

RPi.GPIO Module Usage

Configureable pins: 3V3

Import the module:

Out and 3V3 tolerant In

import RPi.GPIO [as string] - as "IO" is assumed in the following**Pin numbering:**a choice is **required** to specify **BCM** or **BOARD** to designate pins/channels: Note that for all intents a "PIN" means the same thing as a "CHANNEL": (see diagram on page2) **IO.setmode(IO.BCM)** or **IO.setmode(IO.BOARD)****Setup:** Every pin that is to be used must be defined as in or out:**IO.setup(channel, IO.IN) or IO.setup(channel, IO.OUT)**An initial state can be set by adding: **initial=IO.HIGH** or **IO.Low**For example: **IO.setup(channel, IO.OUT, initial=IO.HIGH)**

Multiple channels can be set at once using a list or a tuple:

chan_list = [11,12] or chan_tuple = (11,12)For example: **IO.setup(chan_list, IO.OUT)****Read or write (set) pins:****IO.input(channel)** (returns: 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 numberFind the function of a channel: **func = IO.gpio_function(pin)**

Returns: IN, OUT, SPI, I2C, HARD_PWM, SERIAL, or 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) set up detection

[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.**4. Remove Event Detection:****IO.remove_event_detect(channel)****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)****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:** Pulse Width Modulation - analog signal, **Hardware** available on (BCM / board)**PWM0: 12/32, 18/12; PWM1: is used for audio 13/33 - so use PWM0: GPIO12/Pin32**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)**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**Using 1-wire:** A single channel: **GPIO**

[4] is 1-wire capable for low speed sensor input; Rpi must be configured to utilize alternate pin functions like this!

**A Small RPi 2835 BCM\GPIO Glossary of Terms**

BCM: Broadcom; BCM = GPIO in pin numbering

CE0/CE1: SPI Chip Select 0/1

DPI: Display Parallel Interface - uses 28GPIO pins

GPCLK: General Purpose Clock

I²C/I2C/i2c/IIC: Inter-Integrated Circuit; serial bus;

SCK or SCLK: Serial Clock, master to slave; SCL: BSC Master clock line; SDA: serial data pin; ID_SC: connection to SCL0; ID_SD connection to SDA0

SPI: Serial Peripheral Interface

JTAG: Joint Test Action Group

MSIO/MOSI: Master Slave Out/In

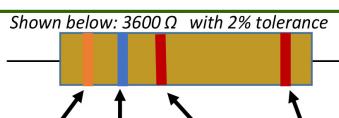
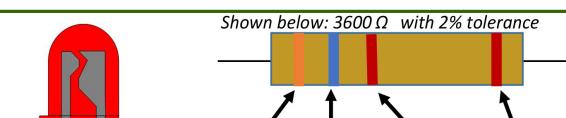
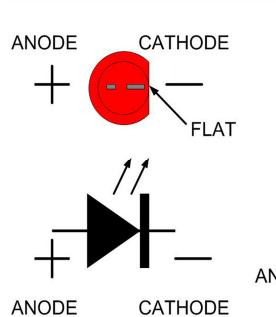
PCM: Pulse Code Modulation

PWM: Pulse Width Modulation

SDIO: SD card interface

W1-GPIO: 1-Wire interface; default is bcm[4]

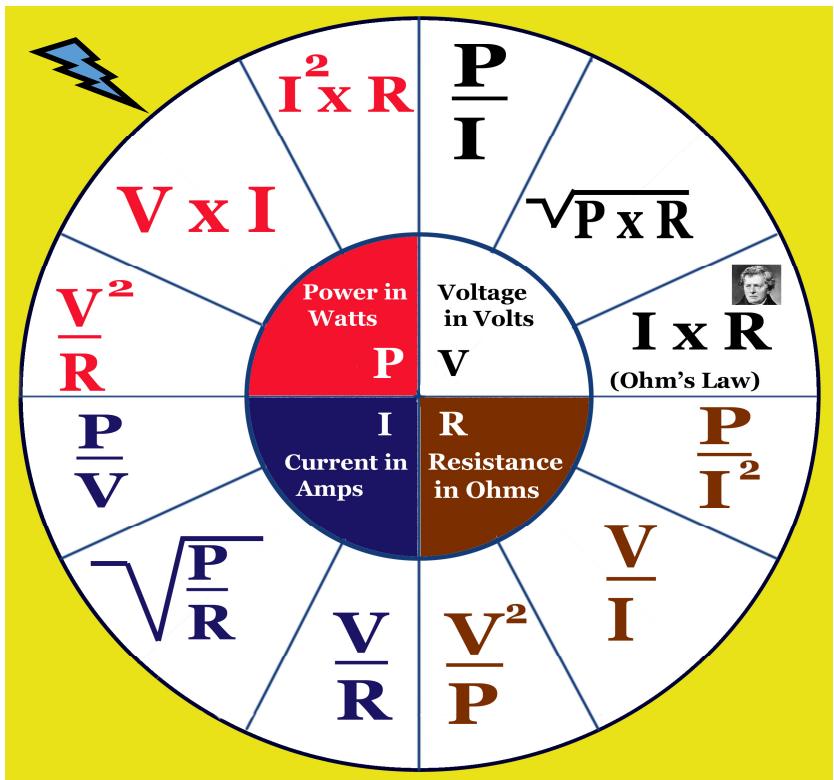
UART: Universal Asynchronous Receiver/Transmitter,
TDX: transmit, GPIO[8]
RDX: receive, GPIO[10]
default is console in/out



1st Digit	2nd Digit	Multiplication	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

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

RPi maximum current to a single pin is 16mA, to all pins is 50 mA. A 3v3 supply is ~ 50 mA

**2835 Raspberry Pi Model B+**

1	3V3 Power	1	2	5V Power	- / -
2	GPIO [2] SDA i2c	3	4	5V Power	8 / -
3	GPIO [3] SCL i2c	5	6	Ground	9 / -
4	GPIO [4] 1 wire GPCLK0	7	8	GPIO [14] UART0-TXD	7 / 15
5	Ground	9	10	GPIO [15] UART0-RXD	- / 16
6	GPIO [17]	11	12	GPIO [18] PCM-CLK / PWM0	0 / 1
7	GPIO [27]	13	14	Ground	2 / -
8	GPIO [22]	15	16	GPIO [23]	3 / 4 wiringpi # system - / 5 12/- 13/6
9	3V3 Power	17	18	GPIO [24]	
10	GPIO [10] SPI0: MOSI	19	20	Ground	
11	GPIO [9] SPI0-MISO	21	22	GPIO [25]	
12	GPIO [11] SPI0-SCLK	23	24	GPIO [8] SPI0-CE0-N	14 / 10
13	Ground	25	26	GPIO [7] SPI0-CE1-N	- / 11
14	ID_SD [0] i2c0 EEPROM	27	28	ID_SC [1] i2c EEPROM	30 / 31
15	GPIO [5]	29	30	Ground	21 / -
16	GPIO [6]	31	32	GPIO [12] PWM0 use this 1	22 / 26
17	GPIO [13] PWM1 audio	33	34	Ground	23 / -
18	GPIO [19] PCM-FS	35	36	GPIO [16]	24 / 27
19	GPIO [26]	37	38	GPIO [20] PCM-DIN	25 / 28
20	Ground	39	40	GPIO [21] PCM-DOUT	- / 29