

GPIO: Light the LED with Python

After setting up the raspberry pi and having hands on practice with the Linux commands, you are now familiar with raspberry pi. Now it's time to work with the GPIO pins of the raspberry pi to have an external interface with the raspberry pi.

Hardware Guide:

Along with the basic setup you will require the following components to get started with the GPIO pins as follows:

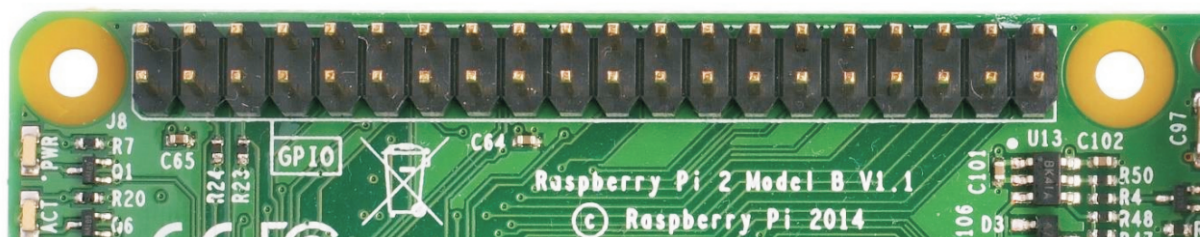
1. LED
2. Resistor
3. Connecting wires
4. Breadboard

Before learning this lesson, you must understand the pin numbering system of the GPIO pins.

GPIO?

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the top edge of the board.

These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Of the 40 pins, 26 are GPIO pins and the others are power or ground pins (plus two ID EEPROM pins which you should not play with unless you know your stuff!)



What are they for? What can we do with them?

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere** and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this.

How the GPIO pins work?

Output

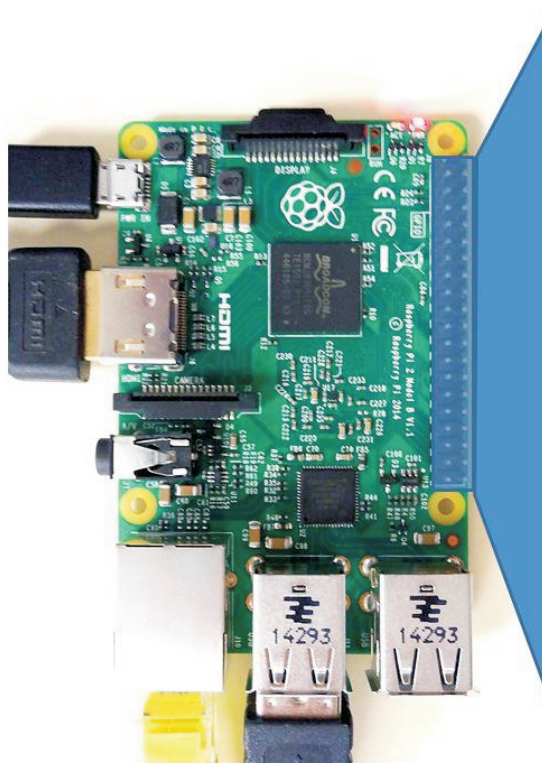
When we use a GPIO pin as an output. Each pin can turn on or off, or go HIGH or LOW in computing terms. When the pin is HIGH it outputs 3.3 volts (3v3); when the pin is LOW it is off.

Input

GPIO outputs are easy; they are on or off, HIGH or LOW, 3v3 or 0v. Inputs are a bit trickier because of the way that digital devices work. Although it might seem reasonable just to connect a button across an input pin and a ground pin, the Pi can get confused as to whether the button is on or off. It might work properly, it might not. It's a bit like floating about in deep space; without a reference, it would be hard to tell if you were going up or down, or even what up or down meant!

Therefore, you will see phrases like "pull up" and "pull down" in Raspberry Pi GPIO tutorials. It's a way of giving the input pin a reference so it knows for certain when an input is received.

Warning: Randomly plugging wires and power sources into your Pi, however, may kill it. Bad things can also happen if you try to connect things to your Pi that use a lot of power.



| GPIO# | 2nd func | Physical Pins | | 2nd func | GPIO# |
|--------|------------|---------------|------|-------------|--------|
| | | pin# | pin# | | |
| N/A | +3V3 | 1 | 2 | +5V | N/A |
| GPIO2 | SDA1 (I2C) | 3 | 4 | +5V | N/A |
| GPIO3 | SCL1 (I2C) | 5 | 6 | GND | N/A |
| GPIO4 | GCLK | 7 | 8 | TXD0 (UART) | GPIO14 |
| N/A | GND | 9 | 10 | RXD0 (UART) | GPIO15 |
| GPIO17 | GEN0 | 11 | 12 | GEN1 | GPIO18 |
| GPIO27 | GEN2 | 13 | 14 | GND | N/A |
| GPIO22 | GEN3 | 15 | 16 | GEN4 | GPIO23 |
| N/A | +3V3 | 17 | 18 | GEN5 | GPIO24 |
| GPIO10 | MOSI (SPI) | 19 | 20 | GND | N/A |
| GPIO9 | MISO (SPI) | 21 | 22 | GEN6 | GPIO25 |
| GPIO11 | SCLK (SPI) | 23 | 24 | CE0_N (SPI) | GPIO8 |
| N/A | GND | 25 | 26 | CE1_N (SPI) | GPIO7 |
| EEPROM | ID_SD | 27 | 28 | ID_SC | EEPROM |
| GPIO5 | N/A | 29 | 30 | GND | N/A |
| GPIO6 | N/A | 31 | 32 | - | GPIO12 |
| GPIO13 | N/A | 33 | 34 | GND | N/A |
| GPIO19 | N/A | 35 | 36 | N/A | GPIO16 |
| GPIO26 | N/A | 37 | 38 | N/A | GPIO20 |
| N/A | GND | 39 | 40 | N/A | GPIO21 |

A note on pin numbering:

When programming the GPIO pins there are two different ways to refer to them: GPIO numbering and physical numbering.

GPIO numbering:

These are the GPIO pins as the computer sees them. The numbers don't make any sense to humans, they jump about all over the place, so there is no easy way to remember them. You will need a printed reference or a reference board that fits over the pins.

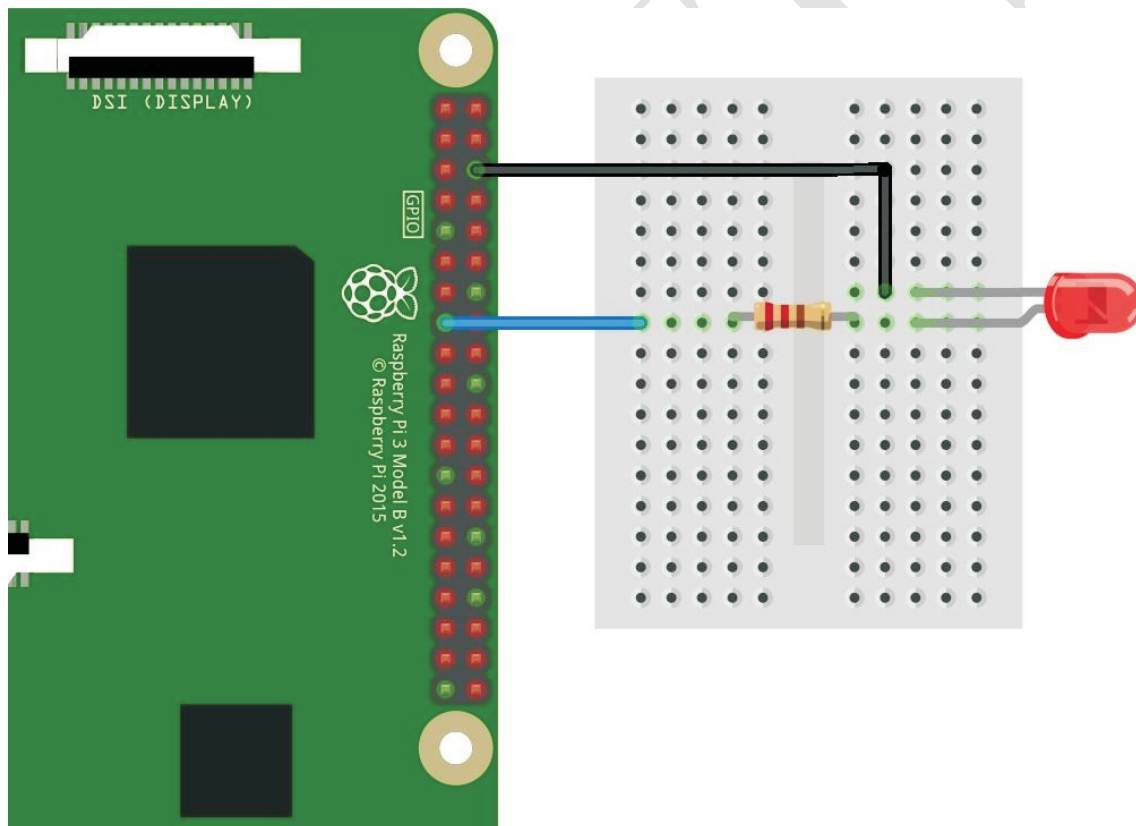
Physical numbering:

The other way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card).

Wiring up your Circuit:

1. Connect the GPIO22 (i.e. Physical Pin 15) Pin of raspberry pi to one end of the resistor.
2. Connect another end of resistor to the positive end (anode) of LED
3. Connect the negative end (cathode) of LED to Ground of raspberry pi.
4. Then Power on your raspberry pi

Circuit Diagram:



Software Guide:

Raspbian OS comes with many preinstalled programming environments. Here we will be using Python for coding.