select the “X” option for the Left LED and the lighted option for the Middle LED.

8. Using this technique set the LED options to match the picture below. When it looks right, click on the check mark as shown:

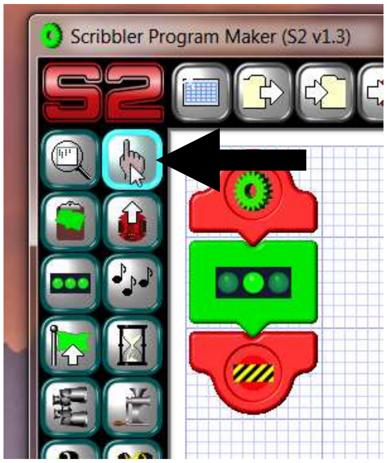


Hint: If you want to cancel the current operation simply click on the “X” button. This will ignore any selections you may have made and will close this settings window.

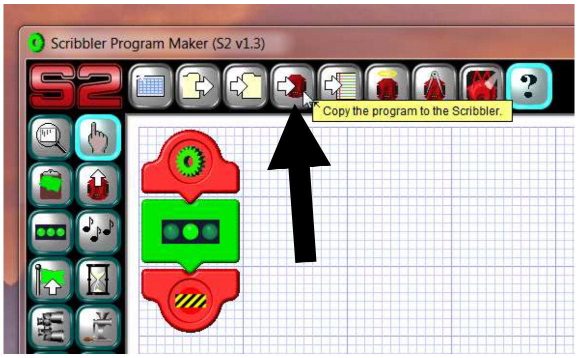
Your screen should now look as shown below:



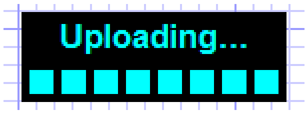
Notice that the LED Action Block is still highlighted (has a blue/green rectangle). If you again move your mouse cursor into a position between blocks in the Program Area, you will see the crosshatching indicating you can drop another LED Block if you want. Since we're done inserting blocks, click on the "Select and Edit" button (looks like a hand with a finger pointing) as shown:



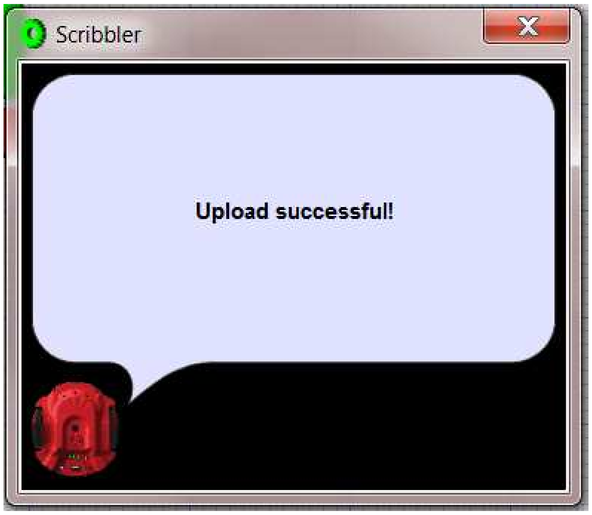
9. Ensure that your S2 is hooked up to the PC and powered on. The blue power LED should be illuminated. In the Menu Bar at the top click on the button entitled "Copy the program to the Scribbler".


Scribbler Program Maker (S2 v1.3)
S2
Copy the program to the Scribbler.

After pressing this button, if all is well you will see this window:

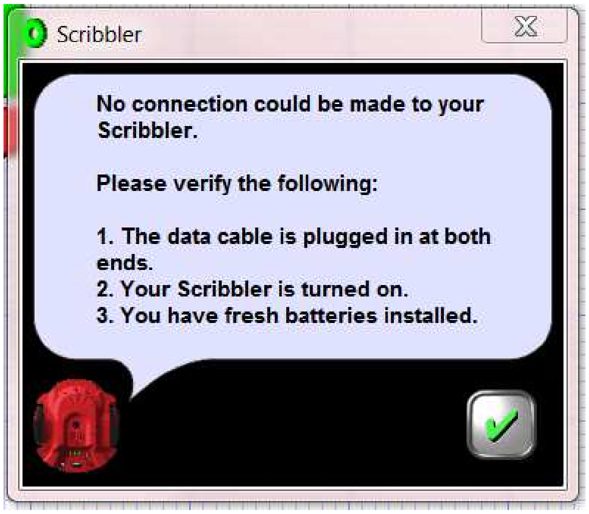

Uploading...

Followed by this window (briefly):



Upload successful!

This signifies that your program has been uploaded to your S2. If instead you see a window like this:



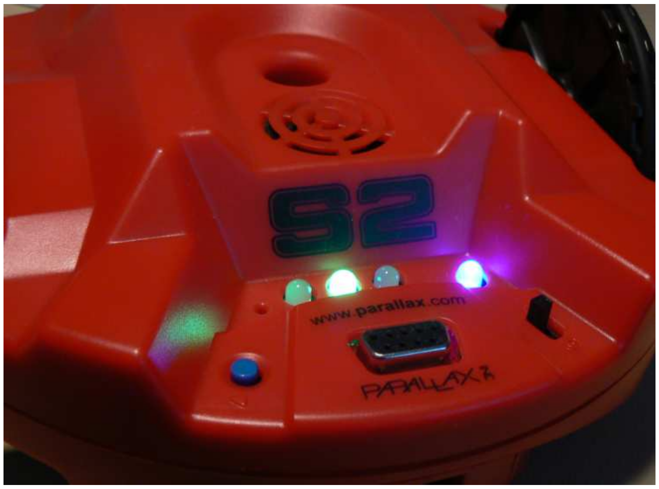
No connection could be made to your Scribbler.

Please verify the following:

1. The data cable is plugged in at both ends.
2. Your Scribbler is turned on.
3. You have fresh batteries installed.

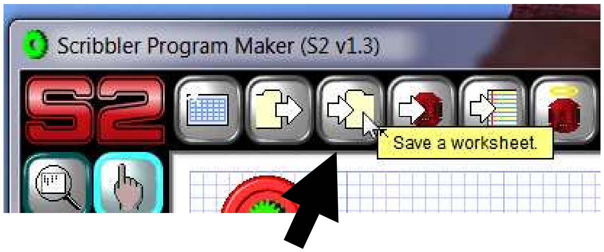
… this means that your PC is having trouble communicating with your S2. Ensure that the cable and adapter are each plugged in at both ends and that the blue power LED is illuminated on your S2. If you are unable to get it to work, ask your instructor for help.

10. Look at the LEDs on your S2, they should appear as shown (adapter and cable removed for clarity, yours can remain plugged in):



Notice that the middle LED is now illuminated, just like in your program. OK this may not be Earth shattering, but you will find in future projects that LEDs can be really useful to indicate what's going on in the mind of your robot.

11. For safe keeping (and later use) save your admittedly not-very-complex program so you can see how it's done. As shown below, select "Save a Worksheet" on the Menu bar:



You will now see a conventional "Save" dialog where you give your program a name and save it to a folder that your instructor suggests.

Hint: It's better to give names to your files that make it easy for you to find later. For example the file for this project would be easier to identify if named like "Bill's Project1.scb" rather than "This is really neat.scb". Also FYI, all worksheet files need to have the ".scb" extension.

Congratulations you just created, uploaded, executed and saved your first S2 program!

Take a deep breath, pat yourself on the back and get ready for…

Project 2, Sound OFF (and ON)

Robots for Beginners

Project 2:Sound OFF (and ON)

Not exactly fine music, but…

The S2 is like a one-man-band, a little bit of everything. In this project we will use another non-motion command for your S2 to create sound. As you'll see you can use pre-programmed musical ditties, play your own notes or even (drumroll please…) create Morse Code.

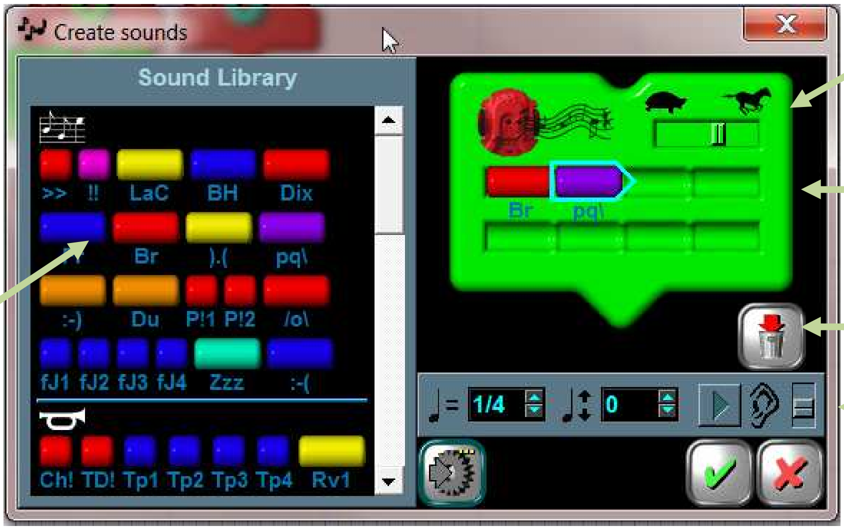
You have seen how to select an Action Block and put it into your program, let's now see how to string a bunch of them together.

***Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil* ☑.**

If you get stuck, try to work through it. If you get REALLY stuck ask for help.

1. Clear the worksheet by pressing the appropriate button on the Menu Bar (no hints this time).

2. Find the "Insert a sound sequence" Action Block and place it into your program. A "Create sounds" window will appear as shown. As illustrated below, there are several sections to this window, each of which allow you to modify what your S2 sound generator will do:



Adjust playback speed

The sounds you have selected, in order

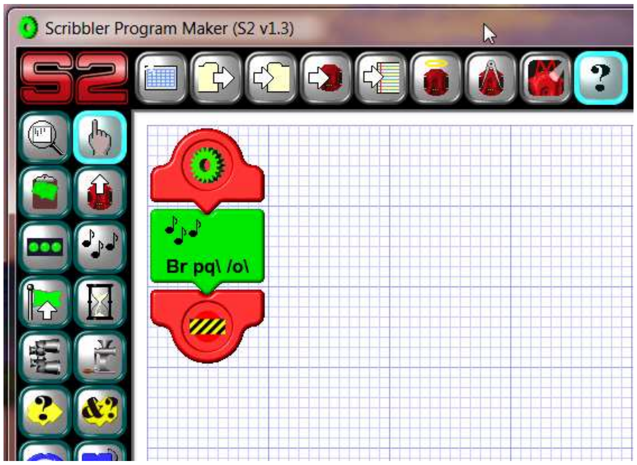
Delete a sound

Settings and preview

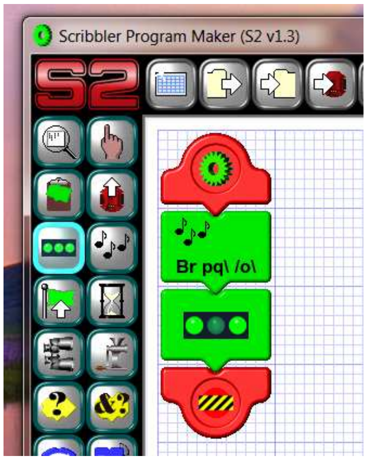
Select Sounds Here

Spend some time here, select a sound, preview it, play with the tempo, volume and speed. Add a few more sounds, sample different types. Learn how to delete one or more sounds.

3. Press the “Check” button and accept your changes. Your screen should now look approximately as shown below. Upload the program to your S2 and let it run. Did it do what you expected? Did your neighbors get irritated?



4. Add an LED Action Block below the Sound Sequence Action Block. Set it to light Left and Right LEDs, with the Middle LED dark. Your screen should now look approximately like this:



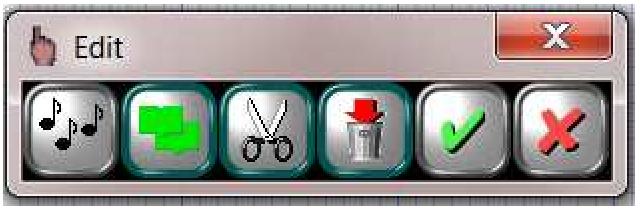
Upload this program to your S2. Notice that the LEDs don’t light up until AFTER the music has finished. This is the sequential nature of programming.

5. Push and release the “Reset” button on the S2 (to the left of the signal cable). Did your program run again? Refer to your Start-Up Guide if you’re not sure where the button is located.

OK, let’s build some more GUI Skills now.

6. Hover your mouse cursor over the Sounds Block in your program and press the **right** mouse button. Notice that the block appears to rise and following window appears:

(Note: If your mouse has only one button (common on some older Mac computers) you can still simulate a right mouse button push. This is often done by pressing the Ctrl key while you click. If you have problems with this please ask your instructor for help.)

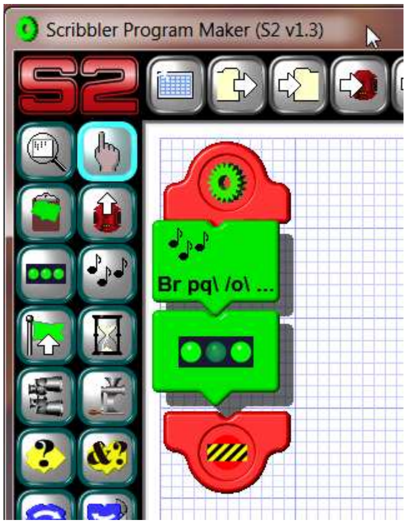


This small window allows you to (in order from the left):

1. Edit the selection (will not be present if you've selected more than one Action Block)
2. Copy the selection
3. Cut the selection
4. Delete the selection
5. Proceed with Copy, Cut or Delete and close this window
6. Cancel and close this window.

Left click on the "Edit the selection" button and notice that the familar "Create Sounds" window appears, filled in with whatever you selected earlier. You can now modify the sounds as you like and either press OK (the check mark) or Cancel (the X mark).

7. In your program left click on the Sounds Action Block and then the LED Action block. Notice that they both appear to rise as shown below:

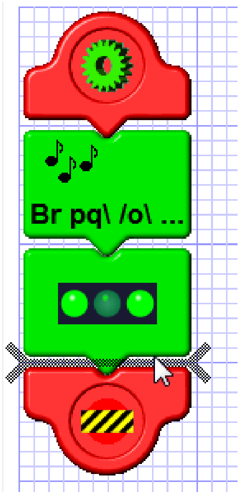


Left click anywhere else in the Program Area and notice that they drop back down again. This is how you select and deselect blocks. Once again select both blocks so they appear as above and while hovering above either block press the right mouse button. Notice that the Edit window appears, but now it only has 5 blocks (think about it, what's the difference?).

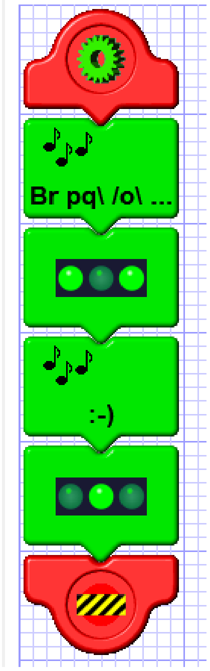
Press the "Copy the selection" button and press OK (the check mark).

8. We now have the two blocks stored ready to paste somewhere in the program, let's go ahead and do that.

Hover your mouse cursor as shown to create an insert location (crosshatching) below the LED Block in your program:



Press your left mouse button and observe that a duplicate copy appears in the new location. Using the method described earlier, modify the contents of the newly pasted blocks so that you can easily spot the difference. After the changes your screen may appear similar to this:



9. Upload and run the program in your S2. Did it do as you expected? Save the worksheet.

10. Spend at least 5 minutes randomly moving, deleting, editing, copying blocks, saving worksheets, loading worksheets. It's ok to make a mess -- this is a good time to explore the GUI.

Robots for Beginners

Project 3:Straight on till Dawn

At this point we will start to move more quickly. As always, if you get stuck refer back to earlier projects, try something, read the help system…eventually you'll figure it out. If you REALLY get stuck there's always the instructor or a lab neighbor.

Baby's first steps

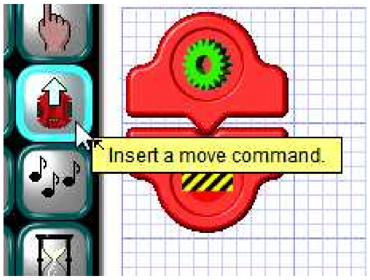
You've waited patiently; you've worked on projects; edited programs, saved and loaded them. Now it's time to make this little bugger move. OK, it's only in a straight line for now, but that's a lot better than just sitting there.

Before we jump right in, let's take a moment to look at the S2 to see what makes it able to move. A quick inspection shows there are two large wheels, one on each side. Turn it over and you see there's a third wheel (not a social comment) just under the signal connector. This allows the robot to sit like a tripod (three legged device), able to handle bumps and uneven surfaces pretty well. Take a closer look at the two bigger wheels, there's a rubber O-Ring around the edge of each -- it acts like a very simple tire, providing improved traction.

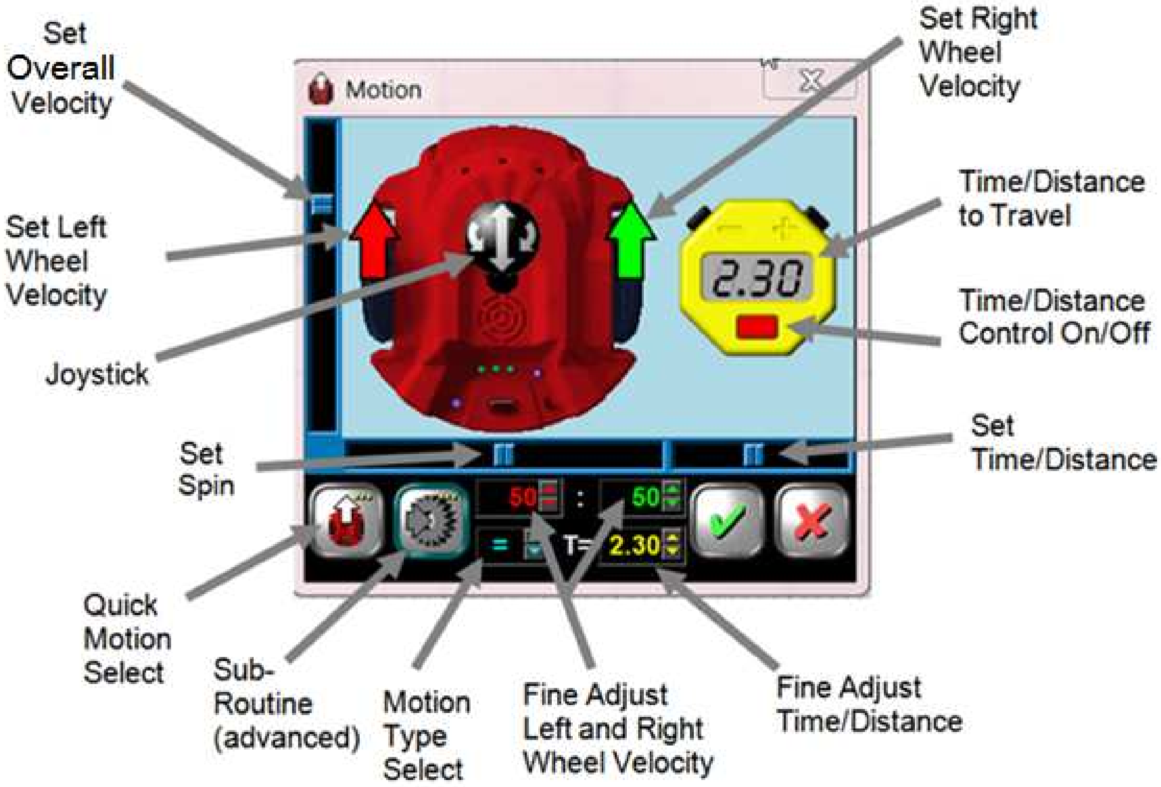
Inside your S2 driving each of these big wheels is a small motor (there's other stuff on the axle that we'll discuss later). Under program control we can tell each of these motors how fast to turn and in what direction. This means we can make the S2 go straight, turn, spin, stop and anything else you can dream up. And just like when you programmed your S2 to play music and then turn on some lights, you can tell your robot to move one way, then another and then another by just inserting the appropriate Action Blocks.

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Starting with a clear worksheet, find and add an "Insert a move command" Action Block. In the Action Bar it looks like a top-view of your S2 with a large white arrow on top of it. Here is a screen shot:



Immediately after inserting the Action Block in your program you will see the following window. It seems to be pretty dense with information, let's review the important parts:



Don't get too concerned, it's really a lot simpler than it looks.

There are two main functions here: how fast is each wheel turning and for how long (optional). The fine folks at Parallax just decided you needed about four ways to do everything.

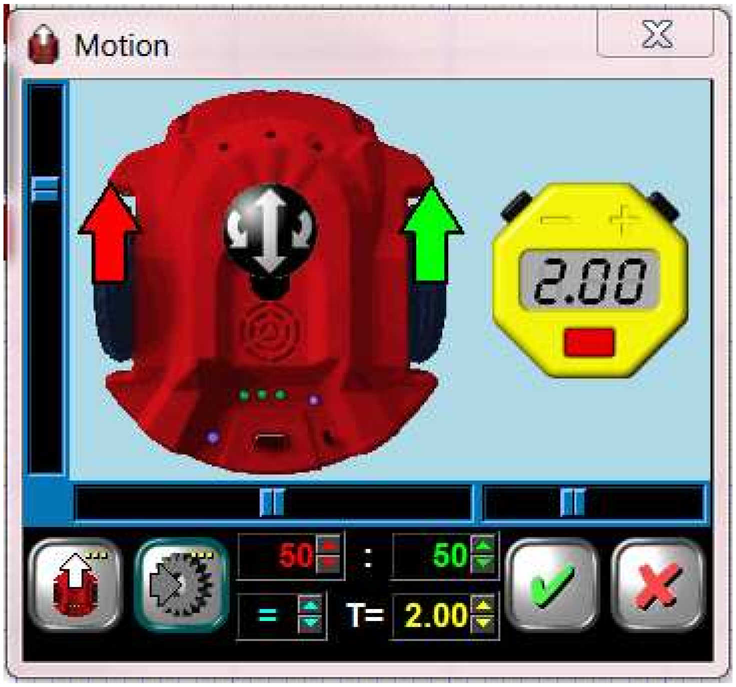
The best way to learn is to simply start dragging controls around and watch the red and green arrows; these indicate the speed and direction (velocity) of each of the two wheels. The red arrow is the velocity of the left wheel, the green arrow is the velocity of the right wheel. When the arrow points up, that wheel moves forward. When the arrow points down, that wheel moves in reverse.

Grab the **Joystick,** it's the black thing in the middle that looks like a shift lever. To grab it, move your mouse over it, click and hold your left button and move around in random directions. Notice you can make the arrows grow and shrink and point up and down as well. Also watch the green and red numbers in the Fine Adjust area, they change too. If you're REALLY careful you may even be able to make both numbers equal to zero (0). That means that both motors are stopped. Also notice that the speed can be negative, that means that motor is turning in reverse.

Lastly, click several times on the "Quick Motion Select" button. Notice it cycles through several useful combinations of settings.

Question for thought – If the motors turn for a certain time at a certain speed, how do we know how far your robot will move?

Since in this project we're only moving in a straight line, set both sides to a speed of (positive) 50 as shown. Set the Time/Distance to 2. With these settings your S2 will travel forward at ½ speed for two seconds and then stop. The Motion settings window should look like this:



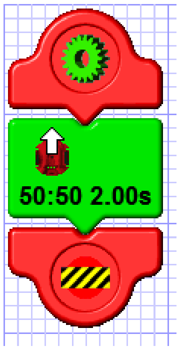
Motion

50 : 50

= T= 2.00

When you're satisfied your settings are correct, click OK.

2. Your program should now look like as shown below. Notice the speed and time values you set are displayed for your convenience:

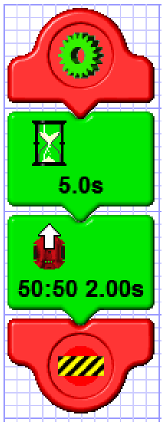


50:50 2.00s

Before we upload this to our S2, we better add some delay at the beginning so you have time to disconnect the cable and set the robot on the table.

Look in the Action Bar for the hourglass, called "Add a pause". Click on that, place it into your program just prior to the Move Block and when the "Wait awhile" window appears, set it for a pause of 5 seconds. To accurately set the time, click in the area immediately above or below the slider to finely increment or decrement the number. Press OK.

Now your program should look like this:



3. When you're satisfied, upload the program to your S2. Once the upload is finished, you've got 5 seconds to disconnect the adapter and place the robot on a surface where it can move for two seconds without falling.

Does it move as you expect? Re-run the program by pressing (and releasing) the reset button. **Use a ruler or tape measure and check how far it travels.** Do this a few times, does it seem to be repeatable (nod your head yes!).

4. Modify the program to run for 2.5 seconds instead of two, but still a speed of 50 for each motor. Measure as accurately as you can how far it travels.

5. Repeat the above, but move for 5 seconds instead. Measure again. How does the distance traveled compare to distance in step 4 above?

6. Repeat step 4 above but with a speed of 100 for each wheel. Measure the distance. What conclusions can you draw about expected behavior? Do you think you can predict how far it will move for other speeds and times, without having to run the program?

7. Based on what you saw above, how far (in inches) do you think the S2 will travel with a speed of 75 on each motor, running for 2.5 seconds? Write your answer here: _________

8. Perform the test described in step 7, was your prediction right? If not, how do you account for the difference?

9. Using the program from step 7, copy the Move block and paste it at the end of your program. Set it to move at the same speed but in the reverse direction (move backwards, make the speeds negative), leave the time at 2.5. Run this program, does it return accurately to the starting point?

Hint -- The S2 has an "encoder" (an optical sensor) in each wheel that measures the number of wheel rotations; it's quite accurate as you're hopefully discovering. This allows you to create programs that move in very controlled ways, without having to worry too much about differences in motors, friction and battery condition.

Question for thought -- Given the above, how accurate do you think the motion will be if one or both wheels slips or skids?

Robots for Beginners

Project 4:A”maze”ing Turns

Sometimes the only way to get ahead is by going sideways

Now that you’re a pro at going straight (and predicting how far you’ll go) it’s time to do some turning. Thankfully your S2 has a couple of cool design features we can exploit. The first is the fact that the two wheels are arranged such that if they rotate in <u>opposite</u> directions at an equal speed, the robot will simply spin on its axis. The second cool design feature is that there is a hole THROUGH the S2 precisely at that axis.

The combination of these features in concert with the fine control of motion that the encoders provide allow us to use the S2 as a sort of X-Y plotter (that’s how “Scribbler” got its name). Just download your program, drop a pen through the hole, and start writing. Of course it’s not quite that simple, but almost.

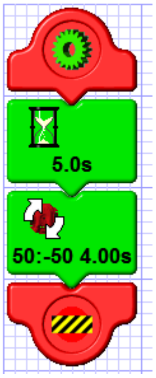
So let’s start turning!

***Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil* ☑.**

1. Starting with a clear worksheet, find and add two blocks: a pause and an “Insert a move command” block.

2. Set the starting pause to the usual 5 seconds to allow you time to get set.

3. Set the velocity on the wheels to: Left = 50 , Right = -50 (yes that’s a MINUS 50). Set the Time/Distance to 4 seconds. Upload to your S2. Your program should now look like this:



5.0s

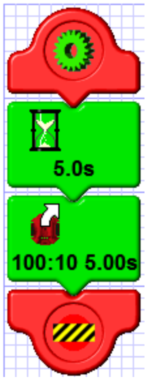
50:-50 4.00s

Caution – We’re about to do some writing with your S2. If you are using an erasable white board, BE SURE TO ONLY USE DRY ERASE MARKERS!!!! Otherwise you can permanently damage the board. Also please cap the pen after each run, that saves it from drying out.

4. Get a pen and a sheet of paper (or dry marker and white board). With your S2 switched OFF place it on the paper or white board, remove the cap from your pen, insert the pen through the hole tip-down. Now without jostling the S2 turn the power on and let go. Once the program has finished, remove the pen first then lift your S2. Look at what was drawn.

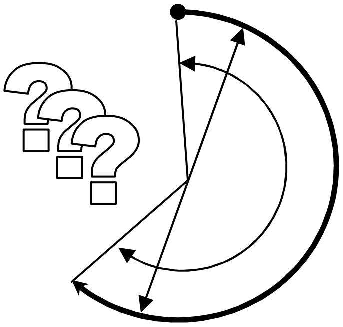
If you entered the program correctly all you will see is a single dot, perhaps a very small circle or squiggle. OK, it's pretty dull but THIS IS GOOD NEWS, it means that your S2 is able to spin on its own center!

5. Now let's learn how to draw a circle. Change your program to have a velocity of 100 on the left and 10 (positive 10) on the right. Set the time to 5 seconds. Your program and drawing will look like this:



5.0s

100:10 5.00s



Upload to your S2, set up the pen and go again. You should now have an arc. What is the diameter of the arc in inches?

What portion of a circle was drawn (there are 360 degrees in a full circle)?

6. Now it's your turn. If your S2 is typical, you drew a little more than a half circle with the time set to the maximum, 5 seconds. Hmmm… but we want a FULL circle, how do we get there? (Think about this BEFORE reading the next sentence!)

.

.

Time to think…

.

.

If you're clever you realized that you can string multiple instances of the same move block, and thus add up the time!

So let's do it. Copy and paste the move block so you now have two identical blocks in a row, upload and run again.

What did it do?

Did it seem to move about twice as far as one single block?

Can you see discontinuity in the line where one block ends and the next one starts?

Did it come back and write (more or less) on top of the original line?

7. Adjust the velocity values and times to produce as nearly a 8” diameter circle as possible, starting and stopping on the same spot. Use multiple move blocks if necessary.

Write down the final values here:

Left Velocity _____ Right Velocity _____ Total Time _______ (add up the blocks)

8. Create a program that will draw a 6” straight line, spin 90 degrees in place to the right and then draw another 6” straight line. The result should be two straight lines joined at a sharp right angle turn.

Write down the values for the two move blocks:

Block 1: 6” Straight Line: Left Velocity _____ Right Velocity _____ Total Time _______

Block 2: Spin 90 degrees to right: Left Velocity _____ Right Velocity _____ Total Time _______

9. *Let’s put this to practice.*

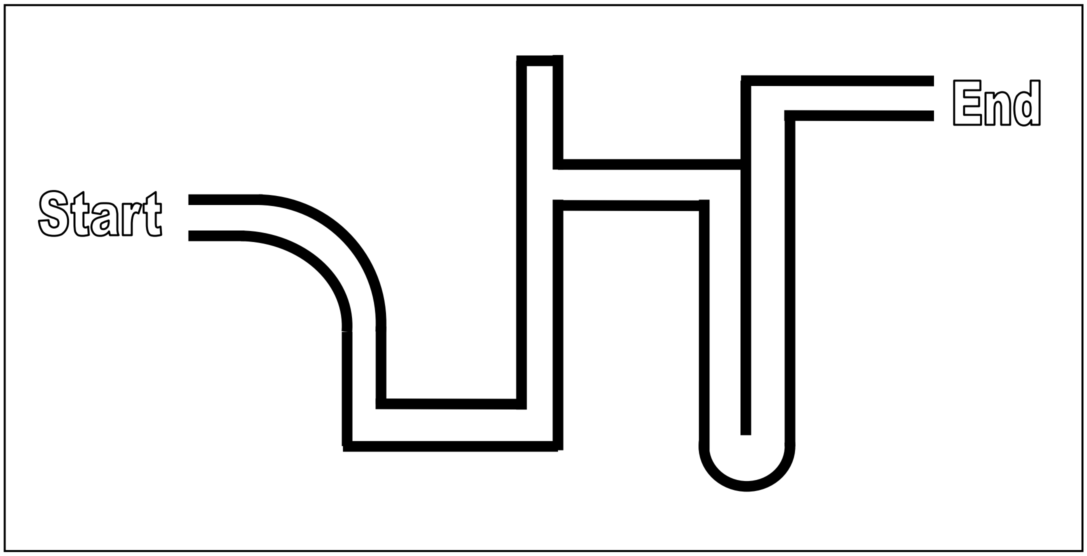
Your instructor will show you the Maze, a white board with two lines that describe the boundaries of a pathway.

Program your robot to turn and move (with a pen) so that the line is always between the lines. You MUST travel the <u>entire</u> path. Be sure to use a dry-erase marker!

Ready, Set…Go!

Hint: If you reach what looks like a dead-end in the path, that means you are to stop, turn 180 degrees, and continue. Or of course, you could always travel backwards…

Here’s an approximate picture of the maze. You are urged to measure the actual maze so you know how far to move and turn:



Robots for Beginners

Project 5:Subroutines

Re-use, re-cycle, re-new

One of the most appealing aspects of programming is that you can easily create groups of instructions that do a specific job and then re-use that “code” many times. This is one aspect of a larger technique called “Structured Programming”.

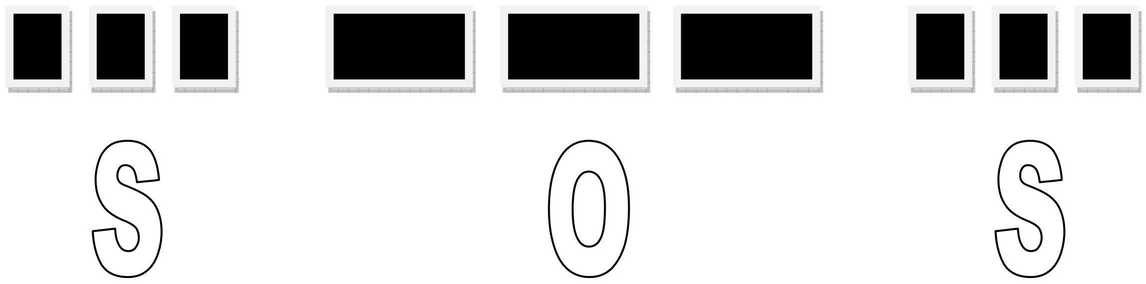
In this project you will write a group of handy “subroutines”, each of which does some smaller part of a larger task. Then you will use these subroutines to do something more complex. But, what IS a subroutine.

A subroutine is a re-useable encapsulated group of program statements (Action Blocks) that perform some useful function.

When your program wants to do that function, instead of having to add all those blocks again and again, you instead “call” the subroutine. Program control then switches to that subroutine and the blocks in that subroutine are executed one-by-one in order. When the subroutine has finished, control returns to whatever called it. It’s that simple.

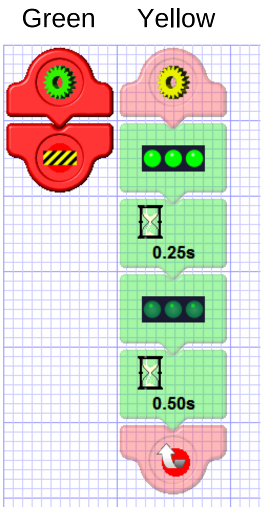
You may have noticed in the S2 GUI many blocks contain a gear icon: ⚙ These indicate that in some way a subroutine can be created or used when selected. You may have also noticed in clicking around that you can change the color of the gear. Each different color refers to a unique subroutine, you have seven to work with: Yellow, Orange, Red, Violet (light purple), Purple, Blue and Cyan (light blue). The Green gear is reserved for the main part of your program.

In this project you will create a program that uses subroutines to display Morse code characters on your LEDs, much like mariners once did ship-to-ship. Let’s also say for simplicity that there are only two characters to send, ‘O’ and ‘S’. So if you want to send “SOS” (the international message for “HELP ME!”) you blink the light 3 times fast, three times slowly and then three times fast again.



Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

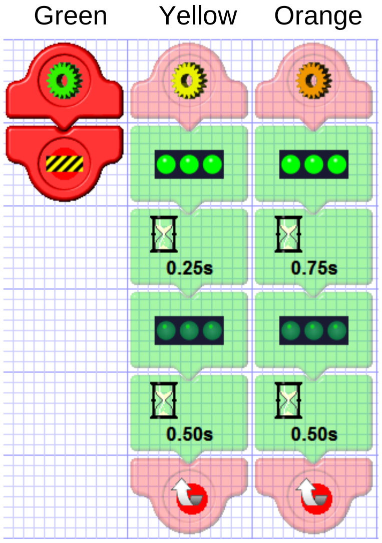
1. Starting with a blank worksheet, create a subroutine (Yellow) that blinks the LED on and off once for a short duration (a “dit”). The way this is done is to place a “Make a new subroutine” block and then populate it with blocks. You will notice that each time you add a new subroutine it is placed to the right of the existing code, and that you get to choose from the available seven colors. The code should now look as follows:



If you study the above, you see that the main part of the program where we usually work (under the green gear) is empty for now; we will add to that soon. More important, you can see the yellow subroutine on the right which turns on the LEDs, waits for a short duration, turns them back off, and waits for another short duration.

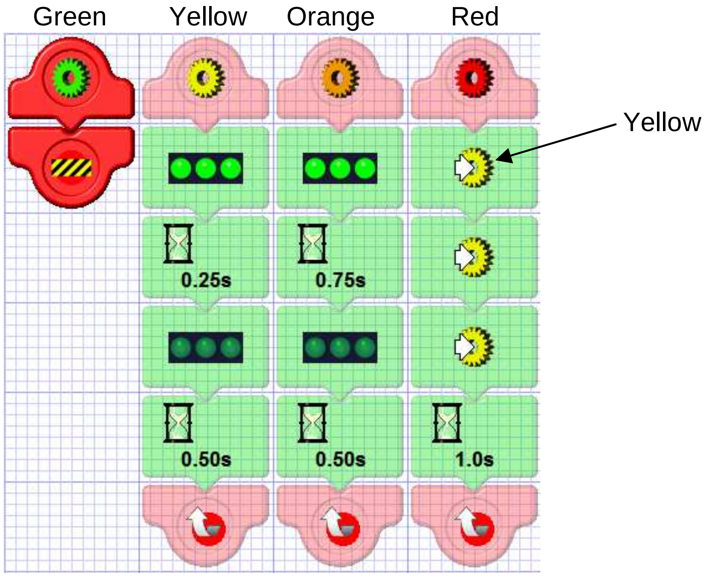
Helper -- For those of you who have black and white printouts of this project, the color of the gears is shown in text near the respective columns.

2. Create a similar subroutine in orange for the longer pulse, the “dah”. It’s pretty easy, just place an orange subroutine block and then copy/paste from the yellow subroutine. All you need to do then is edit the first timer to make it a longer pulse. See the program below:

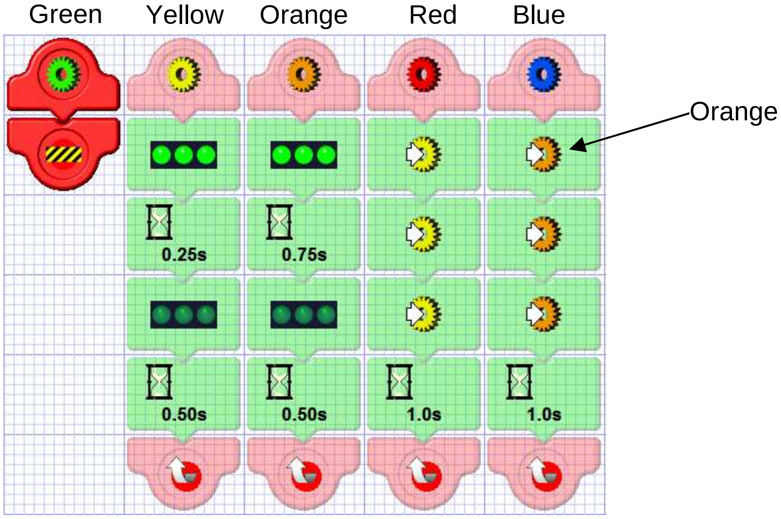


Now we will create subroutines that use the above to send individual letters, the ‘S’ and the ‘O’.

3. Add a subroutine block (red) and inside it, call the yellow subroutine three times followed by a delay. Look for the “Call a subroutine” block in the lower left corner of the Action Bar. When you drop it into place you will be asked to select what subroutine to call, press on the gear icon and cycle to the yellow gear. Then click OK. Do this three times, add an additional 1-second pause to the end. Your program will now look like this:

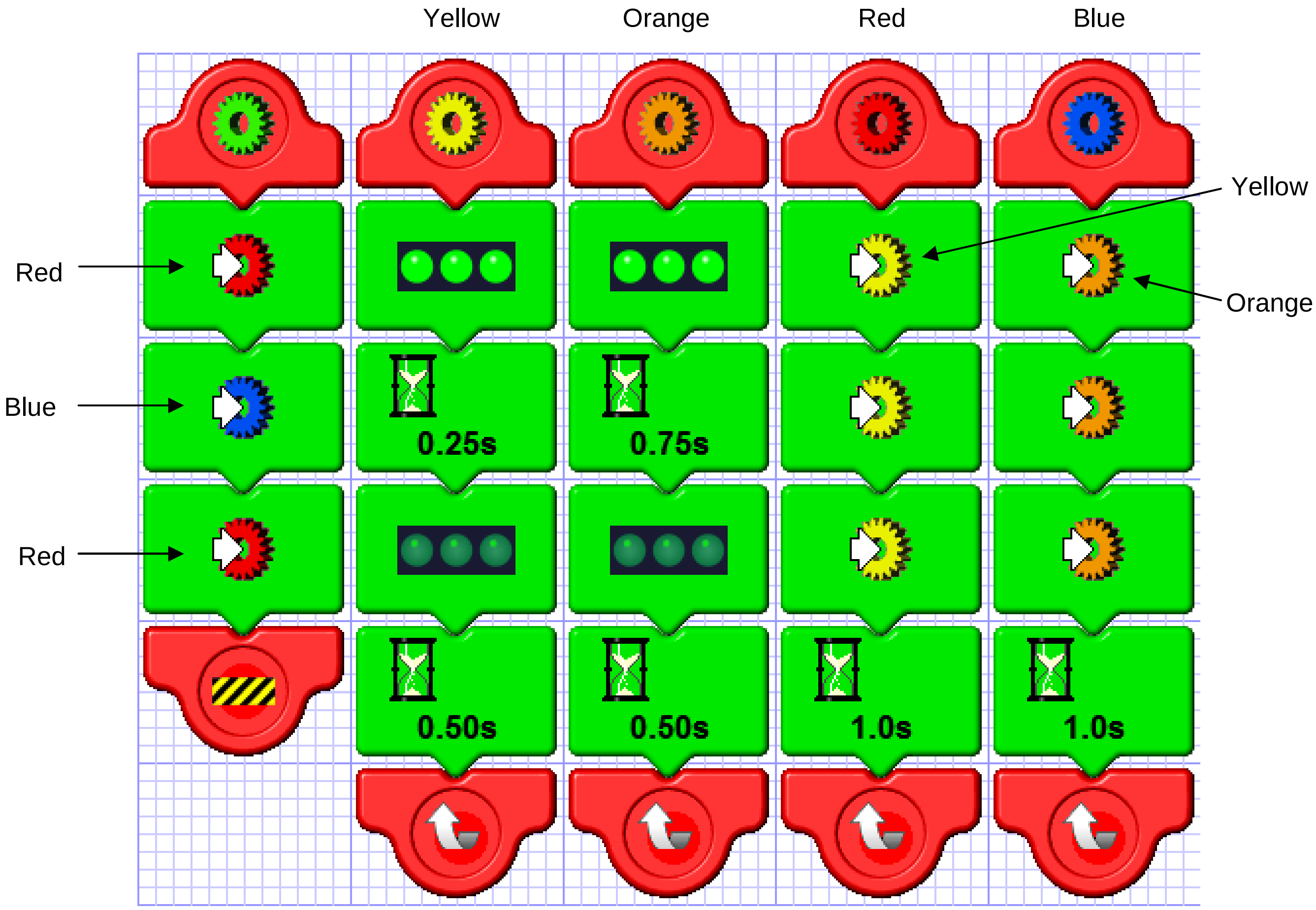


4. Add one more subroutine (blue) and copy/paste the entire contents of the red subroutine into it. Now edit each of the subroutine call blocks to call the orange subroutine instead of yellow. This will generate three long pulses and a pause, the ‘O’ character. Your program now looks like this:



We now have subroutines to make the ‘S’ letter (the red subroutine) and the ‘O’ letter (the blue subroutine). Lets finally put some code in our main section, green, to use them.

5. Insert three subroutine call blocks into the green section (main), call the red subroutine, the blue subroutine and then finally red again (S, O, S). Your finished program should look like this:



Upload the program to your S2. Does it work as expected? Press reset and view it again.

If desired, adjust various timings to make it more pleasing.

Notice that if you want to shorten the length of the “dit” you only have to change it in one place. Imagine all the editing you’d have to do if this program did not use subroutines!

Don’t forget to save your worksheet!

Congratulations you just used your first subroutines. When you’re ready, progress to the next project where you will put them to use drawing figures.

**** Extra credit activity – Create subroutines to send your initials in Morse code. If necessary use Google to look up the Morse patterns for the letters.***

Robots for Beginners

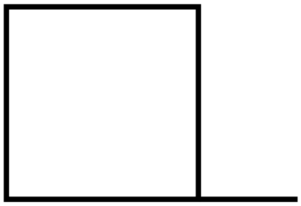
Project 6:Simple figures, Loops

Now that you've drawn graphic "primitives" (lines, curves, angles) and created subroutines we are going to again pick up the pace. In this project you will write your own program that uses all of your newly learned skills, plus you'll learn about loops.

A string of pearls

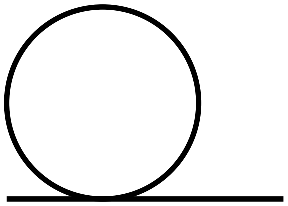
***Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil* ☑.**

1. Starting with a blank worksheet, create a subroutine that draws a 4 inch square. Assume that the starting point is the lower left corner and that you start heading toward the right. When your subroutine finishes be sure your robot is at a point 2 inches from the lower right corner, again heading toward the right. The output of this subroutine should look like this:



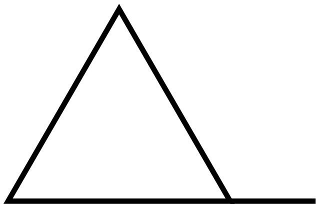
2. In the main section create a call to this subroutine. Upload and run your program using a pen and board/paper. Refine the subroutine to do exactly what you want.

3. Create a subroutine that draws a 4 inch diameter circle. Again assume that you are starting at a point at the lower left of the curve, and that you finish at a point 2 inches from the lower right bottom, heading to the right. The output of this subroutine should look as follows:



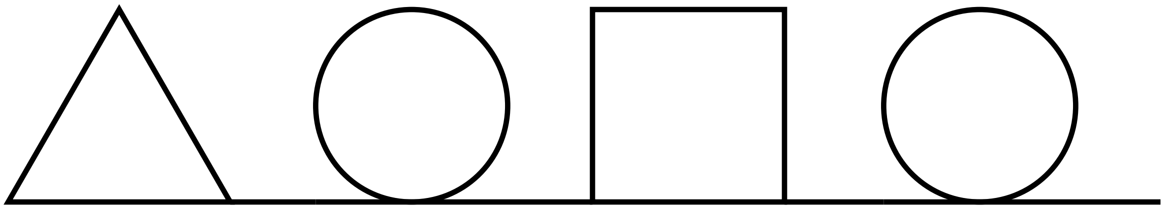
4. Change the call in your main section to call this new circle subroutine. Upload and refine.

5. Create a subroutine that draws a triangle that is 6 inches on a side. Again assume that you are starting at a point at the lower left of the triangle, and that you finish at a point 2 inches from the lower right bottom, heading to the right. The output of this subroutine should look as follows:



6. Change the call in your main section to call this new triangle subroutine. Upload and refine.

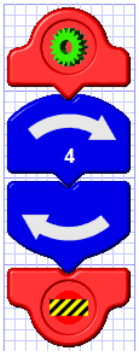
7. Modify main to call all three subroutines in any order you like. Your output may look something like this:



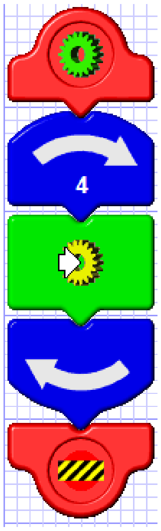
Loops – powerful and easy

You are now going to add a "loop" to your program. A loop is a simple way to repeat a section of code a controlled number of times. In the following example you will use a loop to draw a series of figures.

8. Clear your main program section and insert a program loop. It looks like this: . *Be sure NOT to delete your subroutines.* After you place it in your program a window will appear asking you to set the "Loop counter". If you leave it unset, the loop will repeat indefinitely (more on this later). Set the loop to 4 repetitions and click OK. You will see loop beginning and end blocks as shown:



9. In between the two loop blocks insert a call to your triangle subroutine. Now the main portion will look something like this (not including the subroutines):



Upload and run the program. Did it do as you expected? If not, edit the program and try again.

Draw a sketch of the final output below:

Be sure to save your worksheet!

Congratulations you have now used your own subroutines to create a simple graphic and a loop to repeat an operation.

When you're ready, progress to the next project where you will draw your own initials.

Robots for Beginners

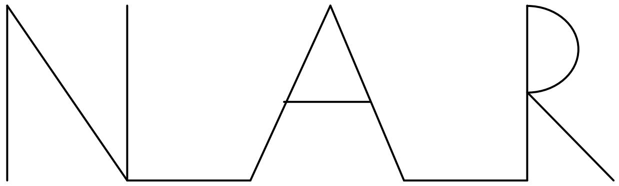
Project 7:Write your Initials

Identify yourself please…

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Starting with a blank worksheet, create a program that that draws your three initials (or if you only have two, invent a third one). Draw in block letters approximately 6 inches high. Be sure to connect each letter at the bottom. Don't worry about dotting your "i"s, and <u>you probably will need to backtrack occasionally.</u>

Here's an example of my initials:



Th-th-th-that's all folks. Go forth and PROGRAM!

Robots for Beginners

Competition 1:Drawing trials

Are you ready to go "robot to robot"?

Operational Note: This "competition" can be used in a group or by yourself (in time trials). If you are in a group, the size of each "alliance" should be either one or two persons per robot, depending on the number of participants and robots available. Your instructor will determine how to divide you up.

The challenge –

Starting with a blank worksheet, write a program that will create the shape that your instructor gives you.

The awards –

Speed Winner: The first group/individual to successfully create a program to create the desired shape.

Quality Winner: The group/individual who creates the most accurate shape.

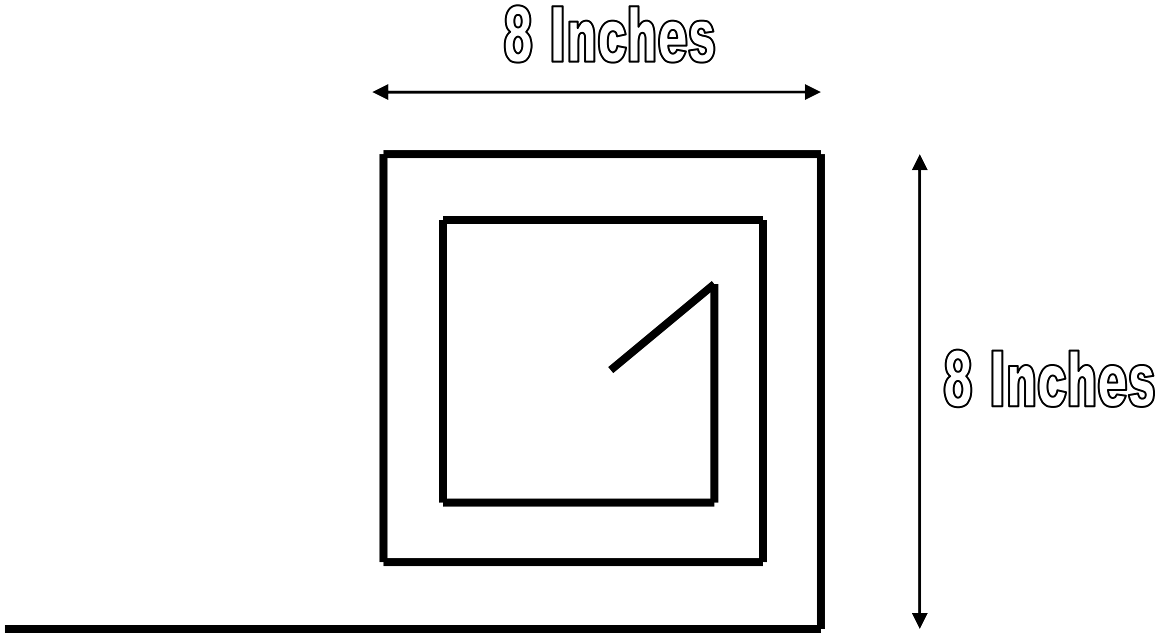
Programming Star: The group/individual who makes the most creative use of subroutines or other clever programming methods.

Rules:

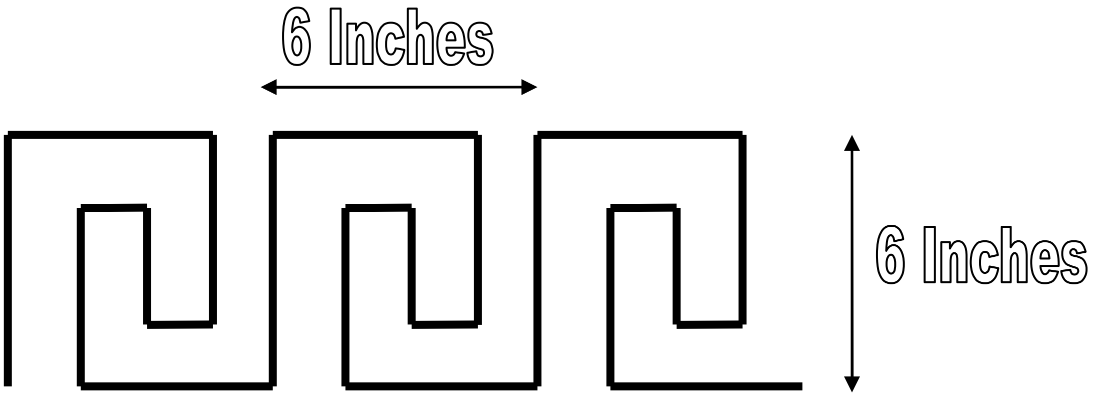
1. No peeking at the shape until the instructor says "Go".
2. You may refer to programs that you (or your partner) wrote earlier.
3. To win, your shape must look essentially the same as the provided pattern. Your instructor has the last word on this.
4. Shapes may be drawn in any order or direction. The final appearance is the only criteria.
5. You should keep going even though someone else may have finished. You can still win the Quality or Programming Star awards.

Prize: ***Bragging rights!***

Shape 1:

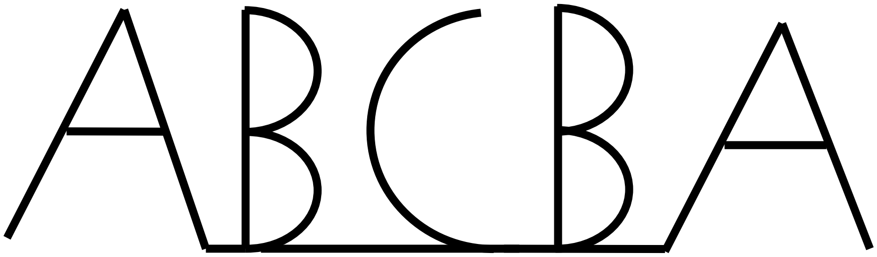


Shape 2:



Shape 3:

(Drawn any size you want)



Robots for Beginners

Project 8: The Monitor (no Merrimack?)

Let's see…really!

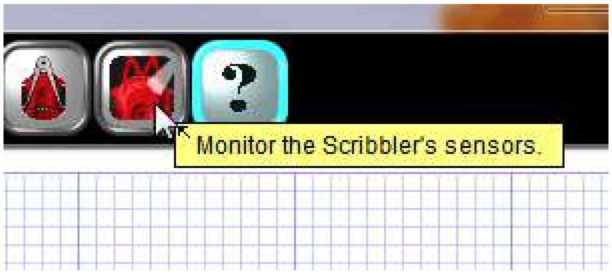
Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Get your copy of the "S2 Robot Startup Guide". While looking at the "TOP VIEW" and "BOTTOM VIEW" diagrams, find the following items on your S2:

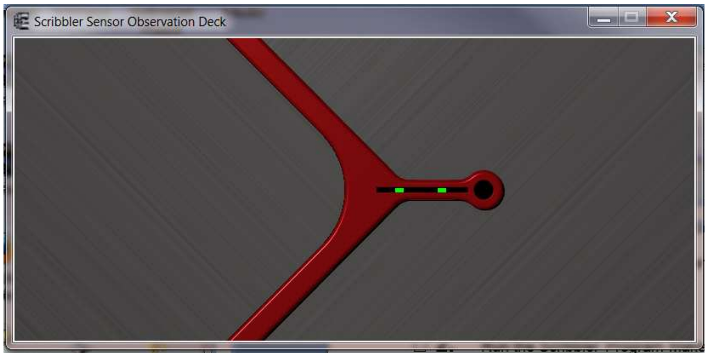
a. Infrared (that's Infra-Red) emitters and detector. These detect obstacles in front of the S2
b. Light sensors. These look up and ahead to see light from left, right and middle.
c. Line sensors. These look down to see dark and light areas on the table/floor.

2. Run the Scribbler Program Maker, ensure that your S2 is attached and powered-on.

3. On the Menu Bar select "Monitor the Scribbler's Sensors" as shown below:

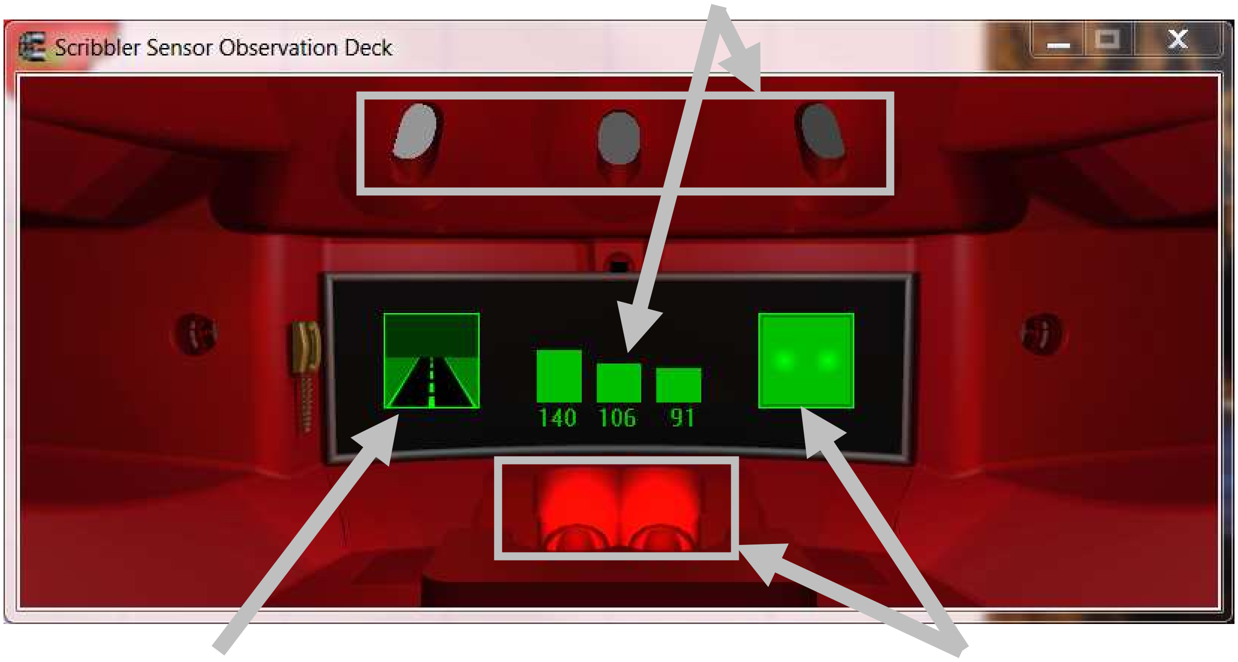


4. The following screen will appear while the "Sensor Observation Deck" is uploaded to your S2:



5. The following screen will appear once the sensor monitoring program is uploaded. Please refer to the labels to better understand which elements refer to which sensors. In the next few steps we'll see how these work.

Light sensors
(left, middle and right)



Obstacle sensors
(left and right)

Line sensors
(left and right)

6. Place your S2 in a location where light can strike the **Light sensors**. With your finger, cover each sensor (one at a time) and see that the light value diminishes for the respective sensor. Also notice that the light sensor values are displayed in two places on the monitor.

7. Place your hand or other object approximately six inches in front of your S2. Notice that the **Obstacle sensors** show a "brick wall" to indicate that the obstacle is there, and as that you move the obstacle (or S2) left and right, the wall shows that the obstacle is on the left, straight ahead, to the right, or not there at all. Play with this for a while to see how it operates.

8. Place your S2 over a sheet of paper with a black line on it. As you move the paper into view of the **Line sensors**, you should see changes in the Line sensor displays. If the sensor sees white, the monitor shows it as "lit". If the sensor sees black, the monitor shows it as "unlit". Move your S2 around over the line and sheet, get the feel for what it sees.

Question for exploration: Since the S2 can see light, objects and lines, what kind of Action block do you think allows the S2 to *<u>use</u>* *that information to affect its behavior?*

Hint: Search the Action Bar for a likely candidate.

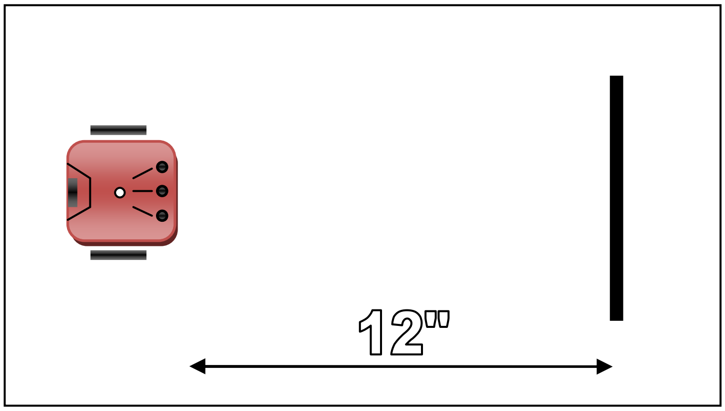
Robots for Beginners

Project 9: Stop at the Line

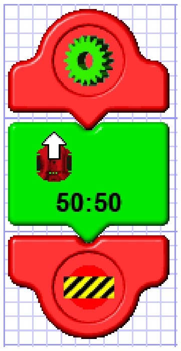
First "exposure" to using line sensors in your programs

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

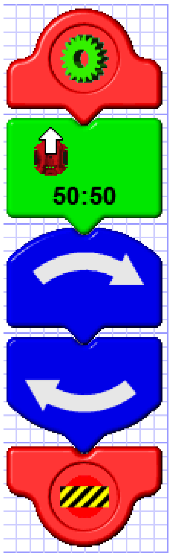
1. Prepare your work area as shown below. You will need room for your scribbler to travel in a straight line, and there should be a line across its path.



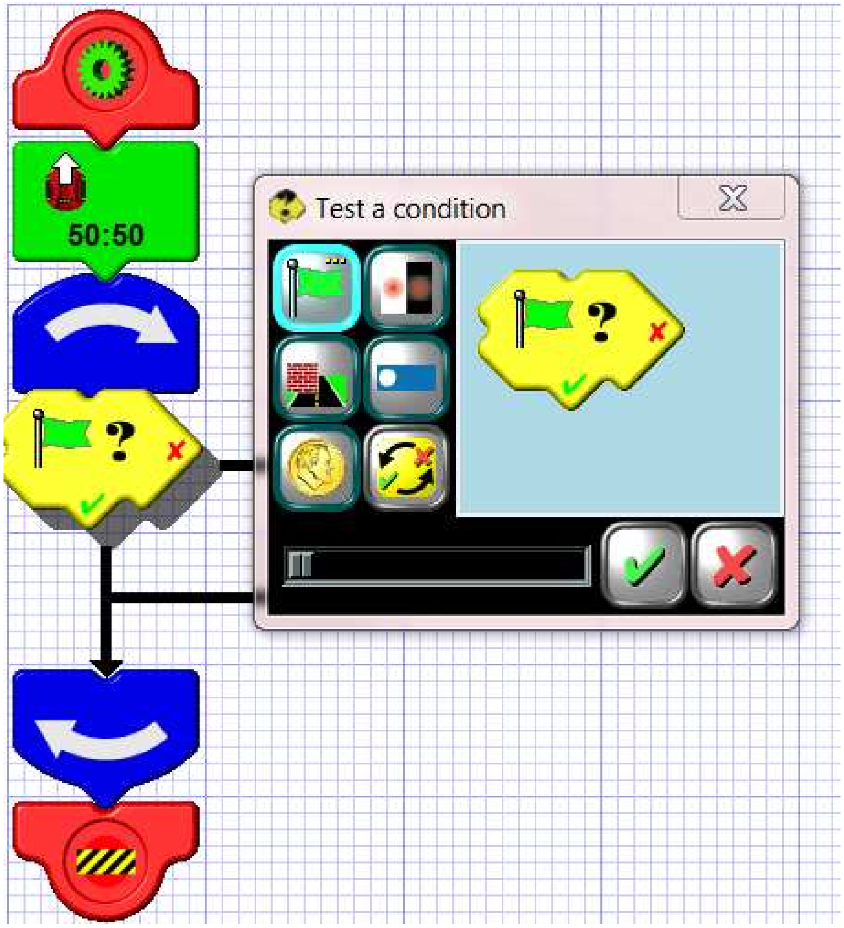
2. Starting with a clear worksheet, create the following program:



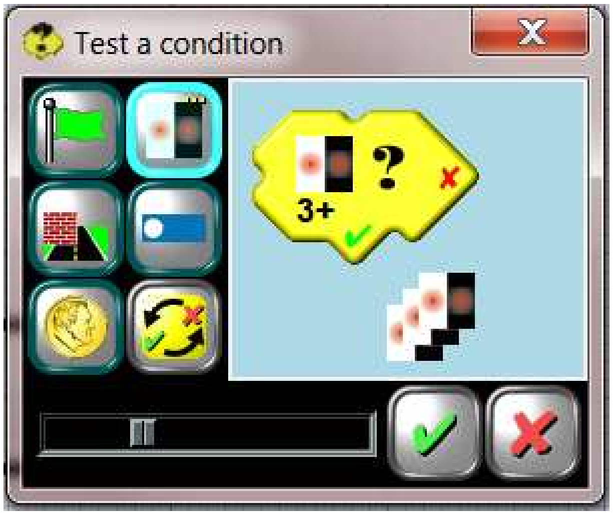
3. After the Move block, insert a loop as shown. Set it for a loop count of "000" (zero), this will make it loop forever:



4. Insert a "Test a condition" block inside the loop, a block and the "Test a condition" settings window will appear:



This is how you instruct the S2 to "branch" (do one thing or another) depending on what is being sensed at the moment. You can also branch depending on the how a flag is set (more on this later). Let's look more closely at this window to see what your choices are:



Condition or sensor to observe

Illustration

Number of required consecutive observations (more on this later)

5. The conditions or sensors you can observe are:

Check status of a flag

Check Line sensors

Check Obstacle sensors

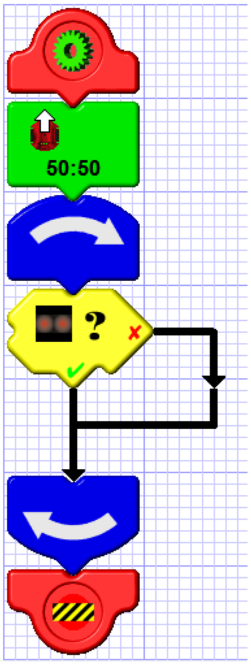
Check Light sensors

Select the Line sensor, click multiple times and notice that you can check for:

White (no line) Left sensor Right sensor Both sensors

There are other settings for the line sensor that pertain to reading bar codes, we will bypass them for now.

Select the icon for **Both sensors** and click OK (the check mark). Your program should look as follows:

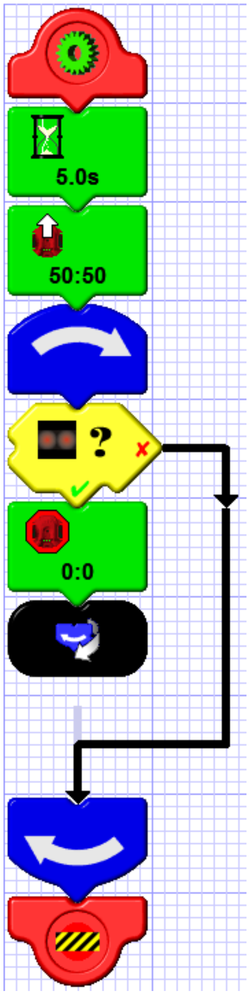


As shown to the left, you have inserted a “Test” block into your program. Look closely at the block. There is a check mark on the bottom and an ‘x’ on the right.

Any blocks you insert below will only be executed if the condition is TRUE, if both sensors see black.

If this condition is NOT TRUE (FALSE), it will branch to the right and execute whatever blocks are there (if any). This condition will occur if one or both sensors don’t see black.

Since we want our robot to stop when both line sensors see black at the same time, we add code for the motors to STOP when this occurs.



6. Insert a Move block below the Test block to tell your motors to stop turning. Immediately below that, insert an “Exit from loop” block.

Lastly, as we have done before, add a 5 second pause to the start of your program to give you time to get set up. Your program will now appear as shown to the left.

Observe that if the condition is TRUE, your motor will stop.

Questions:

1. Why did we add the “Exit from loop” block?

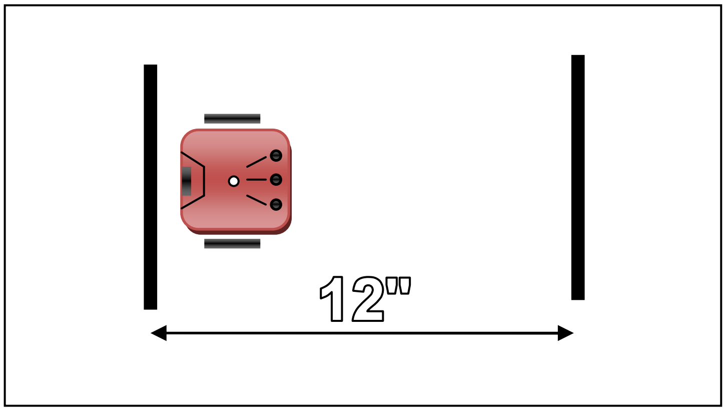
2. What happens if the condition is FALSE?

7. Upload this code to your S2 and test. Does it behave as expected?

8. Add code to your program to have your robot turn around 180 degrees and face the other direction when it comes to the line. Upload and test.

9. Add code to your program to have the robot travel approximately 12 inches after turning around. Upload and test.

10. Add a line to your setup as shown below. Modify your program to make your robot travel to a line, turn around, and travel to the next line and stop. Upload and test.



11. Save your program and when ready begin project 10.

Revision: 1.0 Date: 6/7/2011

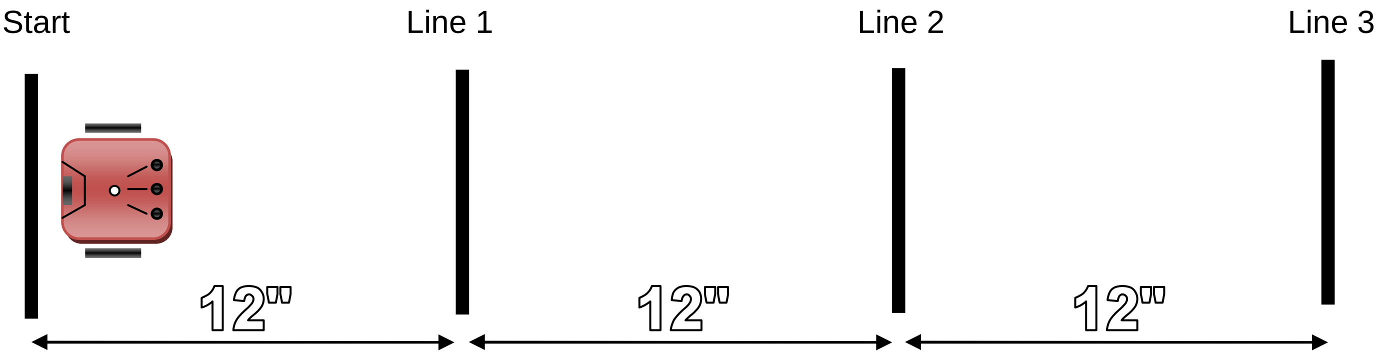
Robots for Beginners

Project 10: Running Ragged (some it "Suicide")

More structure to your programs

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Prepare your test track as shown below.



2. Create a program that does the following:

A. Beginning in front of Start, travel to Line 1.
B. At Line 1, turn around and return to Start.
C. At Start, turn around and travel to Line 2.
D. At Line 2, turn around and travel to Start.
E. At Start turn around and travel to Line 3
F. At Line 3, turn around and travel to Start.
G. At Start, stop.
H. Do a little dance while playing music (not you, silly, your robot!)

3. Demonstrate the finished program to your instructor.

Hint: You may find subroutines and loops helpful.

Save your worksheet and move to project 11.

Robots for Beginners

Project 11: Follow the Black Brick Road (sorry Dorothy)

Intro to Line Following

You know by now that your S2 can tell when one or both sensors are over a dark area, such as a line. In this project you will use the fact that you have a pair of sensors, one on the left and one on the right, to create a program that follows a curvy line, making corrections as needed.

Assume that your S2 starts over a line, that the line is wide enough to cover both sensors, and that your S2 begins to move forward in a straight line. Initially, your robot sees black with both sensors.

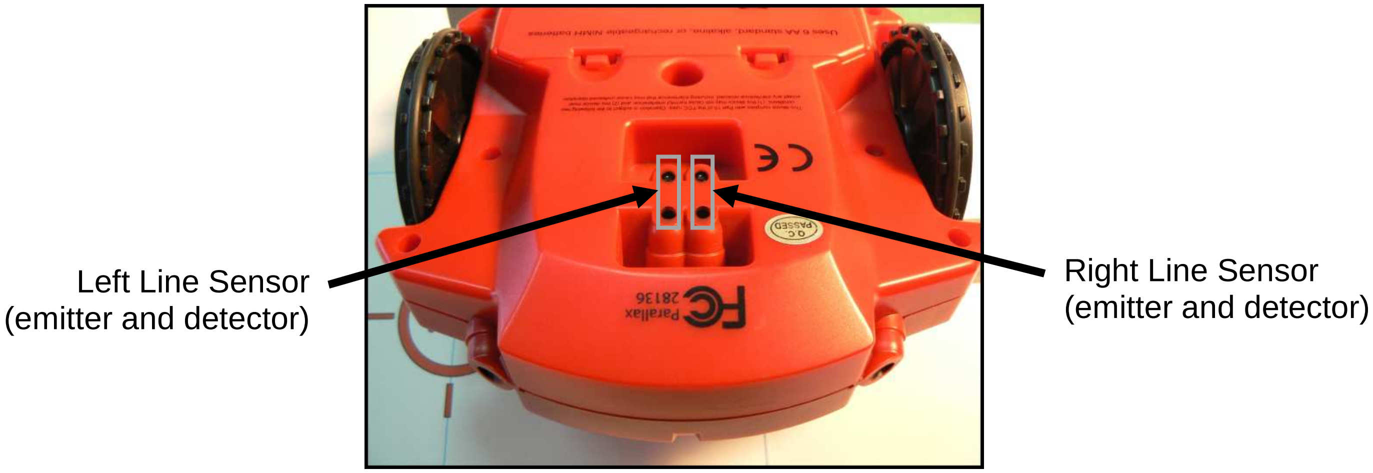
Consider the following questions:

1. If your left sensor suddenly stops seeing the line (goes white), what does that mean? What should your S2 do to compensate?
2. Conversely if your right sensor suddenly sees white, what does that mean? Again, what should your S2 do to get back on track?
3. If your S2 suddenly sees no line at all, what can you do to try to get back to the line?

Let's explore questions 1 and 2 above, and save number 3 for a future project.

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Pick up your S2, flip it over and look at the **bottom**. You will see a pair of optical emitter/detectors as shown in this photo:

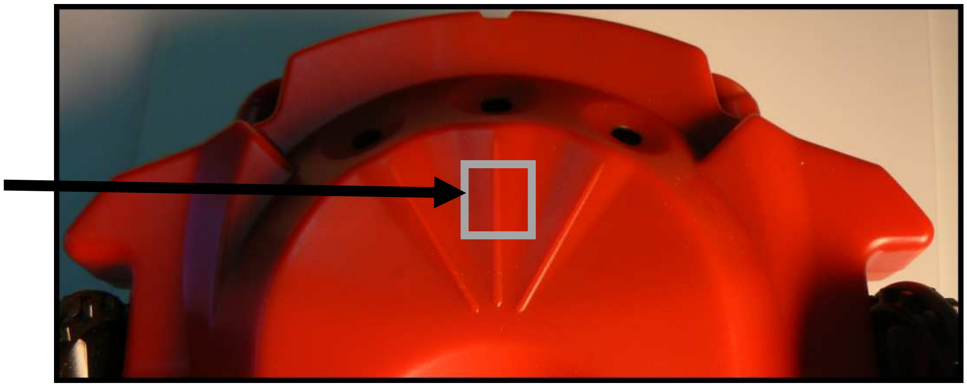


Left Line Sensor
(emitter and detector)

Right Line Sensor
(emitter and detector)

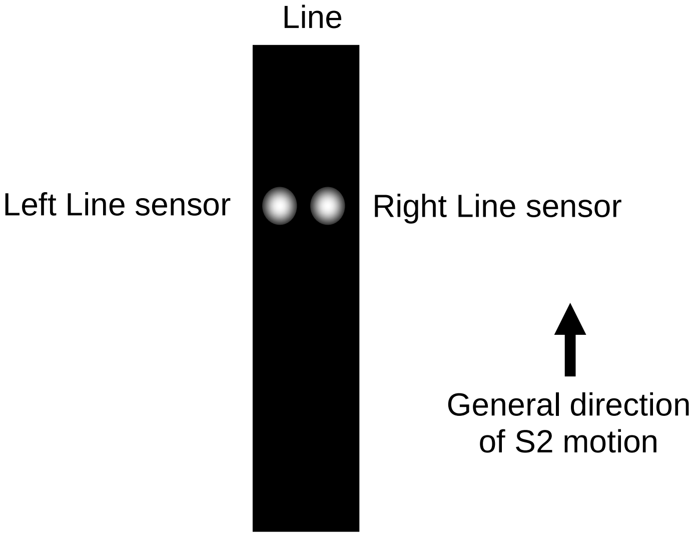
2. Turn the S2 back over and make a mental note of the spot on the TOP of the S2 which is directly above the Line sensors. If necessary, turn the S2 over and back a few times to confirm. This will help later when you want to put the S2 over a line, or analyze what it's seeing while watching it move. If it helps you can make a mark on the top at this position.

As seen in this photo, the sensors are approximately below this position as shown:



Line sensors are directly below this area

3. Look at the following illustration, what do the sensors see?

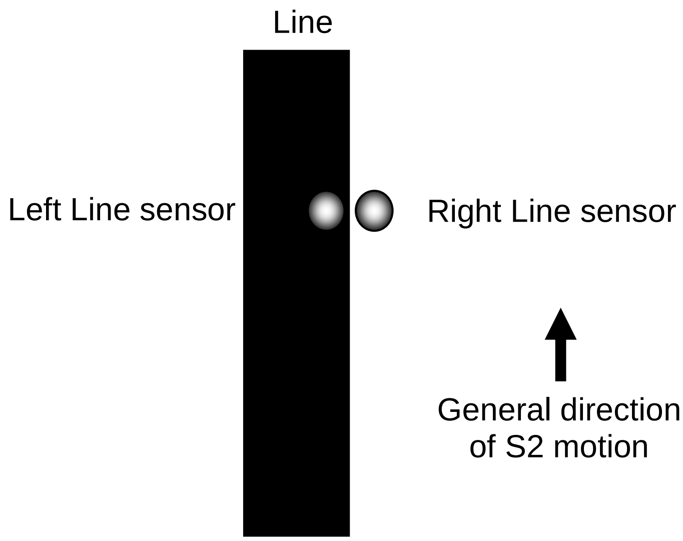


If your goal is to have your S2 follow the line, what should it do in the above instance?

Can you imagine programming logic that will cause this to happen?

Hint: Remember the Test blocks?

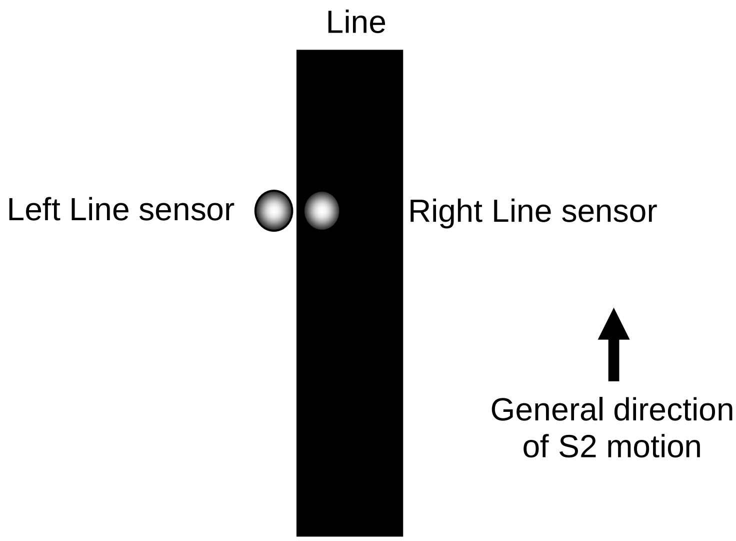
4. Look at the following illustration, what do the sensors see?



If your goal is to have your S2 follow the line, what should it do in the above instance?

Can you imagine programming logic that will cause this to happen?

5. Look at the following illustration, what do the sensors see?

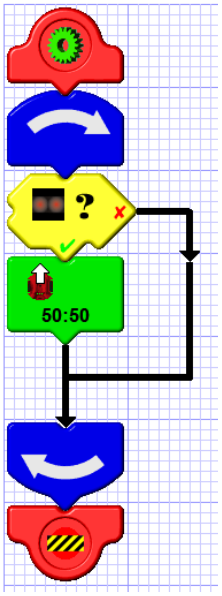


If your goal is to have your S2 follow the line, what should it do in the above instance?

Can you imagine programming logic that will cause this to happen?

Note: As mentioned earlier, we will ignore for now the case of when NEITHER Line sensor is over a line (both see white). That's something we will consider in a future project.

6. Starting with a blank worksheet enter the following program:



Notice that we now have an infinite loop (never stops), with a single test inside it.

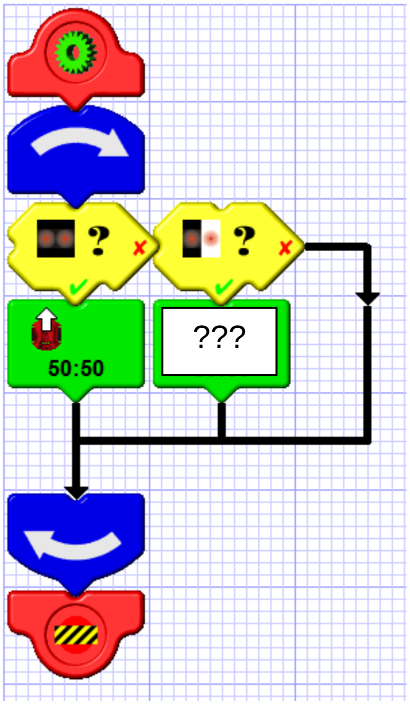
What does this test look for?

What will the S2 do if the test is TRUE?

What will the S2 do if the test is FALSE?

If you look at this short (unfinished) program you will see that it tells the S2 to go straight (and keep going straight) as long as both Line sensors see black. Now let's add code to the right of the test to handle what happens if some other condition exists.

7. Add a second Test block to the program as shown below, with another Move block. Note that in this illustration the Move block is obscured. It is up to you to determine what kind of move is appropriate for this case:



Observe how a second Test block has been added to the right of the first one. This allows you to string a number of tests, each one set to watch for a certain situation.

Based on your earlier conclusions, edit the Move block to cause the S2 to move in a manner that will (hopefully) compensate for the new case.

What other cases do we need to test for?

8. Insert additional Test and Move blocks to take care of the remaining cases.

9. Upload your program to the S2 and test it on the simple oval test track. Does it follow the line? What do you need to do to your program (adjusting move speeds, other?) to make it perform better.

10. Continue adjusting your program until you are satisfied that it can reasonably follow the simple track.

11. Try your S2 on a more complex shape. How well does it perform? What happens when it gets to a sharper turn?

Congratulations, you just made your first line following program.

Save the worksheet and when ready proceed to Project 12.

Robots for Beginners

Project 12: Get Better at Following

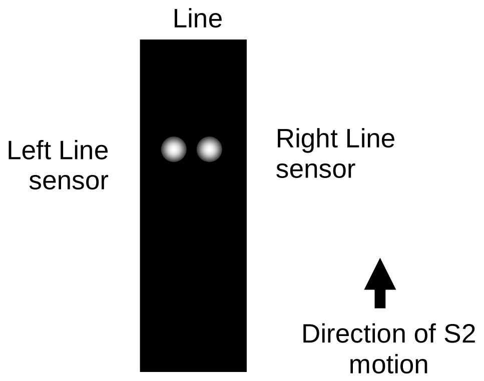
How to bullet-proof your algorithm (my "algo" what?)

Hopefully you found in project 11 that your S2 can follow lines reasonably well. You also probably noticed that there were times when your robot got "lost", moved off the line entirely, and never came back. In this project we will explore a couple of ways to improve the situation which, by the way, will also expose you to some pretty powerful programming techniques.

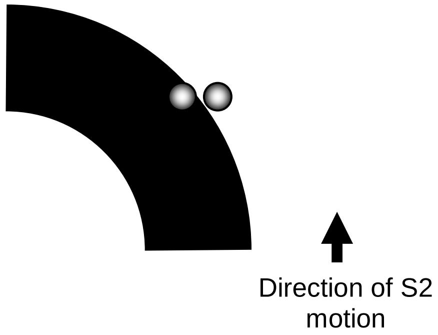
In case you didn't know by now, we're beginning to develop what is called an "algorithm", which is defined as "a process or set of rules designed to achieve a given task, usually within a computer program".

Let's look at a series of events in your line following that needs special attention.

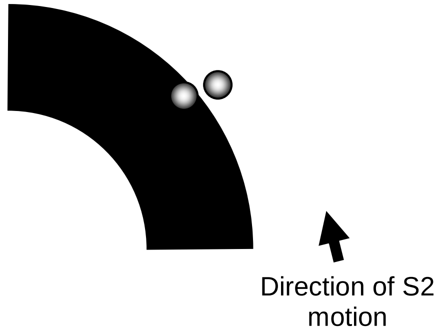
Imagine that your robot is tracking nicely on a straight line, as shown below:



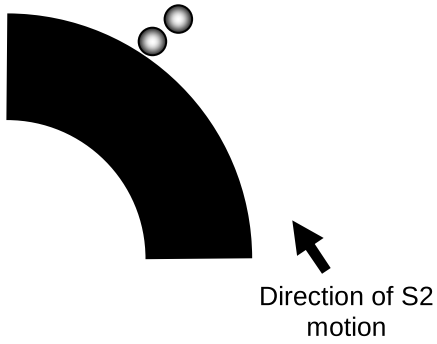
Now imagine that your S2 comes to a sharp curve in the line as shown:



The right Line sensor is now seeing white, so your simple program instructs your robot to start turning left to compensate. This is shown in the illustration below:



Unfortunately the line curves faster than your robot is turning and your robot comes completely off the line as shown:



If nothing changes, your robot has now permanently lost the line. What will you do now?

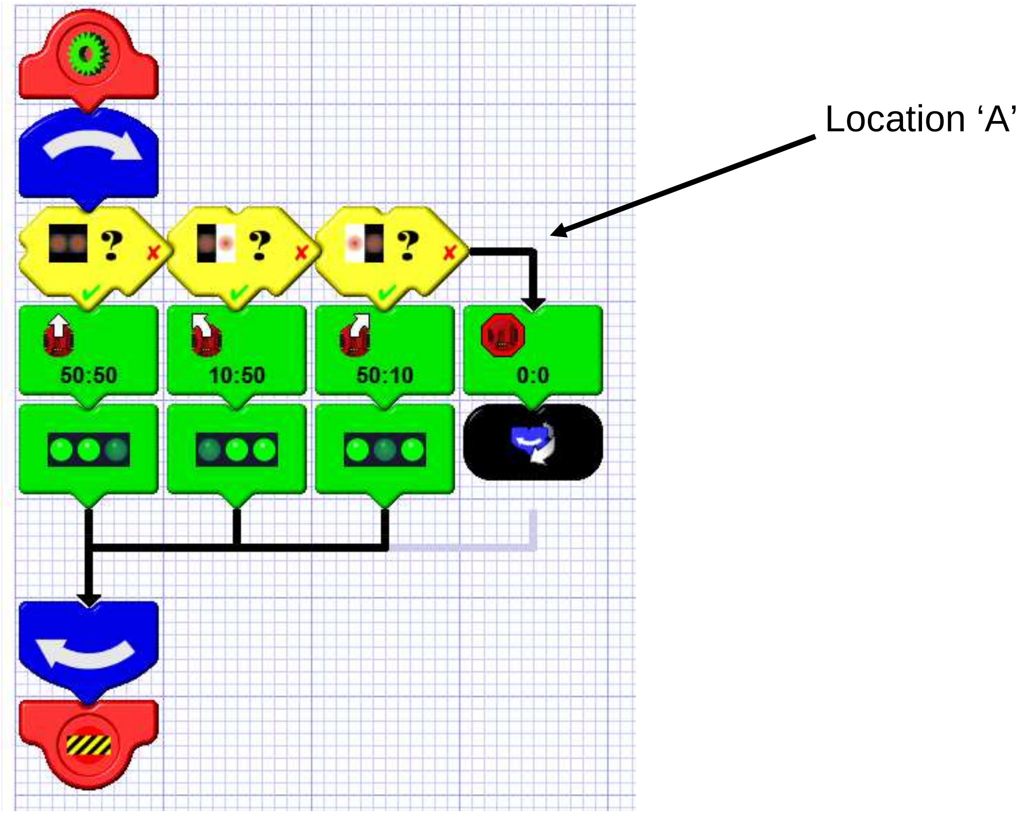
This is where algorithms come in. We have a situation that requires a bit of head scratching and innovation. If you guessed we need to “hunt” to find the line you are correct.

Think of the events that led up to the loss of line. Now that we’re off the line, what could the robot do to find the line again and get back on track?

In the following steps we will explore one algorithm that can take care of this situation.

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Load the final program you wrote for project 11. It probably looks something like this:



Just to review, in this program the possible conditions and resultant behaviors are:

Both sensors see black → go straight
Left sees black, Right sees white → turn gently left
Right sees black, Left sees white → turn gently right
Both see white → stop and exit loop

Question*: Why isn't there a test for both sensors seeing white?*

Answer*: If you think about it, we've tested for all other possible conditions. The only way we can get to the far right (Location 'A') is when both sensors see white, thus we don't need a special test for it.*

2. Think about how one might get back to the line. Perhaps it will help to consider what we were doing JUST PRIOR to losing the line. In this example, we were turning left, but just not hard enough. One way to get back to the line is to turn much sharper to the left, but how will we know (at the moment we got all-white) what we were doing most recently. This is where "flags" come in.

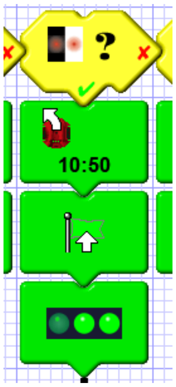
A flag is a small bit of memory that can be "SET" or "CLEARED", and that our program can look at later to see its status.

Let's add code to the program that will SET or CLEAR a flag each time we are turning left or right. We will state (arbitrarily) the following:

SET indicates we most recently turned left
CLEARED means we most recently turned right

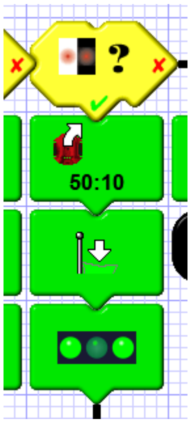
Look at the Action bar for an icon that looks like this: This is the "Raise or Lower a Flag" action block. Raised means SET, Lowered means CLEARED.

Modify your code to include a SET flag as shown in the left turning section:



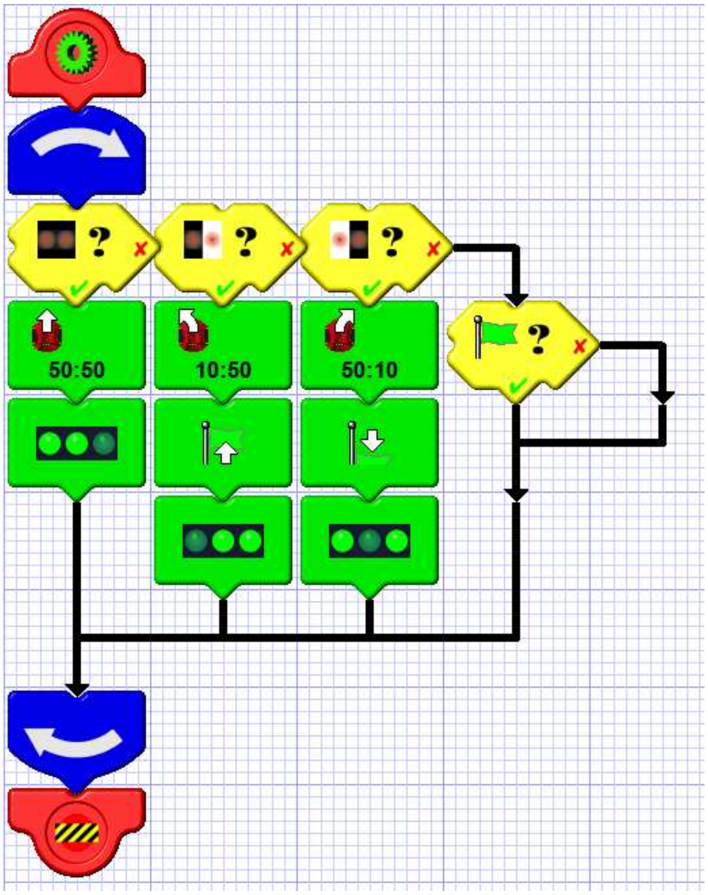
This will SET the flag each time your sensors cause your code to turn left.

3. Modify your code to CLEAR the flag in the right turning section as shown:



We now have a flag whose condition reflects the most recent turn.

4. Replace the old "all white" code on the right with a Flag Test block. If the flag is SET, the code below the test will be executed. If the flag is CLEARED the code to the right will be executed. Your code should look like this:



Information: The green flag is just one of several color coded flags that are available. If you have additional information you want to store in a flag, you can use one of the other colors.

5. Add code to each leg of the Flag Test block. It's up to you what you want your robot to do, what do you think will make the robot get back to the line?

6. Upload and test your code, modify as necessary so that it successfully navigates a test track with tight turns, both left and right.

Question for further thought: What happens if your robot is going straight and both sensors suddenly see white at the same time? This could happen at a sudden right or left turn, or if the line just ends. What changes or additions might you make to your algorithm to compensate? What conditions might cause that new algorithm to fail? These are the kinds of things professional programmers think about.

Congratulations, you just improved your line following program.

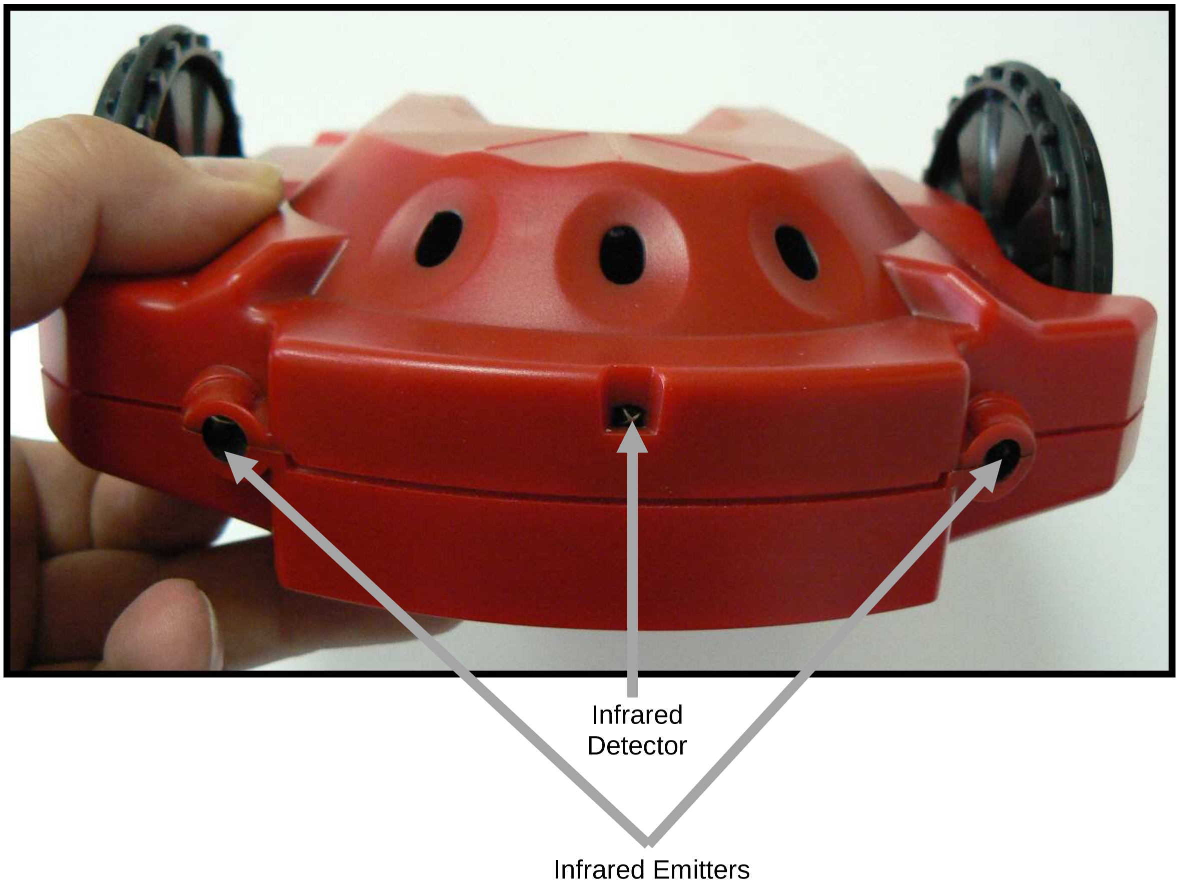
Save the worksheet and when ready proceed to Project 13.

Robots for Beginners

Project 13: Feeling your way

First experience with front-mounted sensors

No robot would be complete without a way to detect obstacles, and the S2 is no exception. If you look at the front of your robot, you see the obstacle sensor array (handsome little devil, isn't it?):



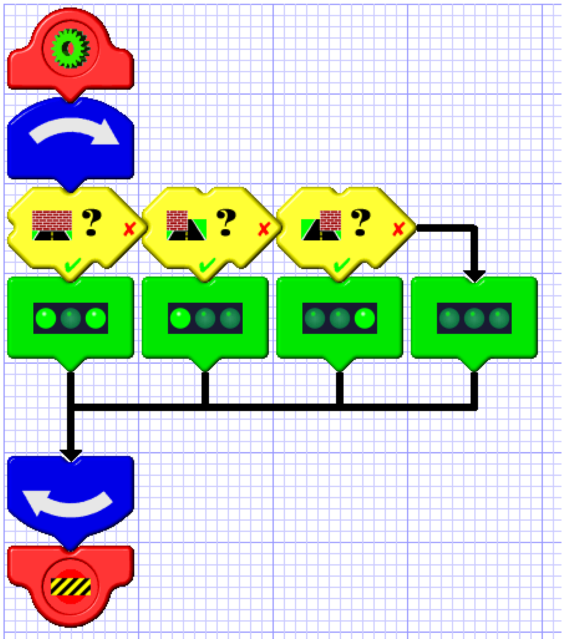
According to the Parallax documentation, the emitters bathe the scene in front of the robot, one on the front left, the other on the front right. The detector receives reflected light coming back from an object or surface that's nearby.

By alternating the pulses of infra-red light between left and right emitter, the detector can tell whether the object is ahead to the left or the right, and your program can read out this information.

For example, if the right emitter sends out light and the detector receives it back from the "bounce", it knows that an object is ahead to the right.

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Clear your worksheet and enter the following program. Study it until you understand how it works:



Just to review, in this program the possible conditions and resultant behaviors are:

Both sensors see an object → left and right LEDs lit
Left sees object, Right sees none → left LED lit
Right sees object, Left sees none → right LED lit
Neither sees and object→ no LEDs lit

Question*: Why isn't there a test to see if <u>neither</u> sensor sees an object?*

2. Upload the program to your S2 and test. Put your hand in front of the S2 on one side, then the other, and then directly in front. Put your S2 near a wall, does it behave as expected? Repeat this until you have a clear idea of how the sensors work.

3. Modify the program to do the following:

a. When an object is detected by BOTH sensors, back up a little, turn around 180 degrees and go straight the other way.

b. When an object is seen by ONE sensor, back up a little, turn 90 degrees away from that side and then go straight.

c. When no object is seen, just keep going straight.

4. Upload and test on the floor, being sure to protect your robot from falls or being stepped on!

Questions:

A. Did your S2 get stuck anywhere? If it did, how would you change the program to make it adapt better?

B. Did your S2 travel around or did it pretty much stay in one area? How would you make it more able to "tour" the environment?

C. What happened when your S2 hit a wall or other object? Did it keep pushing?

In the next (and final) project we will improve the touring algorithm to make your S2 perform better in a complex environment.

Save the worksheet and when ready proceed to Project 14.

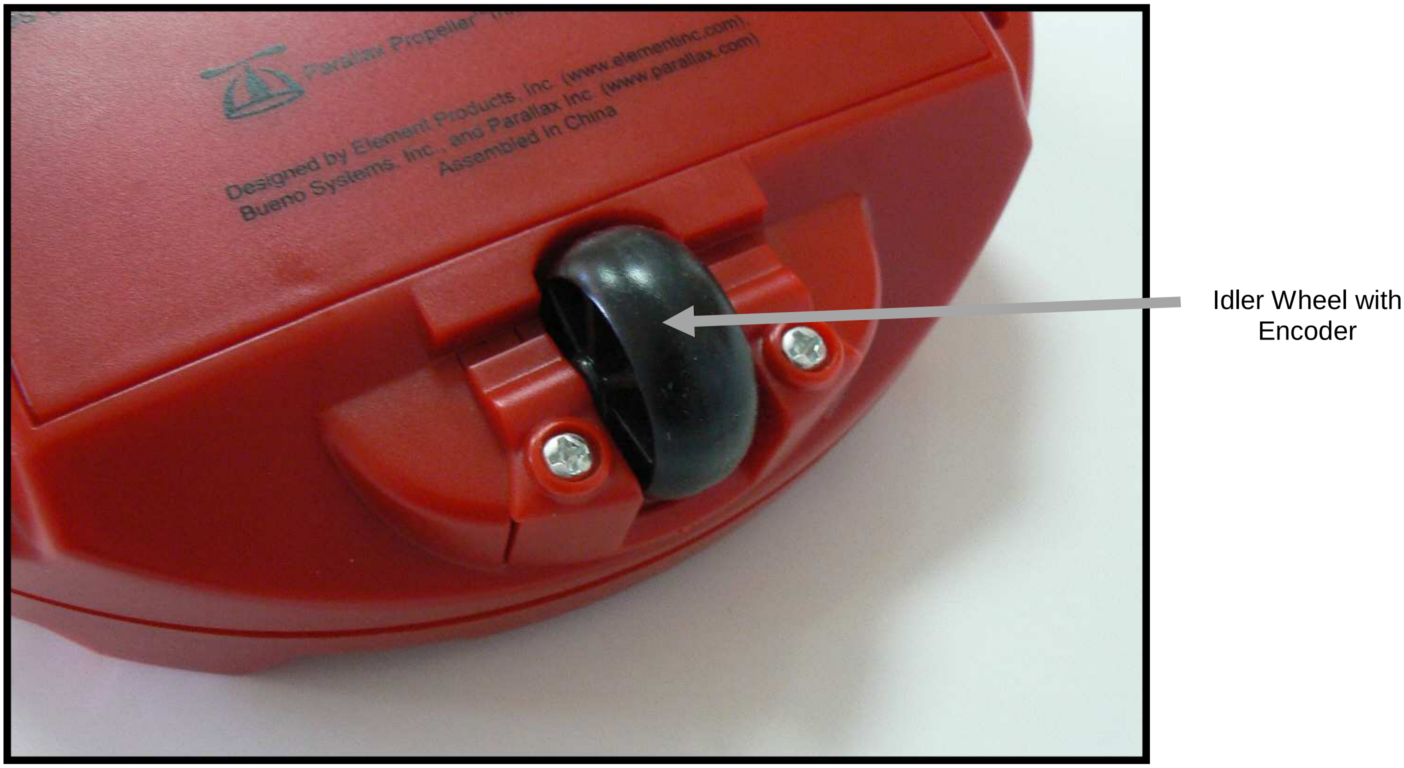
Robots for Beginners

Project 14: Obstacle Avoidance

Roomba's got nothing on us, baby.

Here's your chance to explore obstacle avoidance and sharpen your algorithm skills. In this project you will improve your "touring" algorithm (with only a little guidance). The main thing is to watch CLOSELY as your robot moves around the floor, encountering objects and reacting to them.

There's one more basic feature you need to know about which we will explore first. Take a look again at the bottom of your S2, you see the two main driving wheels and a third free-turning idler wheel in the back of your robot. The idler wheel has an "encoder" that tells the robot when and how fast the robot is moving.

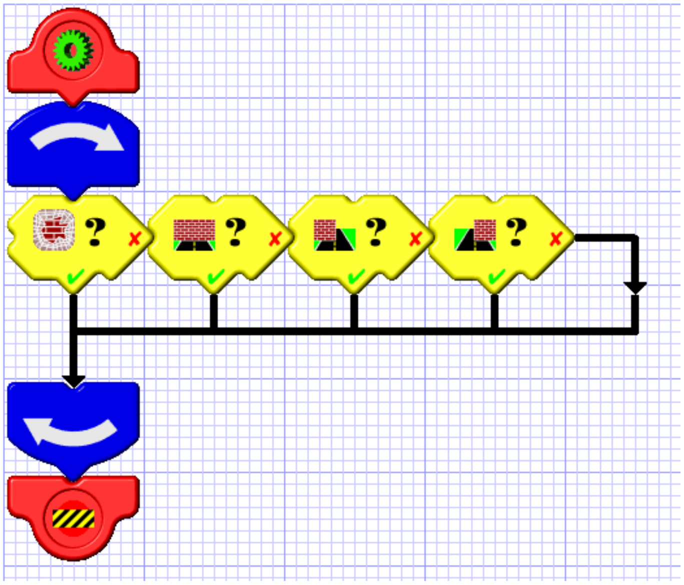

Parallax Propeller
Designed by Element Products, Inc. (www.elementinc.com),
Bueno Systems, Inc., and Parallax Inc. (www.parallax.com)
Assembled in China
Idler Wheel with Encoder

If by some circumstance your robot is unable to move, the idler wheel will not be rotating and the robot brain will know there's a problem.

This is called the Crash sensor, and like other S2 sensors, it can either be TRUE or FALSE, where TRUE means the robot is stuck, and FALSE means it's not.

Please do the following in step-by-step fashion. When you have finished each step please check it off with a pencil ☑.

1. Create a clear worksheet, add the following blocks:



Notice the addition of the "Crash" test block at the far left. The reason it's placed there is to make sure that it is seen, it's really important. If it was further to the right, it might not even be tested (think about it).

2. Create code to handle each of the five possible situations (yes five!). Experiment with your algorithm, try different options, speeds, tests and so on.

The goal is to make your robot able to move around the room, never get stuck, turn as little as needed and explore as much as possible.

Save the worksheet and SHOW OFF YOUR WORK!

Congratulations on completing the final project, now it's time for the final competition.

Robots for Beginners

Competition 2: The Candy Run

First one home…YUM!

Operational Notes: The size of each "alliance" should be either one or two persons per robot, depending on the number of participants and robots available. Your instructor will determine how to divide you up.

If not all participants have finished through Project 14, your instructor will modify the challenge to only include material through project 12. In this case, someone must monitor the robot and manually stop it, give a treat, and turn it around at the end of the track.

The challenge –

Your instructor will lay out two equal tracks side by side. Each track has a line to follow and an obstacle at the end. In a series of "heats" one robot competes against another, each starting at the same time.

In each heat the robots travel from the starting line to the obstacle and stop (for 5 seconds). The first robot to get to the stop has a treat added to its cup. After the 5 second pause, the robots turn around and travel back to the start, all without operator intervention.

The winner of each heat is promoted to the next round, the loser is eliminated.

The champion gets a special treat (tbd).

Rules:

1. You may have as many practice runs on the track as you like.
2. You may refer to programs that you (or your partner) wrote earlier.
3. To win, you must get to the endpoint, stop, and return to the start without help.
4. If in a given heat the winner does not make it back to the start, and the other robot does make it back, the second place robot becomes the winner, claims the treat and goes on to the next round.
5. If neither robot finishes, the heat is canceled and repeated when the teams are ready.

Prize: ***Munchies!***