Using Morse Code to Encode Your Secret Message

Most of what we will discuss today with regard to setting up our telegraph was due to the Raspberry Pi Foundation's excellent primer on Telegraphs. The link is shown below. We will not go into as much detail here, and will focus mostly on hooking up the circuitboard.

In [7]:
```
from IPython.display import HTML
HTML('<iframe src=http://www.raspberrypi.org/learning/morse-code/ width
=700 height=350></iframe>')
```

Out[7]:

<iframe src="http://www.raspberrypi.org/learning/morse-code/" width="700" height="350">
Raspberry Pi Learning Resources

MORSE CODE VIRTUAL RADIO


</iframe>

The Morse Code

The Raspberry Pi site has an excellent discussion on the history of the Morse Code and it's role in WWI. It is a great read! We'll just touch on some technical elements here.

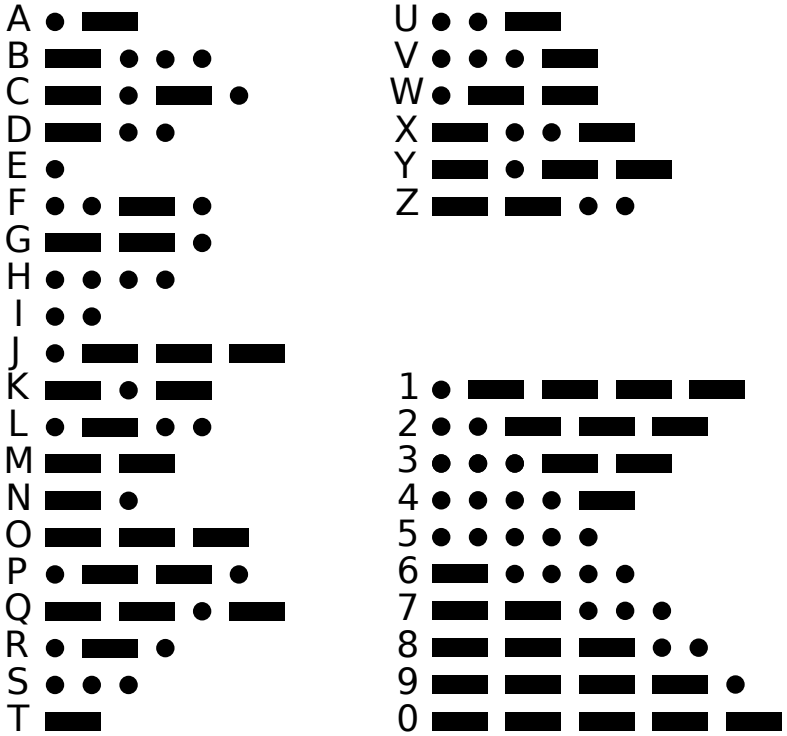
So, the idea behind the Morse Code is that one could translate each letter of the alphabet into a series of dots and dashes that could be communicated with a simple electrical machine called a ***telegraph***.

This machine has a very simple design. When the button is pressed, it makes a noise, and is otherwise silent. If you hold it down for a 'long' time, that is considered a **dash**. If you hold it down for a 'short' time, it is a **dot**. If you pause for a second (actually 1.5 seconds in our code). Then that is the end of your letter. If you wait longer, that is the end of your word.

The translation of Morse Code into letters has a very similar structure to the cipher that we discussed in the previous section. The Morse Code is listed below.

International Morse Code

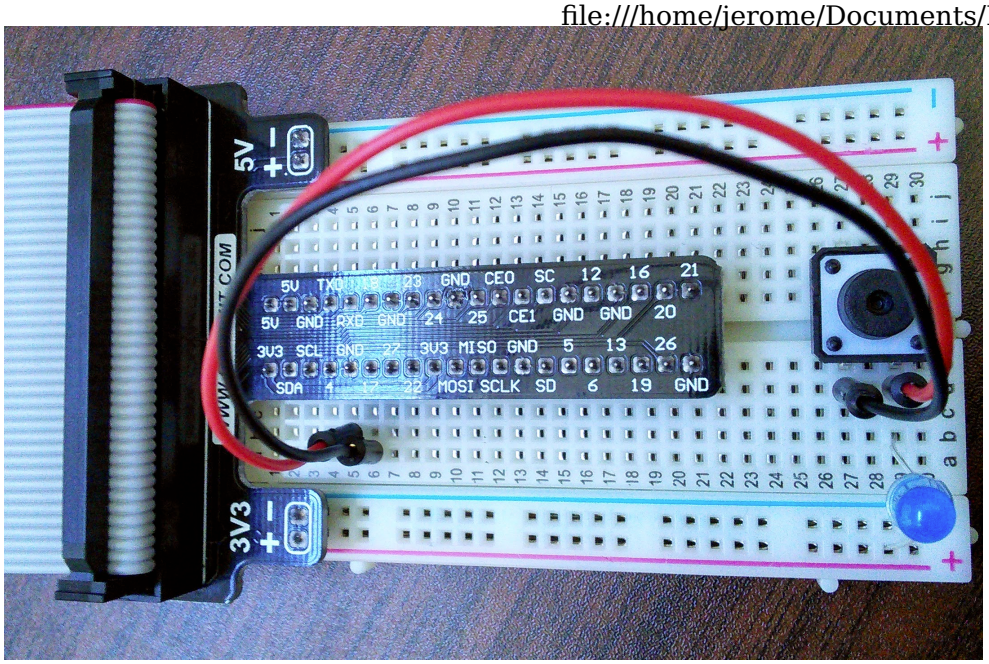
1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.



Can you write your name in Morse Code? Can you sound it out?

The Canakit Telegraph

We will not use an actual telegraph switch, but we will use the parts provided in our Canakit. There is a lot of electrical engineering behind all of this, so it might be easiest to just hook it up, and see if we can get it to work! In that spirit, I want to show you what is happening. Below is a diagram of the circuit board, or ***breadboard***.



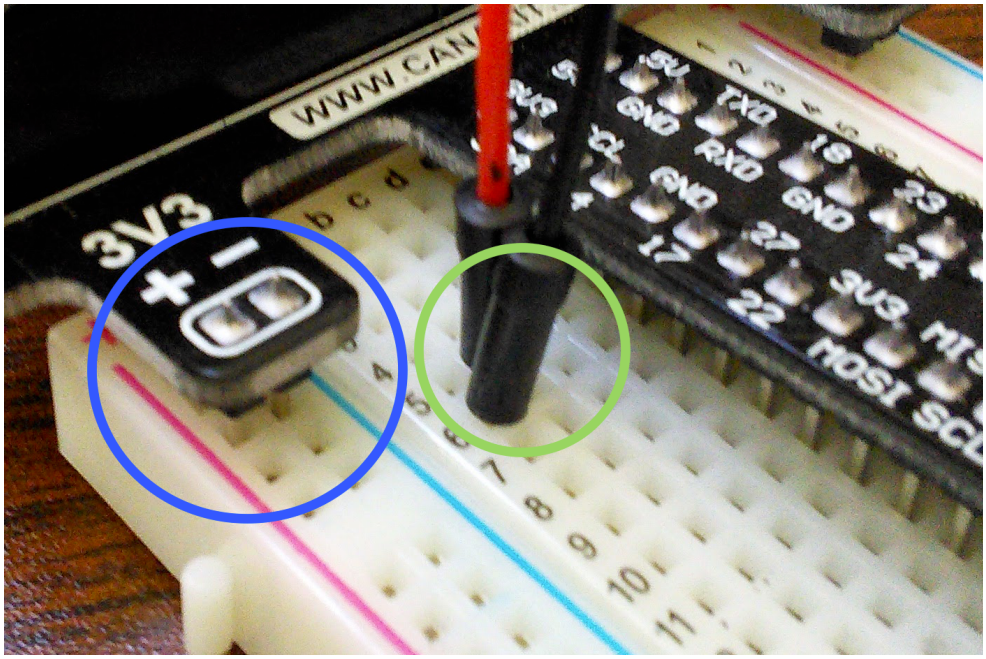
The red wire is the 'hot' wire, the black is the ground, and the button is what we will use to send our messages. The blue diode (or LED, or 'light') will light up when we press the button (if it all works correctly). Before jumping into hooking it up, I want to show you a couple of closeups.

Please Be Careful!

NOTE:

- Make sure this is **UNPLUGGED** while you are setting up the circuit!
- Make sure that the teacher reviews your work **BEFORE** plugging it in!

It doesn't take too much to break these while plugging in circuits, so we want to be extra careful setting them up.



The pins under the portion of the board labeled '3V3' must lay down in the rows next to row 3. All of the other pins should line up easily to the holes in the rest the breadboard. All pins should should be pressed firmly into the board.

If it all works, then the red wire can be placed on any hole in row 5, which corresponds to GPIO 4.

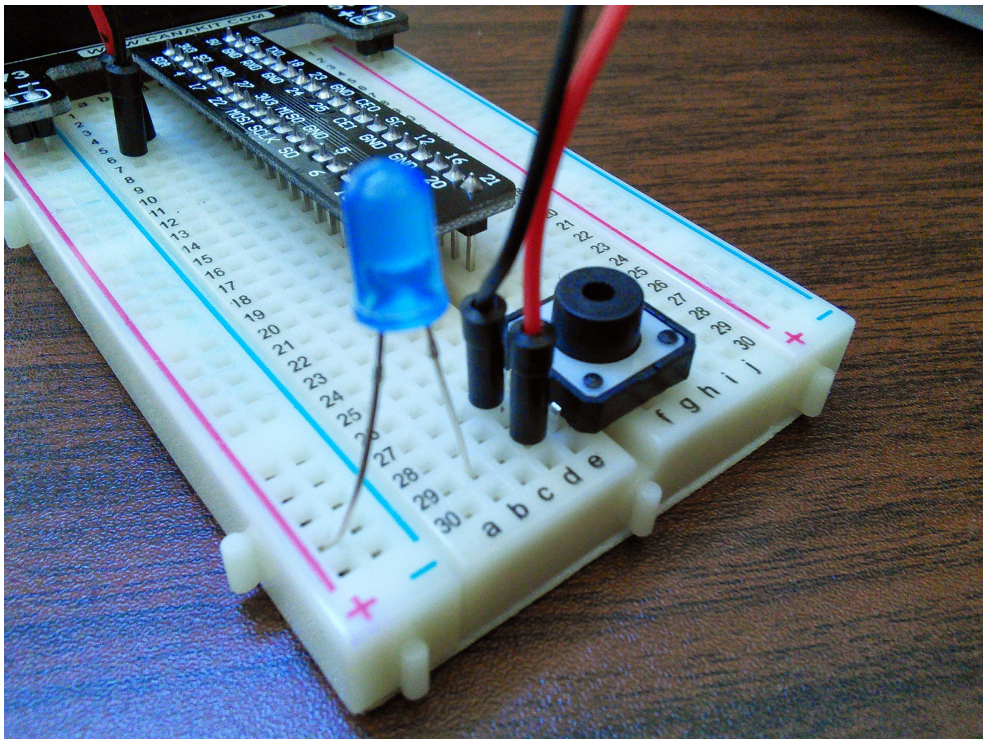
Here is the Canakit GPIO diagram for reference.

We won't be looking too much into the code, but GPIO 4 talks to 'pin 7' on the Pi, which is what we set up in the code. GPIO (General Purpose In/Out) is the name of the board and the wires connecting to the Pi, while "pin" refers to the pins connected to the motherboard.

The knee bone is connected to the thigh bone....

Also, note that the black wire connects to GND, which means 'ground'. We'll talk briefly in class about what ground means in circuits.

The bottom portion of the circuit looks like this.



It is not as important to have the pins on a particular number here, but I have the hot wire on row 29, and the ground on row 27.

An important thing to remember: The rows on the center pieces of the breadboard are all equivalent, and the columns of the outer pieces are equivalent. If this sounds confusing (and it should), we'll discuss it in class.

The button should line up with these pins, and straddle the center gap of the breadboard.

The LED has two wire lengths. The long wire should go into the '+' column of the breadboard. Anywhere on this column will be fine, but I have it on row 29. The short side of the diode should be on the same row as the hot wire (also row 29, but in the center).

Finally, the short wire of the LED should be on row 29 near the button.

Have your teacher come by before plugging in. Once it is plugged in and booted up. The light should flash on when you hold down the button and turn off when you release it.

Did it work?
Woohoo!

Encoding and Decoding Your Message with the Telegraph

To encode the pangram type: **sudo python secret_code.r2.py morse encode pangram1**

Which is pretty much the same thing as before, but the word 'morse' tells the program to ask you to key in the secret word with the telegraph. Don't forget to use **sudo** at the beginning! This tells the Pi that you're a '**Su**per **Do**-er! No, really, that's what it means!

TIPS: - While it is usually a good idea to have a long complicated secret word, let's try to use a small one here until we get the hang of using the telegraph. 'SOS' is a good place to start. - Write your word down in Morse code, and even try to memorize it. You'll need to practice a few times. - Make your 'dots' *rly shrt*, and your 'dashes' *reeeaaallllly long*. It will help to keep the program from getting confused.

If it works, you should get the same output as before. You can decode in the same way.

Okay, good luck! I can't think of anything else for now!
Cheers, J

In []: