

P3  
RA9© 2019 John A. Oakey  
V041219c

## RPi.GPIO Module Usage

### Import the module:

```
import RPi.GPIO [as string] - as "IO" is assumed in the following
```

**Pin numbering:** a choice is required to specify which system you will use for channel designation:

**IO.setmode(IO.BOARD)**

...Or...

**IO.setmode(IO.BCM)**

(see reverse side for diagram)

**Setup:** for every channel that is to be used:

**IO.setup(channel, IO.IN)**

**IO.setup(channel, IO.OUT)**

You can specify an initial state for the pin:

**IO.setup(channel, IO.OUT, initial=IO.HIGH)**

Or setup a group at once:

**chan\_list = [11,12]** add multiple channels

can use tuples instead, i.e.: chanlist = (11,12)

**IO.setup(chan\_list, IO.OUT)**

**Read or write (set) pins:** ( NOTE: a "pin" is the same as a "channel")

**IO.input(channel)** (0=False=IO.Low, 1=True=IO.High)

**IO.output(channel, state)** (states same as above)

Can output to several channels with one command:

**chanlist = [11,12]** <- this also works with tuples

**IO.output(chanlist, IO.LOW)** <- this sets all in chanlist to LOW

**IO.output(chanlist, (IO.HIGH, IO.LOW))** <- this sets first

HIGH and the second LOW

### Environmental information:

**GPIO.RPI\_INFO** about your RPi

**GPIO.RPI\_INFO['P1\_REVISION']** Raspberry Pi board revision

**GPIO.VERSION** RPi.GPIO version number

Find the function of a channel:

**func = IO.gpio\_function(pin)**

will return a value from:

IO.IN, IO.OUT, IO.SPI, IO.I2C, IO.HARD\_PWM, IO.SERIAL,

IO.UNKNOWN

### Pull UP / Pull DOWN:

Unconnected pins float.

Default values (High or Low) can be set in **software** or with **hardware**

#### Hardware:

Pull Up:

Input channel -> 10K resistor -> 3.3V

Pull Down:

Input channel -> 10K resistor -> 0V

#### Software:

**IO.setup (channel, IO.IN, pull\_up\_down = IO.PUD\_UP)** or **IO.PUD\_DOWN** or **IO.PUD\_OFF**)

**Edge detection:** change of state event — 3 ways to handle

**1. wait\_for\_edge()** function - stops everything until an edge is detected.

**IO.wait\_for\_edge (channel, IO.RISING)** can detect edges of type IO.RISING, IO.FALLING or IO.BOTH

**2. event\_detected()** function - use in a loop with other activity — event triggers priority response. Example:

**IO.add\_event\_detect(channel, IO.RISING)**  
activity detection on a channel

[your loop activity here]

if **IO.event\_detected(channel)**:

print('Button pressed')

**3. threaded callbacks** - RPi.GPIO runs a second thread for callback functions. This means that callback functions can be run at the same time as your main program, in immediate response to an edge. For example:

def my\_callback(channel):

print('Edge detected on channel %s'%channel)

print('This is run in a different thread to your main program.')

**IO.add\_event\_detect(channel, IO.RISING, callback=my\_callback)** add rising edge detection on a channel

...the rest of your program...

If you want more than one callback function:

def my\_callback\_one (channel):

print ('Callback one')

def my\_callback\_two (channel):

print ('Callback two')

**IO.add\_event\_detect(channel, IO.RISING)**

**IO.add\_event\_callback(channel,**

**my\_callback\_one)**

**IO.add\_event\_callback(channel,**

**my\_callback\_two)**

Note that in this case, the callback functions are run **sequentially, not concurrently**. This is because there is only one thread used for callbacks, and every callback is run in the order in which it is defined.

**Switch debounce:** solutions to a button event causing multiple callbacks

**Hardware:** add a 0.1uF capacitor across your switch.

**Software:** add the bouncetime= parameter to a function where you specify a callback function. bouncetime= should be specified in milliseconds.

**IO.add\_event\_detect(channel, IO.RISING, callback=my\_callback, bouncetime=200)** or

**IO.add\_event\_callback(channel, my\_callback, bouncetime=200)**

### Remove Event Detection:

**IO.remove\_event\_detect(channel)**

**Cleanup:** resets all channels and clears the pin numbering system at the end of a program. Just good practice.

**IO.cleanup()**

Or cleanup selected pins:

**IO.cleanup(channel)**

**IO.cleanup( (channel1, channel2) )** <-tuple

**IO.cleanup( [channel1, channel2] )** <-or list

**PWM in RPi.GPIO** is an analog signal, Pulse Width Modulation - available ONLY One of the Pi's pins: board #12 = BCM #18 ; is hardware used mostly for audio

To create a software instance of PWM on any in/out pin:

**p = IO.PWM(channel, frequency)**  
To start PWM: **p.start(\*dc)**

\*dc is the duty cycle (0.0 <= dc <= 100.0)

Pin #7/GPIO [4] is 1 wire capable for low speed sensor input

To change the frequency:

**p.ChangeFrequency(freq)** freq is the new frequency in Hz\*

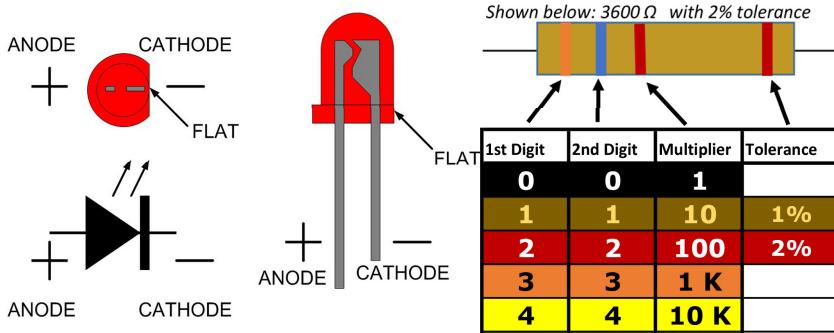
To change the duty cycle:

**p.ChangeDutyCycle(dc)** where 0.0 <= dc <= 100.0

To stop PWM:

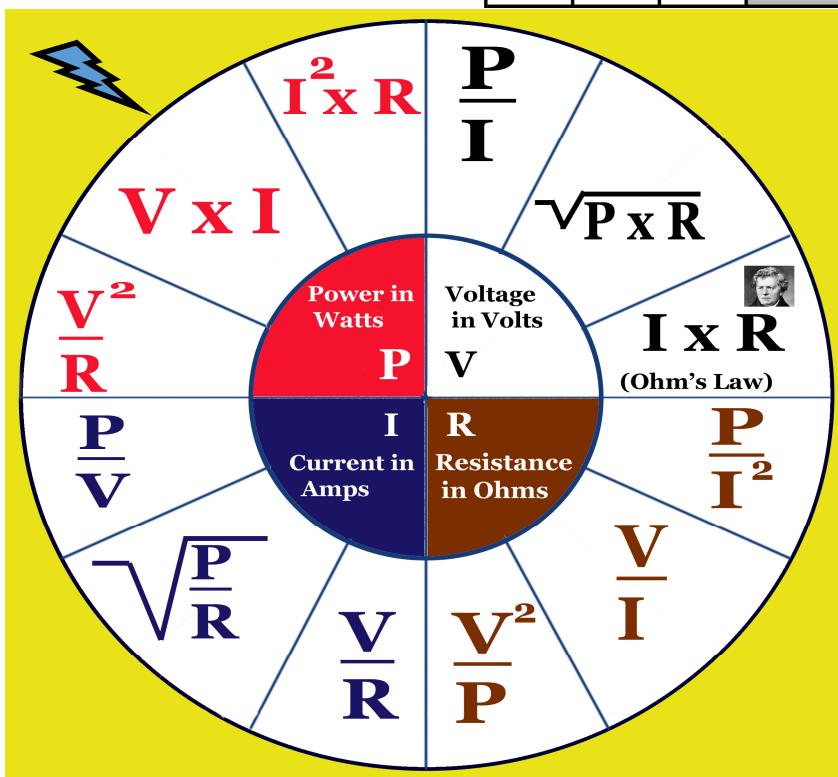
**p.stop()**

\*100 = 100 times a second, .5 = once every 2 seconds, .1 is every 10 seconds, .0167 = once a minute



270Ω -> red, purple, brown  
330Ω -> orange, orange, brown  
10KΩ -> brown, black, yellow

Note: RPi maximum current to a single pin is 16mA, to all pins is 50mA



Raspberry Pi Model B+			
3V Power	1	2	5V Power
GPIO [2] SDA1 12C	3	4	5V Power
GPIO [3] SCL1 12C	5	6	Ground
GPIO [4] 1 wire capable	7	8	GPIO [14] UART0-TXD
Ground	9	10	GPIO [15] UART0-RXD
GPIO [17]	11	12	GPIO [18] PCM-CLK
GPIO [27]	13	14	Ground
GPIO [22]	15	16	GPIO [23]
3V Power	17	18	GPIO [24]
GPIO [10] SPIo-MOSI	19	20	Ground
GPIO [9] SPIo-MOSO	21	22	GPIO [25]
GPIO [11] SPIo-SCLK	23	24	GPIO [8] SPIo-CEO-N
Ground	25	26	GPIO [7] SPIo-CE1-N
ID_SD [0] 12C ID EEPROM	27	28	ID_SC [1] 12C ID EEPROM
GPIO [5]	29	30	Ground
GPIO [6]	31	32	GPIO [12]
GPIO [13]	33	34	Ground
GPIO [19] PCM-FS	35	36	GPIO [16]
GPIO [26]	37	38	GPIO [20] PCM-DIN
Ground	39	40	GPIO [21] PCM-DOUT

Comments and suggestions appreciated:  
[john@johnnoakey.com](mailto:john@johnnoakey.com)

#### SERIAL PERIPHERAL INTERFACE PINS

19 MOSI-master output, slave input  
21 MISO-master input, slave output  
23 SCK-serial clock 24 & 26-slave select pins  
UART - UNIVERSAL ASYNCHRONOUS RECEIEVER/TRANSMITTER pins  
8 UART-TDX & 10 UART-RDX