

# MAKER WORKSHOPS

## INTRO to RASPBERRY PI



Instructors: Raphael Kopala, Shawn Nook

Unlondon Digital Media Assoc.

# UnLondon

## Digital Media Association

### **Enabling Exploration, Creativity, and Excellence In Art+Make+Tech**

Challenging and embracing ideas related to new technologies and social platforms through the education, entertainment and engagement of our membership and the community-at-large.

- 121Studios: Coworking for Creatives
- Unlab: Hackerspace
- Events: STEAM Outreach & Edu., ExplodeConf, Nuit Blanche

# Raphael

## Day Job

### **Mechanical Engineer**

- Working in the medical device industry
- Experience in medical device R&D and Manufacturing
- Teaching SolidWorks CAD at Fanshawe in evenings

# Raphael

## The Fun Stuff

### **Thinkier, Jack of all Trades – Master of None**

- Arduino & RPi for Fun, and Odd Jobs
- 3D Printer Hobbyist
- PC Builder & Gamer
- Fish keeper

# Shawn

## Day Job

### **Freelance Embedded Systems Engineer**

- Indoor location tracking w/ Bluetooth
- Keychain / Fitness Band Widgets
- Joystick for VR
- Remote Controls
- Internet of S\*#t

# Shawn

## The Fun Stuff

**Hacker, Church of the Weird Machine, Odd Duck**

- Arduino compatible implant
- EEG Games / Toy Hacking
- Brain Stimulation
- Be Weird, Make Weird, Have Fun!

# RASPBERRY PI

## Essentials



# RASPBERRY PI

## ESSENTIALS

### What is it?

The Raspberry Pi is a low cost credit-card sized computer.

Wide range of possibilities

- Normal computing capabilities (web browsing, spreadsheets)
- Interact with external world through sensors and actuators

Created by Raspberry Pi Foundation





# RASPBERRY PI ESSENTIALS

## Accessories



**Display**



**Camera**



**Motor Controller**



**Remote Control**



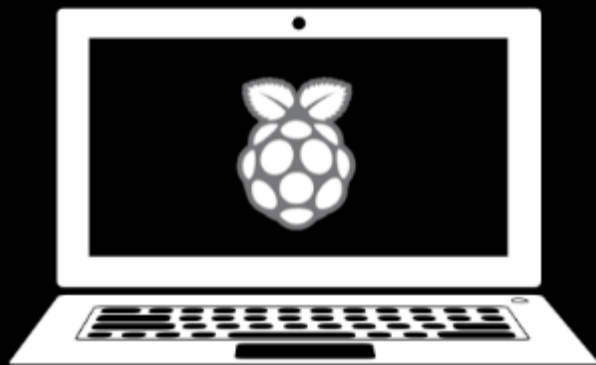
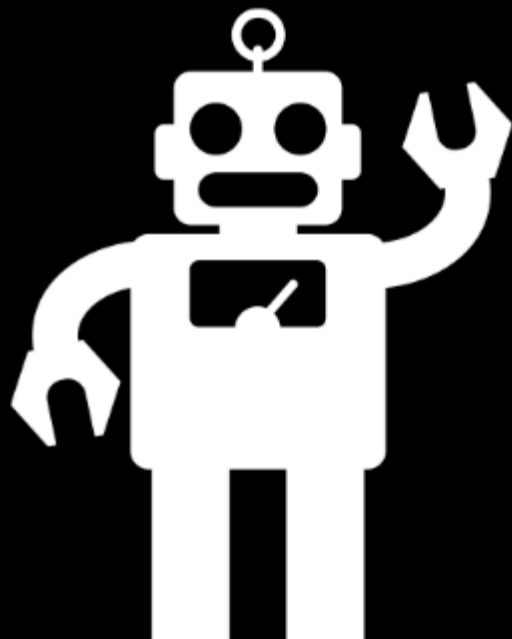
**WiFi**



**Case**

# RASPBERRY PI

## WHAT CAN YOU DO



# RASPBERRY PI

## What Can You Do?

### Projects



**Mini-Laptop**



**Robot**



**Mini-Arcade**



**Home Automation**



**Radio**

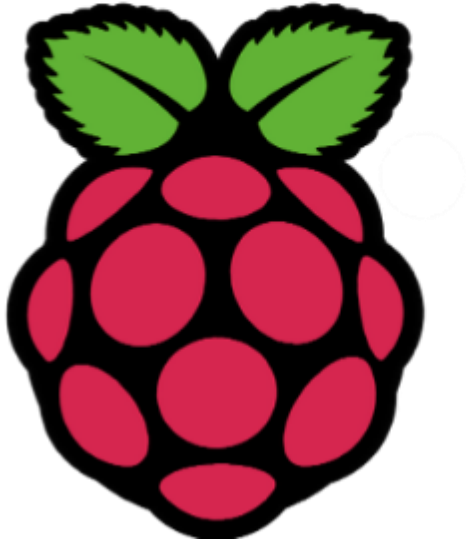


**LED**

# RASPBERRY PI

## What Can You Do?

**Brainstorming time!**



+

\_\_\_\_\_

=

?

# RASPBERRY PI SETUP



# RASPBERRY PI SETUP

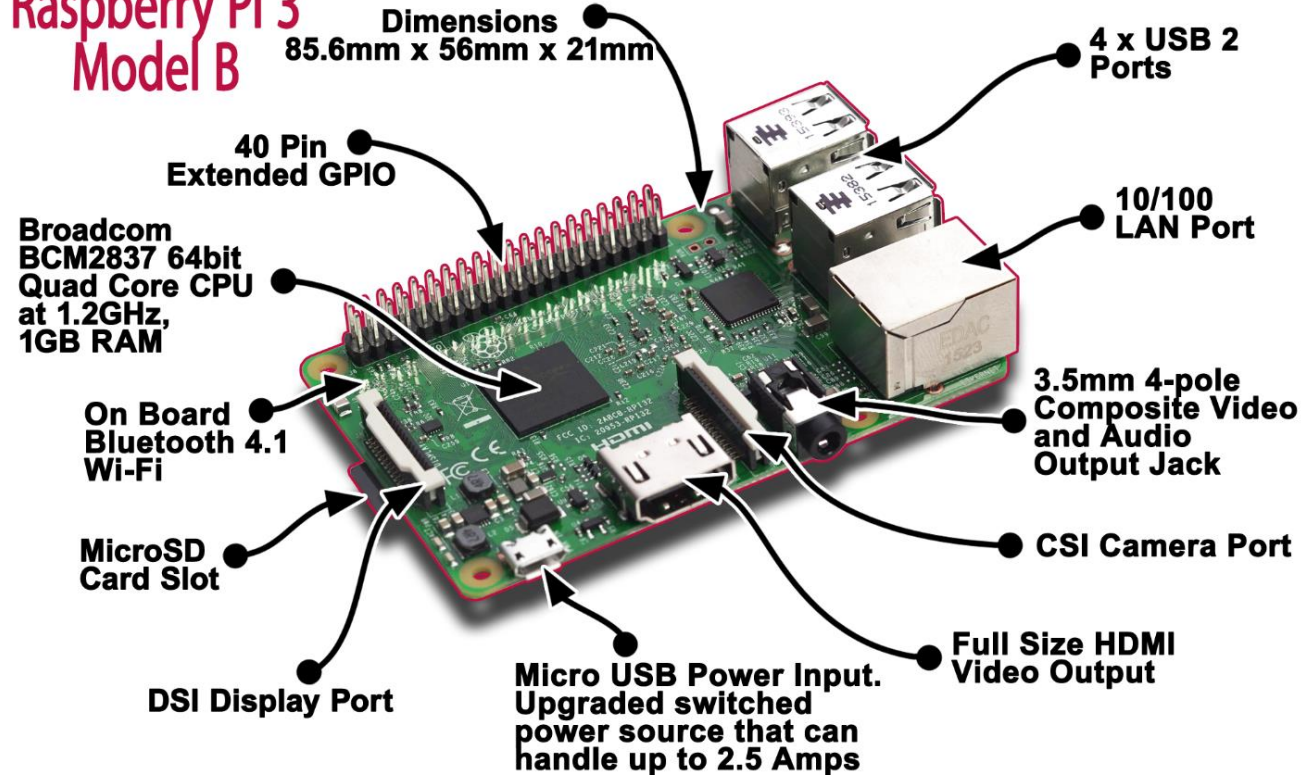
## Your Kit



# RASPBERRY PI ESSENTIALS

## Components

### Raspberry Pi 3 Model B



<http://www.memoryexpress.com/Products/MX61461>



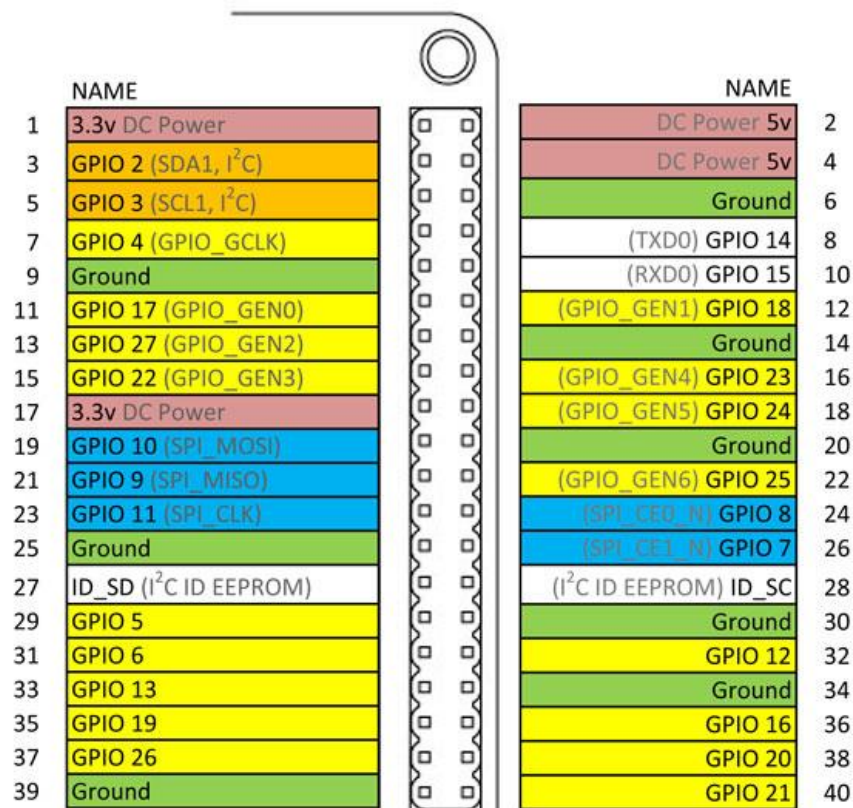
# RASPBERRY PI GPIO

## KEY

Power
Ground
UART
I <sup>2</sup> C
SPI
GPIO

<http://blog.mcmelectronics.com/image.axd?picture=%2F2016%2F03%2FGPIO-Chart2.jpg>

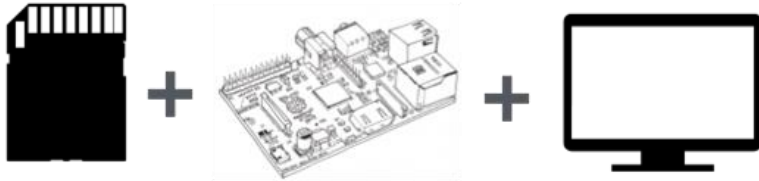
## Raspberry Pi 3 GPIO Pin Layout





# RASPBERRY PI SETUP

## 1. GET CONNECTED



## 2. PLUG IT IN



# RASPBERRY PI LINUX BASICS

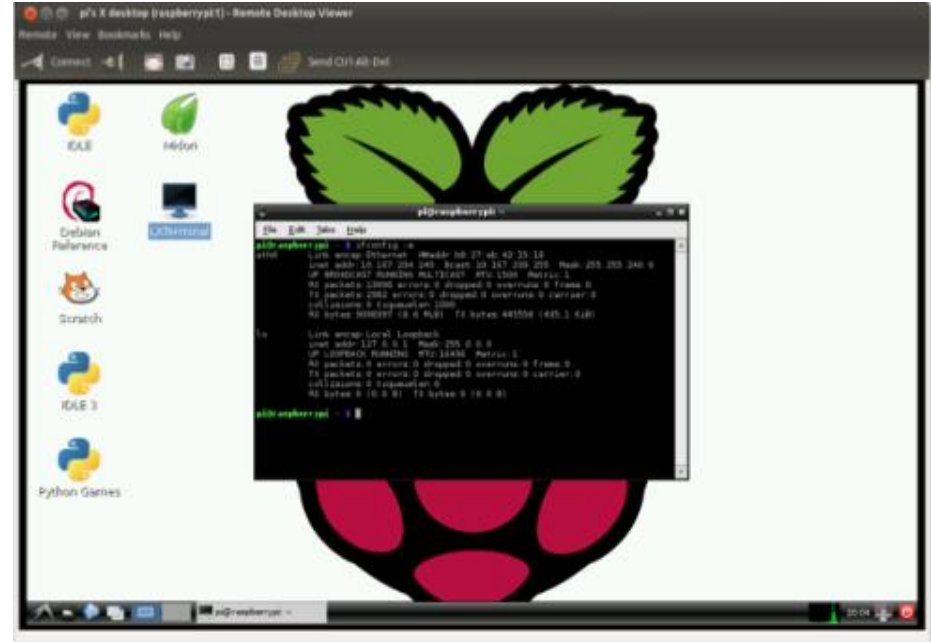


# RASPBERRY PI

## Linux Basics

- Similar to Windows or Mac
- GUI or console
- Good linux intro:

[www.ee.surrey.ac.uk/Teaching/Unix/](http://www.ee.surrey.ac.uk/Teaching/Unix/)



Good starting point: [www.ee.surrey.ac.uk/Teaching/Unix/](http://www.ee.surrey.ac.uk/Teaching/Unix/)

# RASPBERRY PI

## Linux Basics

Linux Command	Description
<code>cd <i>'directory'</i></code>	Change to <i>'directory'</i>
<code>cd ..</code>	Change to parent directory
<code>ls</code>	List current directory contents
<code>pwd</code>	Print name of current directory
<code>mkdir <i>'directory'</i></code>	Create <i>'directory'</i>
<code>rmdir <i>'directory'</i></code>	Remove <i>'directory'</i>
<code>rm <i>'file'</i></code>	Remove <i>'file'</i>

# RASPBERRY PI

## Linux Basics

Linux Command	Description
<code>sudo 'command'</code>	Run a <i>'command'</i> as super user
<code>nano 'file'</code>	Use text editor to create/edit <i>'file'</i>
<code>cp 'file_1' 'file_2'</code>	Copies file_1 as file_2
<code>startx</code>	Start Rasbian Desktop
<code>sudo shutdown</code>	Shutdown
<code>sudo reboot</code>	Reboot
<code>ifconfig</code>	View Networking Information
<code>sudo apt-get update</code>	Update repositories
<code>sudo apt-get upgrade</code>	Upgrade repositories

# Login + Launch GUI

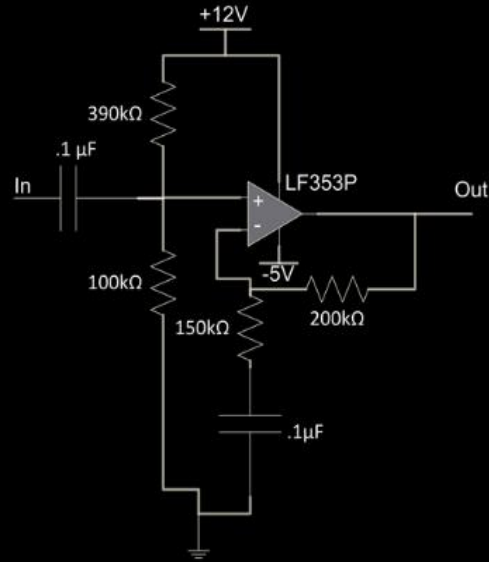
Username: pi

Password: raspberry (\*Note: keystrokes will not show up for the password)

Type startx to start the rasbian gui

# RASPBERRY PI

## Electronics Basics



# RASPBERRY PI

## Electronics Basics

### Current

- Measured in Amps (A)
- Represents the speed, or flow of charge in a circuit

### Voltage

- Measured in Volts (V)
- Represents potential, or pressure applied to a charge in a circuit

### Resistance

- Measured in Ohms ( $\Omega$ )
- Represents the resistance in a circuit that impedes the flow of charge



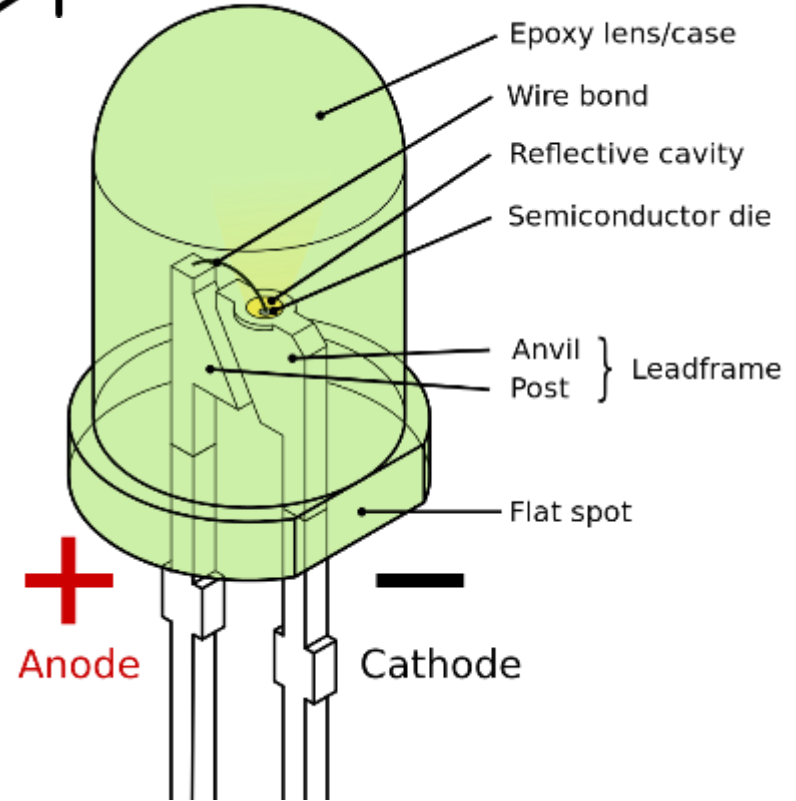
# LED / Diode

Direction/polarity matters!

- Like a one way valve

Positive (+) = Longer Leg

Negative (-) = Shorter Leg



# Resistors



- Direction/polarity doesn't matter
- Each has a fixed value
- Value indicated by coloured bands



# Resistor Colour Bands




Diagram of a resistor with four color bands: red, black, red, and gold. The bands are labeled as 1st digit, 2nd digit, Multiplier, and Tolerance.

1st digit	2nd digit	Multiplier	Tolerance
0	0	x 1	±1%
1	1	x 10	±2%
2	2	x 100	
3	3	x 1K	
4	4	x 10K	
5	5	x 100K	
6	6	x 1M	
7	7		
8	8	x 0.1	±5%
9	9	x 0.01	±10%

Example; what is the resistance of the resistor below?



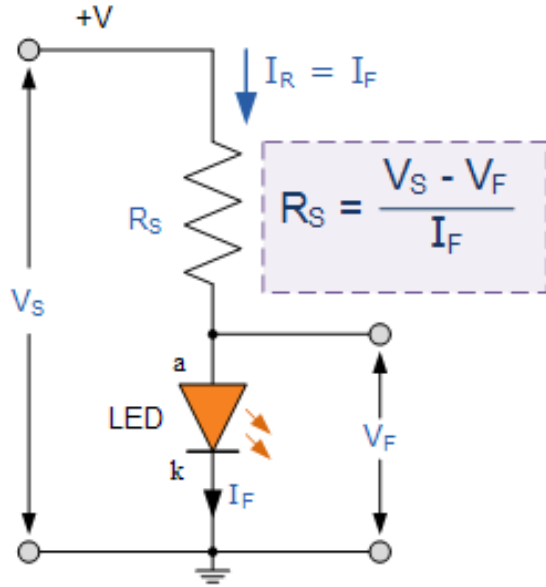
Using the chart:

Brown, Black, Brown, Gold  
1 + 0 x 10 ± 5%

Results in:

100 +/- 5% ohms  
Between (95-105 ohms)

# Calculating Resistors for LED's



Source:

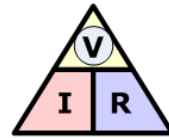
[http://www.electronics-tutorials.ws/diode/diode\\_8.html](http://www.electronics-tutorials.ws/diode/diode_8.html)  
<http://www.electronics-tutorials.ws/dccircuits/dcp23.gif>

Example: 3.3V Supply with a 1.8V LED @ 0.01A

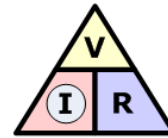
$$R_S = \frac{V_S - V_F}{I_F} = \frac{3.3V - 1.8V}{10mA} = \frac{3.3V - 1.8V}{0.01A} = 150\Omega$$

Most LEDs are specified with a forward voltage ( $V_S$ ) to turn on the LED, and a maximum current draw ( $I_F$ )

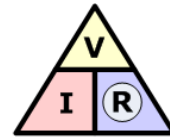
Remember;  
(ohms law)



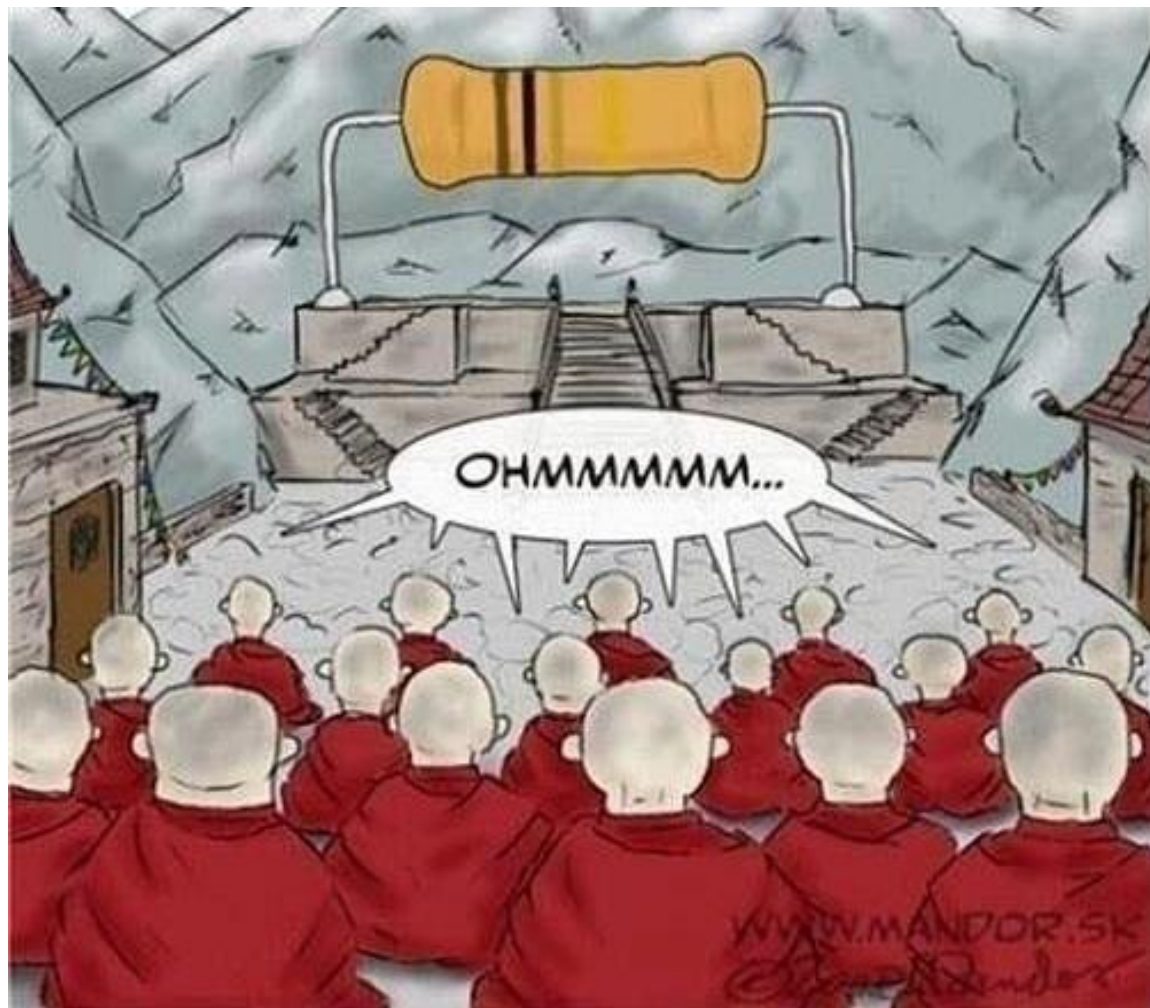
$$\textcircled{V} = I \times R$$



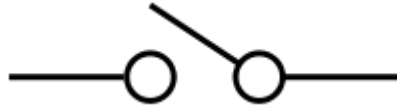
$$\textcircled{I} = \frac{V}{R}$$



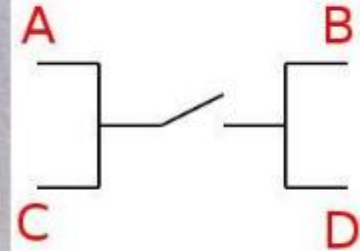
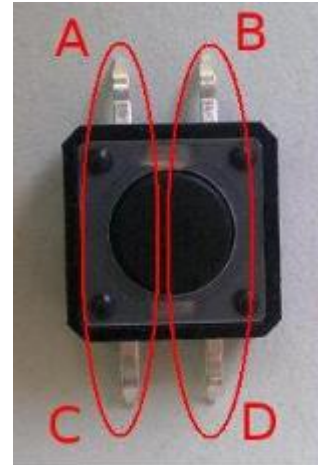
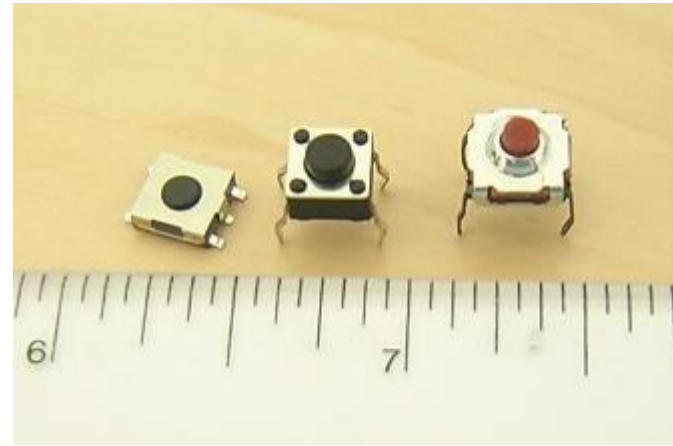
$$\textcircled{R} = \frac{V}{I}$$



# Switch



- On/Off
- Momentary vs. Toggles

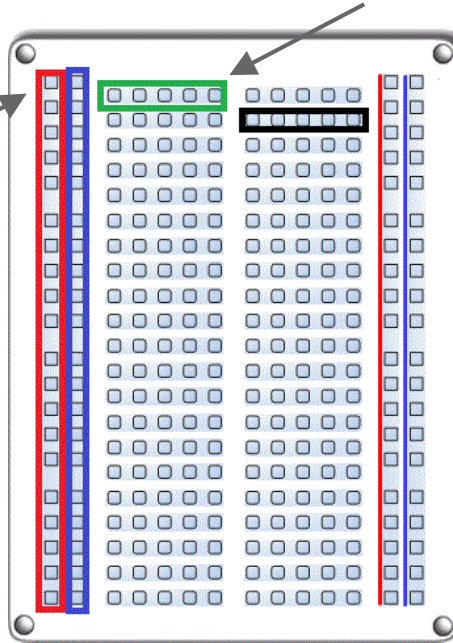


Pins A & C, and B & D are connected always.

When pressed. All pins are connected.

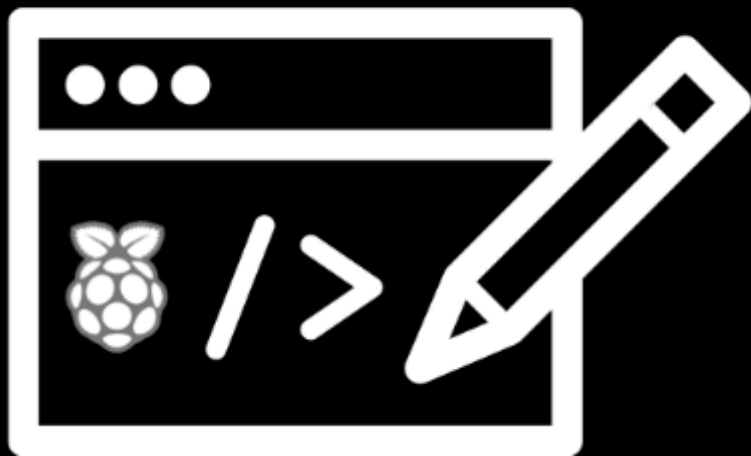
# Breadboards

Short horizontal lines are connected



Long vertical lines are connected. Usually for power(+) and ground (-)

# RASPBERRY PI DEMO APP





# Programming

- We'll try a few basic programs to see how we can interact with lights and switches
- Programs are created using the python programming language



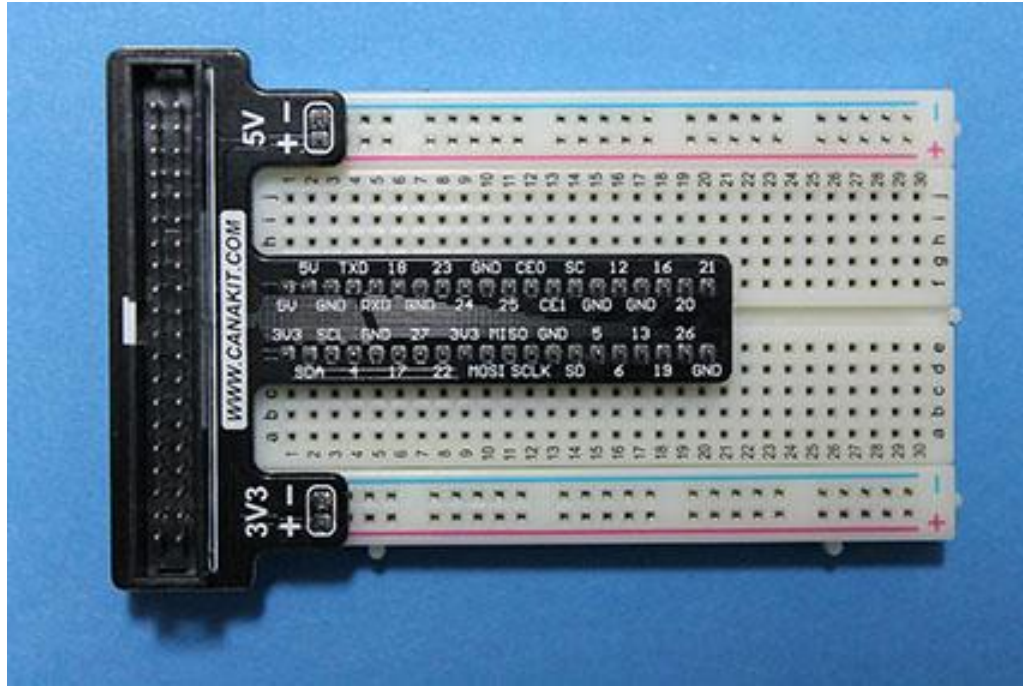
# Caution!



- GPIO pins can supply up to 16 mA of current safely
- Do not exceed more than 50mA across all GPIO pins
- GPIO pins operate on 3.3V, never input more than 3.3V
- Keep this in mind to avoid burning your pi

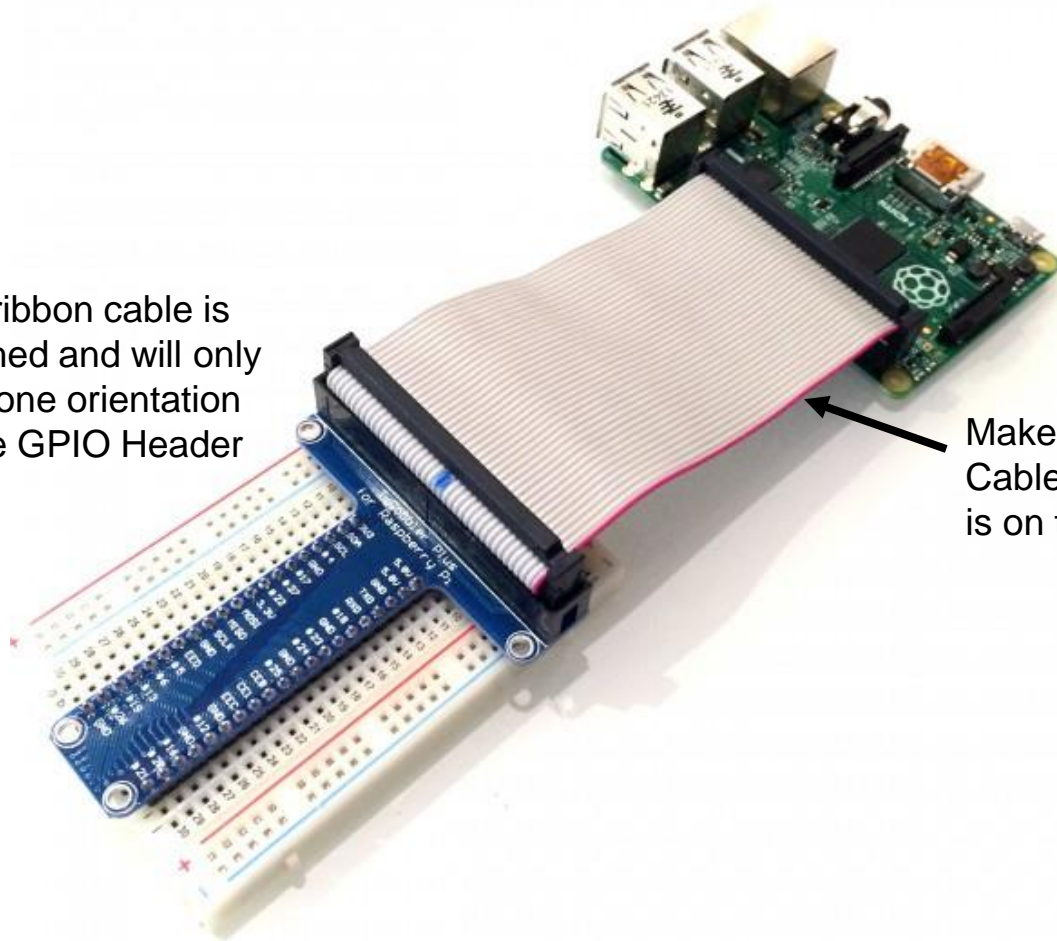


# Connect your GPIO Header Like This



The ribbon cable is notched and will only fit in one orientation in the GPIO Header

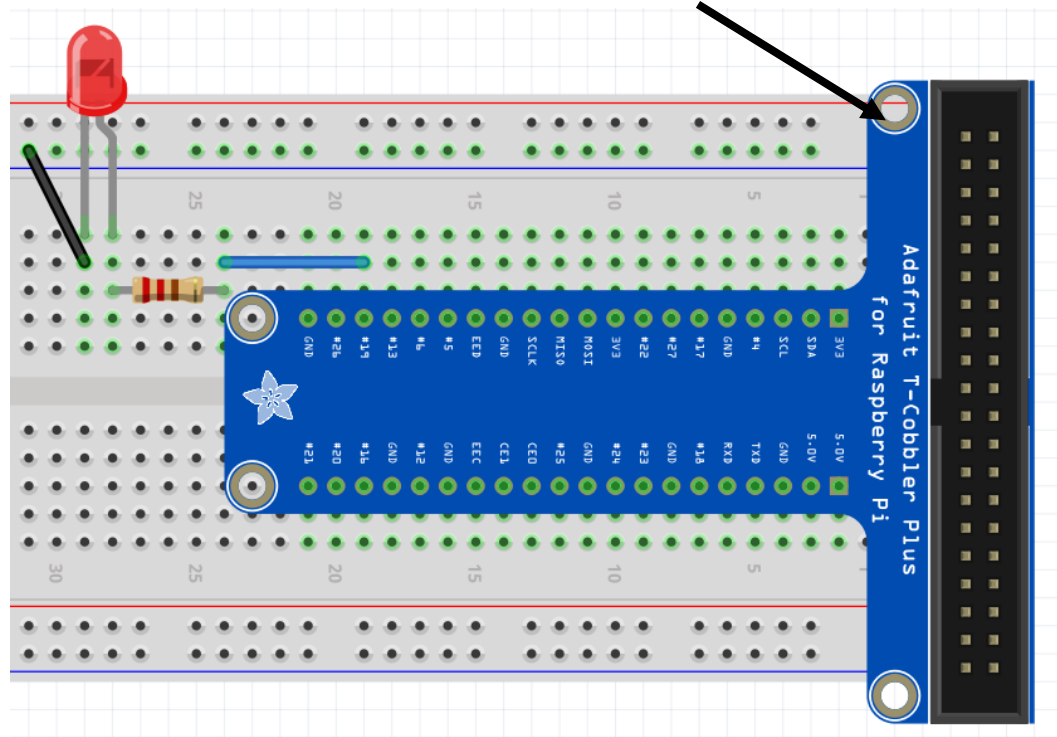
Make sure the Red Cable on the Ribbon is on this side



# Project 1 Blinking LED

- Use a 220 Ohm resistor (Red – Red – Brown)

Your breakout board looks different. The 3.3V will be on this side.



# Project 1 Blinking LED

In a terminal, type

`sudo nano blinker.py`

Type the code →

```
import RPi.GPIO as GPIO
import time
import atexit

ledPin = 19

def cleanup():
    GPIO.cleanup()

atexit.register(cleanup)

GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)

while (True):
    GPIO.output(ledPin, True)
    time.sleep(0.5)
    GPIO.output(ledPin, False)
    time.sleep(0.5)
```

Ctrl-O to save, Ctrl-X to exit

Type '`sudo python blinker.py`' to run . > **Press Ctrl-C to cancel**

# Programming Note: Variables

**Python variables do not need to be explicitly defined as a particular data type. When assigning variables the python interpreter allocates the appropriate memory for each variable (integers, decimals, chars).**

```
<variableName> = <initial value>
```

# Programming Note: Functions

A function is grouping reusable code to perform an action and allow the re-use of existing code. Functions may be input a value(s) upon which an operation is made, and a value may be returned.

```
def functionName(parameters) :  
    code  
    return [expression]      (optional)
```



# Programming Note: Python Modules

A module can be a grouping of functions, variables and classes.

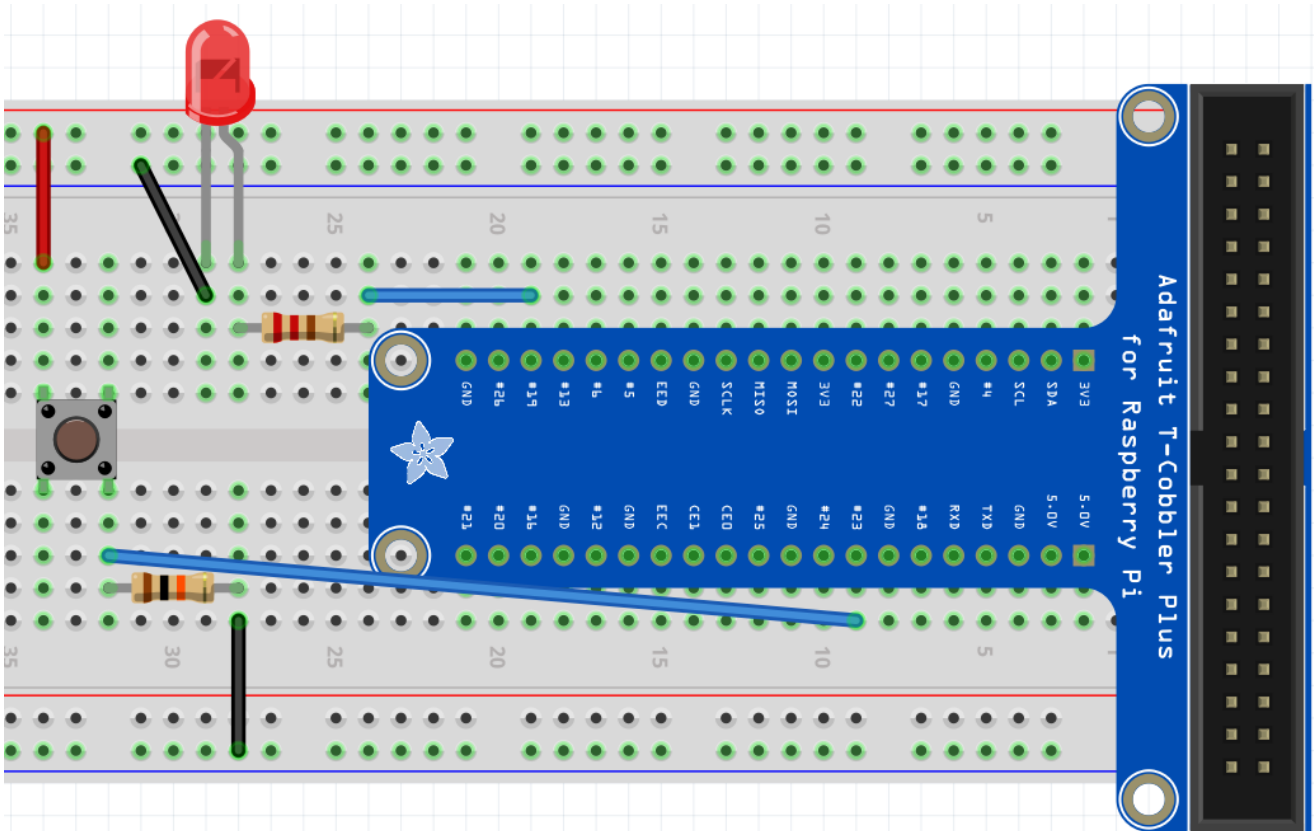
Modules may be imported and loaded into memory at the start of your code:

```
import module
```

Variables and functions within the module can then be accessed using the . (dot) operator.

```
module.function1(paramter)
```

# Project 2 Button



# Project 2 Button

In a terminal, type

*sudo nano button.py*

Type the code →

- Ctrl-O to save, Ctrl-X to exit
- Type '*sudo python button.py*' to run. > Press Ctrl-C to cancel

```
import RPi.GPIO as GPIO
import atexit

ledPin = 19
buttonPin = 23

def cleanup():
    GPIO.cleanup()

atexit.register(cleanup)

GPIO.setmode(GPIO.BCM)

GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(ledPin, GPIO.OUT)

lastState = False

while (True):
    currentState=GPIO.input(buttonPin)
    if lastState != currentState:
        GPIO.output(ledPin,currentState)
        print(currentState)
        lastState = currentState
```

# Project 2 Button

## GPIO.PUD\_DOWN vs GPIO.PUD\_UP

- What's this?  

```
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```
- Raspberry Pi has internal pull up, or pull down resistors configurable withing the programming
- This ensures:
  - the Pi is always reads a 1 or a 0 (floating pin)
  - you are not shorting 3.3v to ground (path of least resistance)
- Configured as either:
  - GPIO.PUD\_DOWN or GPIO.PUD\_UP
- This can be instead done by physical resistors

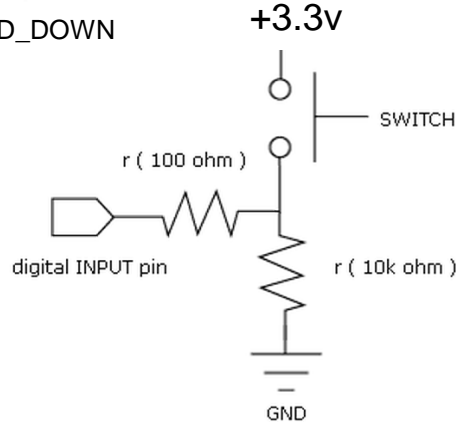
(next slide)

# Project 2 Button

## Pull Up vs Pull Down Resistors

Switch with "pull-down" resistor

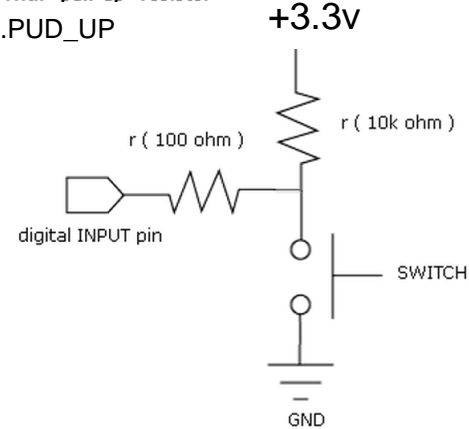
GPIO.PUD\_DOWN



The Pi reads a false (0) initially.  
When the switch is pressed, true (1) is read.

Switch with "pull-up" resistor

GPIO.PUD\_UP



The Pi reads a true (1) initially.  
When the switch is pressed, true (0) is read.

# Programming Note: If Statements

Decision making can occur within `if` and `if-else` statements. When an `if` statement is determined to be true, it's contents are executed. If the statement is determined to be false, the inner code is skipped and an optional `else` statement is executed if the `if` statement was false.

```
if statement:
```

```
    code executed if statement is true
```

```
else:
```

```
    code executed if statement is false
```

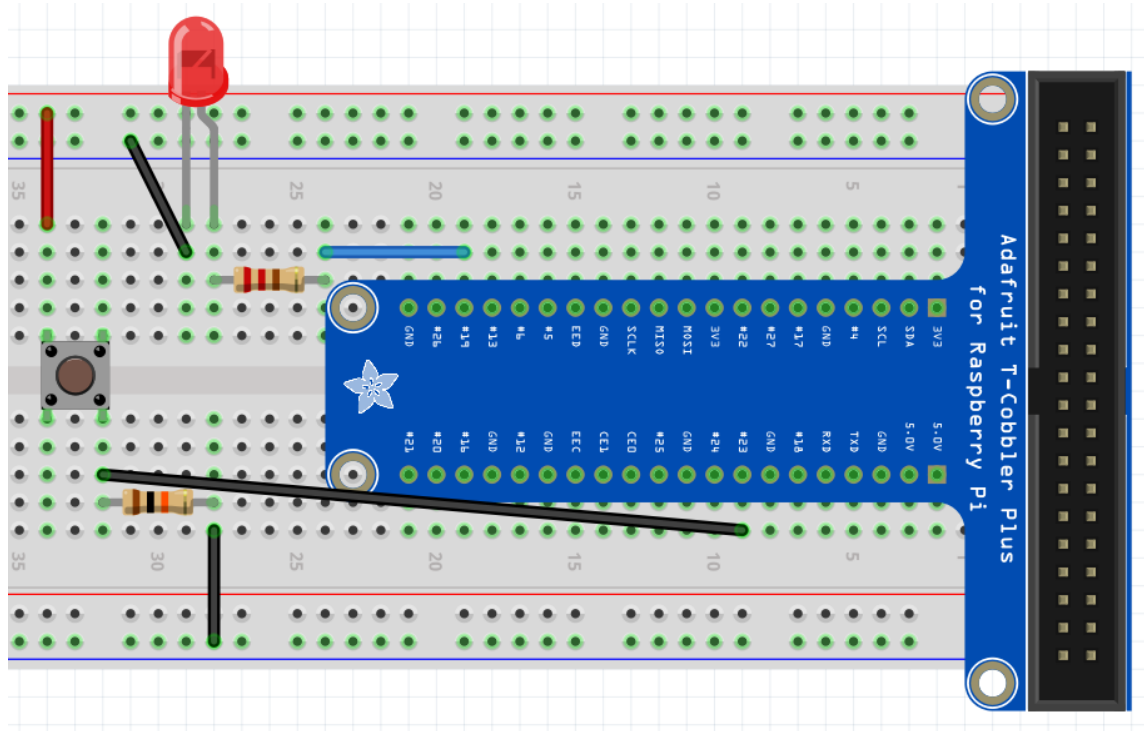
# Programming Note: Comparison Operators

**Comparison operators are used within statements to test conditions. The result is either a true, or a false result.**

Operators	
==	Equal to
!=	Does not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

# Project 2 Button

## Physical Wiring Pull Down (FIY)

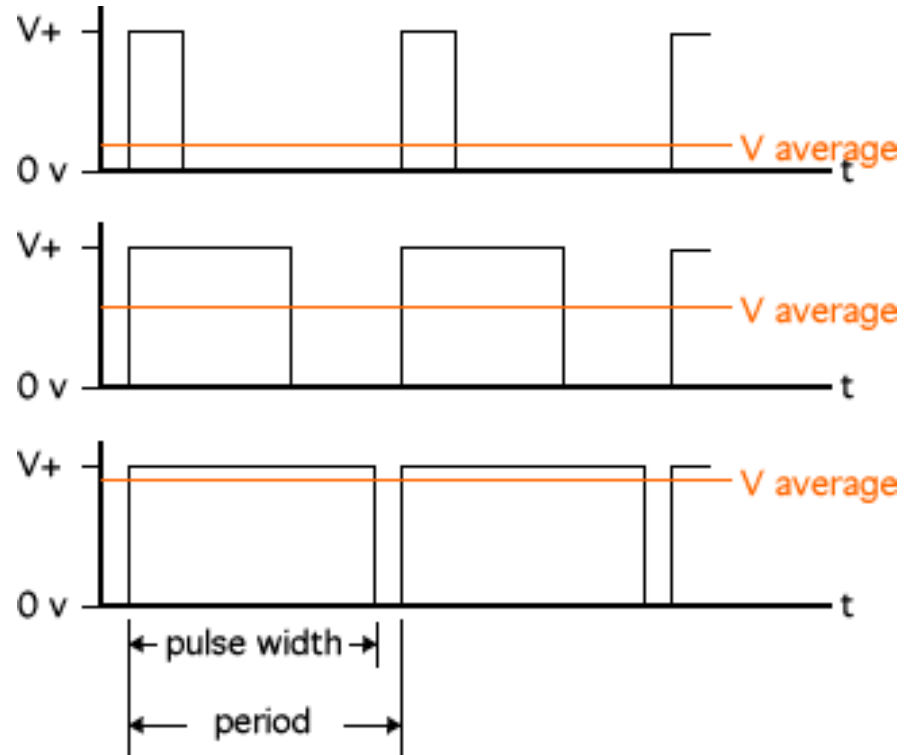




# Project 3 PWM

## Pulse Width Modulation

- We can use our last circuit
- Pulse width modulation turns the pin on and off very fast:
  - The period defines the frequency of the pulse
  - The pulse width is the percent of time the signal is on (also called duty cycle).
  - This can make an LED appear dimmer, or motor spin slower



# Project 3 PWM

In a terminal, type

*sudo nano pwm.py*

Type the code →

```
import RPi.GPIO as GPIO
import atexit

ledPin = 19

def cleanup():
    GPIO.cleanup()

atexit.register(cleanup)

GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)

pwmLed = GPIO.PWM(ledPin, 500)
pwmLed.start(100)

while True:
    input = raw_input("Enter brightness (0-100):")
    duty = int(input)
    if (duty>=0 and duty<=100):
        pwmLed.ChangeDutyCycle(duty)
    else:
        print "Try Again"
```

- Ctrl-O to save, Ctrl-X to exit
- Type '*sudo python pwm.py*' to run. > Press Ctrl-C to cancel

# Project 4 GUI Interface

- We will be using the Tkinter (“Tk interface”) module to create a GUI interface from within Rasbian
- The interface will change the PWM of the LED, toggle the LED state from on to off, and notify us if the button has been pressed.

# Project 4 GUI Interface

- For this project, lets use the Python IDLE (Integrated Development Environment)
- Accessed from Menu > Programming > Python 2
- Create File, File > New
- Save File
- To Run, Run > Run Module

## Project 4 GUI Interface (Code 1 of 4)

```
from Tkinter import *
import RPi.GPIO as GPIO
import atexit
import tkMessageBox

ledPin = 19
buttonPin = 23
storedPwm = 0
```

## Project 4 GUI Interface (Code 2 of 4)

```
def cleanup():
    GPIO.cleanup()
    win.destroy()

def update(duty):
    pwm_led.ChangeDutyCycle(int(duty))

def toggle():
    if scale.get() > 0:
        global storedPwm
        storedPwm= scale.get()
        scale.set(0)
        pwm_led.ChangeDutyCycle(0)
    else:
        pwm_led.ChangeDutyCycle(int(storedPwm))
        scale.set(storedPwm)
```

## Project 4 GUI Interface (Code 3 of 4)

```
atexit.register(cleanup)

GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(buttonPin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

pwm_led = GPIO.PWM(ledPin, 500)
pwm_led.start(100)

win = Tk()
```

## Project 4 GUI Interface (Code 4 of 4)

```
win.title("LED PWM Switch")
win.geometry("500x100")
win.protocol("WM_DELETE_WINDOW", cleanup)

scale = Scale(win, from_=0, to= 100, orient=HORIZONTAL, length = 400,
              tickinterval=10, command=update)
scale.pack()
scale.set(100)

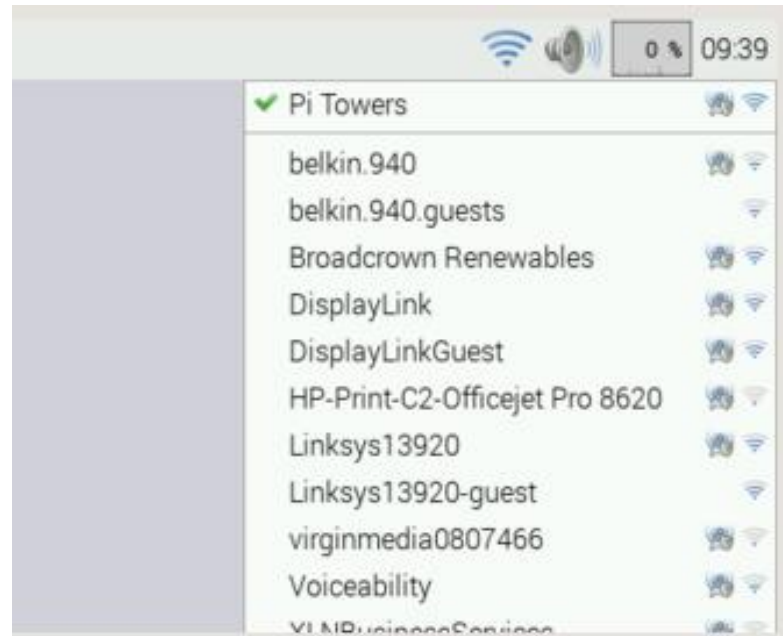
toggleButton = Button(win, text="Toggle", command=toggle)
toggleButton.pack()

while 1:
    task()
```



# Setting Up Wifi

- Select Wifi dropdown from top bar




# Find IP Address

- Run '*ifconfig*'

```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:cd:b7:0e
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:0f:55:bb:c8:4a
          inet addr:192.168.1.90  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:46653 errors:0 dropped:12928 overruns:0 frame:0
          TX packets:15552 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15116654 (14.4 MiB)  TX bytes:2173262 (2.0 MiB)
```



# Remote Login – SSH, Port 22

Windows - Install and Run 'Putty'

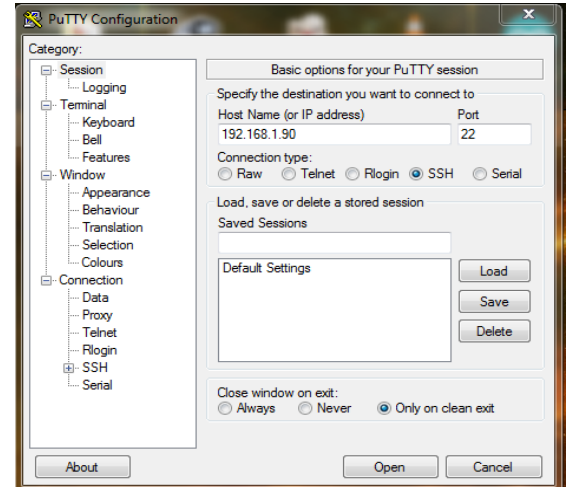
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Mac, Type:

```
ssh <ip address of your Rpi> -l <username of RPi user you set up previously>
```

Once logged in, try running the `blinker`, `button`, or `pwm` programs remotely.

username: pi, password: raspberry



# FORMATTING SD CARD

In the event you want to start fresh, following these steps.

1. Download & Install SD Formatter from

[https://www.sdcard.org/downloads/formatter\\_4/](https://www.sdcard.org/downloads/formatter_4/)

2. Format SD card with SD Formatter

3. Download NOOBS or your choice of distribution from

<https://www.raspberrypi.org/downloads>

4. Extract contents of distribution zip file to SD card

NOOBS v1.9 - Built: Mar 18 2016



Install (i)



Edit config (e)



Wifi networks (w)



Online help (h)



Exit (Esc)



Raspbian [RECOMMENDED]

A community-created port of Debian jessie for the Raspberry Pi



Disk space

Needed: 3617 MB

Available: 13764 MB

Language (l):



English (US)



Keyboard (9):

us





## Confirm



Warning: this will install the selected Operating System(s). All existing data on the SD card will be overwritten, including any OSes that are already installed.



Yes

No



# Welcome to your Raspberry Pi



We're currently setting up your SD card but don't worry, you'll be able to start programming very soon.

Raspbian: Extracting filesystem

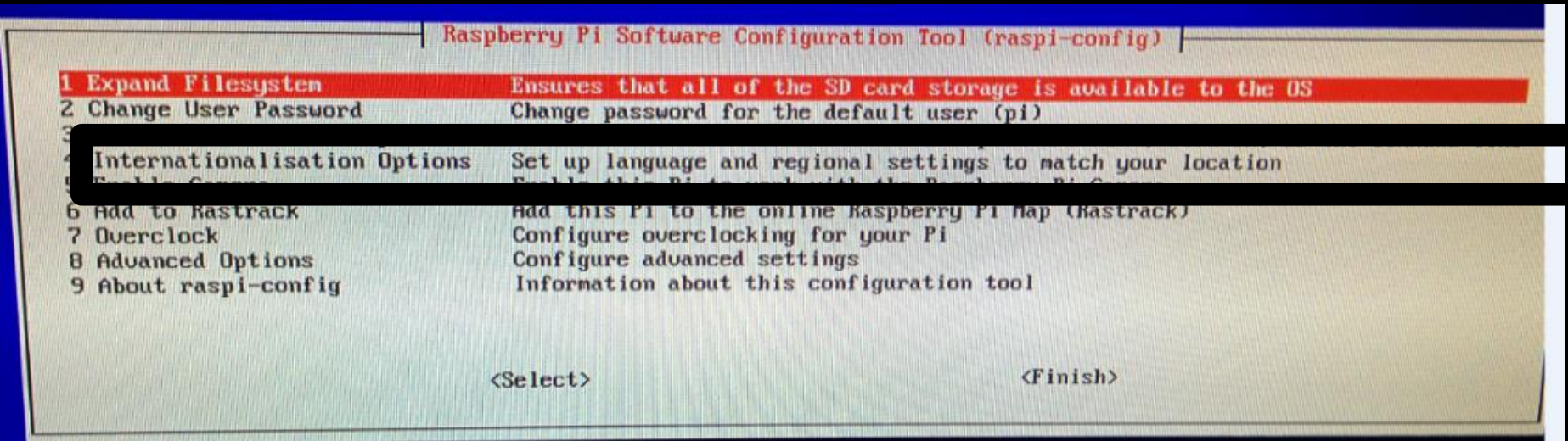


0%

0 MB of 2349 MB written (0.0 MB/sec)

If you forgot to select English language & US Keyboard. Follow these steps. Otherwise proceed to press Finish.

This can be accessed by typing *raspi-config* at the terminal





## Raspberry Pi Software Configuration Tool (raspi-config)

I1 Change Locale Set up language and regional settings to match y

I2 Change Timezone Set up timezone to match your location

I3 Change Keyboard Layout Set the keyboard layout to match your keyboard



Generic 104-key PC



Keyboard layout:

English (US)

## Raspberry Pi Software Configuration Tool (raspi-config)

**I1 Change Locale** Set up language and regional settings to match y

I2 Change Timezone

I3 Change Keyboard Layout Set the keyboard layout to match your keyboard



en\_US.UTF-8 UTF-8  
[ ] en\_US ISO-8859-1  
[ ] en\_US

# RASPBERRY PI QUESTIONS?



# **RASPBERRY PI**

## **THE END**

**Thanks for Joining Us!**

