

# Snek Lesson #1: Use Snekboard to Control LEGO Robots with Snek

## Table of Contents

License .....	1
Acknowledgments .....	1
1. Overview .....	1
2. First Look at Snekboard .....	2
3. Mu Editor (mu-editor) .....	3
4. Snekboard Cables and Sensors .....	6
5. Snekboard Outputs .....	8
6. Snekboard Inputs .....	9
7. Hooking Things Together .....	9
8. Switches .....	11
9. Servos .....	12
10. Next Steps .....	13

## License

Copyright © 2020 Keith Packard, Michael Ward

This document is released under the terms of the [GNU General Public License, Version 3 or later](https://www.gnu.org/licenses/gpl-3.0.en.html) [https://www.gnu.org/licenses/gpl-3.0.en.html]

## Acknowledgments

Thanks to Jane Kenney-Norberg's students for helping develop the material in this lesson.

Keith Packard  
[keithp@keithp.com](mailto:keithp@keithp.com) [mailto:keithp@keithp.com]  
<https://sneklang.org>

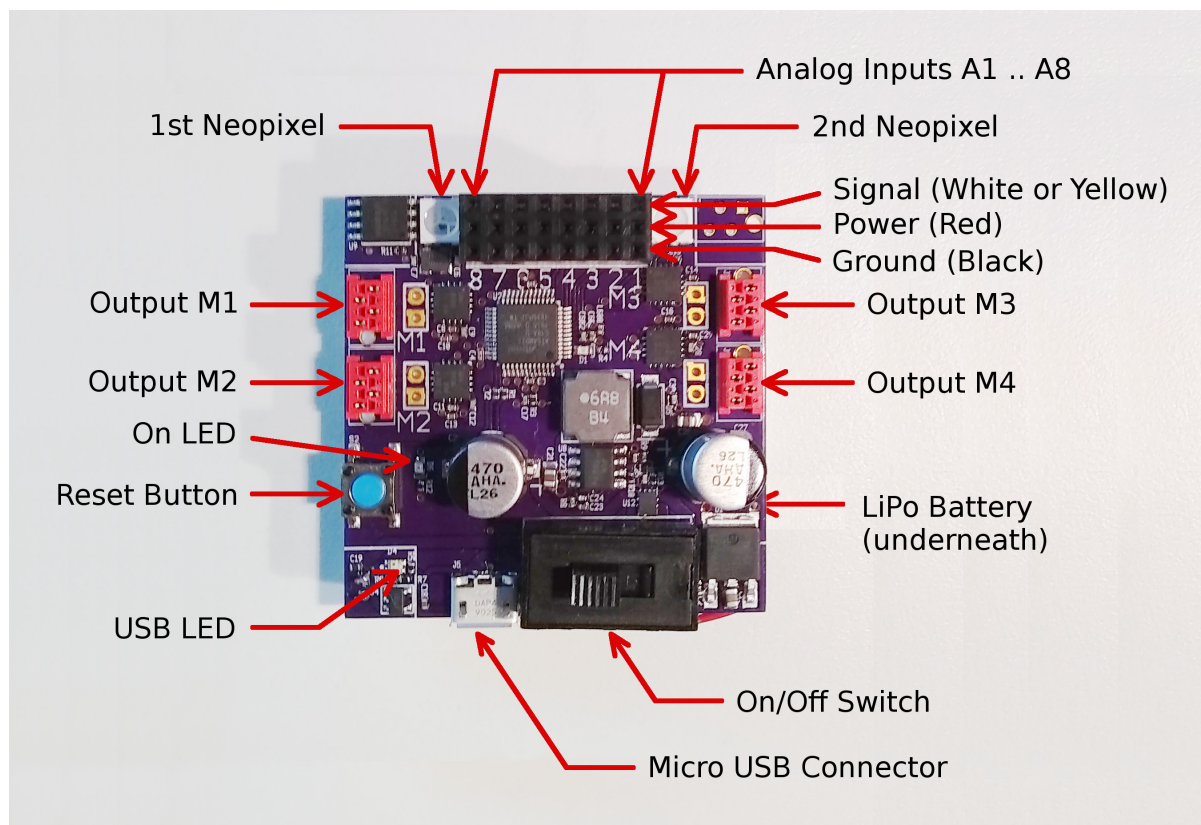
## 1. Overview

Snekboard is a small computer with integrated peripherals designed to work with

LEGO components enabling the construction of small autonomous robots. Snekboard supports three different programming environments:

- Snek. This small Python-inspired language is ideal for people new to robotics and programming.
- Circuit Python. A larger Python compatible language with more sophisticated control over the system and a more complete implementation of the Python language.
- Arduino. This C++ based development environment offers full access to the hardware at a primitive level. It is more challenging to learn and use, but offers the highest performance.

## 2. First Look at Snekboard

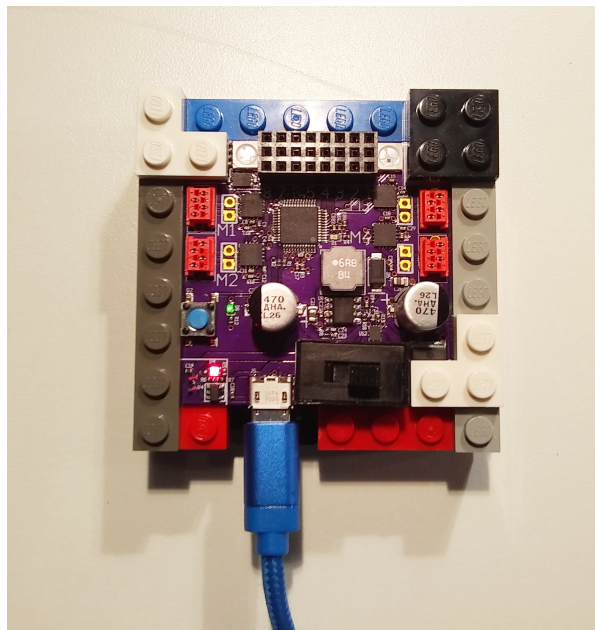


Snekboard has:

- A lithium-polymer (LiPo) battery so it can run with no USB connection.
- A green LED that is lit when the board is on.
- A blue LED that is lit while a Snek program is running.
- A USB LED that is:

- Unlit when there is no power connection.
- Red when charging.
- Green when the battery is charged.
- A reset button to restart the snekboard (starting the program in memory).
- 8 analog inputs A1 through A8 that return values from 0 to 1.
- 4 outputs M1 through M4, each with two directions (left or right) and power from 0 to 1.
- 2 NeoPixels that combine a red, green, and blue LED each of which can be lit with a power from 0 to 1.

Here is what it looks like when in its Lego case:



*Figure 1. Snekboard in Enclosure*

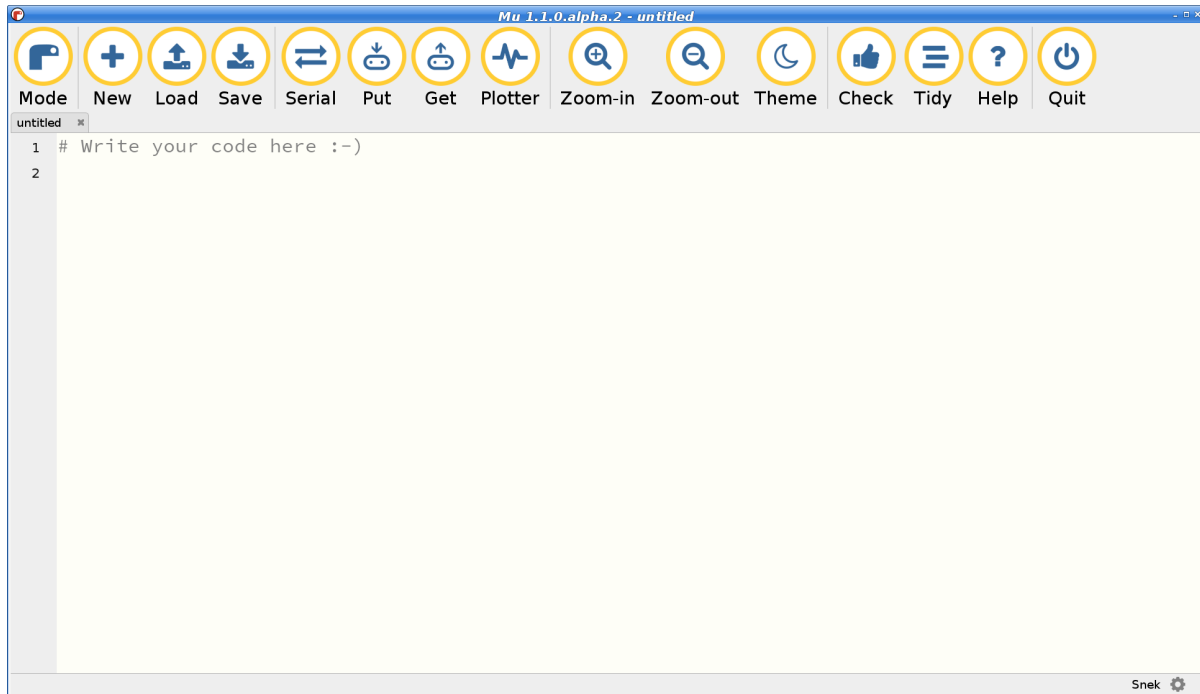
### 3. Mu Editor (mu-editor)

Find the Mu Editor icon on your desktop or in the system menu on your laptop. Launch the application and wait patiently for it to start. Depending on the speed of the machine, this can take a few moments.

When ready, your screen should look something like this:

## 4 Snek Lesson #1

---



*Figure 2. Mu Editor Startup*

At this point the interface is not connected to the snekboard yet. To connect to the snekboard:

1. Connect the snekboard to the computer with a micro USB to USB cable. (The USB LED should light.)
2. Slide the snekboard power switch so that the snekboard On LED is lit. (Be ready for the program saved on the snekboard to run!)
3. Click the Serial button. The editor will search for a snekboard connection and connect to it.

If all goes well, the editor will transform the bottom portion of the window into a interpreter terminal that runs commands on the snekboard:

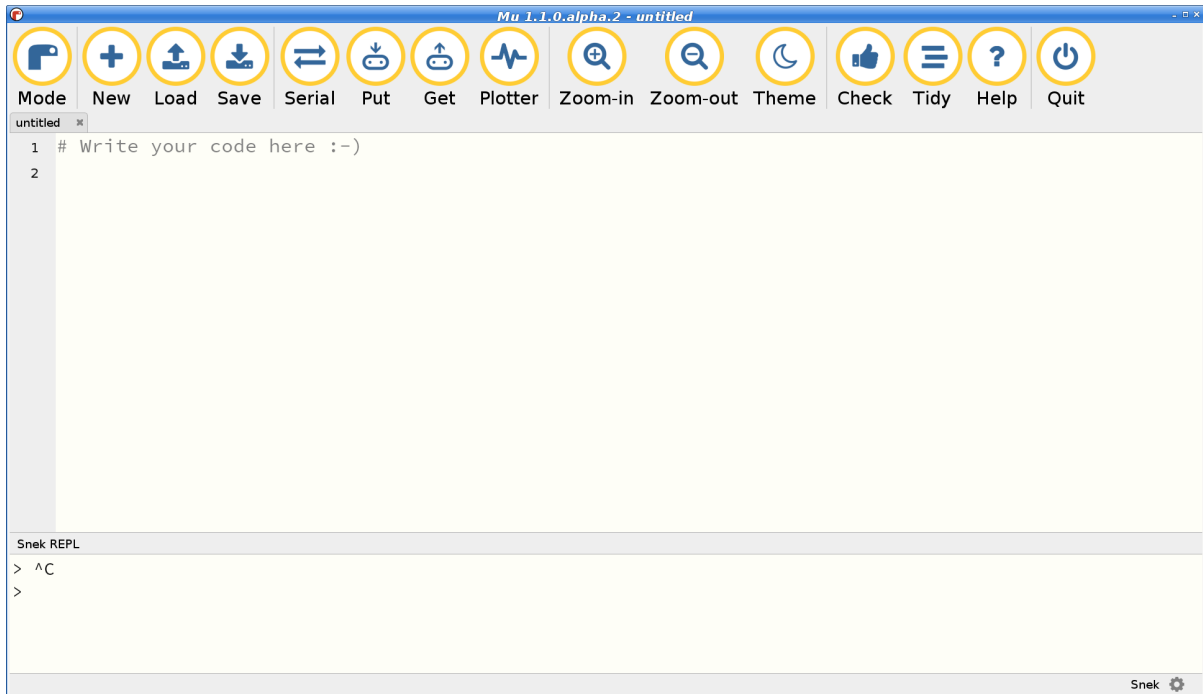


Figure 3. Mu Editor with Serial Pane

You can type commands into interpreter to execute immediately or click in the program frame to edit the snek program. The editor knows about snek and will automatically highlight keywords, pop-up menus to help you complete names, pop-up info windows to help you complete function calls, and automatically indent when you hit the enter key (use tab to indent more, and backspace to indent less). The editor provides lots of (online) help and can be used to develop python programs for hardware other than the snekboard as well.

A typical development sequence might look like this:

1. Start the mu-editor and connect to a snekboard as described above.
2. Click in the program editor frame and write some code:
  - a. Change the comment to something useful.
  - b. Define a function to perform some action. (Write a function named Hello to print a message).
  - c. Be sure to have a blank line at the end of the code!
3. Click on the Save button to save it to a file. (Notice the tab changes to the name of the file.)
4. Click on the Put button to transfer it to the snekboard. (Notice the “Welcome” message to indicate the transfer and the “>” prompt indicating successful reboot and running of the program on the snekboard. Problematic programs generate an error message and a “+” prompt that can require pressing the *Enter* key

several times)

5. Click in the interpreter terminal and run the newly defined function. (In this case the result is just the “Hello World!” message, but functions that run motors, flash lights, and/or read sensors would do their thing).

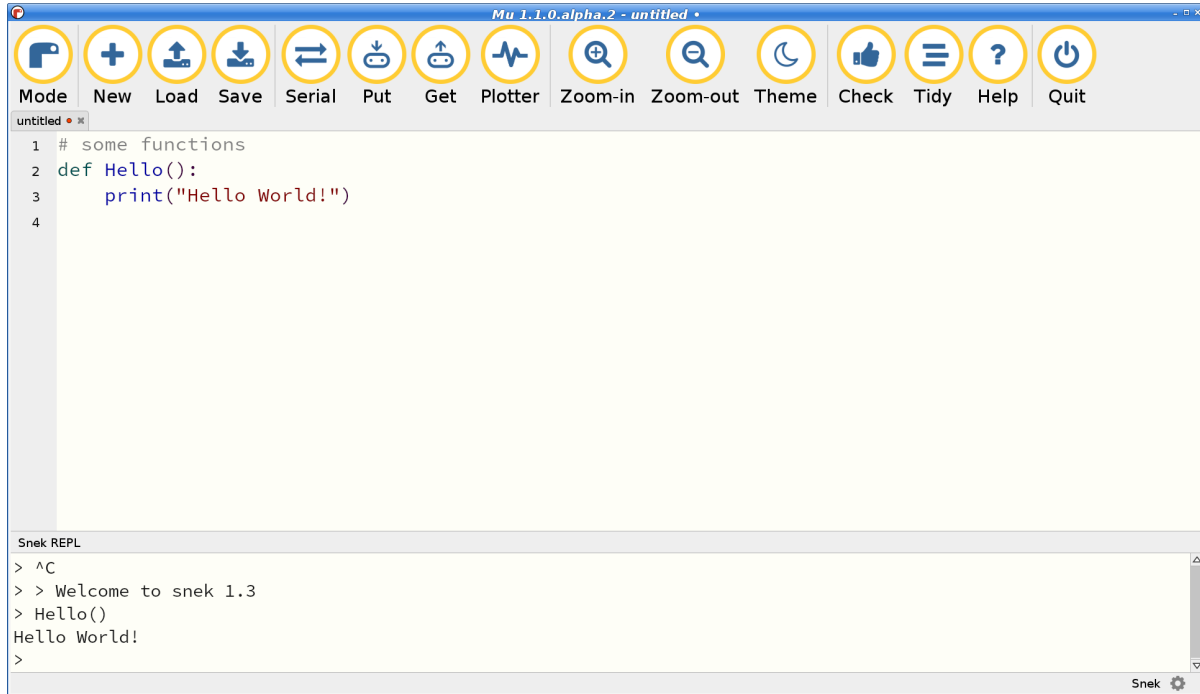


Figure 4. Mu Editor with Program

To run the function when the snekboard boots (including after Put), add the function call at the end of the program code (don't forget a blank line after it). Notice that using snek will consist primarily of:

1. Writing and testing expressions and commands in the interpreter.
2. Defining functions in the program editor.
3. Figuring out how to get expressions, commands, and functions to work.

## 4. Snekboard Cables and Sensors



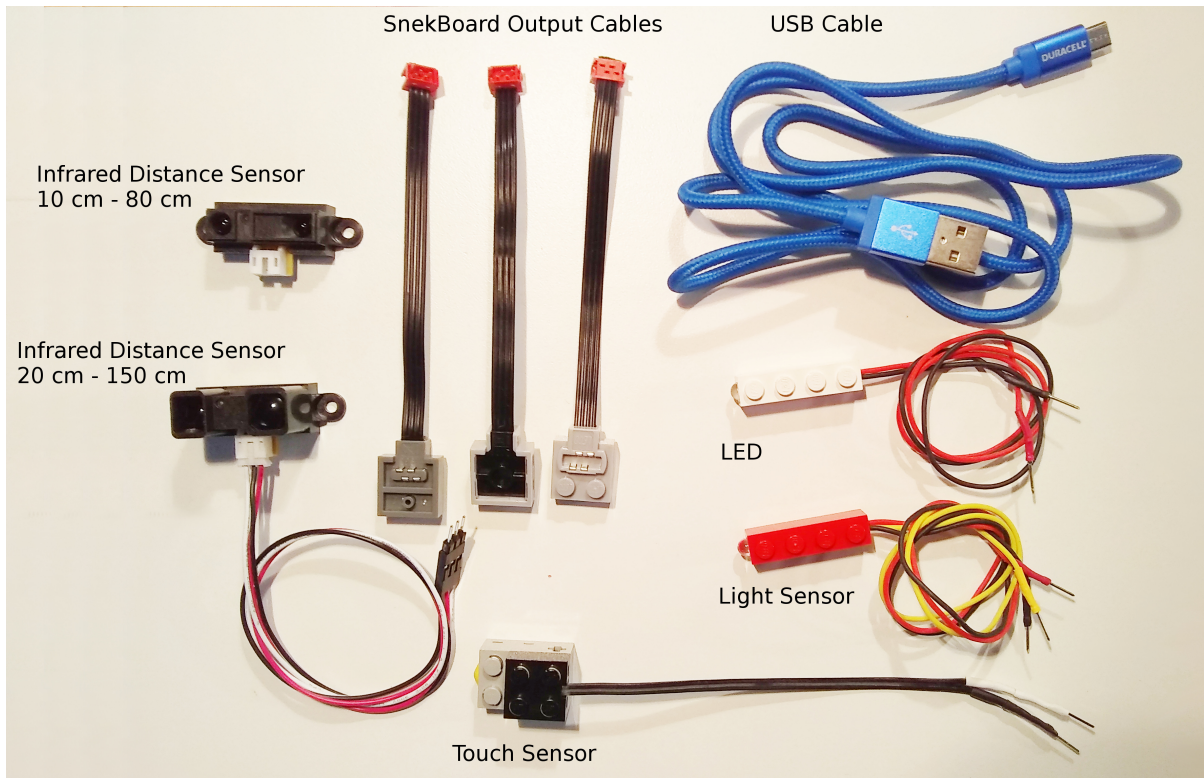


Figure 5. Cables and Sensors

Notice that:

- There are two kinds of snek output cables. Light gray connectors have power functions on top and brick underneath. Dark gray connectors have power functions on top and bottom. Connect the red connectors to the M1..M4 output ports so that the cables immediately exit the board.
- The light sensors are in red 1x4 Bricks. The white 1x4 brick is an LED, not a sensor, and has no white or yellow sensor signal wire.
- There are two kinds of distance sensors. The long range sensor (150cm or about 5 feet) is larger, and the short range one is smaller. The cables are the same for both.
- The touch sensor needs a special cable that has a brick connector on one end and (at least) two pins on the other. If it has only two pins, you need to pull up or pull down the signal line in software (depending on how it is connected) for proper function. See the section on Switches.
- The sensors have either 2 or 3 colored wires. With 3-wire sensors, be careful to connect sensor pins to an analog input column so the colors match the labels of the photo on page 1.

Here is a well connected Snekboard:

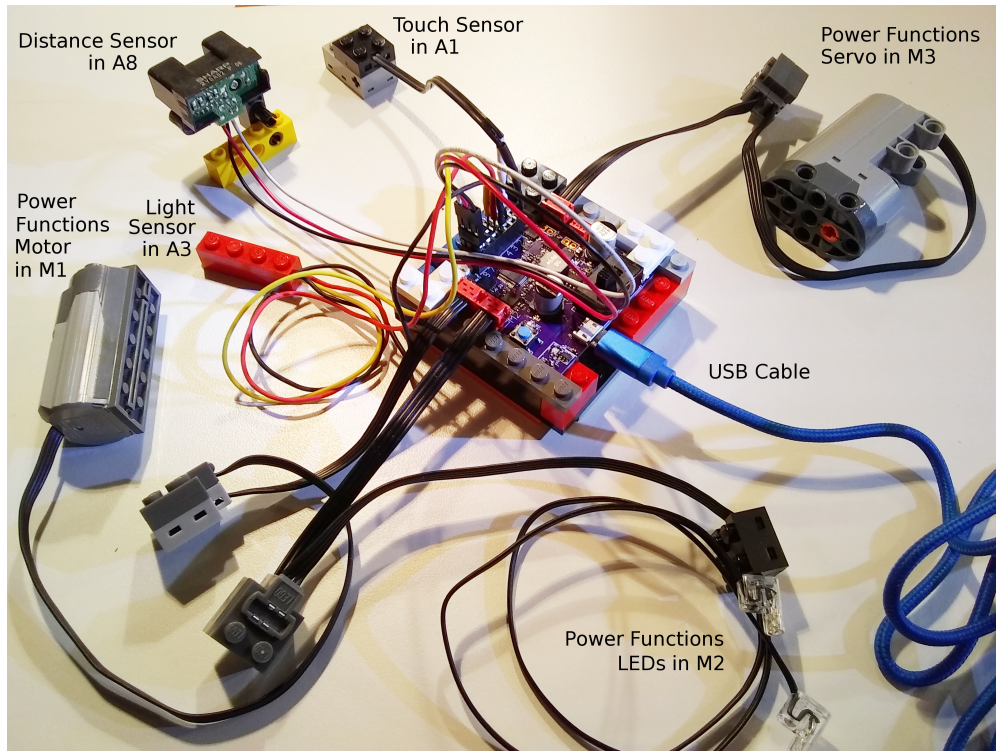


Figure 6. Wired Snekboard

## 5. Snekboard Outputs

### Outputs on Snekboard

- Are used to drive or control actuators. Actuators are devices that physically act in the world, like motors, servos, lights, and speakers.
- Use ports M1 to M4 and NEOPIXEL.
- Use these functions to control output ports M1 to M4:
  - `talkto(M#)`: Remembers output port # for other output commands.
  - `setpower(P)`: Sets the output power to P with  $0 \leq P \leq 1$ .
  - `setleft()`: Sets the output power to flow left-wise through port #.
  - `setright()`: Sets the output power to flow right-wise through port #.
  - `on()`: Turns the power to port # on.
  - `onfor(S)`: Turns the power to port # on for S seconds (approximately).
  - `off()`: Turns the power to port # off.
- The neopixel LEDs work a bit differently. See the snek manual.



## 6. Snekboard Inputs

- Are used to read sensors. Sensors are devices sense or measure physical conditions in the world such as light intensity, distance (as reflected by IR light), and touch.
- Use analog input ports A1 to A8 to provide values that range from 0 to 1. “Analog” indicates that the values provided represent continuous values.
- Use the read function to get values from ports A1 to A8:
  - `read(A#)`: Gives the value (between 0 and 1) of port #.
  - Advanced Topic: A 2-wire touch sensor needs `pullup(A#)` or `pulldown(A#)`. More on this later. A 3-wire touch sensor does not.

## 7. Hooking Things Together

Now we'll write a Snek program to show how to use a distance sensor to control the speed of a motor.

1. Connect a motor to Snekboard output M1
2. Connect a distance sensor to Snekboard input A1

We can test the distance sensor and the motor with some Snek commands. Use the Mu Serial pane to enter the text in **bold** below. The text in a normal face is what Snek writes back to you:

```
> read(A1)  
0.3484738
```

Your Snekboard will probably show a different value. If you like, you can experiment with how the distance sensor reacts when you move things in front of it:

```
> while True:
+   print(read(A1))
+   time.sleep(1)
+
0.3484738
0.3829738
0.2694738
0.4214738
0.6064738
0.4194738
0.3469738
0.4659738
0.3039738
0.3074738
Ctrl+C
```

You'll need to hit *Enter* after **time.sleep(1)** to get the program to run. Hit *Ctrl+C* when you're done testing to stop the program.

Now we'll test the motor:

```
> talkto(M1)
> on()
```

This turns the motor on at full speed. To reduce the speed, try:

```
> setpower(0.5)
```

Now let's hook the input sensor up to the motor:

```
> talkto(M1)
> on()
> while True
+   setpower(read(A1))
+
```

Experiment with moving things in front of the distance sensor to see how the motor responds.

## 8. Switches

Using switches (also known as “touch sensors”) with Snekboard is slightly tricky. That’s because Snekboard inputs are normally set to read sensors that produce a range of values, like distance and light sensors. These kinds of input devices are called *analog sensors*. We want switches to either be 1 or 0, which is called a *digital sensor*.

Here’s what you can get if you hook a switch up to Snekboard.

1. Connect a switch to Snekboard Input A1 between A and +

```
> read(A1)
0.08229548
> read(A1)
0.9997558
```

The first value was read with the switch **off**. The value is very nearly 0, but not quite. It has to be exactly 0 for Snek to treat it as False. The second value was read with the switch **on**. This value is very nearly 1, but again, not quite.

When a switch is **on**, the two leads are connected together. When the switch is **off**, the two leads are not. Let’s explore how this works by connecting the switch differently.

1. Connect a switch to Snekboard Input A1 between A and -

```
> read(A1)
0.1269841
> read(A1)
0.0007326007
```

This time, **both** values are small, but neither is exactly zero. What happened here? When the switch is **off**, the A lead isn’t connected to anything, so Snekboard isn’t being given a value and it makes one up on its own. In this case, the value isn’t terribly useful.

You can tell Snekboard how to fix this by using either of two new Snek functions, `pulldown(I)` or `pullup(I)`. These functions tug the input line gently to either 0 or 1 when the switch is **off**. When the switch is **on**, it pulls strongly to whichever value you’ve connected the other lead to (+ pulls to 1, - pulls to 0). When you use either of these functions, you also tell Snekboard that you’re using a *digital sensor*, making the output be either 0 or 1, depending on which is closest.

1. Connect a switch to Snekboard Input A1 between A and +

```
> pulldown(A1)
> read(A1)
0
> read(A1)
1
```

You can try pullup too by changing the switch connections again:

1. Connect a switch to Snekboard Input A1 between A and -

```
> pullup(A1)
> read(A1)
1
> read(A1)
0
```

To get A1 back to normal mode, just use the `pullnone` function:

```
> pullnone(A1)
```

## 9. Servos

Snekboard can also control a LEGO™ Power Functions Servo Motors. These work differently from regular motors; they don't rotate continuously, but rather can be told to rotate to a specific position. This is done by controlling both the power and direction. The power setting controls how far the servo rotates while the direction setting controls which way, either clockwise or counter clockwise, from the center position.

1. Connect a servo to Snekboard output M1

The servo should reset to the center position. Let's move it all the way to the left:

```
> talkto(M1)
> setleft()
> on()
```

Now we can switch the direction and move it all the way to the right:

```
> setright()
```

Finally, we can move halfway between center and right:

```
> setpower(0.5)
```

## 10. Next Steps

After this lesson, you can go on to the Line Bug, Washing Machine or Bumper Car. Those are designed to be done in any order. Have fun making Robots!

