# Raspberry Pi
# GPIO for Dummies

**Part 1**

Bruce E. Hall, W8BH

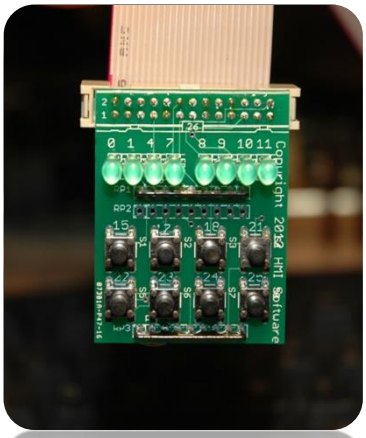
## 1) INTRODUCTION

The amazing Raspberry Pi is a credit card size computer, built for learning and fun. In addition to the USB and Ethernet ports, it can connect to other hardware through its general-purpose IO (GPIO) interface. The Raspberry Pi GPIO is a bank of 26 pins on the top-left corner of the circuit board, which includes 17 digital input/output connections.

If you want to use GPIO on the Raspberry Pi, there are a few things you must know. First, these unbuffered lines connect directly to the processor. They do not tolerate much abuse: a spark from your fingertip, excessive current drain, or an incorrect voltage is enough to fry your pi. Next, you should learn what each pin is for, and be careful how they are connected. Using the wrong pin, or accidentally shorting the pins, can ruin your whole day.

Fortunately, help is available. There are simple add-on boards that plug directly on the GPIO connector. I wholeheartedly recommend learning with one of these boards. For this tutorial I will be using the 'Push your Pi' kit, available for $9.99 at MyPiShop.com. Why?

- It's compact: Just pop it on top of the pi.
- It's simple: no need for a breadboard or wires.
- It's safer: electrical connections are already worked out.
- It's a great excuse to get out your soldering iron.
- It's an easy way to learn about GPIO on your pi
- Like the Pi, it's incredibly cheap! Maximize your bang-for-the-buck.



If you like learning by doing, don't hesitate. Just buy one of these add-on boards and follow along.

## 2) ASSEMBLY

I enjoy a good excuse to melt solder. My bag of parts arrived safely, and well packaged. This kit does not contain a huge number of parts. Make sure you have all of the parts, including 8 green LEDs, 8 switches, 2 black resistor arrays, the 26-pin socket, and the PCB. Use whatever method you like for keeping parts off the floor, sorted & secure in your workspace. Some people like egg cartons or muffin tins. I use a shoebox lid.

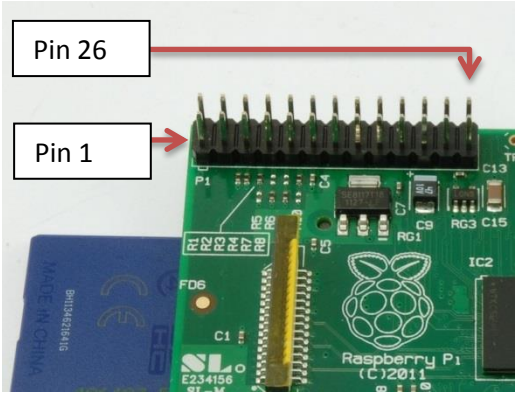
The kit does NOT come with any documentation. Print out the online instructions and follow along. It's pretty easy. Remember to put your 26-pin connector on the bottom of the board, and the rest of the components on the top. I suggest starting with the switches first, since these are the easiest components to place. Next, mount the resistor arrays. Tack down one -- and only one -- pin with solder, then turn the board over and check your alignment. Straighten if necessary. Reread the construction note and confirm the label faces the bottom of the board. Solder the remaining leads. Next, install the LEDs, making sure the longer leads are toward the top of the board. Tack down one lead for each LED, check alignment, then solder the remaining lead. I suggest soldering the 26-pin connector last. Why? Its plastic body will melt if your soldering iron happens to touch it while soldering the LEDs. Don't ask me how I know!

When everything is connected, double-check your soldering job. There should be a row of unused holes labeled RP2. The only extra part you should have is a circular, clear-plastic 'foot' with adhesive backing. Shut down your pi, unplug the power, then mount the board on top of the GPIO header. The board should mount over your pi, not away from it. Make sure that the header and socket are firmly mated. It's easy to misalign the connection, so make sure there are no visible pins on either side of the socket. Attach the foot to bottom of the board where is overlies the Pi's tall electrolytic capacitor. The foot gives the PCB stability, as well as providing electrical insulation between the board and your Pi.

Reattach power and boot your Pi. You may notice one or more LEDs light up. Don't worry - we will get control of them soon enough.

## 3) THE GPIO CONNECTOR

The GPIO connector is a block of 26 male pins in the upper-left corner of the Raspberry Pi board, arranged 2x13. Pin 1 is lower-left pin (marked P1) and pin 26 is upper-right. The bottom row are odd-numbered 1 though 25, and the top row are even-numbered 2 through 26. The pins are spaced 0.1" apart. Search "26p female connector" to find mating connectors, which cost about $0.50 each. With a little care, even oversized, 40-pin computer ribbon cables can be attached. If you want to use just a few lines, female-to-male breadboard jumber wires are also available.



Here is a diagram of the pins and their functions. There are four pins for power and five pins for ground, leaving 17 pins for digital input/output. Of these, 9 (shown in blue) have alternative, specialized uses: 2 for I2C, 2 for UART, and 5 for SPI data communication. The remaining 8 lines in green are for general-purpose IO.

*On version 1 boards, pins 3, 5, and 13 were designated GPIO0, GPIO1, and GPIO21, respectively.

| | | | |
|---|---|---|---|
| 3.3V | 1 | 2 | 5V |
| (SDA) *GPIO2 | 3 | 4 | 5V |
| (SCL) *GPIO3 | 5 | 6 | GROUND |
| GPIO4 | 7 | 8 | GPIO14 (TxD) |
| GROUND | 9 | 10 | GPIO15 (RxD) |
| GPIO17 | 11 | 12 | GPIO18 |
| *GPIO27 | 13 | 14 | GROUND |
| GPIO22 | 15 | 16 | GPIO23 |
| 3.3V | 17 | 18 | GPIO24 |
| (MOSI) GPIO10 | 19 | 20 | GROUND |
| (MISO) GPIO9 | 21 | 22 | GPIO25 |
| (SCKL) GPIO11 | 23 | 24 | GPIO8 (CE0) |
| GROUND | 25 | 26 | GPIO7 (CE1) |

## 4) THE VIRTUAL FILE SYSTEM

OK, that's enough background. Now let's use it. You can manipulate each of the 17 I/O lines directly from the command prompt, using the virtual file system. In Raspian, each GPIO port can be addressed like a file: open the port by creating the file(s), change the port value by writing to the file, read the port by reading the file, and close the port by deleting the file. These files are located at /sys/class/gpio. Log in as root (you must be root to create the GPIO files) and go to this directory:

```
$ sudo su
# cd /sys/class/gpio
# ls
```

The directory will contain at least two files, export and unexport. To open a port, send the port number you want to the export file. To close a port, do the same with unexport. For example, lets open GPIO4. Send '4' to the export file using the echo command:

```
# echo 4 > export
# ls
```

Look at the directory. There is now a 'gpio4' directory, which indicates that the port is open. If you look in the gpio4 directory, you will see several files. One file is named direction, and another is value. To write a '1' to GPIO4, we first must set the direction to 'out' then write set the value. Try the following:

```
# echo out > gpio4/direction
# echo 1 > gpio4/value
```

Did anything happen? The third LED, which is connected to GPIO4, should now be on. To turn it off, just send a 0 to the same value. Close the port when you are finished, by sending 4 to the unexport file. The gpio4 folder will be deleted.

```
# echo 0 > gpio4/value
# echo 4 > unexport
# ls
# exit
```

You'll need to troubleshoot your connections and board if the LED didn't light. Nothing that follows will work until you get LED 3 working.

Let's try a bash script. This is a collection of bash commands, forming a program that can be executed from the command prompt. In addition to echo, all we need is a time delay (sleep) and a counting loop (for). Fire up an editor (I use nano) and enter the following lines:

```
#!/bin/bash
cd /sys/class/gpio

echo "Opening port" 4
echo 4 > export

echo "And making it an output"
echo out > gpio4/direction

for I in {1..5}
do
    echo "Setting value to 1"
    echo 1 > gpio4/value
    sleep 1
    echo "Setting value to 0"
    echo 0 > gpio4/value
    sleep 1
done

echo "Closing port" 4
echo 4 > unexport
cd /home/pi
```

Save it as ledcheck. Make the file executable with chmod, then run it as root:

```
$  sudo su
#  chmod +x ledcheck
#  ./ledcheck
```

LED 3 should blink 5 times. Make it even cooler by replacing each '4' in the file with a '$1' token. Now you can run it from root like this: ./ledcheck <port>, where <port> is the GPIO# that you want to toggle. Try the following numbers: 2,3,4,7,11,…

```
#  ./ledcheck 7
```

If you want, remove the cd commands at the top and bottom, and change all the file references to absolute addresses. Change the for loop to an infinite loop. Shorten the timing delays. Season to taste!

That's all for part 1. In part 2 we will get use python to write to the LEDs and read from the switches.

## 5) BASH SCRIPT for GPIO, PART 1:

```
#!/bin/bash

#  Name     : LedCheck 1.0
#  Author   : Bruce E. Hall <bhall66@gmail.com>
#  Date     : 20 Mar 2013
#  Platform : Raspberry Pi, Raspian OS
#
#  A bash script for controlling an LED via a GPIO port.
#
#  Call this script with a single parameter = GPIO port#
#  The LED connected to this port will flash.
#  NOTE:  You must run this script as root.

NUMCYCLES=8     #Number of LED flashes
DELAY=0.5       #Flash delay, 0.5 seconds = 1 Hz

#  Do some basic error checking before we start.
#  Make sure we are root & check for a port number.

if [ $(whoami) != 'root' ]
then
    echo "Must be root to run this script."
    exit
fi

if [ $# != 1 ]
then
    echo "Need a single parameter <GPIO port#>. "
    exit
fi

if (( ($1 < 0) || ($1 > 27) ))
then
    echo "Bad GPIO port number."
    exit
fi

#
#  Start the flasher!
#

echo "Starting LED flasher"
echo "  Opening port" $1
echo $1 > /sys/class/gpio/export

echo "  And making it an output."
echo out > /sys/class/gpio/gpio$1/direction

for COUNT in  `seq $NUMCYCLES`
do
    echo "  Setting value to 1 "
    echo 1 > /sys/class/gpio/gpio$1/value
    sleep $DELAY
    echo "  Setting value to 0"
    echo 0 > /sys/class/gpio/gpio$1/value
    sleep $DELAY
done

echo "  Closing port" $1
echo $1 > /sys/class/gpio/unexport
echo "Done."
```