Introduction to the Arduino and Scientific Python

April 1, 2018

1 Introduction

In this lab, we will learn to use the Arduino microprocessor and scientific python, two tools we will be using through out this course.

2 Introduction to the Arduino

For this lab, we will use Arduino Uno. The Arduino Uno is fairly robust and inexpensive, so you should definitely be comfortable to experiment with them. However, to avoid the potential to damage the boards, you should *NOT* supply any external power to the Arduino (e.g. from your proto-board PB-503, bench-top power supply, or function generator.)
https://www.arduino.cc/en/Guide/ArduinoUno

Step 1: The main Arduino website, which includes lots of supporting documentation, is located at `http://arduino.cc`. Acquaint yourself with this site.

Step 2: The lab PCs have the Arduino software installed, and each team will have an Arduino to work with. Follow the instructions for using the "Arduino Desktop IDE" located here: `https://www.arduino.cc/en/Guide/ArduinoUno`.

Step 3: Adjust the pattern of the LED blink example to a slower rate.

Step 4: The Arduino tutorials and built-in-examples are often excellent starting points for you own projects, browse the tutorials here:
https://www.arduino.cc/en/Tutorial/HomePage

Step 5: Read, build and upload the DigitalReadSerial example. You will need the Arduino proto-shield, which includes a pushbutton and LEDs, to complete this part. Note: to use the pushbutton switch labeled S1 on the Prototype Shield v.5, note that hole labeled (+) is shorted to ground when the push button is pressed. Your protoboard shield should already have a wire soldered at this point. Use a pull-up resistor as shown if Fig. **??**.

Step 6: Modify it to only output to the serial port after a change in value.

Step 7: Read, build and upload the Read Analog Voltage example. You will need a potentiometer (variable resistor) for this and the following step.

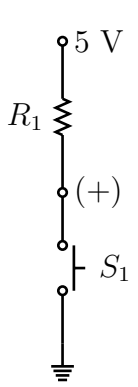

Figure 1: On the proto-board shield, the push-button switch S1 shorts the point labeled "+" to ground when pressed. Use a pull-up resistor $R_1 = 10$ kΩ to set the voltage at point + to 5 V unless the push-button S_1 is engaged.

Step 8: Read, build, and upload the Analog Input example.

3 Introduction to Scientific Python

Much of our work will be using scientific python in an interactive notebook environment. On some of the newer lab PCs, this is called "Jupyter Notebook" while on some of the older PCs, this is installed under it's older name as "IPython Notebook". Simply find the icon on your Desktop to start the notebook, click it, and start a new notebook. Most often, we will want inline plots that show up as output in your notebook. You enable this by using the special command `%pylab inline` at the start of your notebook, as demonstrated in Fig. **??**. This command also imports matplotlib and numpy as "plt" and "np" so you will be able to work through the tutorial here:
`https://matplotlib.org/users/pyplot_tutorial.html`
by simply omitting the "import" directives from the example code snippets. Note that some of the code snippets are not complete examples. You should only try to run the complete examples which include the expected output and begin with an "import" command (which you will omit.) Also, note that our installation does not include the `logit` scale, so simply replace `logit` with `log` for the last example. See Fig. **??** to get started.

4 Lab Report

There is no lab report for this lab, simply inform your TA when you have completed everything.

IP[y]: Notebook Untitled1

File Edit View Insert Cell Kernel Help

Code Cell Toolbar:

None

In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```

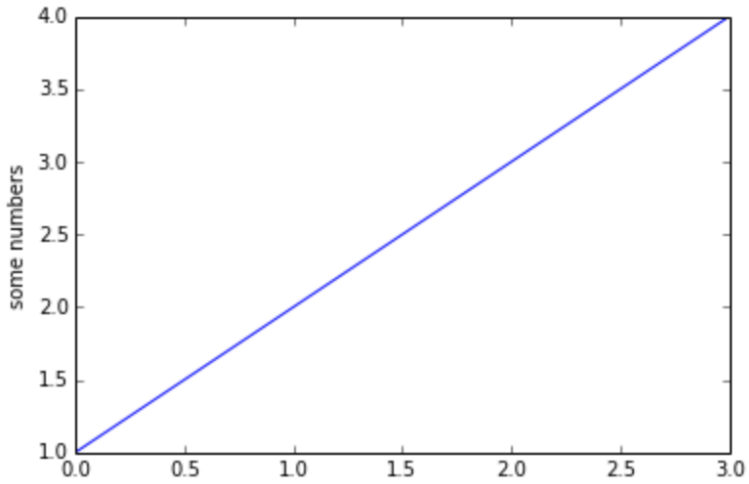

Figure 2: Using Jupyter notebook in inline mode to start the Matplotlib tutorial.