BIG DADDY'S


P1
RA7


V052217a

# PYTHON TOOLBOX For 3.5

## RPi.GPIO Module Usage

### Import the module:

**import RPi.GPIO** (**as** "whatever" if desired - **as IO** is assumed in the following)

### Pin numbering:

a choice is required to specify channel designations:
**IO.setmode**(IO.BOARD)
...Or...
**IO.setmode**(IO.BCM)

(see reverse side for diagram)

### Setup:

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 bunch at a time:
**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 to IO.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, in which every callback is run in the order in which they have been 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)**

# BIG DADDY'S PYTHON TOOLBOX For 3.5



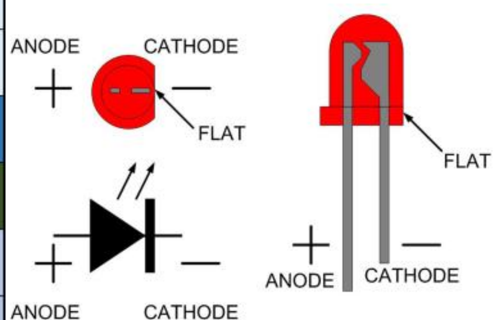
**Cleanup:** resets all channels and clears pin num-bering system at the end of a program - just do it.
**IO.cleanup()**
Or cleanup just select pins:
**IO.cleanup(channel)**
**IO.cleanup( (channel1, channel2) )** <-tuple
**IO.cleanup( [channel1, channel2] )** <-or list

**PWM in RPi.GPIO** is an analog signal, **P**ulse **W**idth **M**odulation - available ONLY on one of the Pi's pins: board #12 = BCM #18 ; 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)** where dc is the duty cycle (0.0 <= dc <= 100.0)
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()**

Common breadboard numbering

| Raspberry Pi Model B+ | | | | |
|---|---|---|---|---|
| 1 | 3V Power | 1 | 2 | 5V Power |
| 2 | GPIO2 SDA1 12C | 3 | 4 | 5V Power |
| 3 | GPIO3 SCL1 12C | 5 | 6 | Ground |
| 4 | GPIO4 | 7 | 8 | GPIO14 UART0-TXD |
| 5 | Ground | 9 | 10 | GPIO15 UART0-RXD |
| 6 | GPIO17 | 11 | 12 | GPIO18 PCM-CLK |
| 7 | GPIO27 | 13 | 14 | Ground |
| 8 | GPIO22 | 15 | 16 | GPIO23 |
| 9 | 3V Power | 17 | 18 | GPIO24 |
| 10 | GPIO10 SPI0-MOSI | 19 | 20 | Ground |
| 11 | GPIO9 SPI0-MOSO | 21 | 22 | GPIO25 |
| 12 | GPIO11 SPI0-SCLK | 23 | 24 | GPIO8 SPI0-CE0-N |
| 13 | Ground | 25 | 26 | GPIO7 SPI0-CE1-N |
| 14 | ID_SD 12C ID EEPROM | 27 | 28 | ID_SC 12C ID EEPROM |
| 15 | GPIO5 | 29 | 30 | Ground |
| 16 | GPIO6 | 31 | 32 | GPIO12 |
| 17 | GPIO13 | 33 | 34 | Ground |
| 18 | GPIO11 PCM-FS | 35 | 36 | GPIO16 |
| 19 | GPIO26 | 37 | 38 | GPIO20 PCM-DIN |
| 20 | Ground | 39 | 40 | GPIO20 PCM-DOUT |

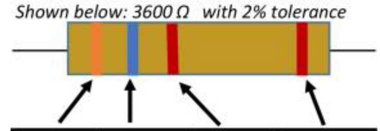
**SERIAL PERIPHERAL INTERFACE PINS**
**19** MOSI-master output, slave input
**21** MISO-master input, slave output
**23** SCK-serial clock **24** & 26-slave select pins
**UART - UNIVERAL ASYNCHRONOUS RECEIEVER/TRANSMITTER pins**
**8** UART-TDX & **10** UART-RDX



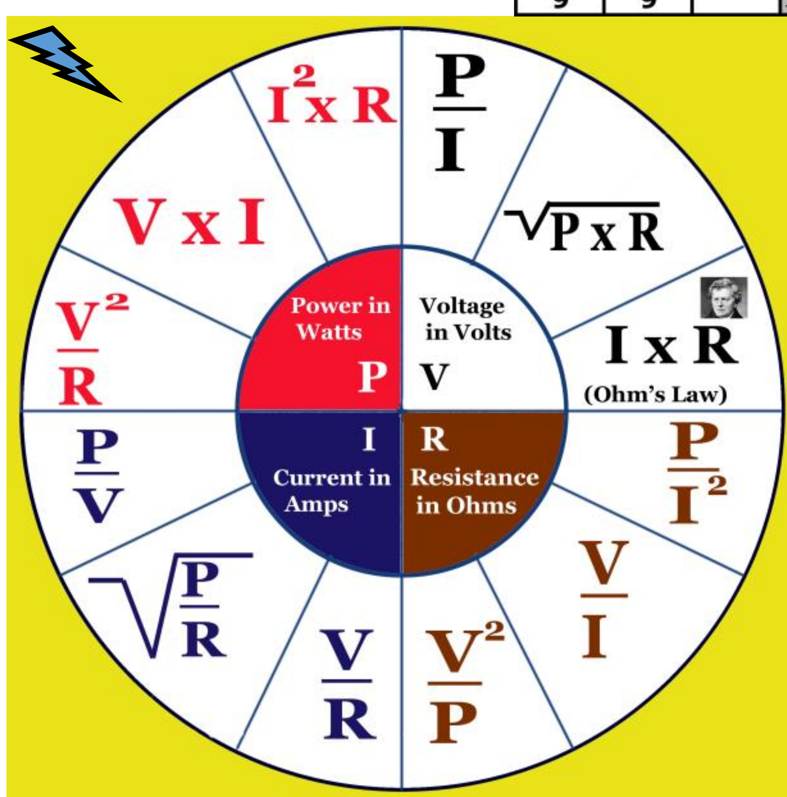

**270Ω -> red, purple, brown**
**330Ω -> orange, orange, brown**
**10KΩ -> brown, black, yellow**

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

*Shown below: 3600 Ω with 2% tolerance*



| 1st Digit | 2nd Digit | Multiplier | Tolerance |
|---|---|---|---|
| 0 | 0 | 1 | |
| 1 | 1 | 10 | 1% |
| 2 | 2 | 100 | 2% |
| 3 | 3 | 1 K | |
| 4 | 4 | 10 K | |
| 5 | 5 | 100 K | |
| 6 | 6 | 1 M | |
| 7 | 7 | 10M | |
| 8 | 8 | | 5% gold |
| 9 | 9 | | 10% silver |




comments and suggestions appreciated:
john@johnoakey.com