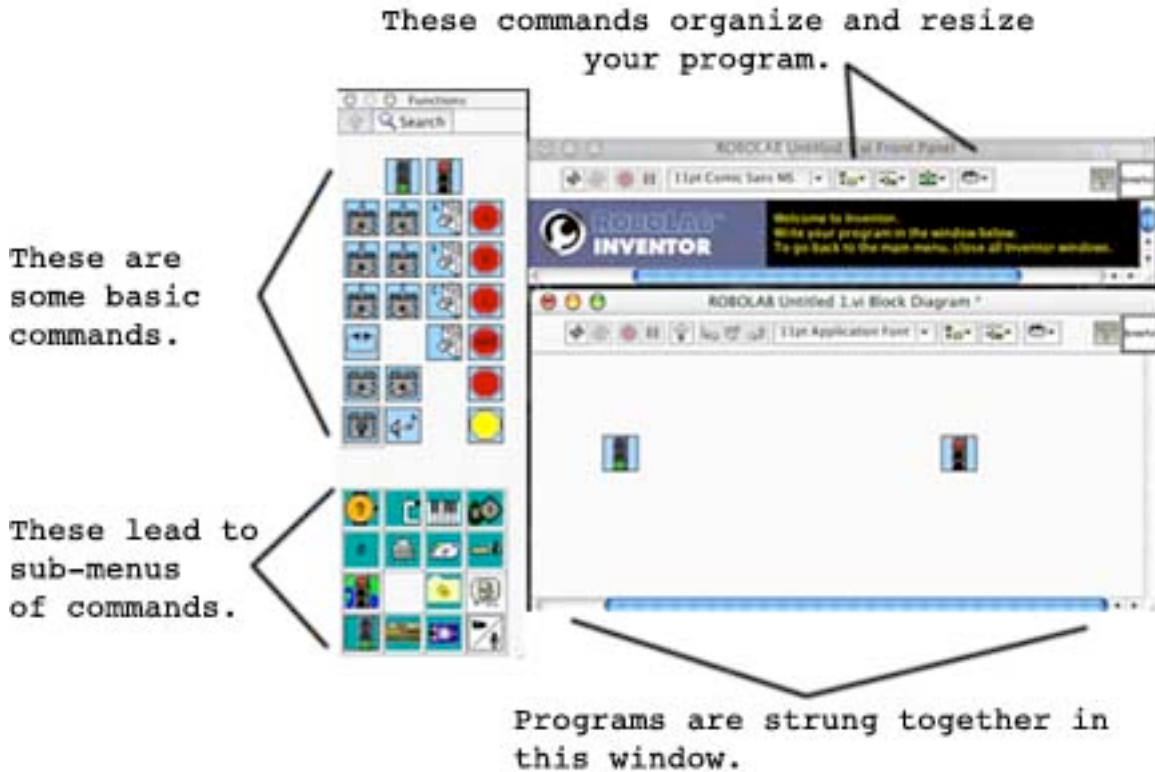


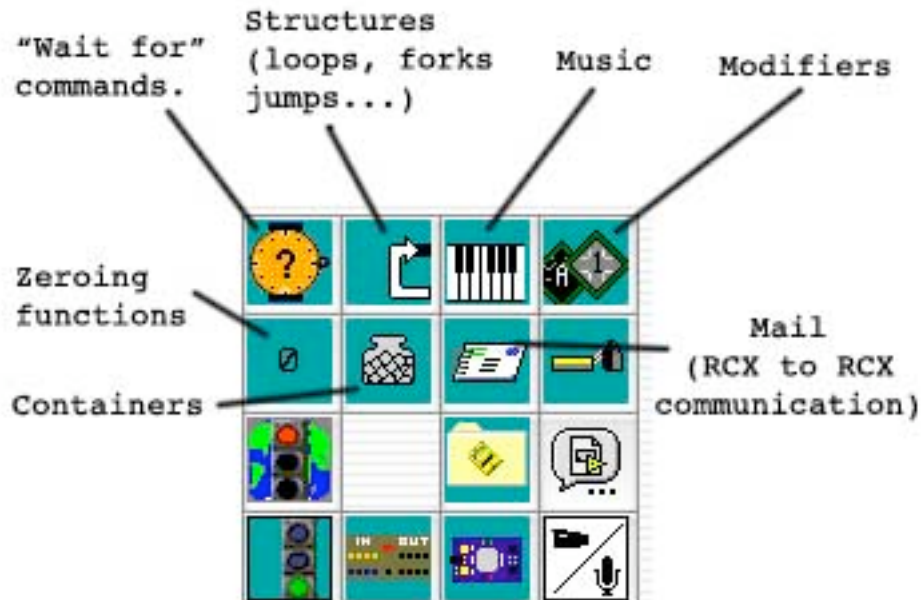
Introduction To Robolab

All programs made for this lab will be in Robolab Inventor Level 4. This offers the greatest amount of flexibility with a graphical interface. Help is easily accessible by opening the context help window (Help> Show Context Help). Simply hold the cursor over a command to see detailed information about it in the context help window.

Three main windows are used in Robolab:



The most commonly used sub-menus:

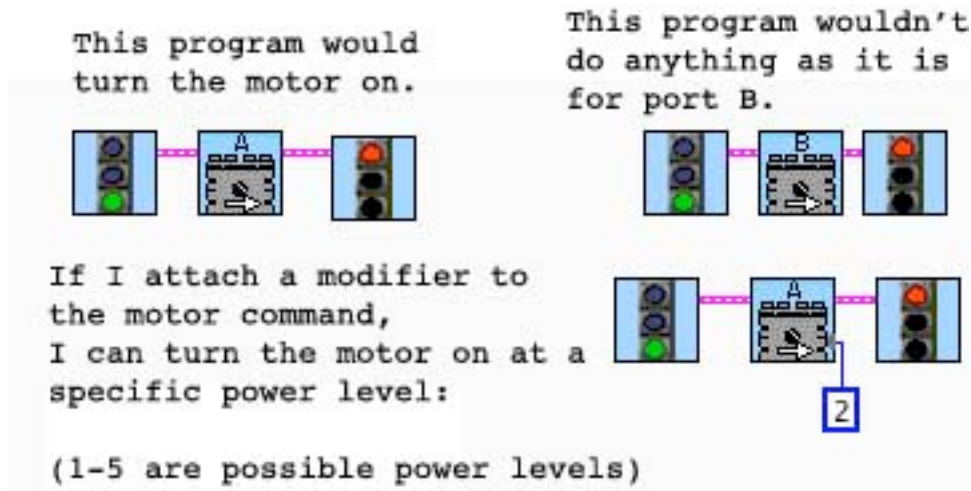


In Robolab the cursor is four tools: Select tool, Hand tool, Text tool, and String tool. The Select tool allows you to select icons to move or delete. The Hand tool is used to drag and drop commands from the functions palette to the block diagram window. The text tool allows you to insert comments or values. The String tool is used to string commands and modifiers together. All programs in Robolab require an unbroken string of commands between the Start and Stop stoplights. To toggle between the four cursor tools use the tab key.

All RCXs need firmware in order to compile and run programs. This needs to be downloaded every time the batteries are changed. Firmware will be downloaded when you try to download a program for the first time. To download a program to an RCX simply turn it on, place it in front of an IR tower, and click Run:



All RCXs have three inputs (1 2 3) and three outputs (A B C). This means your program must always specify which port a command refers to. For example, if I have a motor connected to output A:

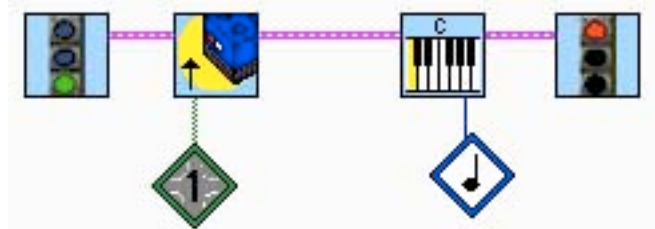


The previous programs tell the RCX to turn on a motor, but not to turn it off. This program turns on motor A and C, waits 4 seconds (letting them run), and turns them off:

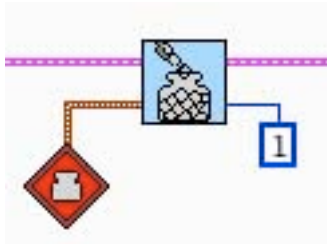


Tip: "Floating" the outputs (yellow stop sign) lets you avoid an abrupt halt by rolling to a stop.

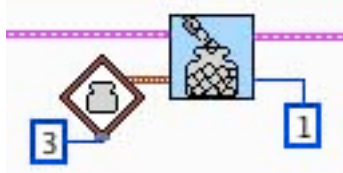
Commands using sensors also have to specify ports. This program tells the RCX to wait for the light sensor connected to port 1 to measure an increase in light, then beep (middle C for one quarter note):



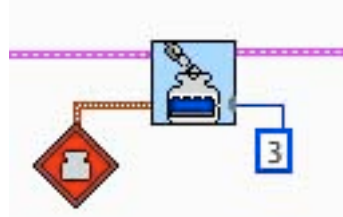
Numeric values are stored in containers in Robolab. There are three coloured containers. You can also use numbers 3 to 22 for generic containers. There are also special containers for sensor values:
This stores a value of 1 in the red container:



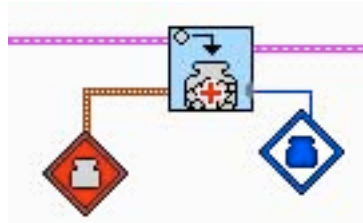
This stores a value of 1 in the generic container number 3:



This stores the value of the light sensor connected to port 3 in the red container:

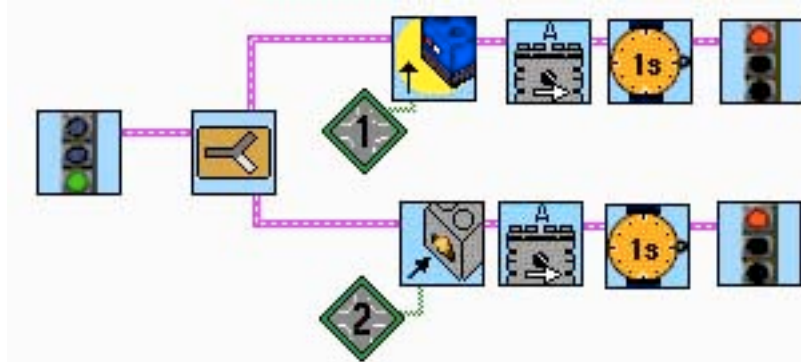


This adds the value of the blue container to the red container:



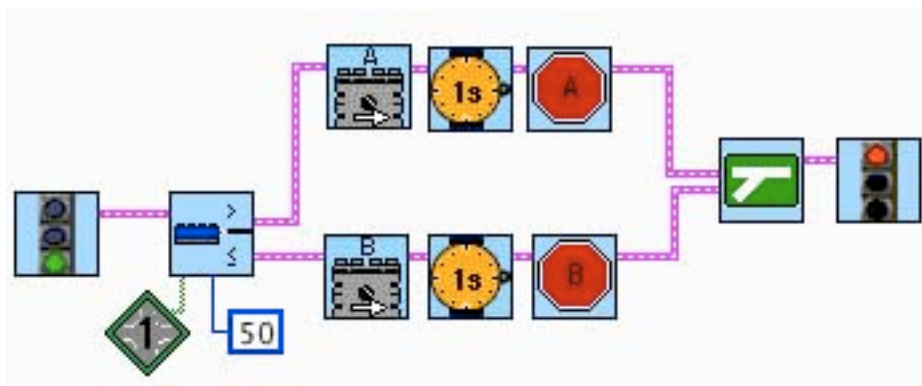
There are a number of structures used in Robolab programming:

To have a robot do two things at once, use a task split. This program waits for the light to increase (as measured by the sensor connected to port 1) and for the touch sensor connected to port 2 to be pressed in. When EITHER event occurs motor A is turned on for 1 second:



Forks can be used to compare values:

In this program, if the value of the light sensor connected to port 1 is greater than 50 motor A turns on for 1 second, when the value is less than or equal to 50 motor B turns on for 1 second:

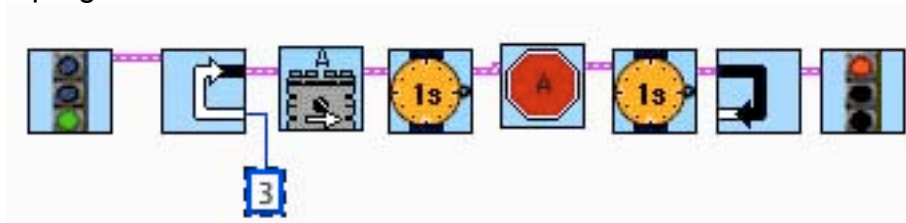


You can move from one part of a program to another using jumps and lands. There are six different coloured jumps.

In this program motor A turns on for 1 second. The robot then waits 1 second before jumping back to earlier in the program. This is an infinite loop.



If you want to execute a section of code a certain number of times use a loop.
 In this program motor A will turn on and off three times:



You can also have a loop execute while a certain condition is met.

In this program motor A runs while the light sensor in port 1 reads a value less than 20 (port 1 is the default port, other modifiers can be used).



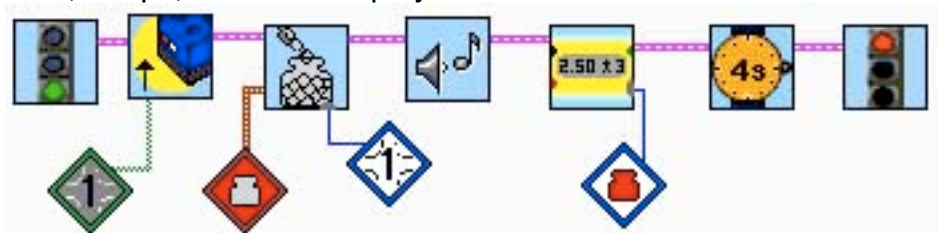
Viewing values:

When sensors are connected you can see the values they are giving by pressing the View button by the RCX display to toggle through all the inputs and outputs. Touch sensors give values of zero or one, and light sensors give a reading between zero and one hundred.

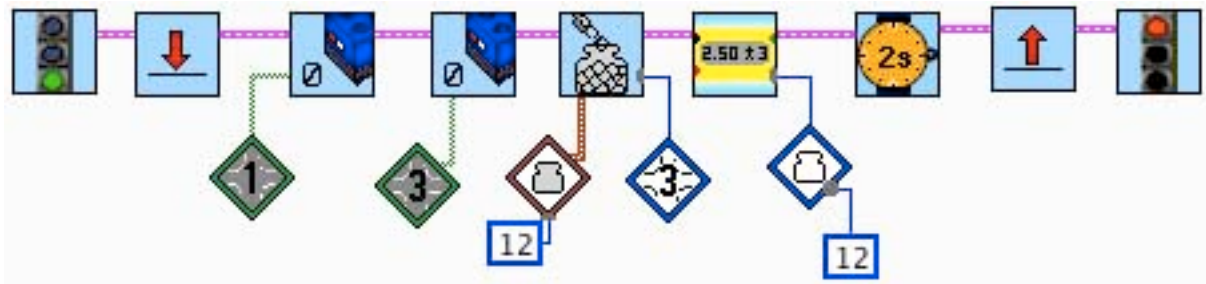
Troubleshooting Light Sensors:

Sometimes a light sensor will give what appears to be an impossible number (often 500-something), even though your program is good. Priming your light sensor should fix this. To prime your light sensor make a short program that immediately and directly calls for your light sensor to be used. Here are two priming programs that I have used in the past:

This program waits for an increase in light, then stores the sensor value in the red container, beeps, and then displays the value for four seconds:



I made this program because I had a robot with two light sensors, one of which was working while the other needed priming! This program zeros both light sensors, stores the value of sensor 3 in container 12, then displays the contents of container 12 on the display for 2 seconds, then repeats:



Questions

1. Consider methods of attaching wheels to your RCX using gears. When would you want a small gear on your motor axle and a large gear on your wheel? When would you want a large gear on your motor axle and a small gear on your wheel?
2. Attach motors to your RCX. Make a program that has it drive forward for 1 second and stop. Make a program that has it drive forward for 1 second and float to a stop. Test them on your robot. Do you notice a difference? When would a stop command work well? When would floating work better?
3. Attach a light sensor to your RCX. What is the sensor value of something black? What is the sensor value of something white? If the values are absurd (eg. over 100) create and use this code to prime your light sensor:



4. Create a short program that has a robot drive and turn. Test it on an RCX with wheels. Test it on the Lab Assistants' treaded robot. When would a treaded robot be advantageous? When might wheels work better?
5. How could you build a robot with a small turning radius?