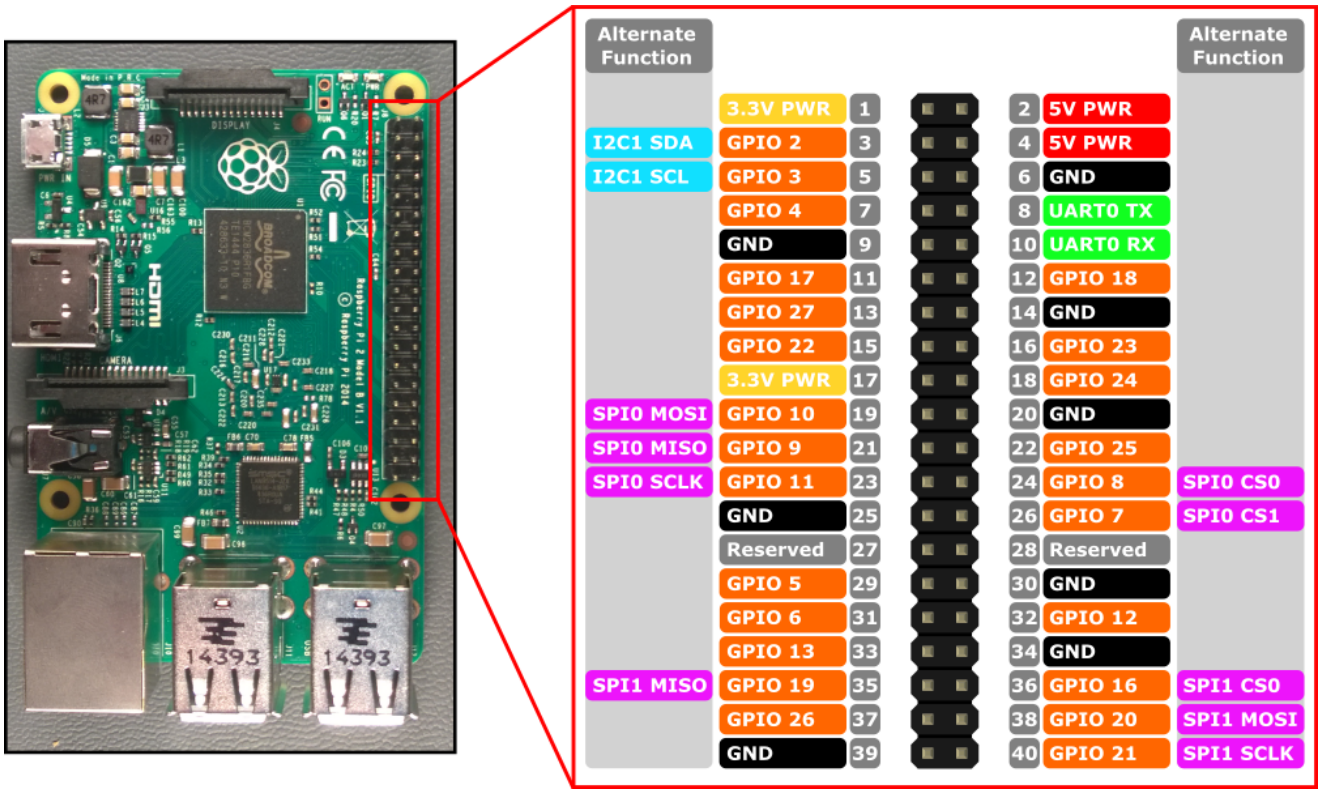
RASPBERRY PI QUICK START

Mr Baumgarten
15 February 2018

GPIO PIN OUT

For Raspberry Pi model 2 and 3



It is unfortunate that each Raspberry Pi as more than one numbering scheme, and the numbers change/move around based on the version of the Pi. You specify in your Python code which numbering scheme you are using with the **`GPIO.setmode()`** command (see example code).

GPIO.setmode() has two acceptable values as follows:

- **GPIO.BOARD** specifies that you are referring to the pins by the number of the pin on the board. These are the numbers in grey, which are by themselves on the above diagram.
- **GPIO.BCM** means that you are referring to the pins by the "Broadcom SOC channel" number, these are the numbers that appear after the word "GPIO" in the orange rectangles on the above diagram.

INSTALL PYTHON TOOLS

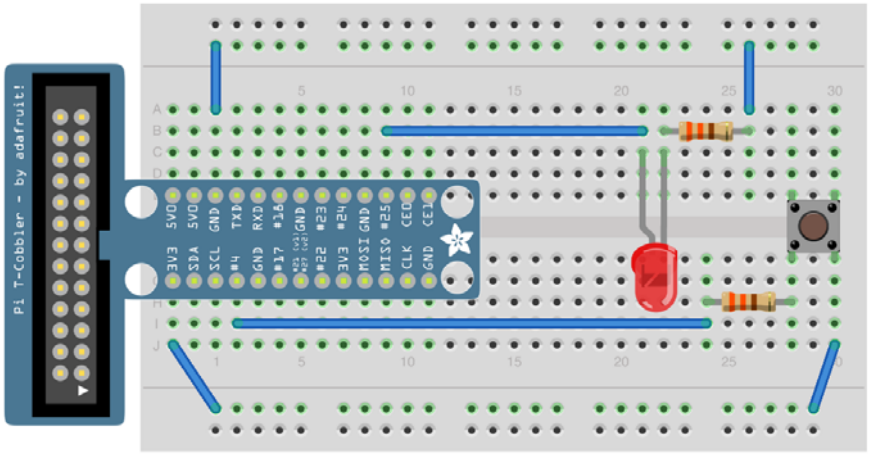
In a terminal:

```
sudo apt-get update
sudo apt-get install python-dev python-pip
sudo pip install --upgrade distribute
sudo pip install ipython
sudo pip install --upgrade RPi.GPIO
```

RASPBERRY PI QUICK START

SAMPLE PYTHON CODE AND WIRING

To control and LED with a push button.




```
import RPi.GPIO as GPIO

# Set board pin numbering system
GPIO.setmode(GPIO.BCM)

# Setup each pin you use: whether it is input/output and if the default is on/off
GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(25, GPIO.OUT, initial=GPIO.LOW)

GPIO.add_event_detect(4, GPIO.BOTH)

def my_callback():
    GPIO.output(25, GPIO.input(4))

GPIO.add_event_callback(4, my_callback)
```

SOME FUNCTION REFERENCES

GPIO.INPUT(CHANNEL)

This will return either 0 / GPIO.LOW / False or 1 / GPIO.HIGH / True.

You can take a snapshot of an input at a moment in time:

```
if GPIO.input(channel):
    print('Input was HIGH')
else:
    print('Input was LOW')
```

To wait for a button press by polling in a loop:

```
while GPIO.input(channel) == GPIO.LOW:
    time.sleep(0.01)  # wait 10 ms to give CPU chance to do other things
```

GPIO.OUTPUT(CHANNEL, STATE)

State can be 0 / GPIO.LOW / False or 1 / GPIO.HIGH / True.

To set an output high:

```
GPIO.output(12, GPIO.HIGH)
 # or
GPIO.output(12, 1)
 # or
GPIO.output(12, True)
```

To set an output low:

```
GPIO.output(12, GPIO.LOW)
 # or
GPIO.output(12, 0)
 # or
GPIO.output(12, False)
```

GPIO.ADD_EVENT_DETECT(CHANNEL, EVENTTOLOOKFOR)

EventToLookFor can be:

- GPIO.RISING — The value has moved to 1 / GPIO.HIGH / True
- GPIO.FALLING — The value has moved to 0 / GPIO.LOW / False
- GPIO.BOTH — The value has changed. Get the GPIO.input() function to determine which way.

```
GPIO.add_event_detect(channel, GPIO.RISING)  # add rising edge detection on a channel
do_something()
if GPIO.event_detected(channel):
    print('Button pressed')
```

GPIO. .ADD_EVENT_CALLBACK(CHANNEL, MY_CALLBACK)

Setup a callback function to be executed whenever a value change is detected on an input pin.

```
def my_callback(channel):
    print('Callback!')

GPIO.add_event_detect(channel, GPIO.RISING)
GPIO.add_event_callback(channel, my_callback)
```

GPIO.CLEANUP()

By returning all channels you have used back to inputs with no pull up/down, you can avoid accidental damage to your RPi by shorting out the pins. Note that this will only clean up GPIO channels that your script has used. Note that GPIO.cleanup() also clears the pin numbering system in use.

BOARD AND LIBRARY INFO

To discover information about your RPi:

```
GPIO.RPI_INFO
```

To discover the Raspberry Pi board revision:

```
GPIO.RPI_INFO['P1_REVISION']
GPIO.RPI_REVISION    (deprecated)
```

To discover the version of RPi.GPIO:

```
GPIO.VERSION
```

RASPBERRY PI QUICK START

REFERENCES

http://raspberry.io/projects/view/reading-and-writing-from-gpio-ports-from-python/

https://raspberrypi.stackexchange.com/questions/12966/what-is-the-difference-between-board-and-bcm-for-gpio-pin-numbering

https://sourceforge.net/p/raspberry-gpio-python/wiki/BasicUsage/