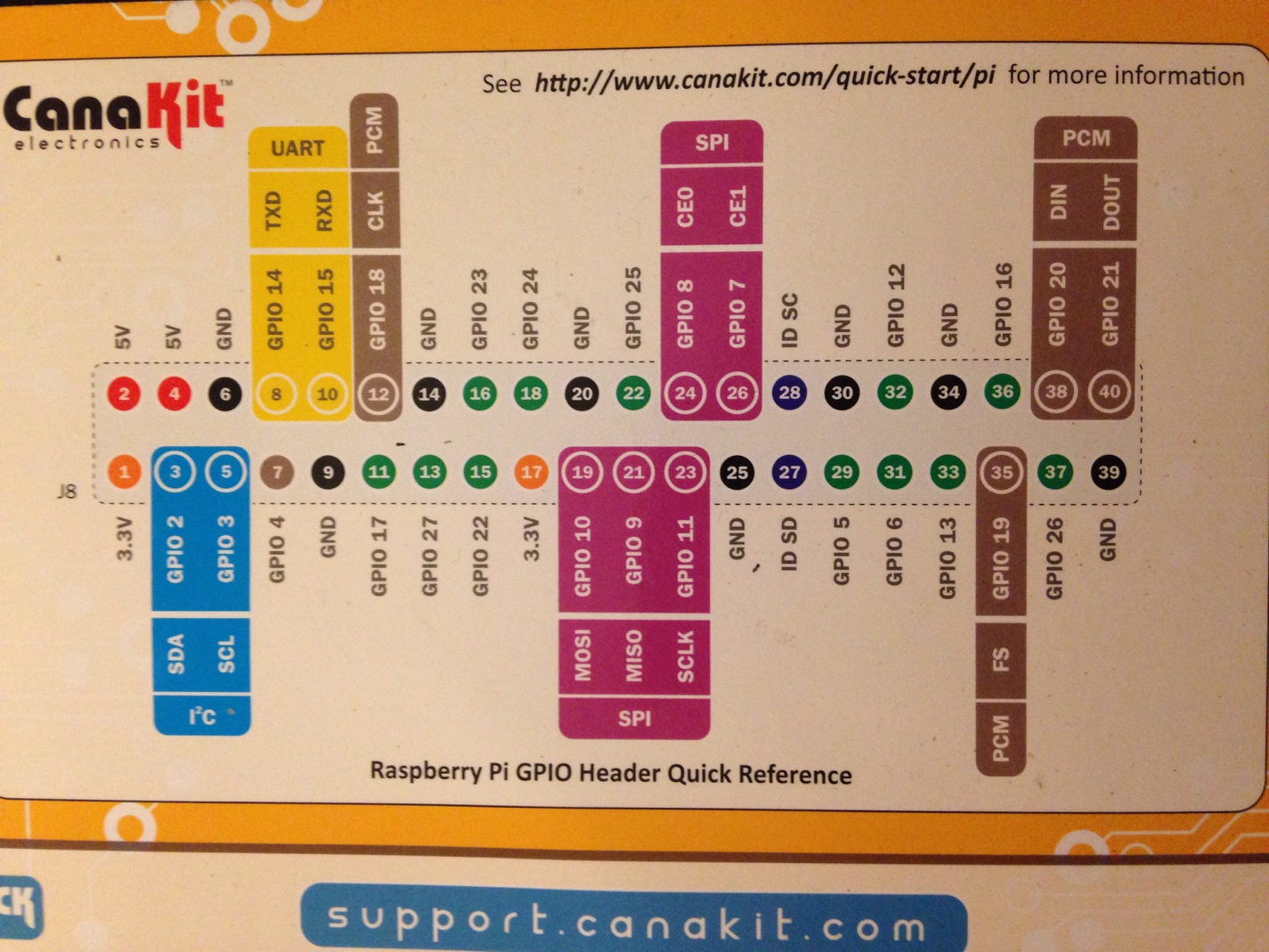
# Raspberry Pi GPIO

The GPIO pins often pose a headache for new users. There are many libraries available for various languages, so it is mostly a matter of finding one that works for you. WiringPi is a popular choice, and Python has a handy library that makes access easy.

There are two primary pin mappings on the GPIO – the hardware numbering and the BCM numbering. This tutorial uses the BCM mapping. If you choose to use WiringPi, it has its own mapping. This CanaKit resource will be useful when making circuits:



We will use the outer labels – for example, GPIO 4 is pin 7 (upper left of the image)

## File Access

The following three links all describe raw file access to the GPIO. All libraries do is provide handy wrappers for these commands.

* <http://www.codefoster.com/pi-basicgpio/>
* <https://sites.google.com/site/semilleroadt/raspberry-pi-tutorials/gpio>
* <http://raspberrypi.znix.com/hipidocs/topic_gpiodev.htm>

As practice, try setting up a simple LED circuit and turn it off and on with the commands in these links. “Simple LED circuit” means a series resistor and LED going from a data pin to ground. In this schematic, the BCM pins are labelled on the inside – I am using GPIO 17 and connecting to the GND on pin 39. Any data pin and ground pin will work.



In the command line, execute the following lines

|  |
| --- |
| Terminal  sudo echo 17 > /sys/class/gpio/export sudo echo out > /sys/class/gpio/gpio17/direction sudo echo 1 > /sys/class/gpio/gpio17/value // LED on sudo echo 0 > /sys/class/gpio/gpio17/value // LED off |

This is digital control of the pins. For PWM and serial communication there are alternative processes that you can find elsewhere (the middle link above provides one method of each).

## Upstream: GPIO Data to Mongo

This section is about reading input from sensors and uploading it to Mongo. You can then read it from Mongo into your webserver.

For the demo project I used Python to read from the DHT22 Temperature and Humidity Sensor. I recommend Python because the pymongo library makes Mongo transactions easy. However, the method of reading data will vary based on your sensor/input. Raspberry Pi pins are purely digital, so you will need an Analog-to-Digital Converter to use an analog input. For QP, stick to digital sensors. Example code for uploading to Mongo is at <https://github.com/WigginWannabe/qp16/tree/master/demo_fall>, in the gpioUP folder. Look for “test\_mongo.py”. “readSensor.py” in the main directory is the final version of this file.

1. Install pymongo with sudo python –m pip install pymongo
2. Google how to use your sensor in Python. You will likely need os.system to read the sensor value in the manner above, or a library for the sensor (I use a library in the demo code)
3. Use the example file to practice sending information into Mongo. The process is generally the same as in PHP (create an array of data in key-value pairs, make a mongo connection and get the database and collection, insert the array into the collection as a document or an update to an existing document), but syntax is different.
4. “readSensor.py” also sets the timezone, which is useful if you want to be a part of your database. Install pytz and use that file as a guide.

## Downstream: Sending GPIO Commands from Webserver

While in your website, certain sensor readings or user input can produce a response on the gpio by sending command line instructions from PHP. This is the exact same concept as using os.system in Python – in PHP, the function is exec.

For example, the line exec(“sudo echo 1 > /sys/class/gpio/gpio17/value”); in a PHP file will turn on GPIO 17 (if it has already been exported and set to output).

The file “GPIO.php” found at … is a simple class for managing GPIO. If you include place it in your directory and include it in a webpage, you can control it from that page. Look at index.php in the “gpioDOWN” folder at <https://github.com/WigginWannabe/qp16/tree/master/demo_fall>. At the bottom there is a very crude example of using this class to light up LEDs based on the most recent temperature reading (it is crude because I should have used loops to manage which LEDs are on, and instead hardcoded it. Bad style! Do not emulate!)

Try forking or copying this file into your directory, and take a look inside. A couple of things to note are that it uses WiringPi’s command line controls in the exec functions, so the gpio access looks different, and that the write function accepts various types of input.

Practice downstream commands by responding to sensor readings – like in the example – or user actions like submitting a form, clicking a button, refreshing a page, or anything else you can do on a website!