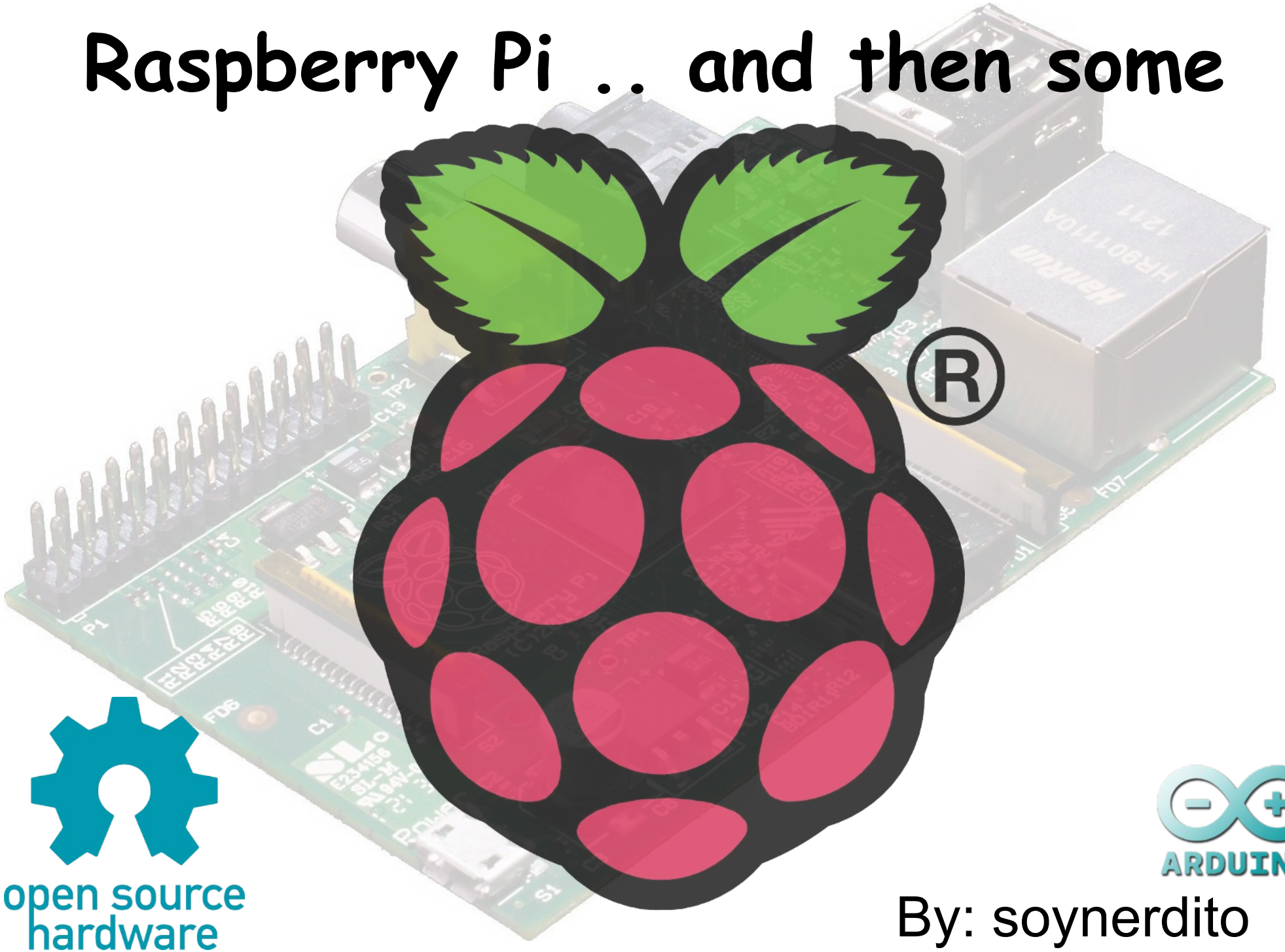
Raspberry Pi .. and then some



open source hardware

ARDUINO

By: soynerdito

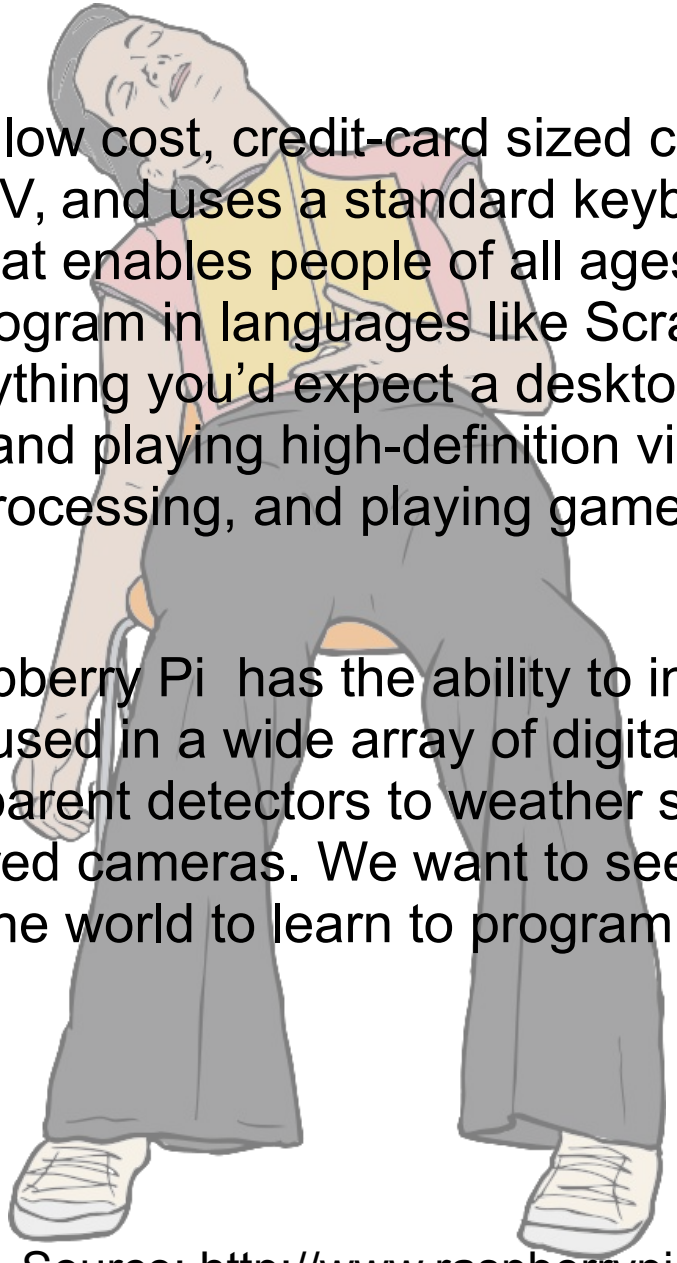
Glossary

- **GPIO**: General purpose input/output. A pin that can be programmed to do stuff.
- **GPU**: graphics processing unit. The hardware the handles the graphics.
- **Distro**: a specific package (“flavour”) of Linux and associated software.

- **BGA**: ball grid array. A type of surface mount packaging for electronics.
- **SoC**: system on chip. A computer on a single chip.
- **Brick**: to accidentally render a device inert by making changes to software or firmware.
- **Pxe**: preboot execution environment. A way to get a device to boot by via the network.
- **PoE**: power over ethernet. Powering a device via an ethernet cable.

Source: http://www.raspberrypi.org/help/faqs/

What is the Raspberry Pi?



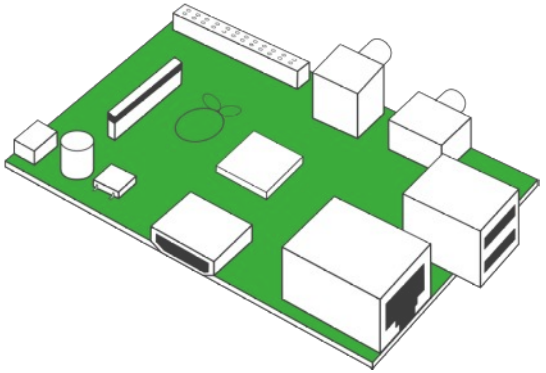
- The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

- What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

Source: http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

What is the Raspberry Pi?

- The Raspberry Pi is a low cost, credit-card sized computer
- Capable of everything you'd expect a desktop computer to do.
- Plugs into a computer monitor or TV
- Uses a standard keyboard and mouse



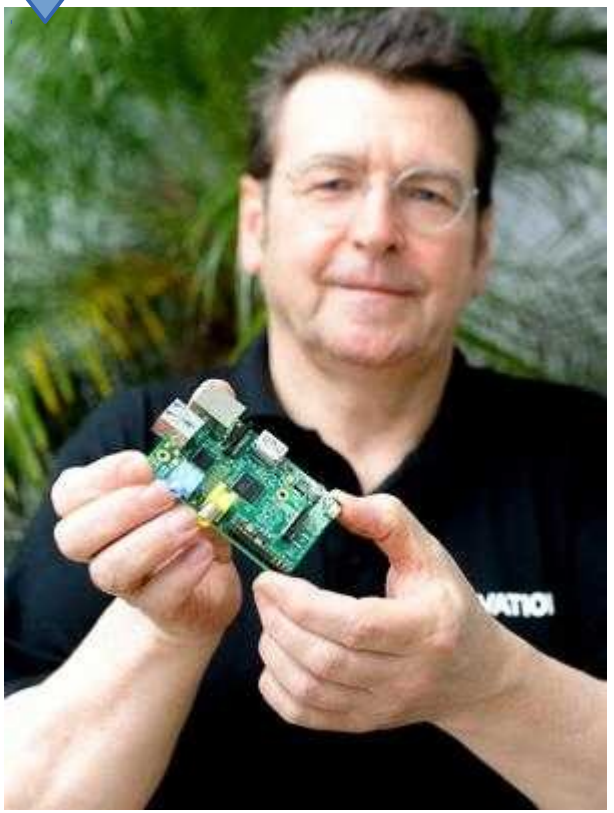
- Raspberry Pi has the ability to interact with the outside world
- "We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work."

Source: http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

Raspberry Pi Foundation (2009) co-founders among others

Eben Upton

Pete Lomas

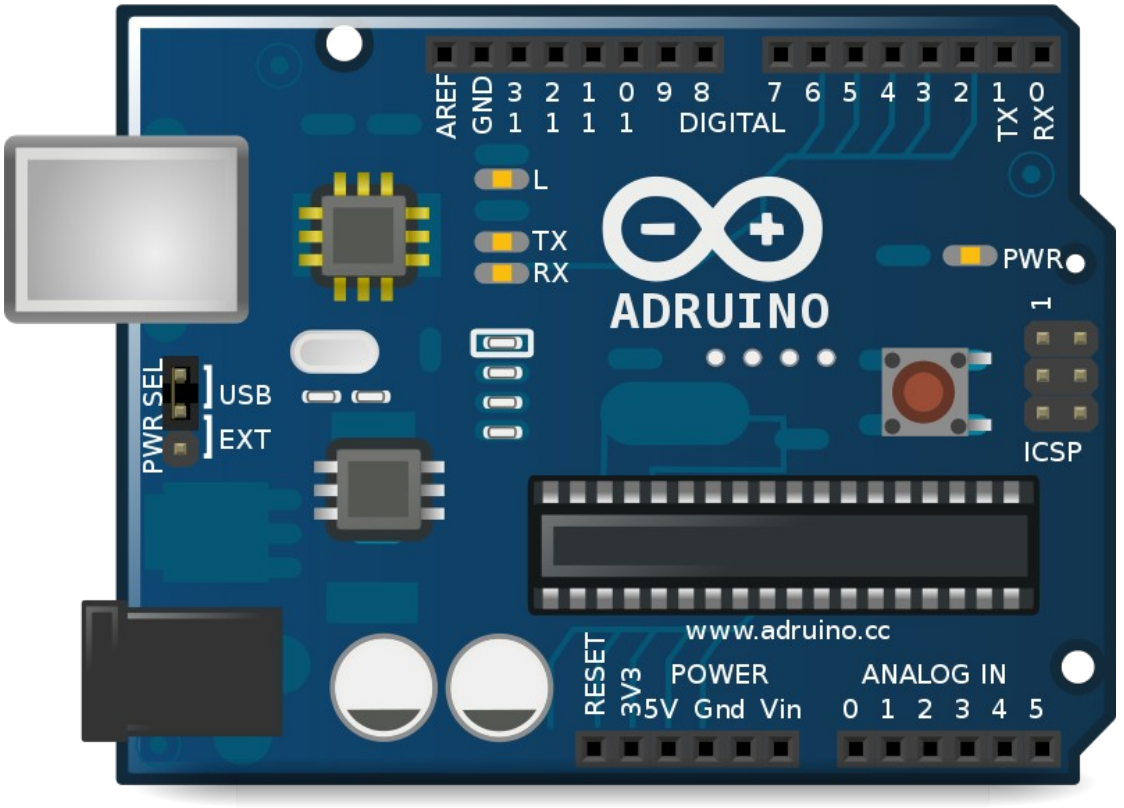


Raspberry PI = Arduino ?

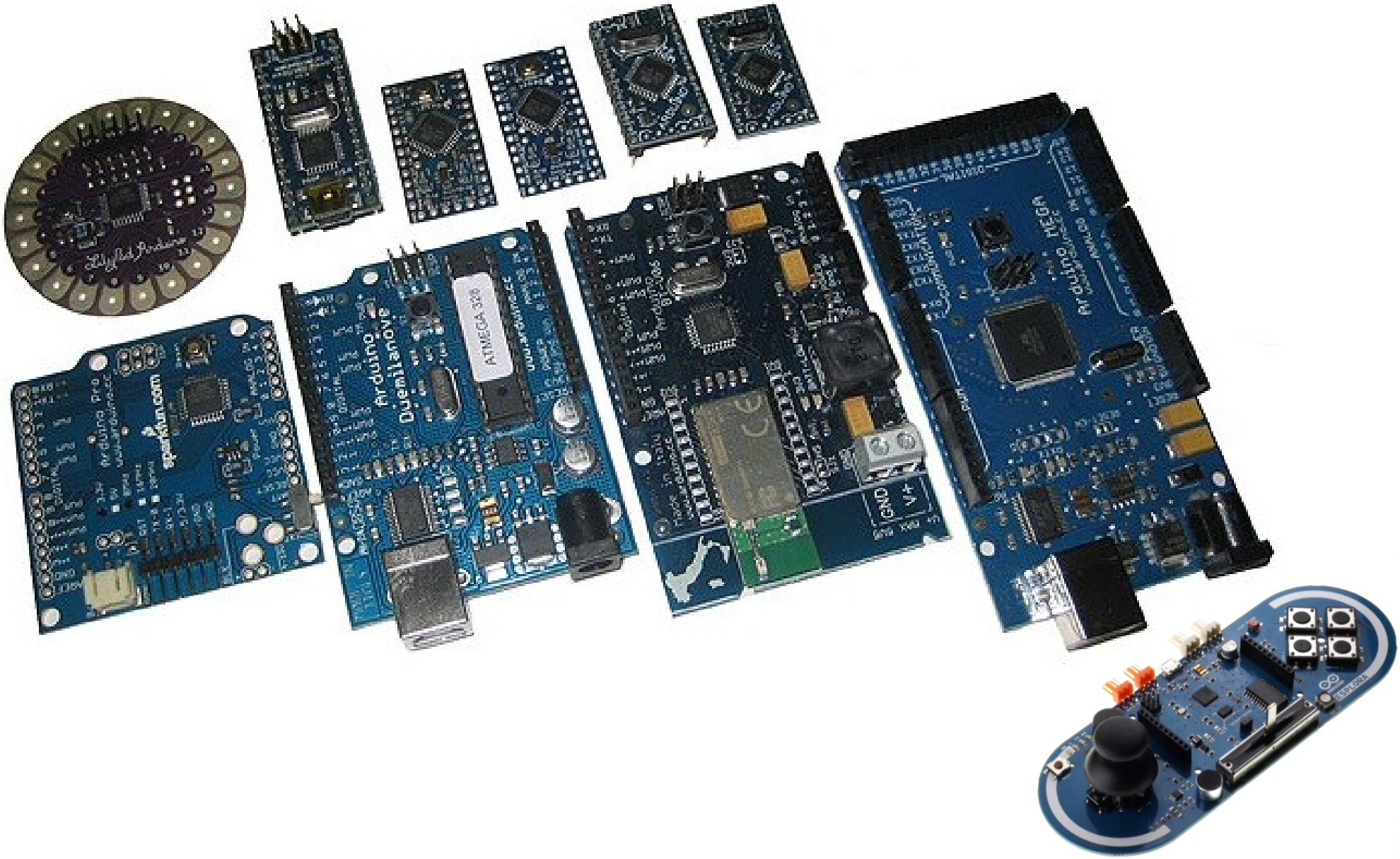
Arduino

Arduino is a single-board microcontroller (wikipedia)

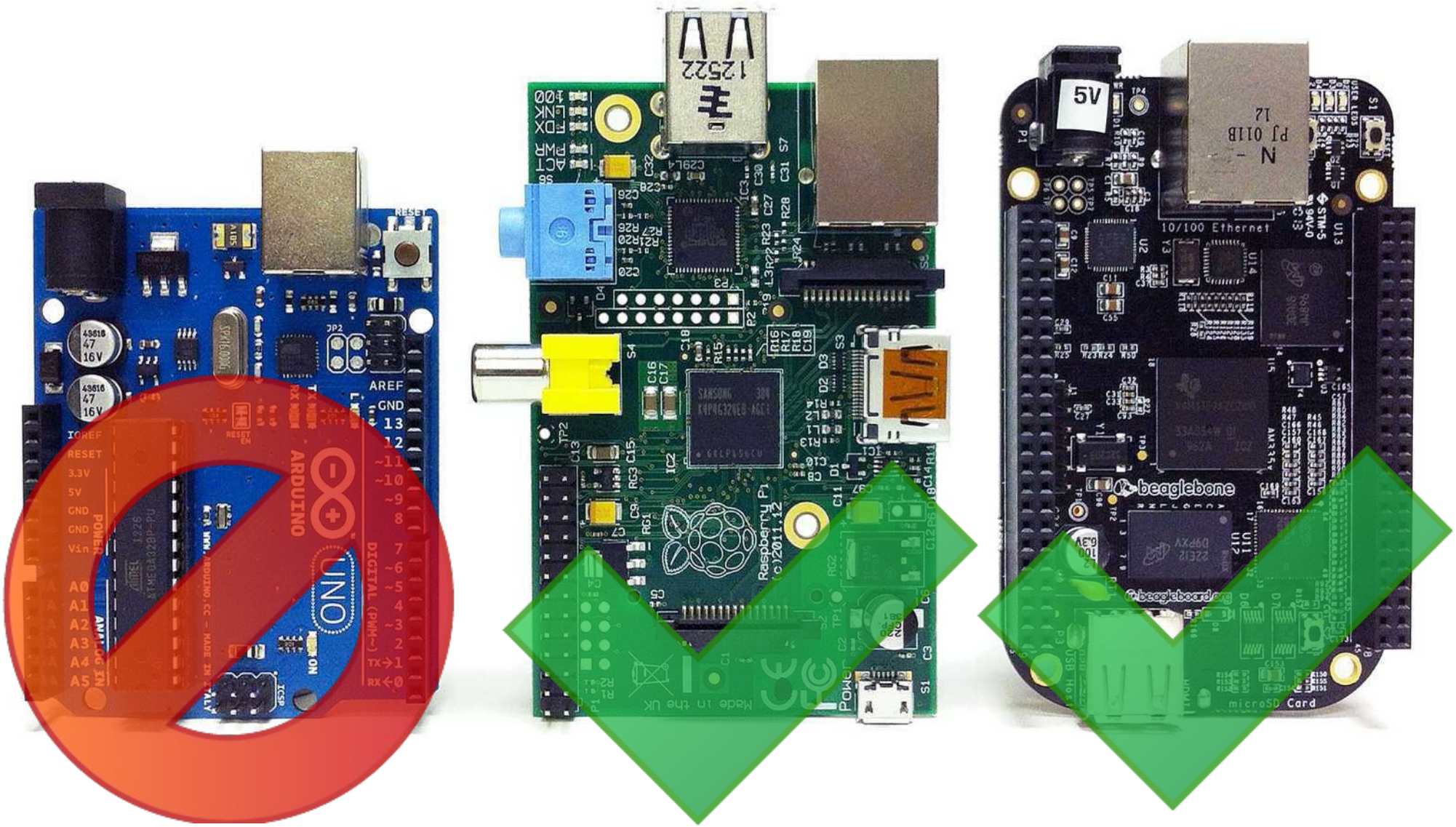
- Arduino UNO
- Microcontroller
- Arduino is more than a board



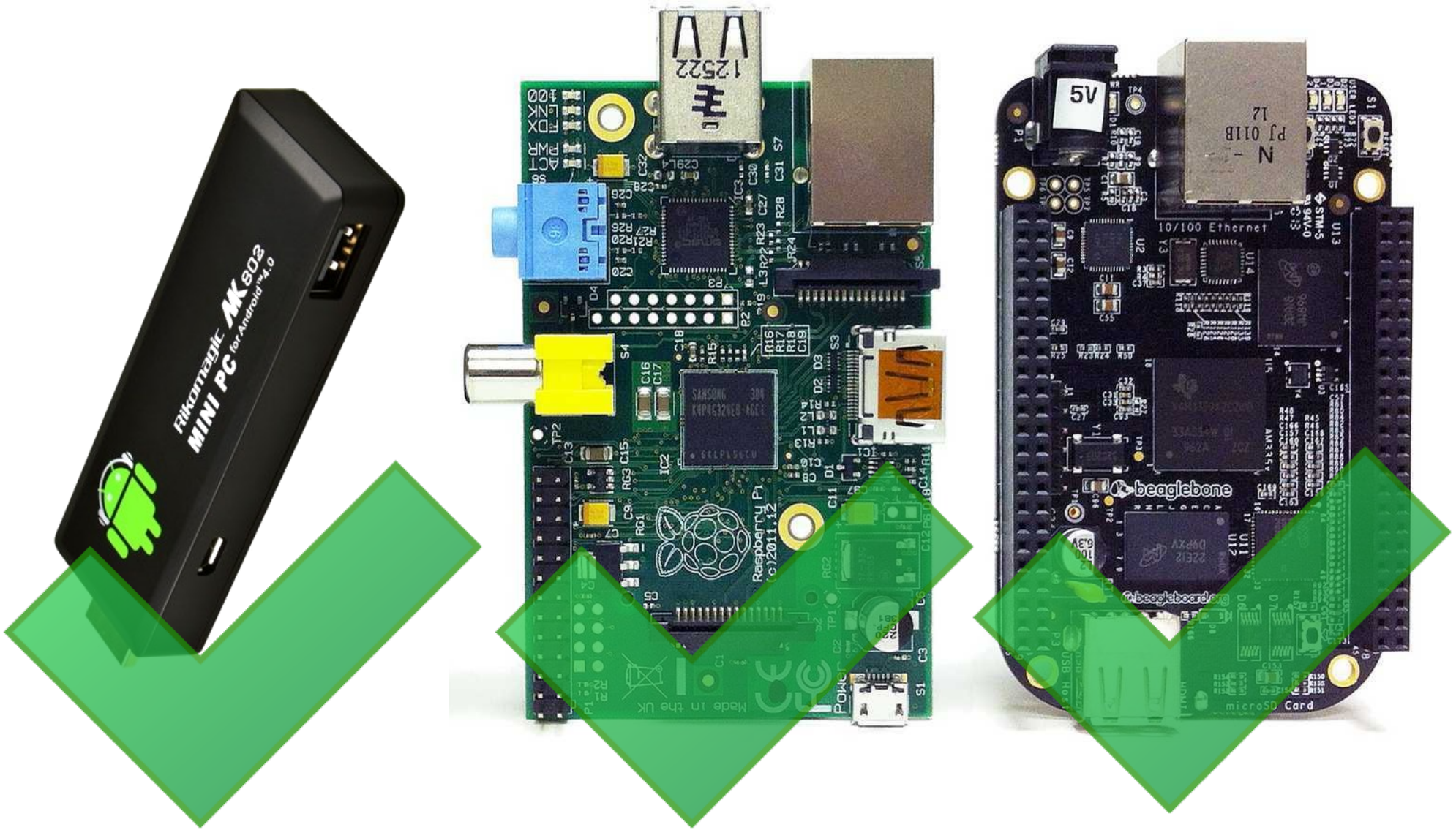
Arduino Family



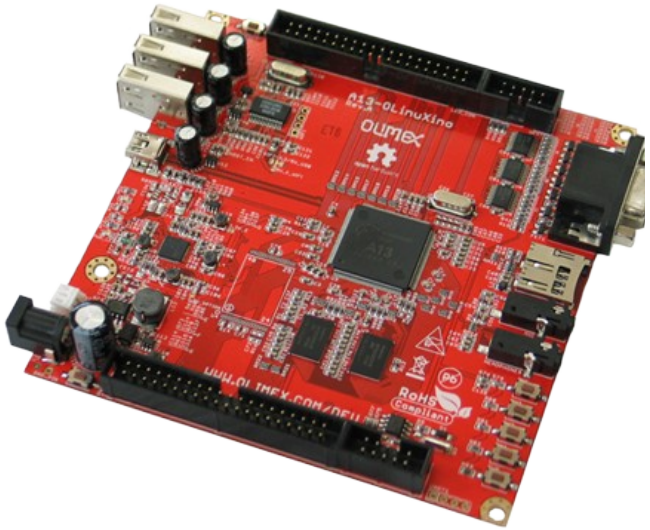
Challengers

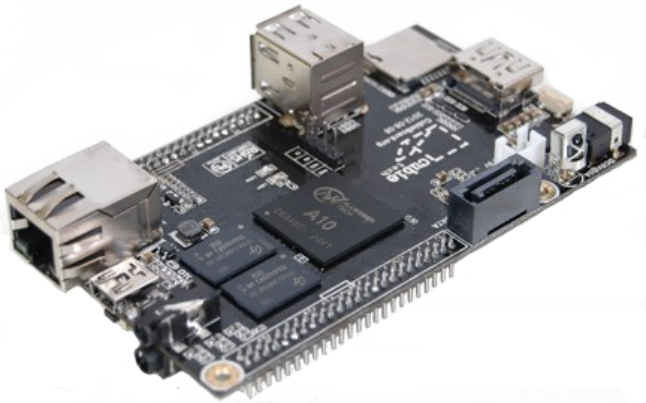


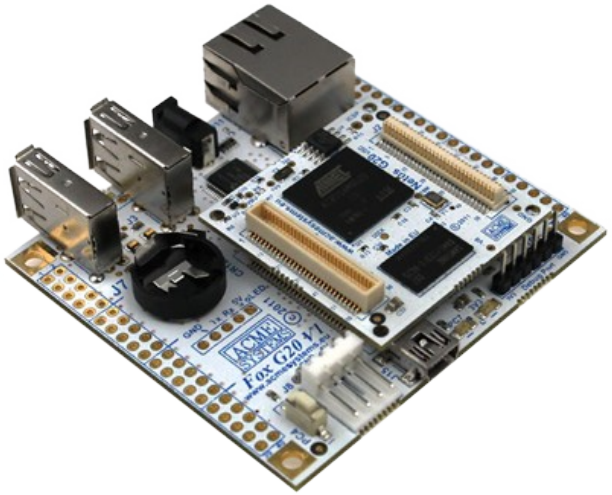
Challengers
Single Board Computers

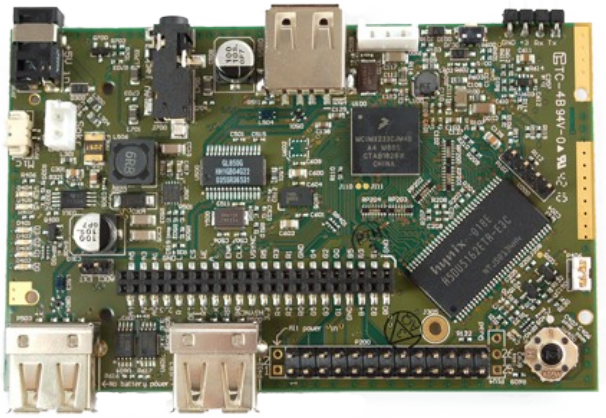


And the list goes on

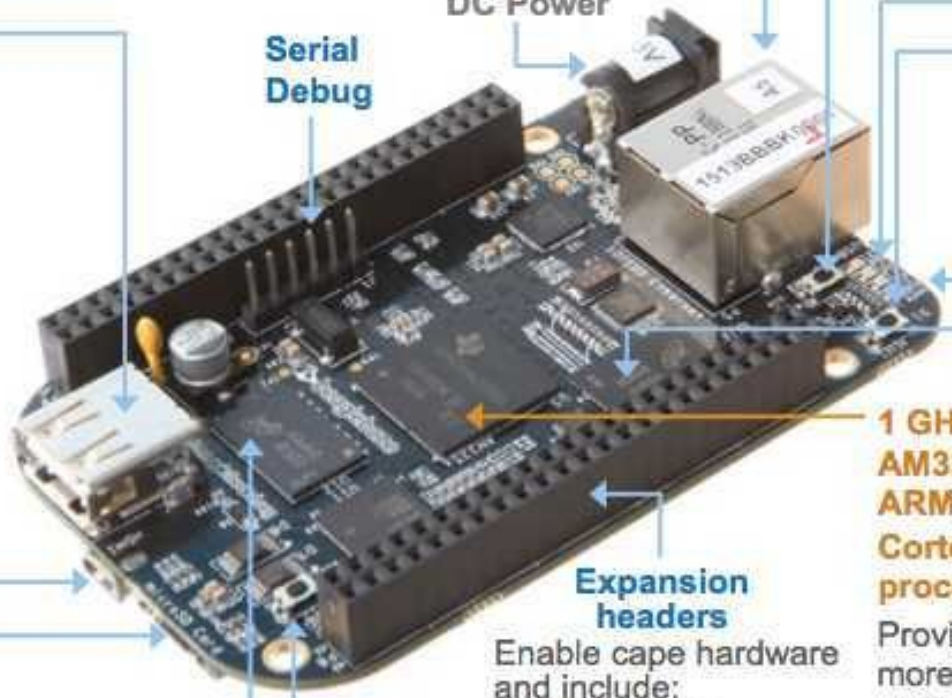








BeagleBone Black
1 GHz performance ready to use for $45



10/100 Ethernet

USB Host
Easily connects to almost any everyday device such as mouse or keyboard

microHDMI
Connect directly to monitors and TVs

microSD
Expansion slot for additional storage

512MB DDR3
Faster, lower power RAM for enhanced user-friendly experience

Serial Debug

DC Power

Boot Button

Power Button

LEDS

Reset Button

USB Client
Development interface and directly powers board from PC

2GB on-board storage using eMMC
- Pre-loaded with Ångström Linux Distribution
- 8-bit bus accelerates performance
- Frees the microSD slot to be used for additional storage for a less expensive solution than SD cards

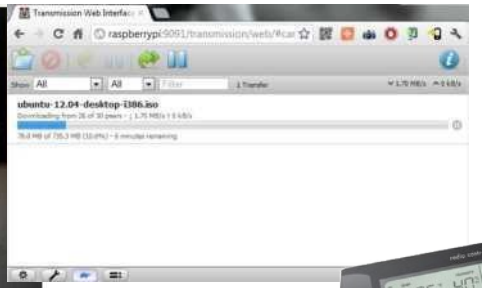
1 GHz Sitara AM335x ARM® Cortex™-A8 processor
Provides a more advanced user interface and up to 150% better performance than ARM11

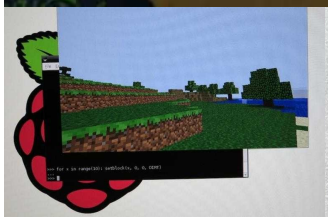
Expansion headers
Enable cape hardware and include:
- 65 digital I/O
- 7 analog
- 4 serial
- 2 SPI
- 2 I2C
- 8 PWMs
- 4 timers
- And much much more!

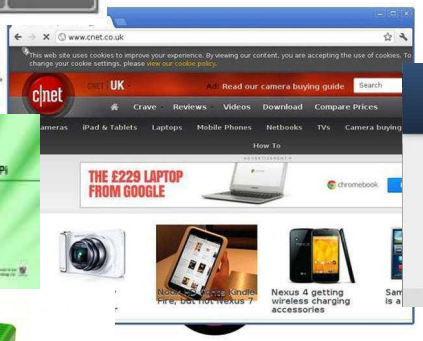
Included in price:
- Power supply ~ $10
- USB network cable ~ $3
- 2GB on-board storage $5-$10
- PRU for real-time tasks typically on FPGA ~ $20

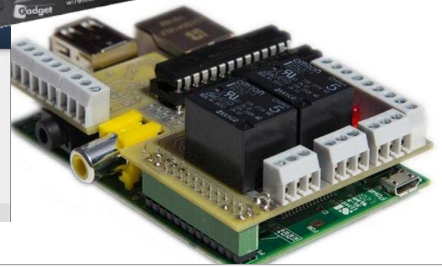
Common projects

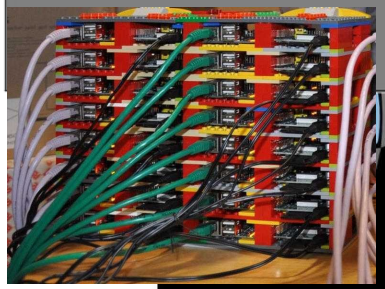
http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

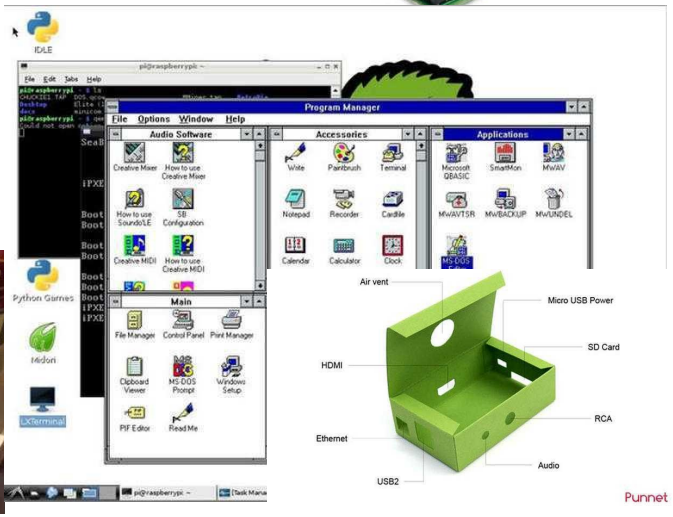












Punnet

Robotics

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Retro Pi

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

10-inch RPi Touchscreen

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Arcade Pi

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Beer can keyboard

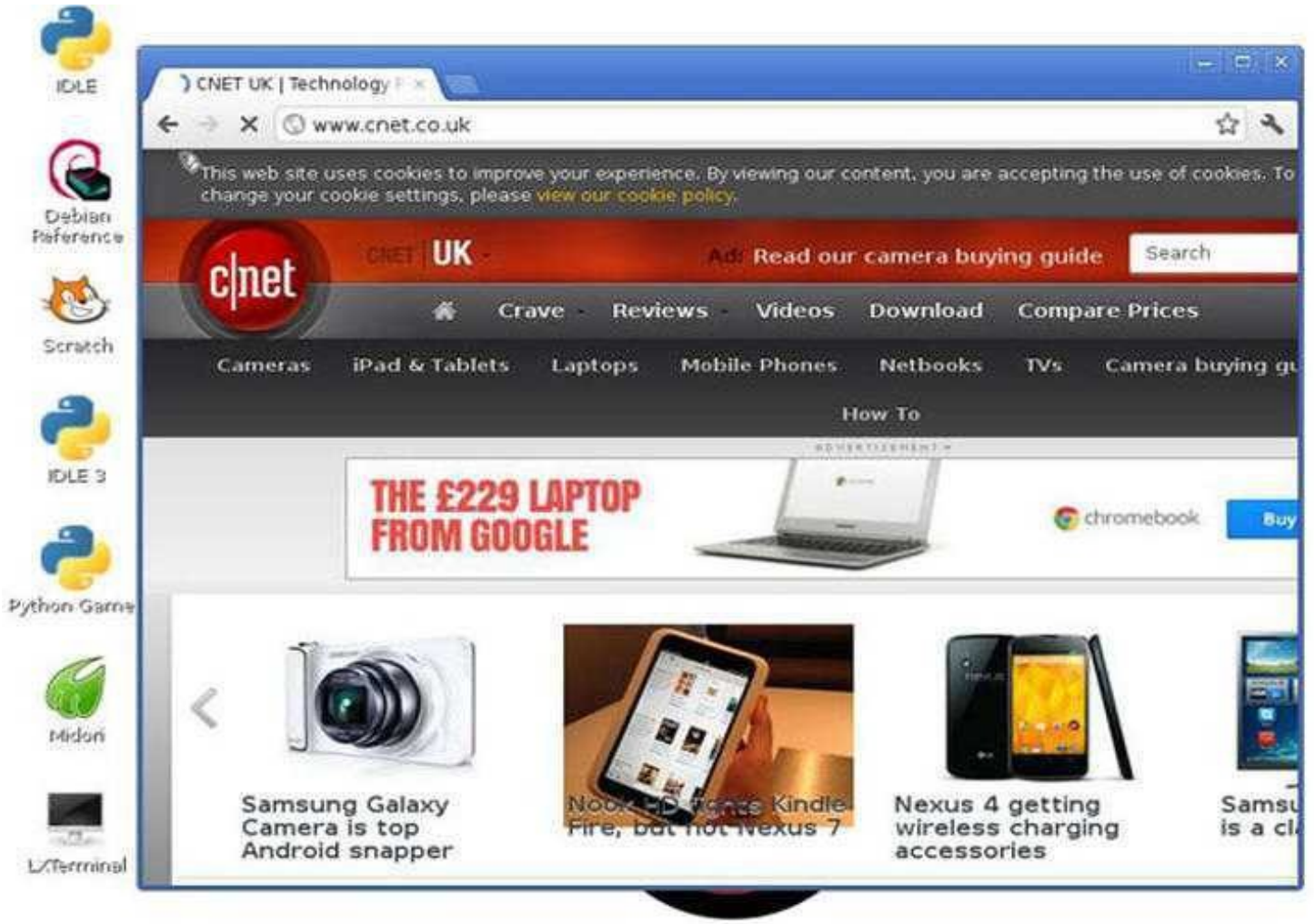
A keyboard made from beer cans? True enough, the Robofun team hooked up an Arduino board to a Raspberry Pi along with many cans of beer.

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

More Robotics

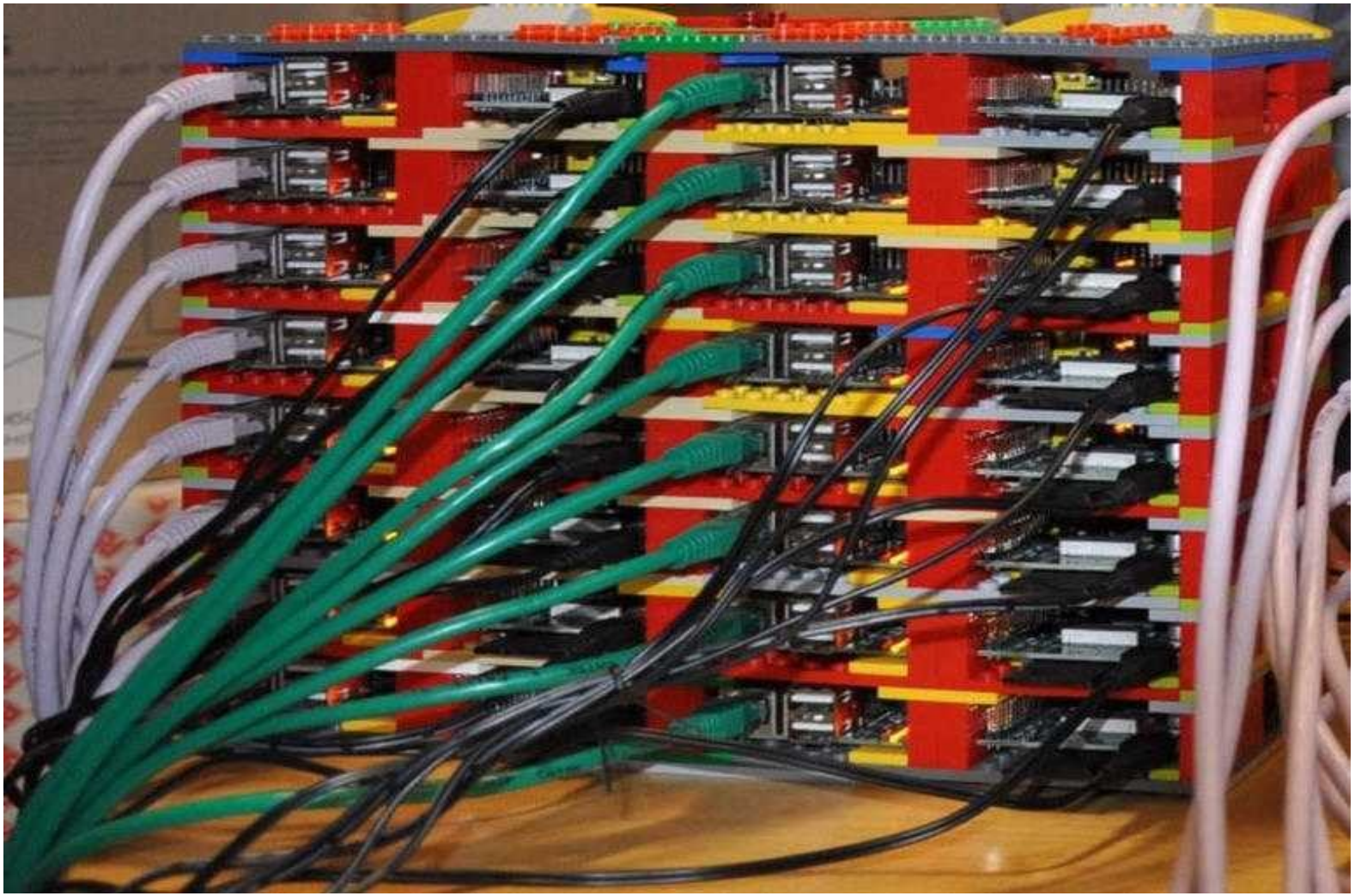
http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

A mini web browser



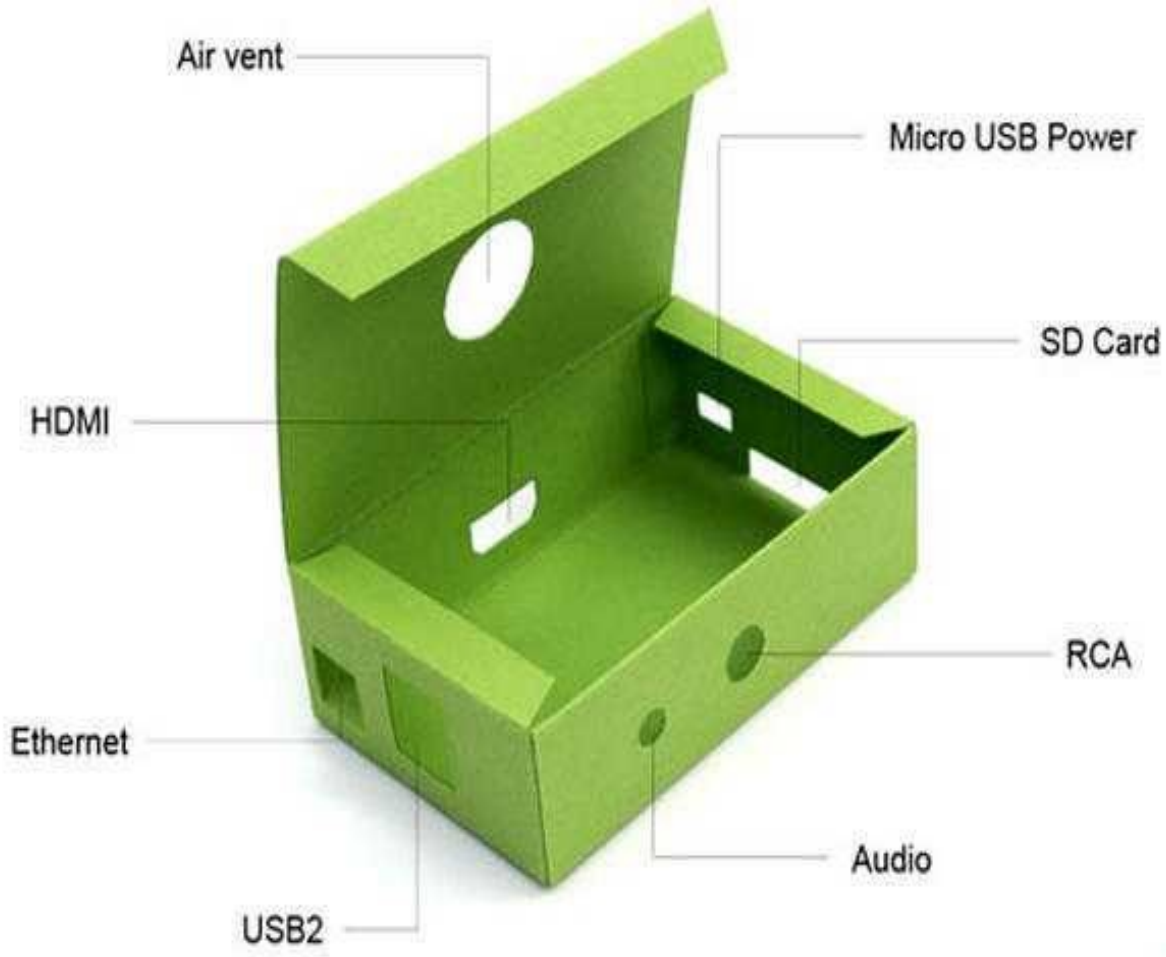
http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

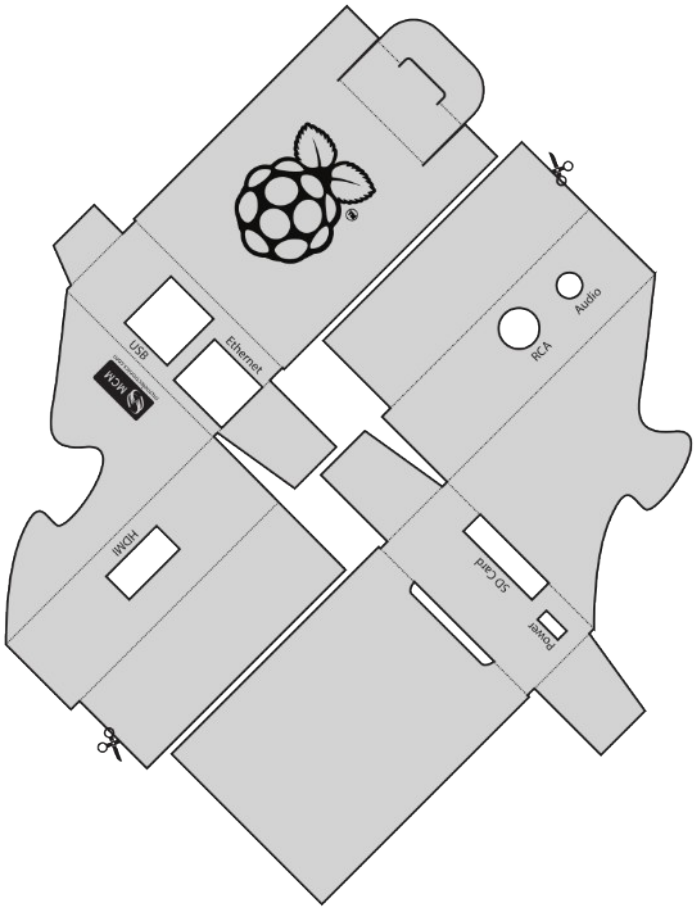
Raspberry Pi Cluster



http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Make your own Pi case





Punnet

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

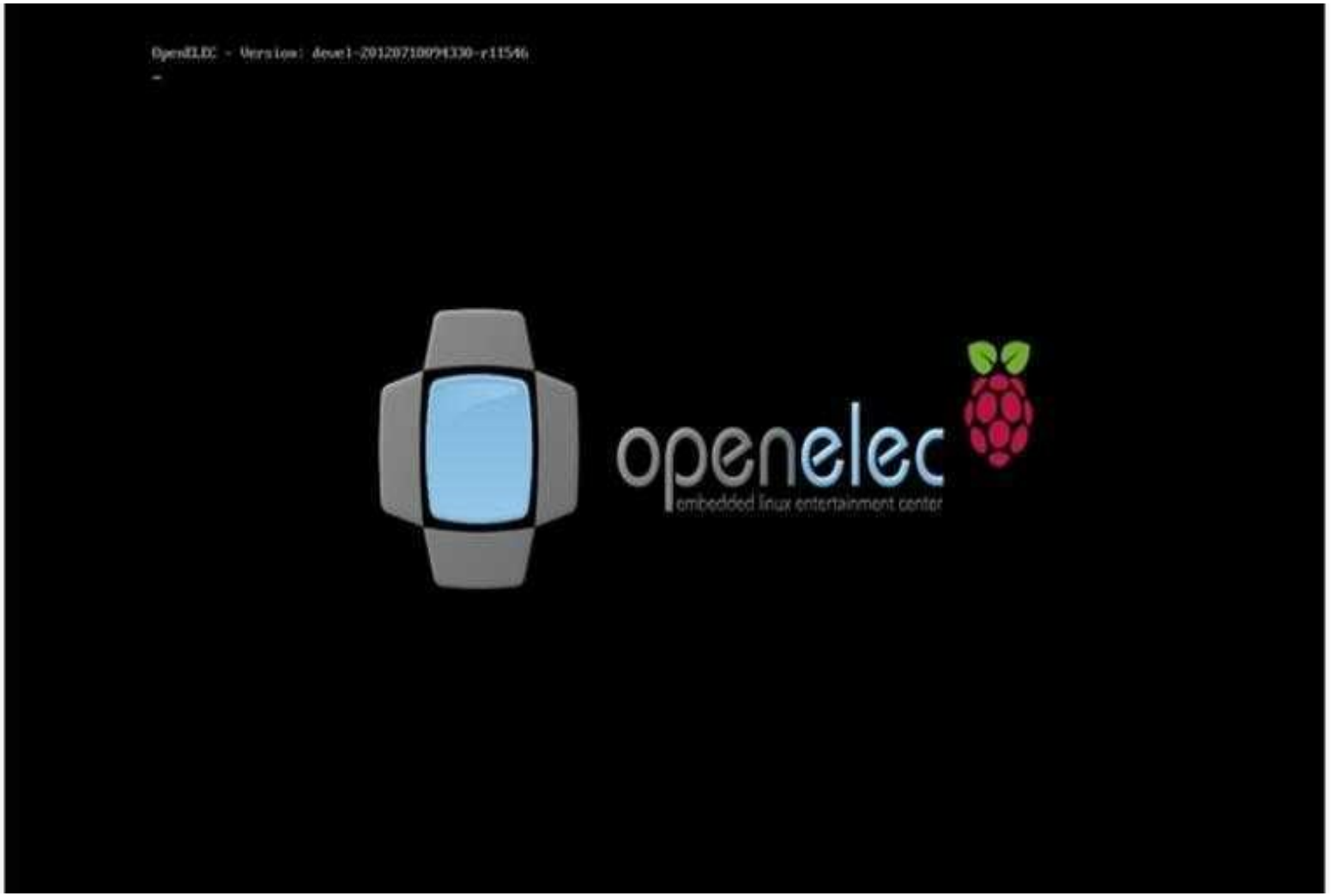
Minecraft

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Mod My Pi

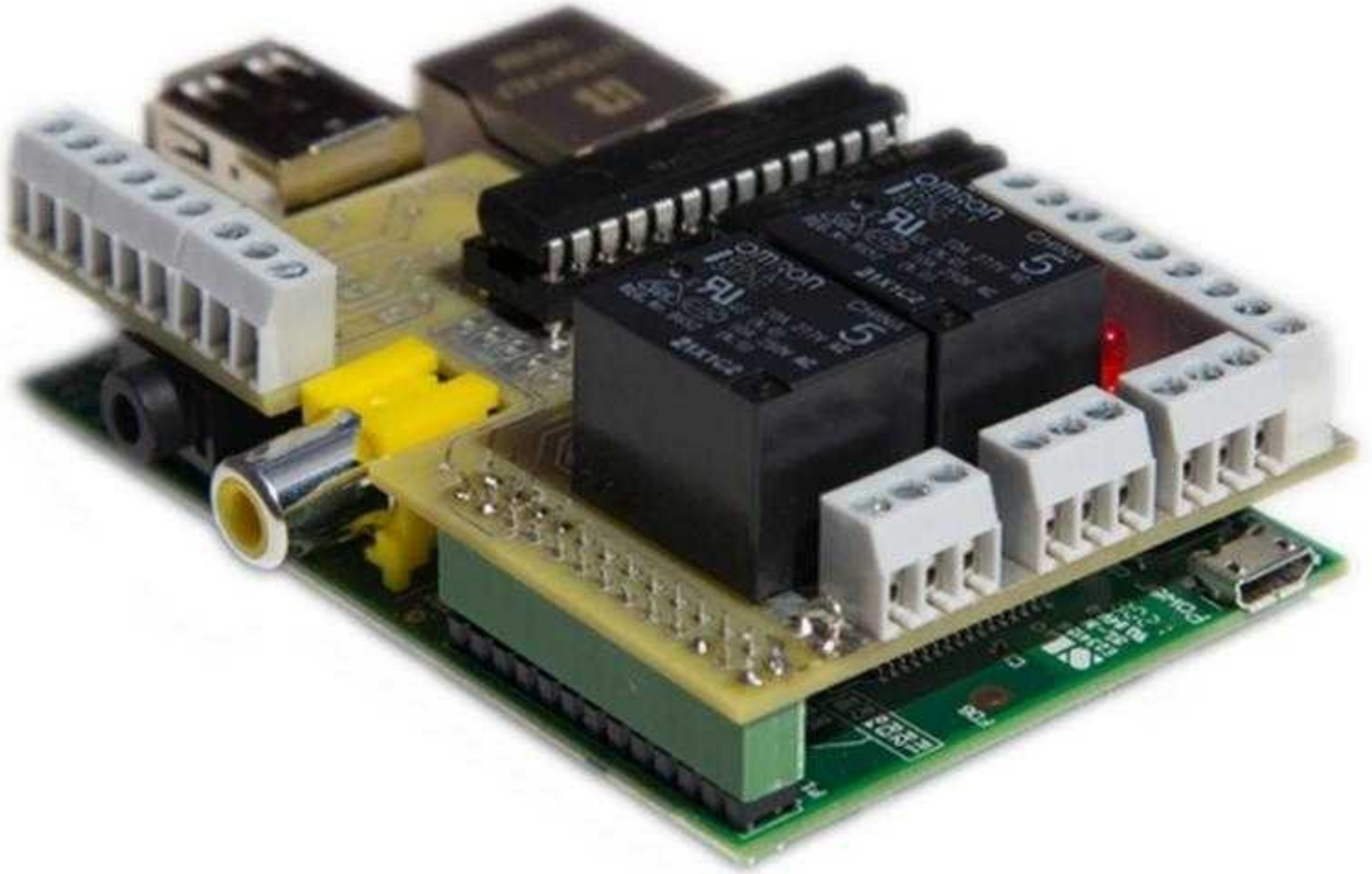
http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Living room PC



http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Home automation



http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Pi Hacker

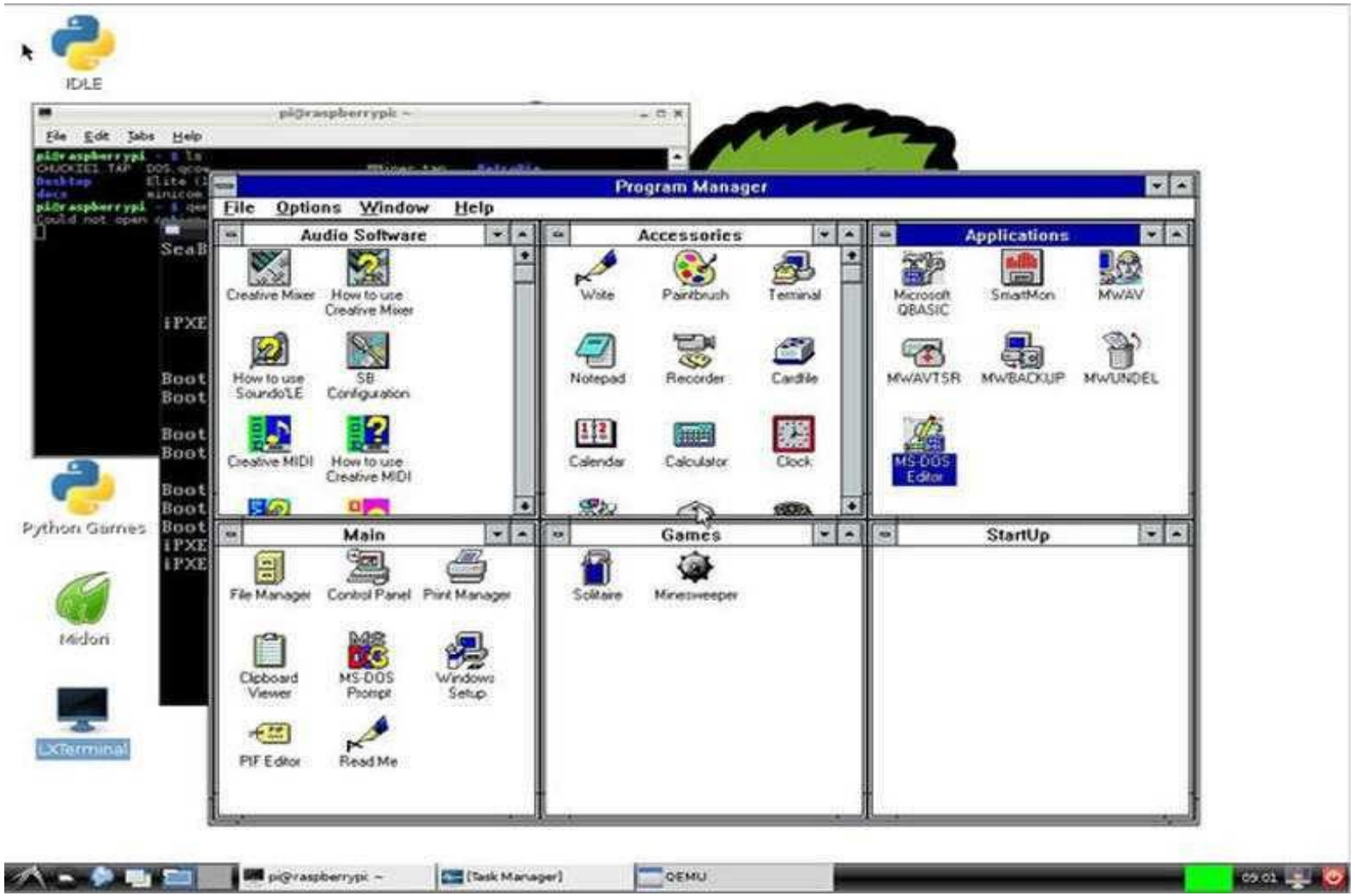
Being as small as it is, the RPi would make an excellent hacking tool. Regardless of the ethics involved, try out this security penetration testing project.

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

RPi cloud server

http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

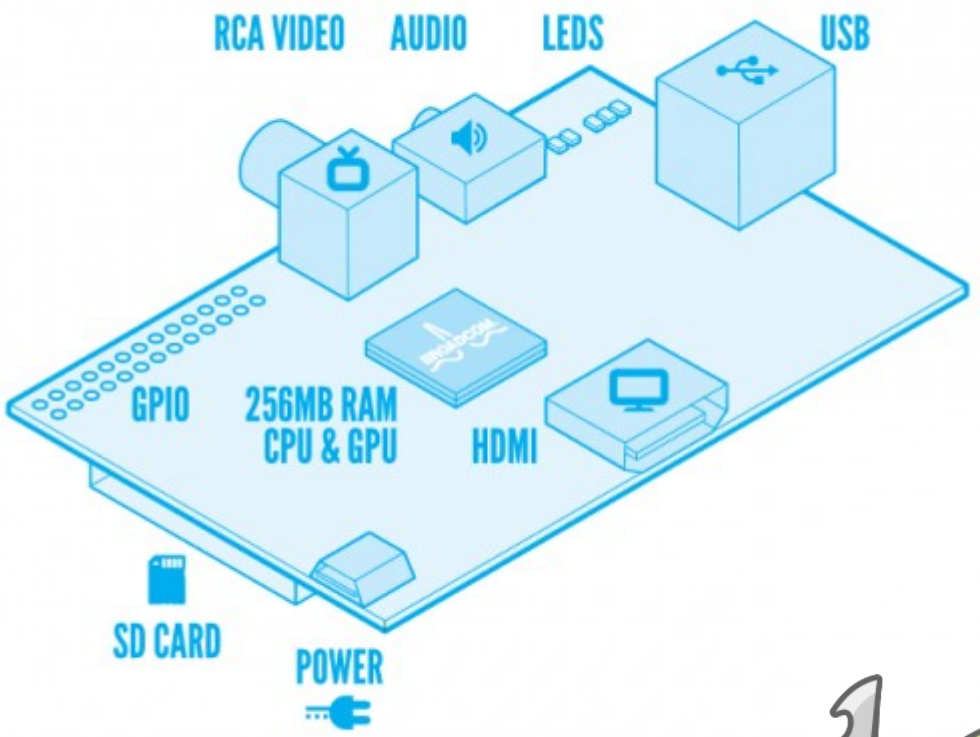
Windows 3.0 on a Pi



http://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

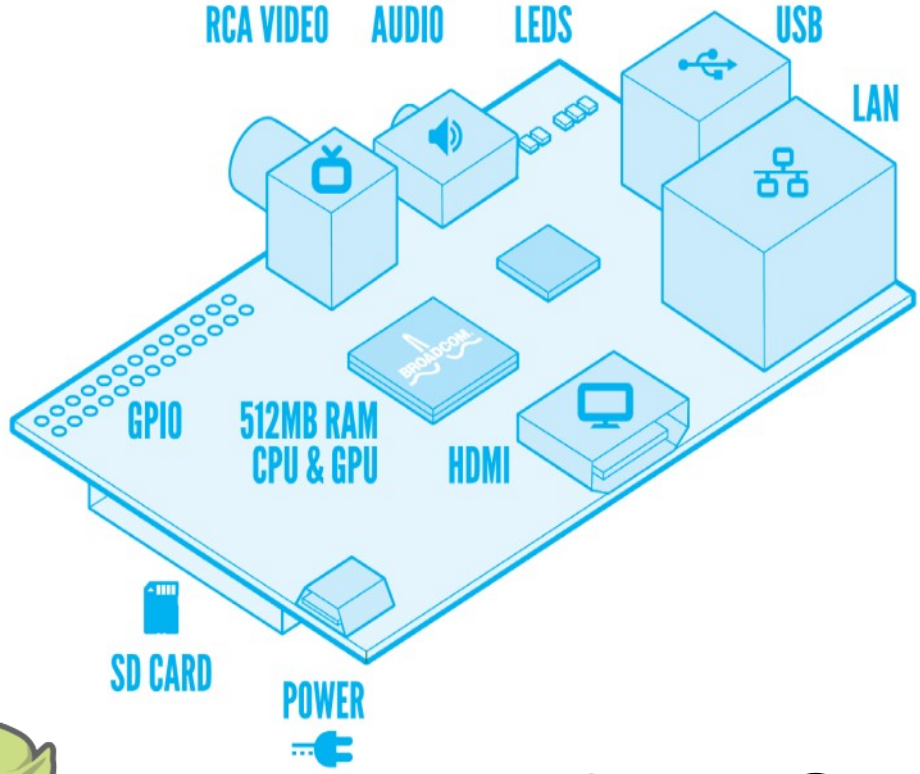
Meet the Enemy

Model A



1 x USB
256 MB RAM

Model B



2x USB
Ethernet
512 MB RAM

No real time clock?

- Raspberry pi has no Real Time Clock
- The Pi remember the last time
  - fake-hwclock

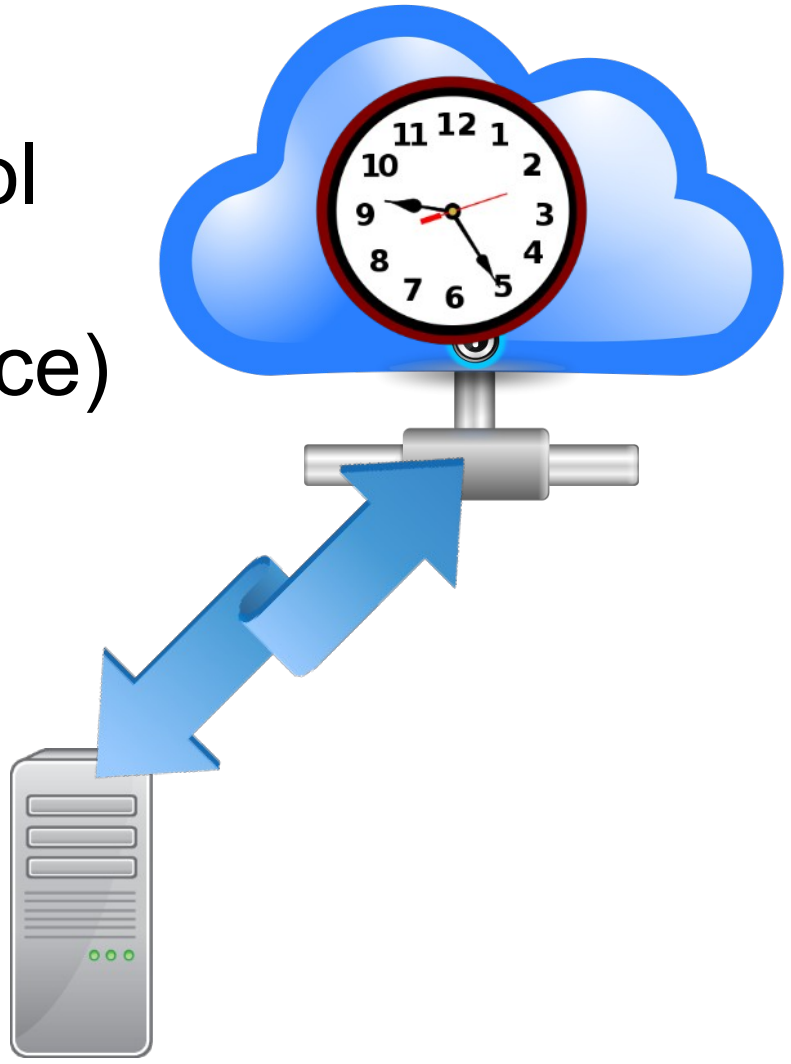
Rasclock – Add on Device

Unofficial add on

Solution

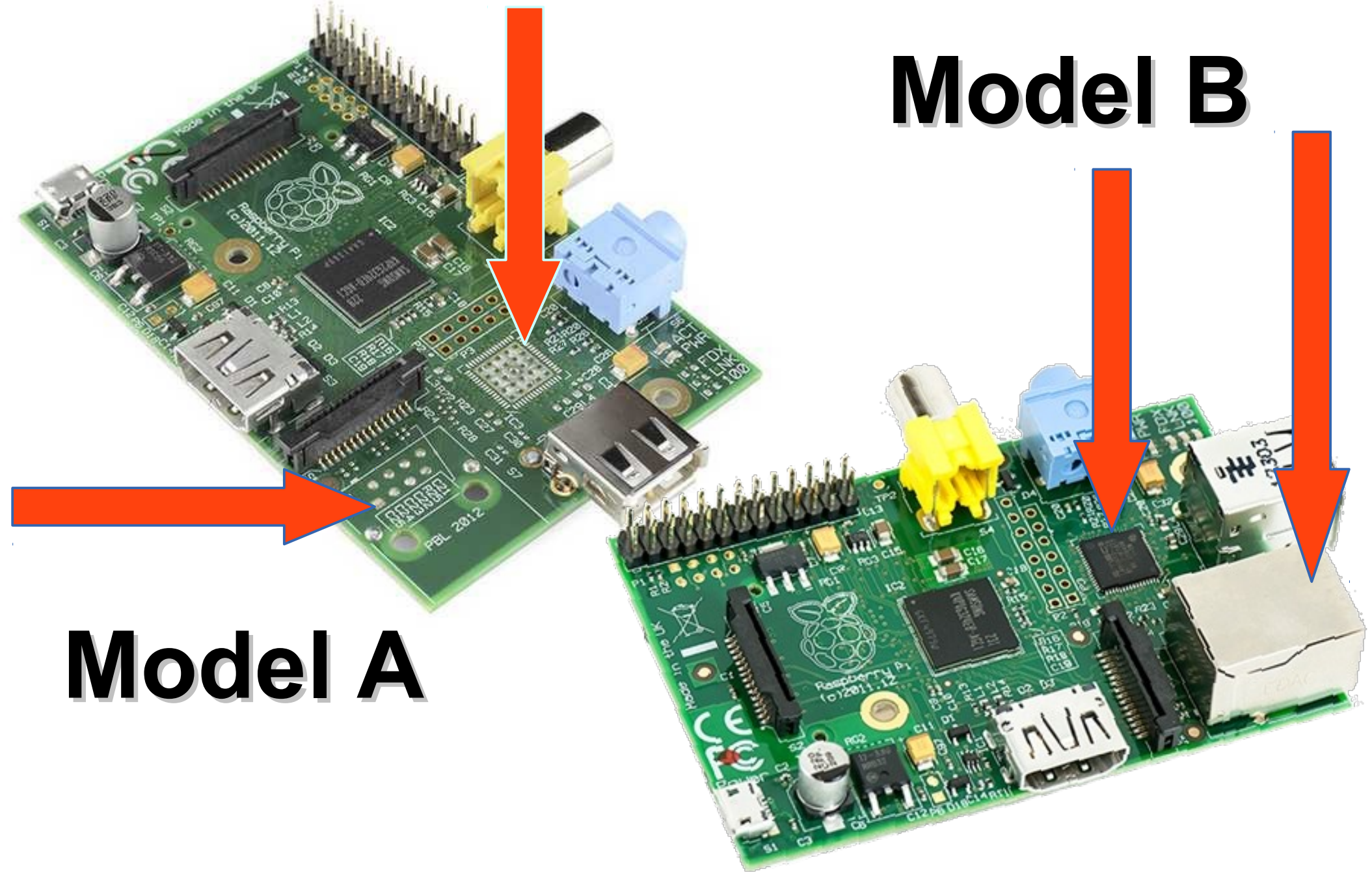
NTP – Network Time Protocol

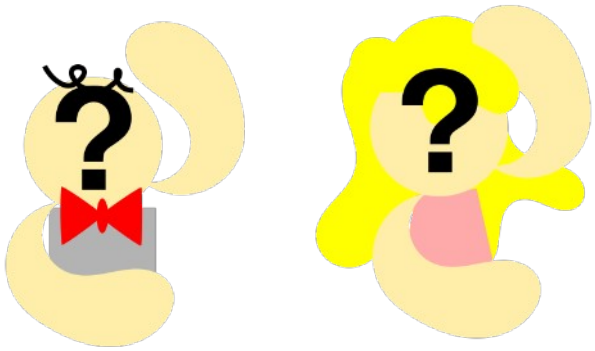
Ntpd → NTP Deamon (Service)



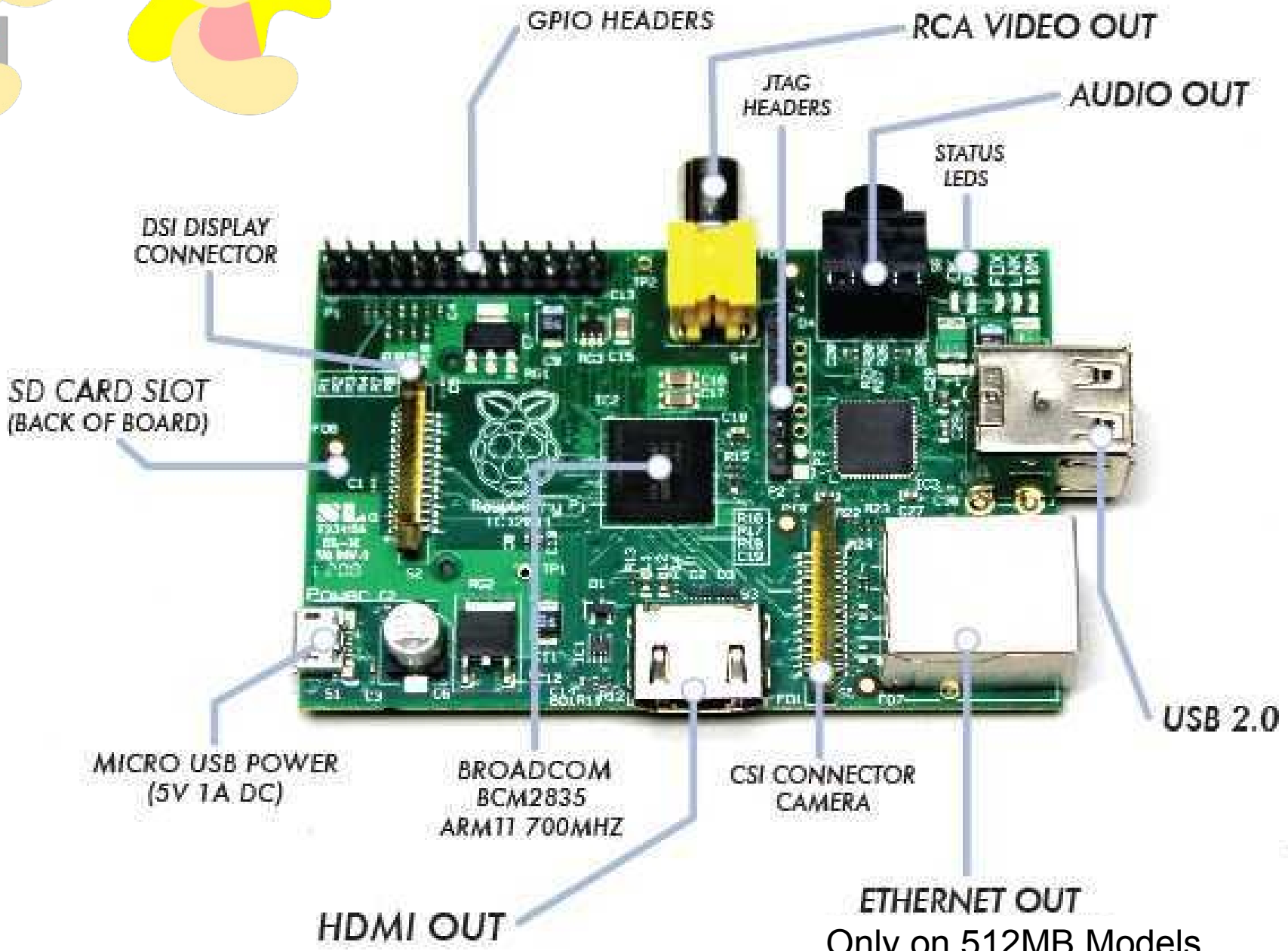
/etc/ntpd.conf
server ntp1.npl.co.uk iburst server 0.dk.pool.ntp.org iburst server 1.dk.pool.ntp.org iburst server 2.dk.pool.ntp.org iburst server 3.dk.pool.ntp.org iburst

How to identify?





What is what

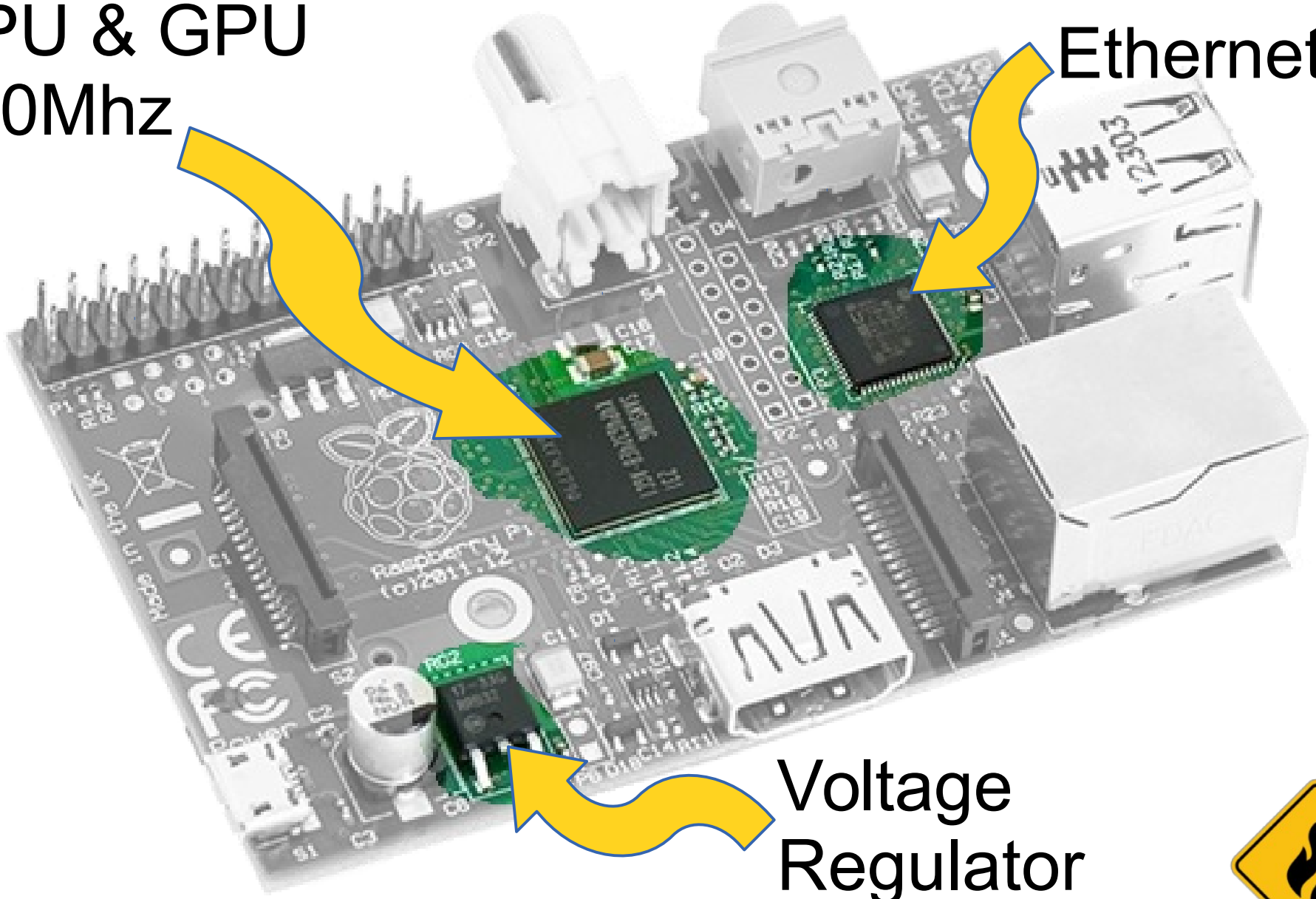


Only on 512MB Models

Warm Components

CPU & GPU
700Mhz

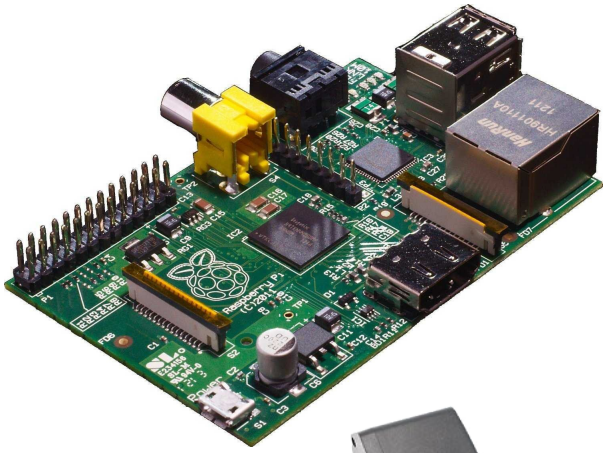
USB host
Ethernet



Voltage
Regulator

Connecting the PI

- Raspberry PI
- SDCard Min 4GB
- Micro USB Power Supply 5V/1A
- Keyboard/Mouse (optional)
- USB Wifi (optional)
- USB Hub (optional)



Problem with HDMI not visible

- /etc/config.conf
- Start with uncommenting "hdmi_force_hotplug=1", if that does not work try hdmi_safe=1. Also try to manually set your monitor to HDMI

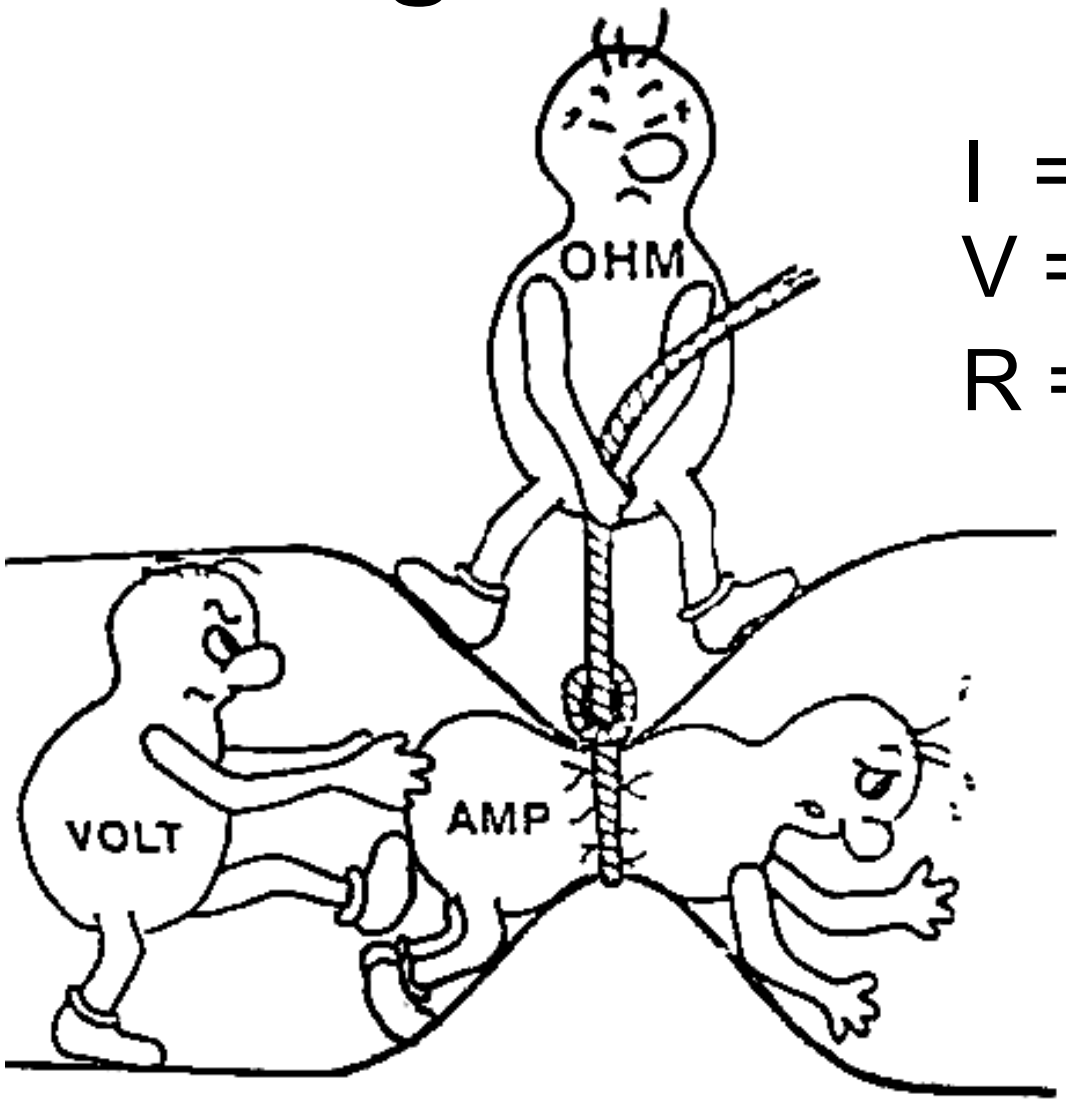
http://elinux.org/R-Pi_Troubleshooting#No_HDMI_output_at_all

The magic smoke producer

- Electronics
- Symbol basics
- Voltage
- Current
- Resistance

Ohms Law: **V = I * R**

Voltage = Current x Resistance

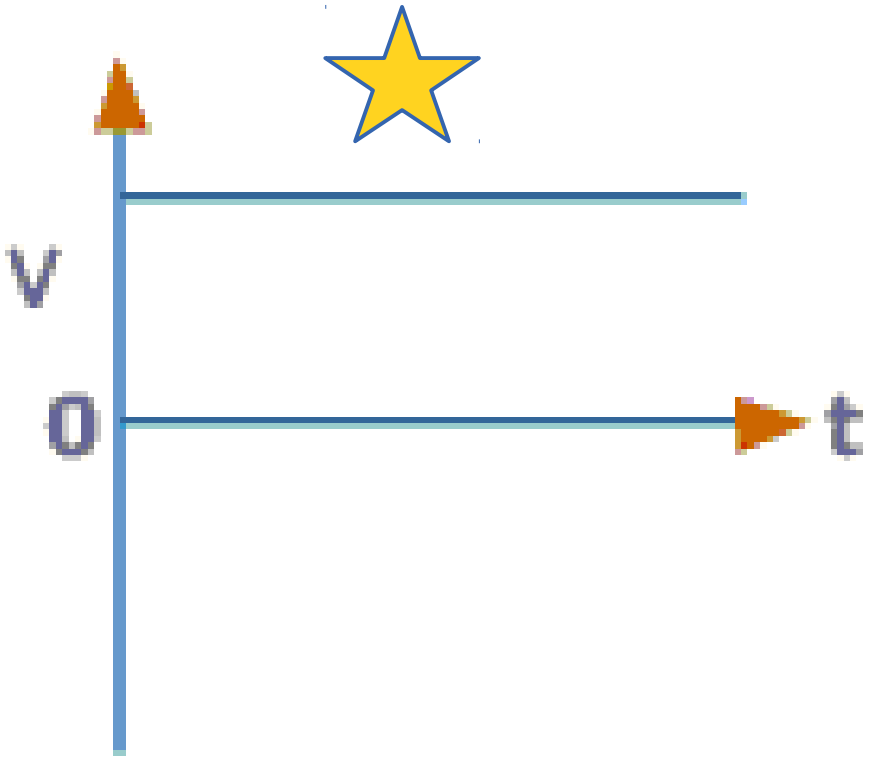


I => Current (Amp **A**)

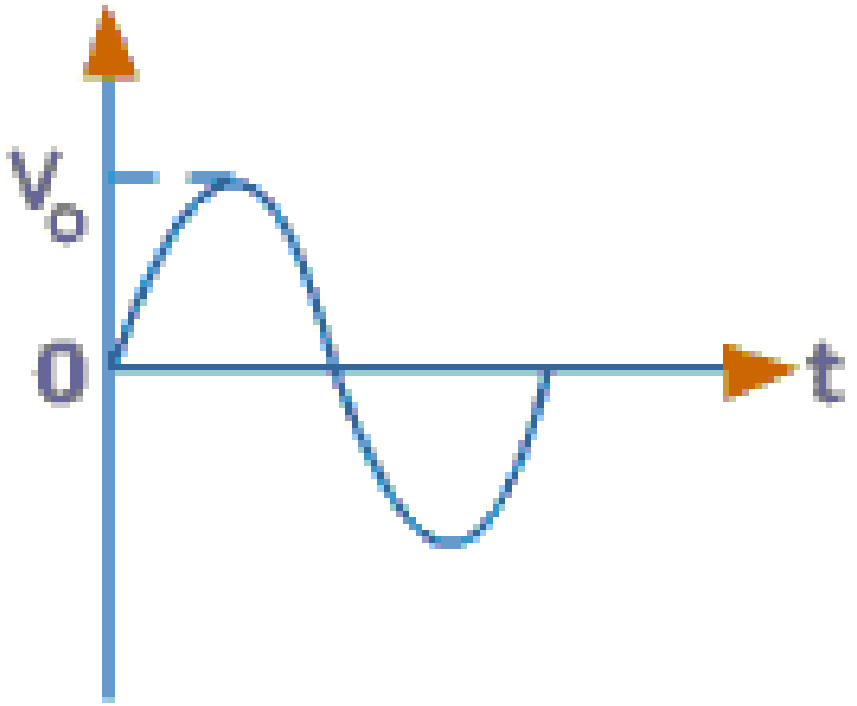
V => Voltage (Volt **V**)

R => Resistance (Ohm **Ω**)

AC DC Voltage

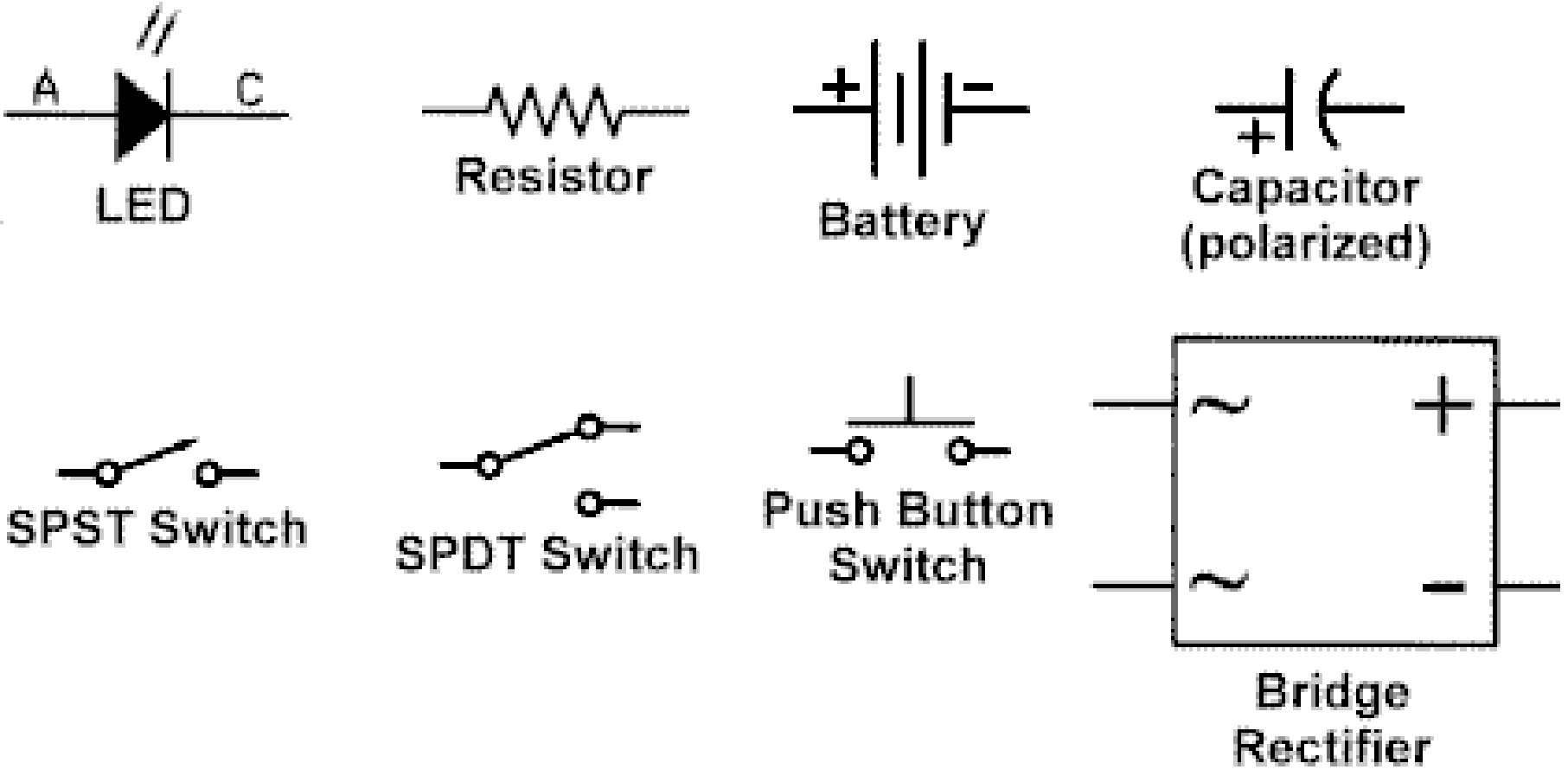


DC Source



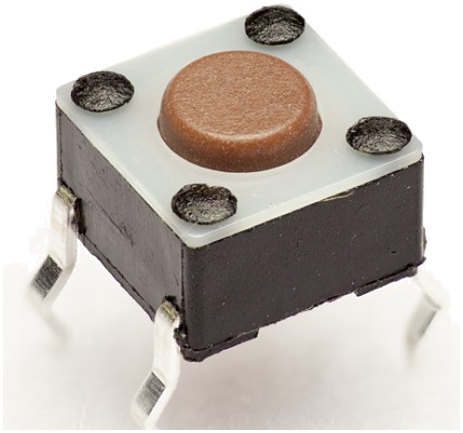
AC Source

Basic Symbols

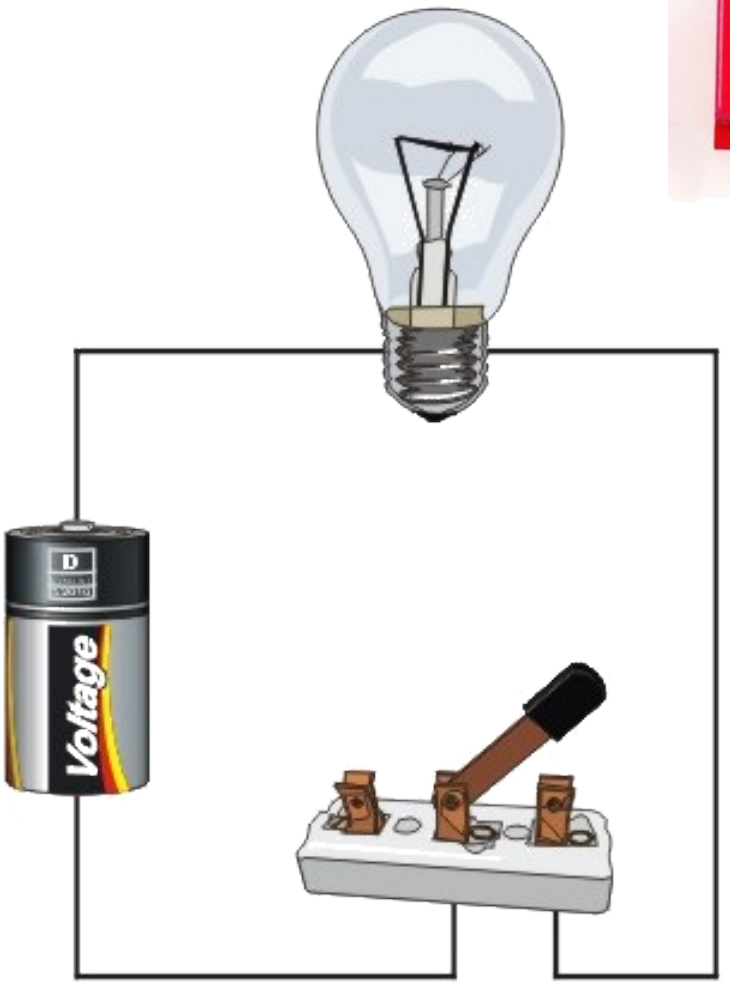


Switch

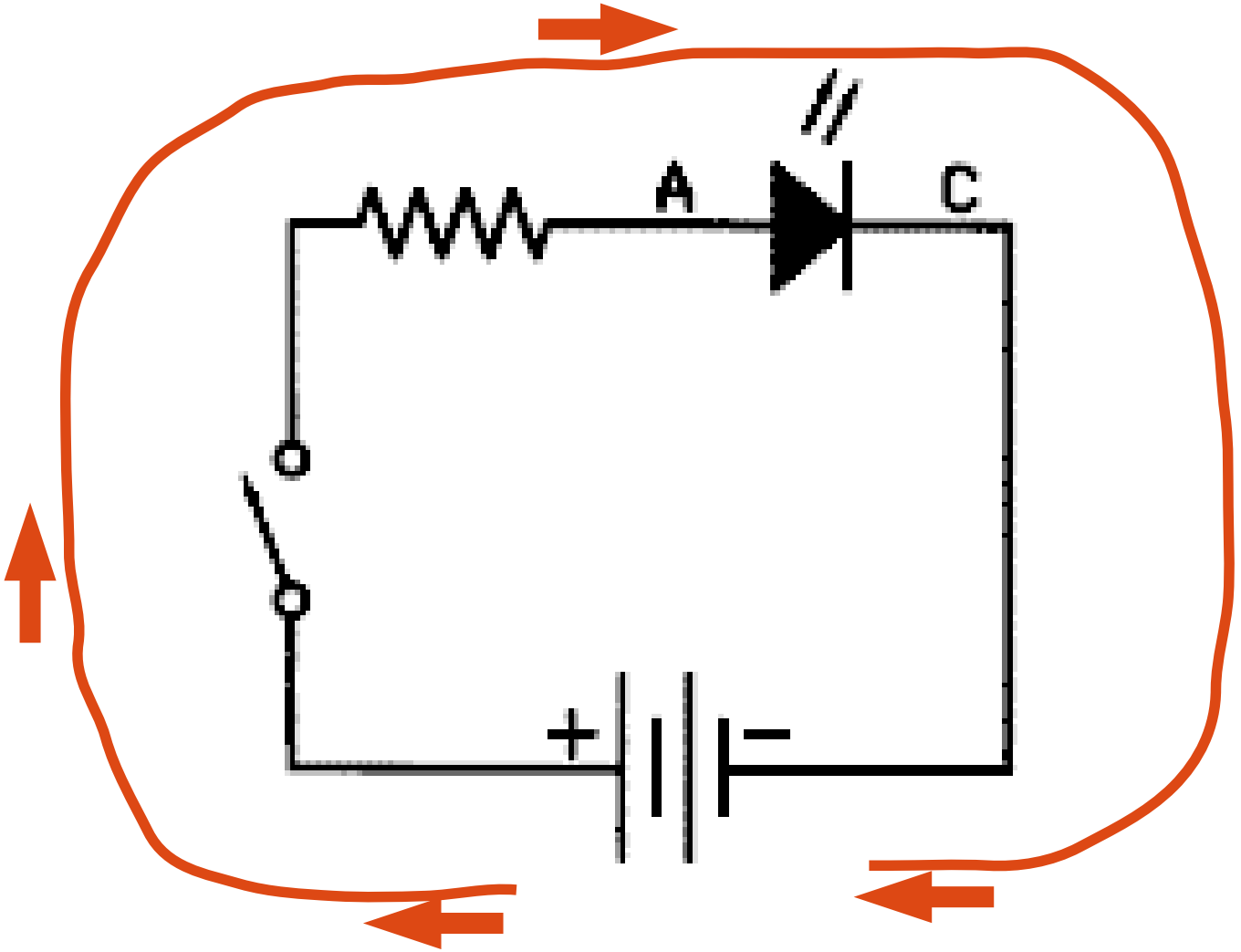
Push button



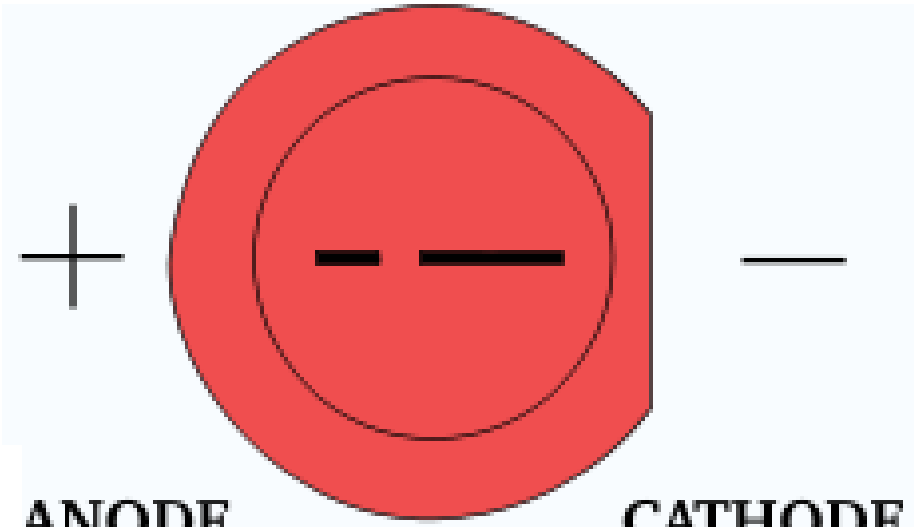
Toggle Switch

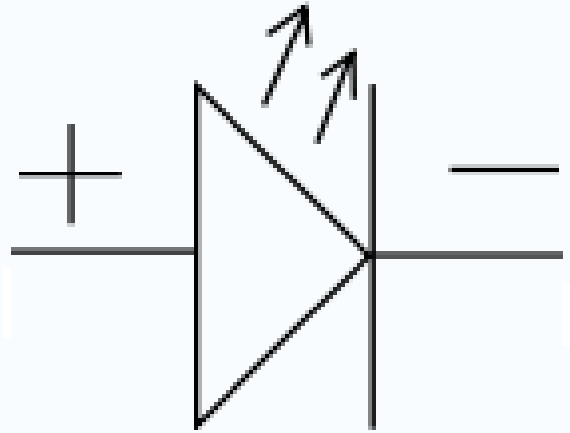


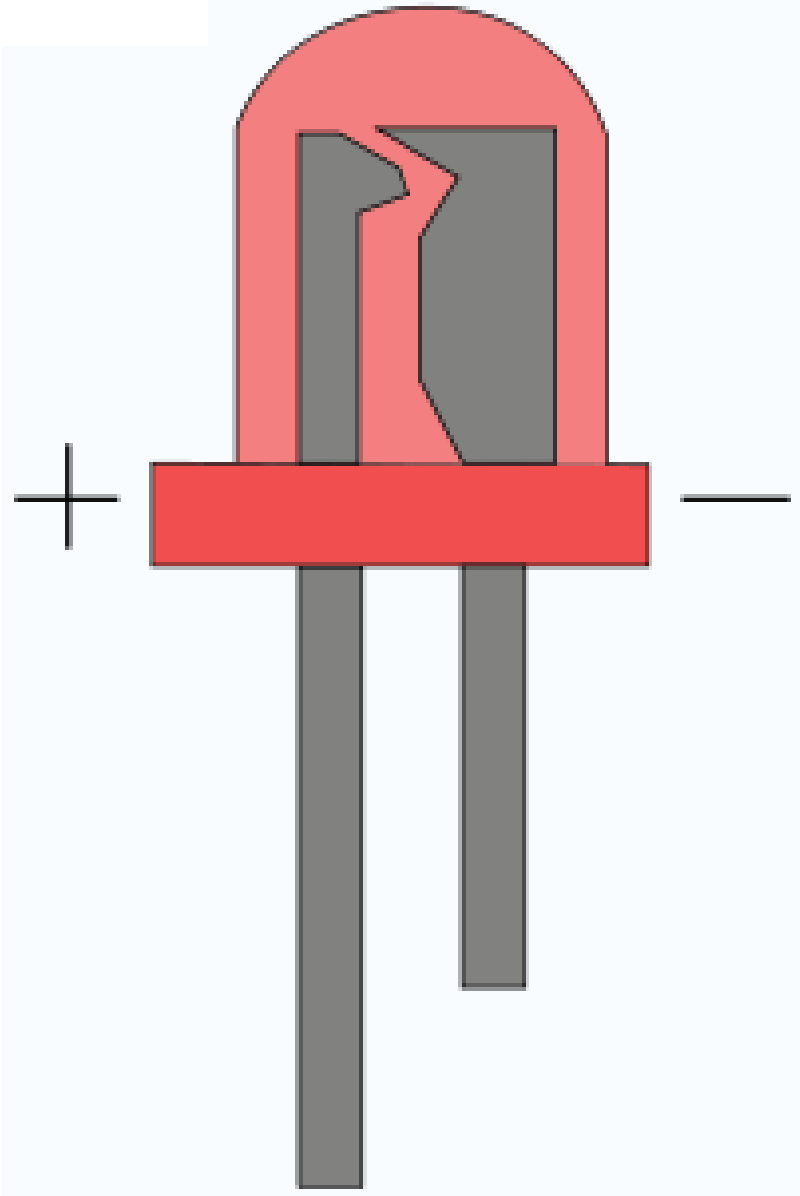
Circuits basic



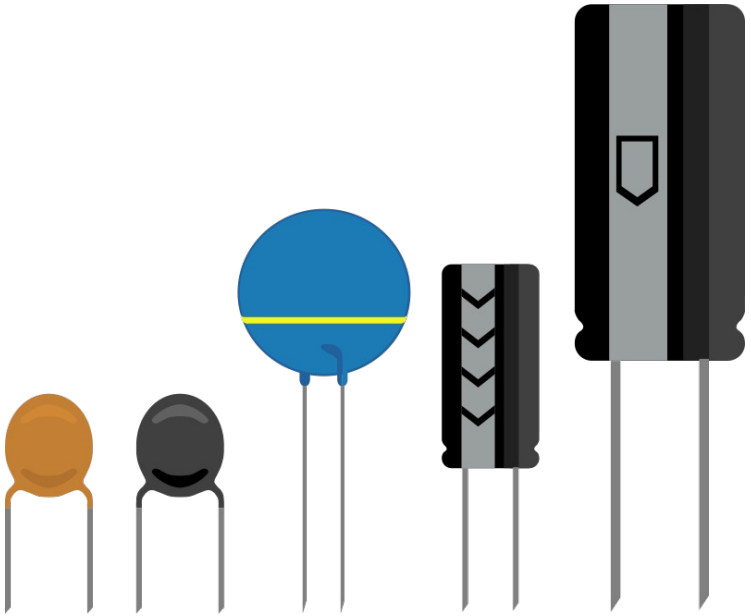
LED
Light Emitting Diode



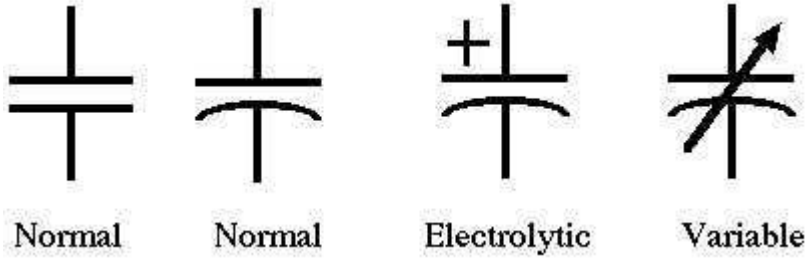


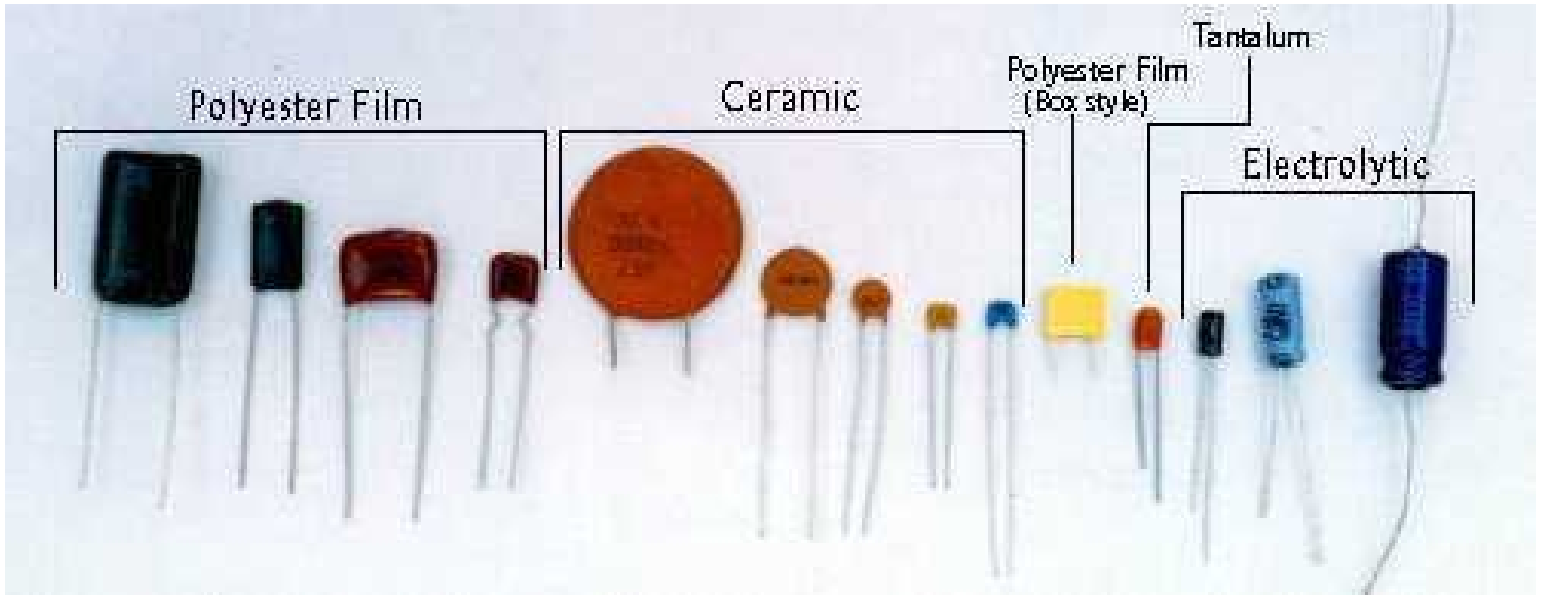


Capacitor

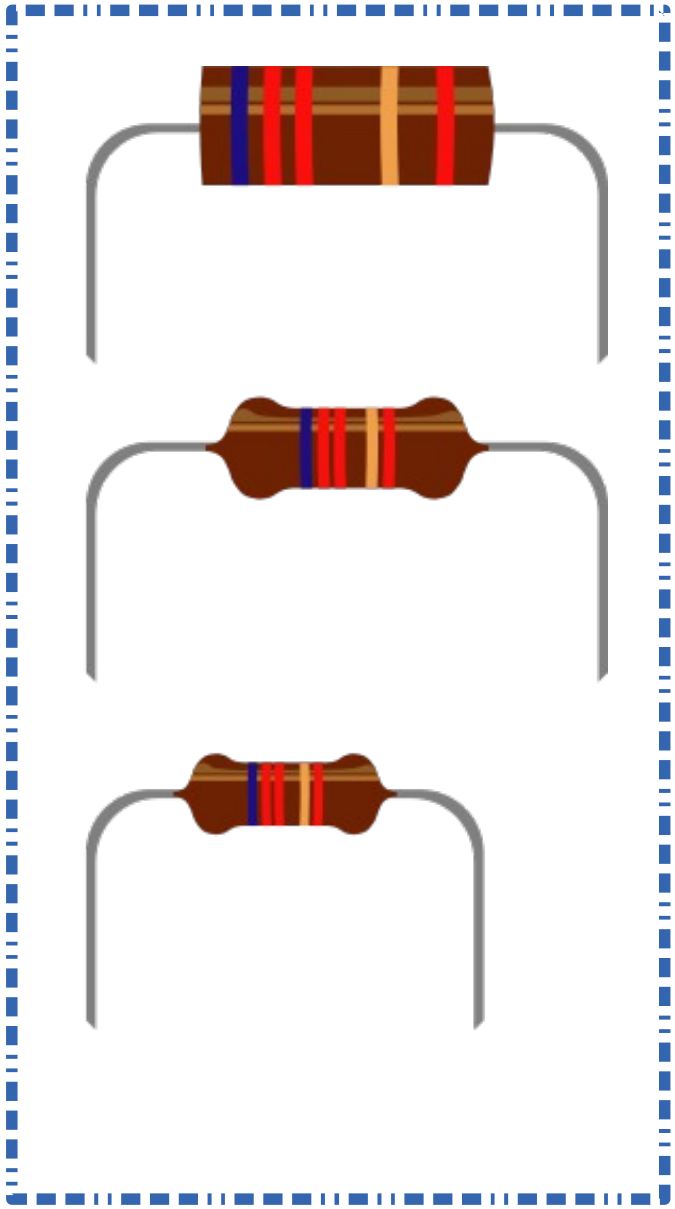


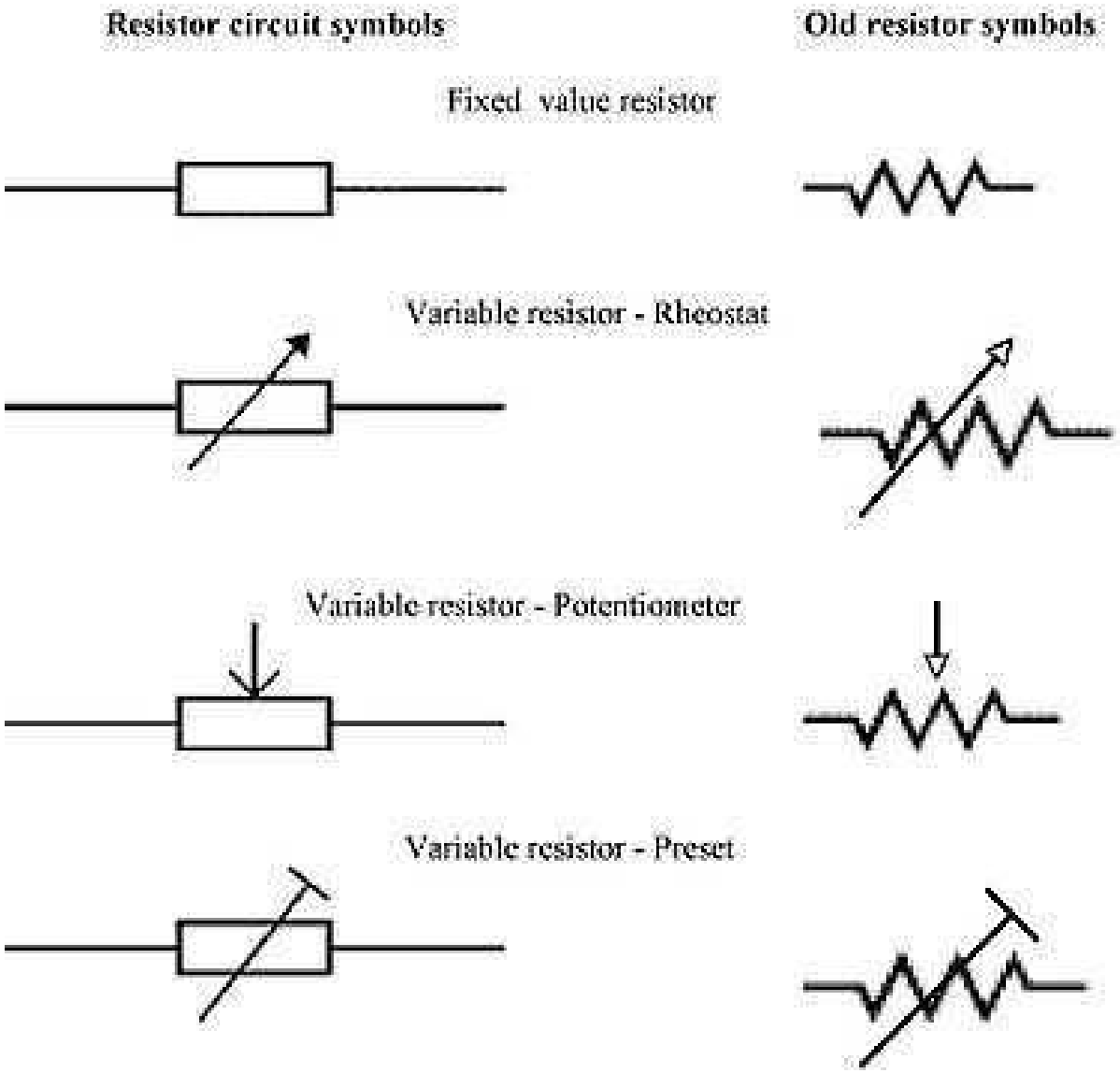
Diagram





Resistor

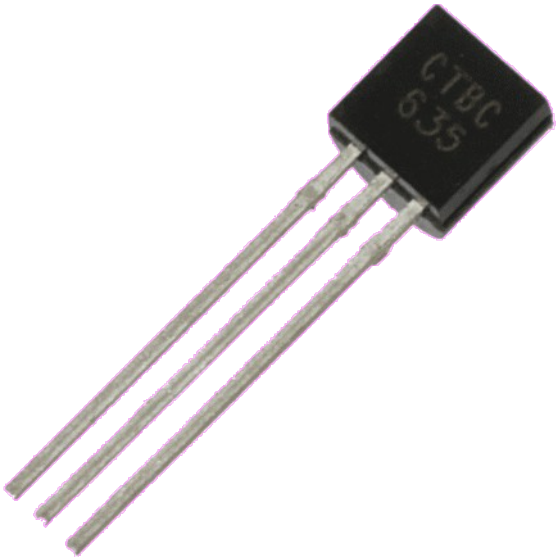


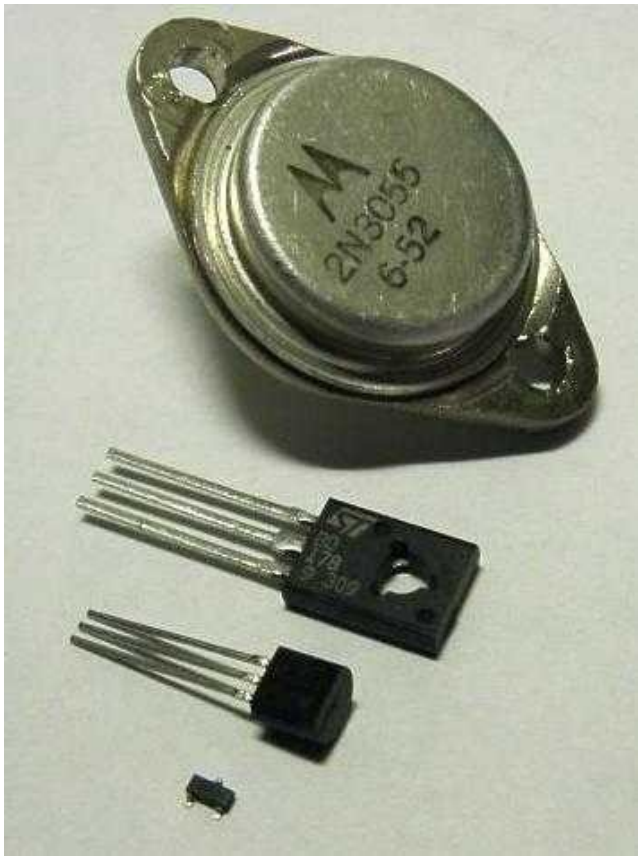

Resistor circuit symbols
Old resistor symbols
Fixed value resistor
Variable resistor - Rheostat
Variable resistor - Potentiometer
Variable resistor - Preset

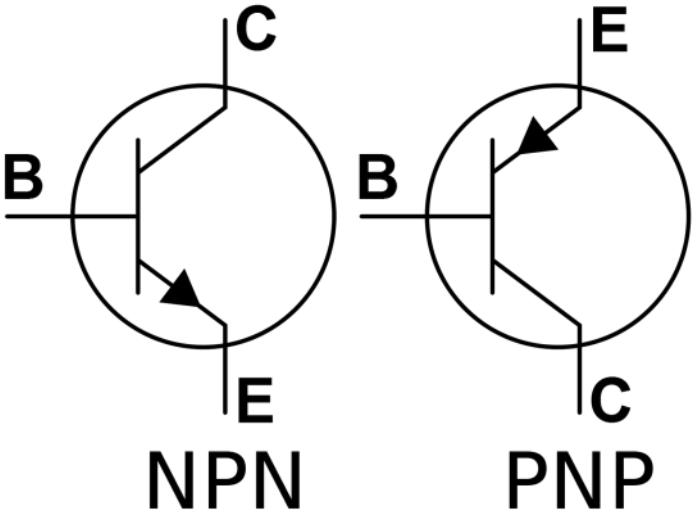
Transistor

Wikipedia says

A transistor is a semiconductor device used to amplify and switch electronic signals and electrical power.



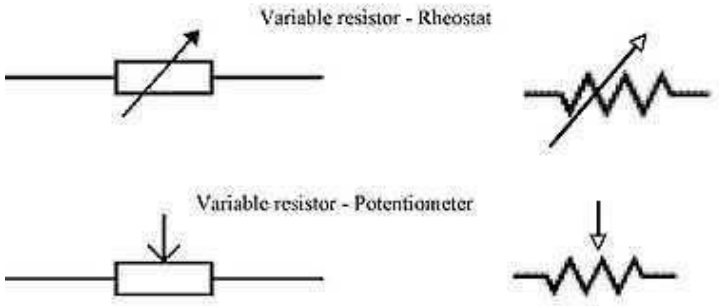




Info: wikipedia pictures: google search

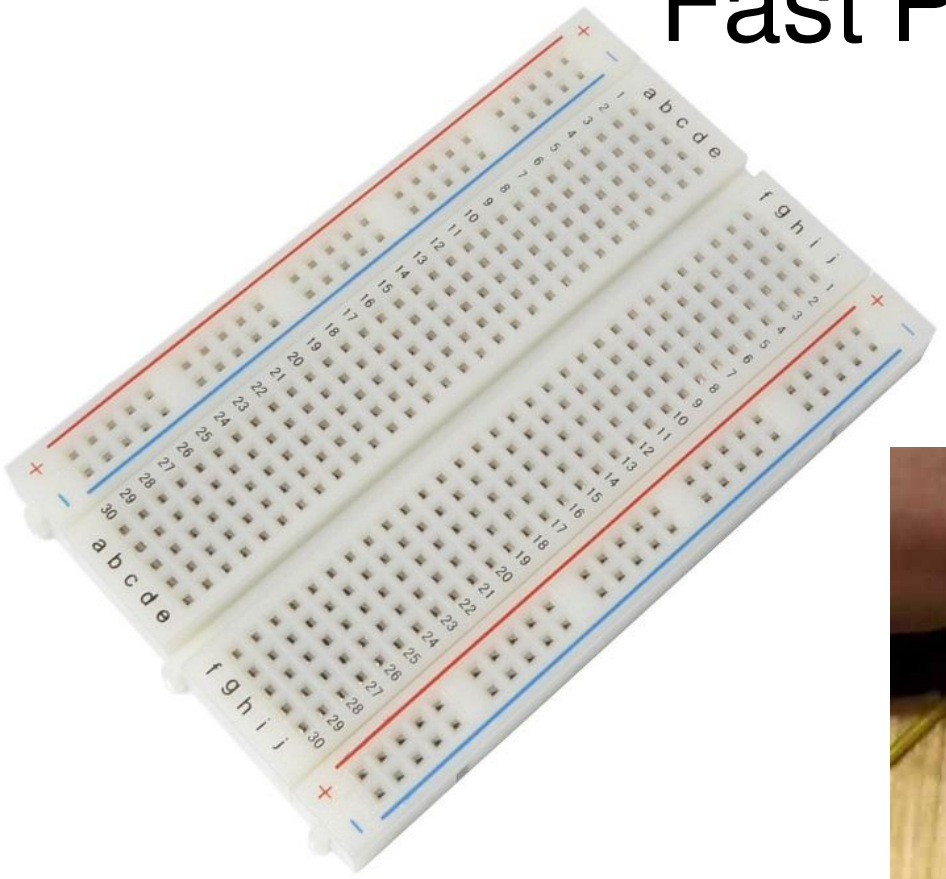
Difference between a Potentiometer and a Rheostat?

- The difference between a potentiometer and rheostat is the number of connections, or terminals. A potentiometer has three connections, and rheostat only has two connections. Rheostats are used in such things as light dimmer switches. A potentiometer can be found in a radio volume control knob.

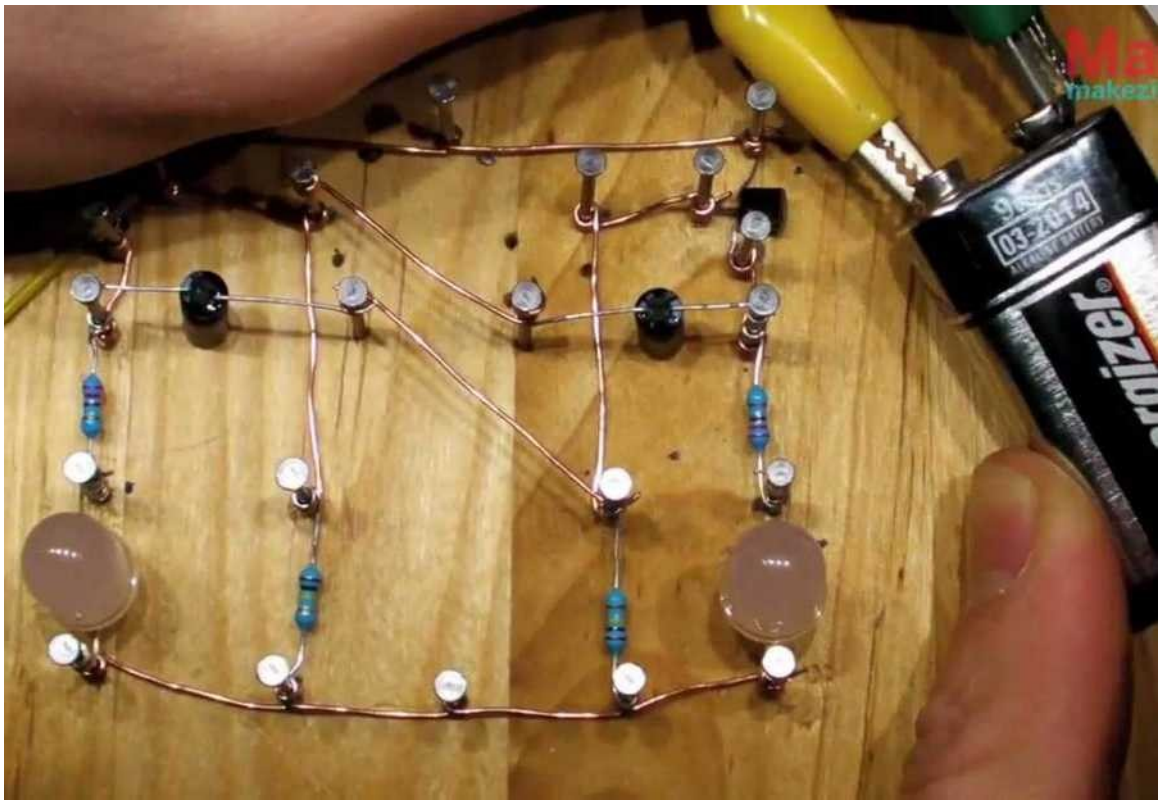


Source: http://www.ask.com/question/difference-between-a-potentiometer-and-a-rheostat

Breadboard
Fast Prototyping circuits

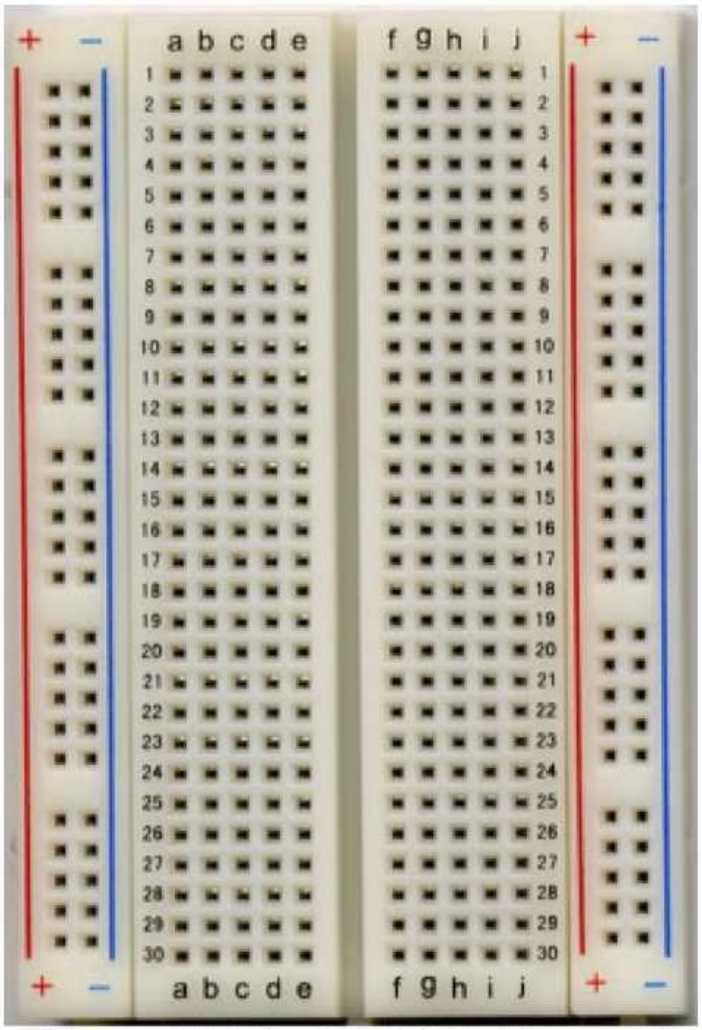


Actual breadboard circuit

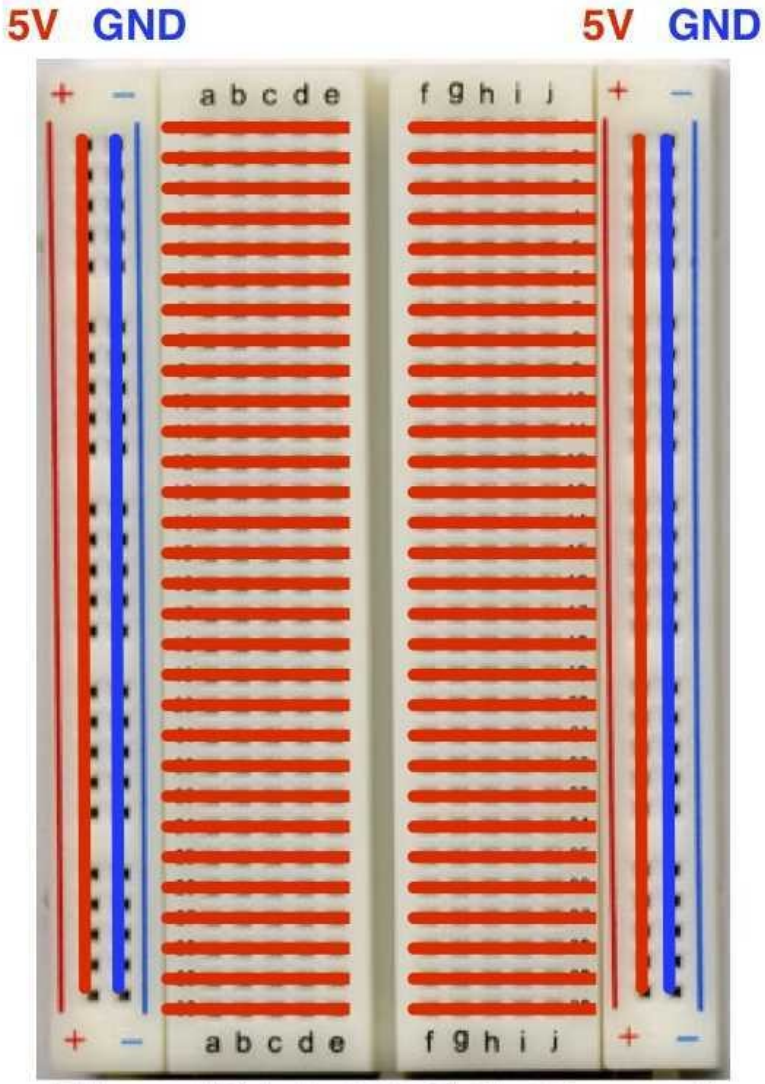


Breadboard

Photo



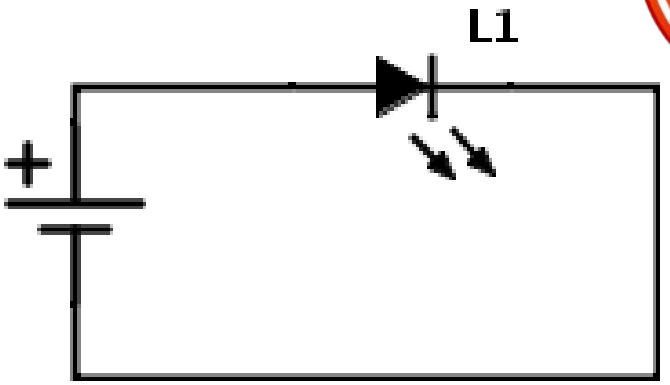
Schematic



Why circuits “burn”?

- More current than it can handle
- Blah
- Blah
- Blah
- More current than it can handle

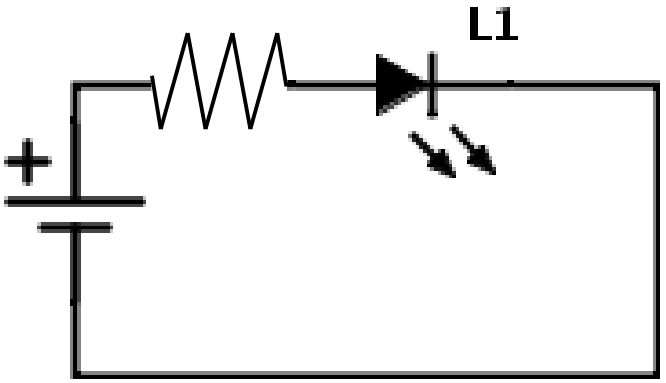
Short Circuit?



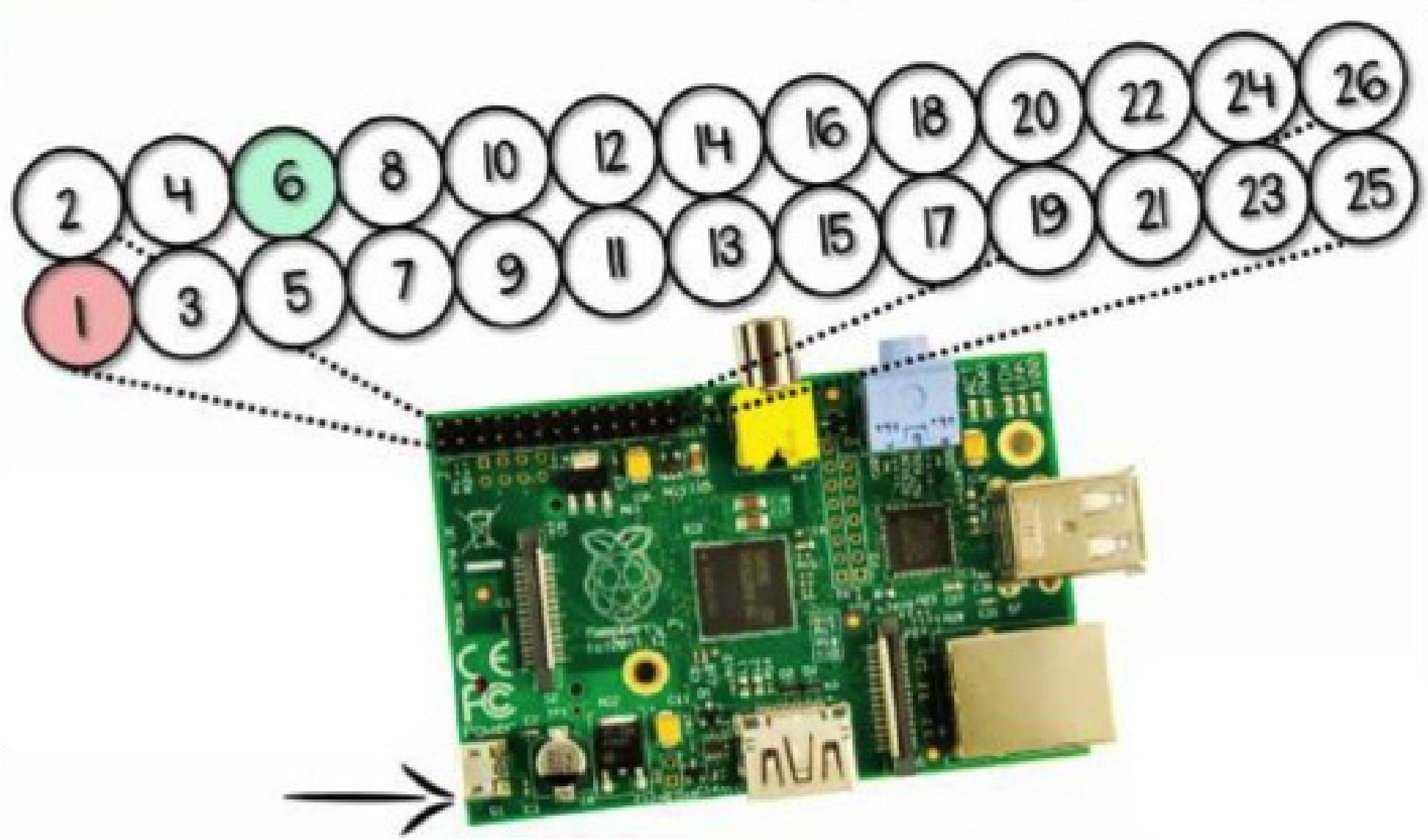
- V = I * R
- I = V / R
- When R = 0 the value of current is infinite
- Infinite Current?!!!!!

NOT a Short Circuit

- V = I * R
- I = V / R



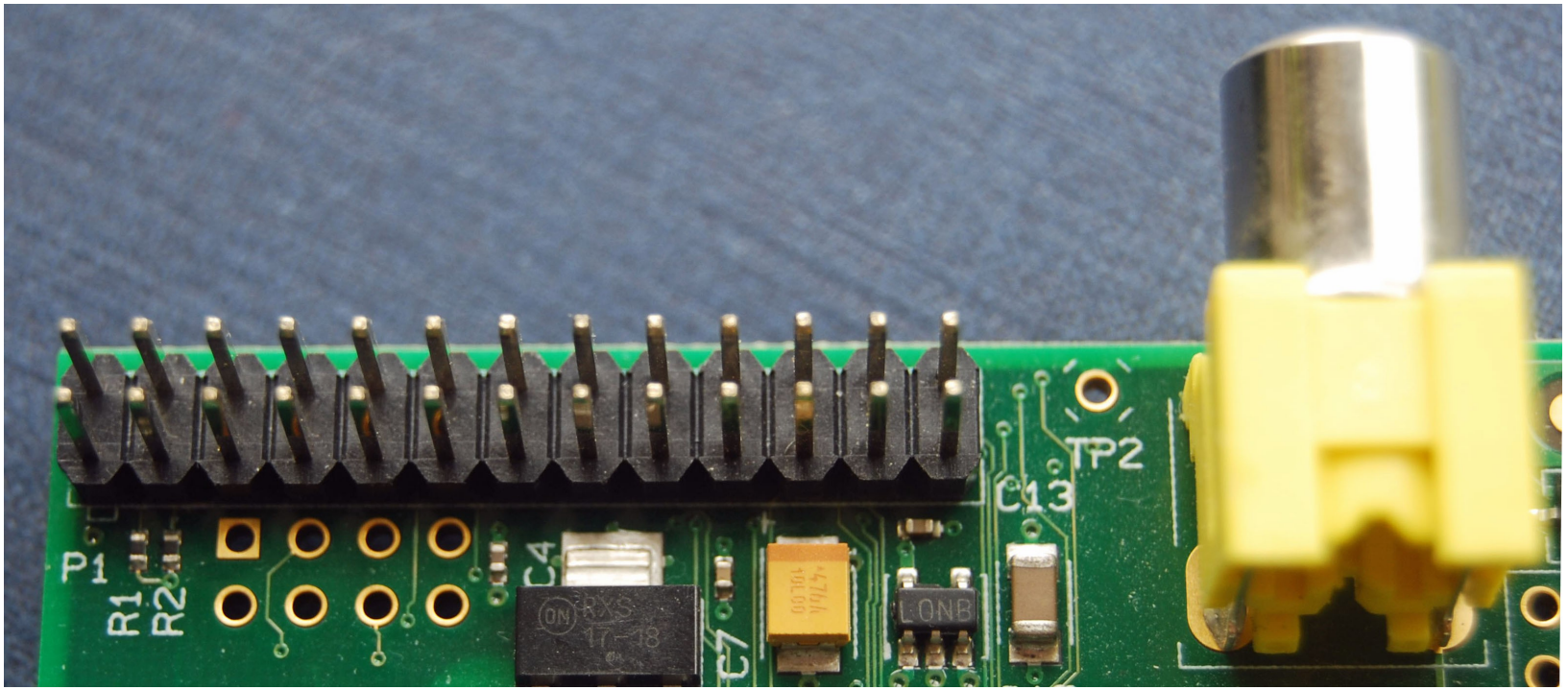
GPIO

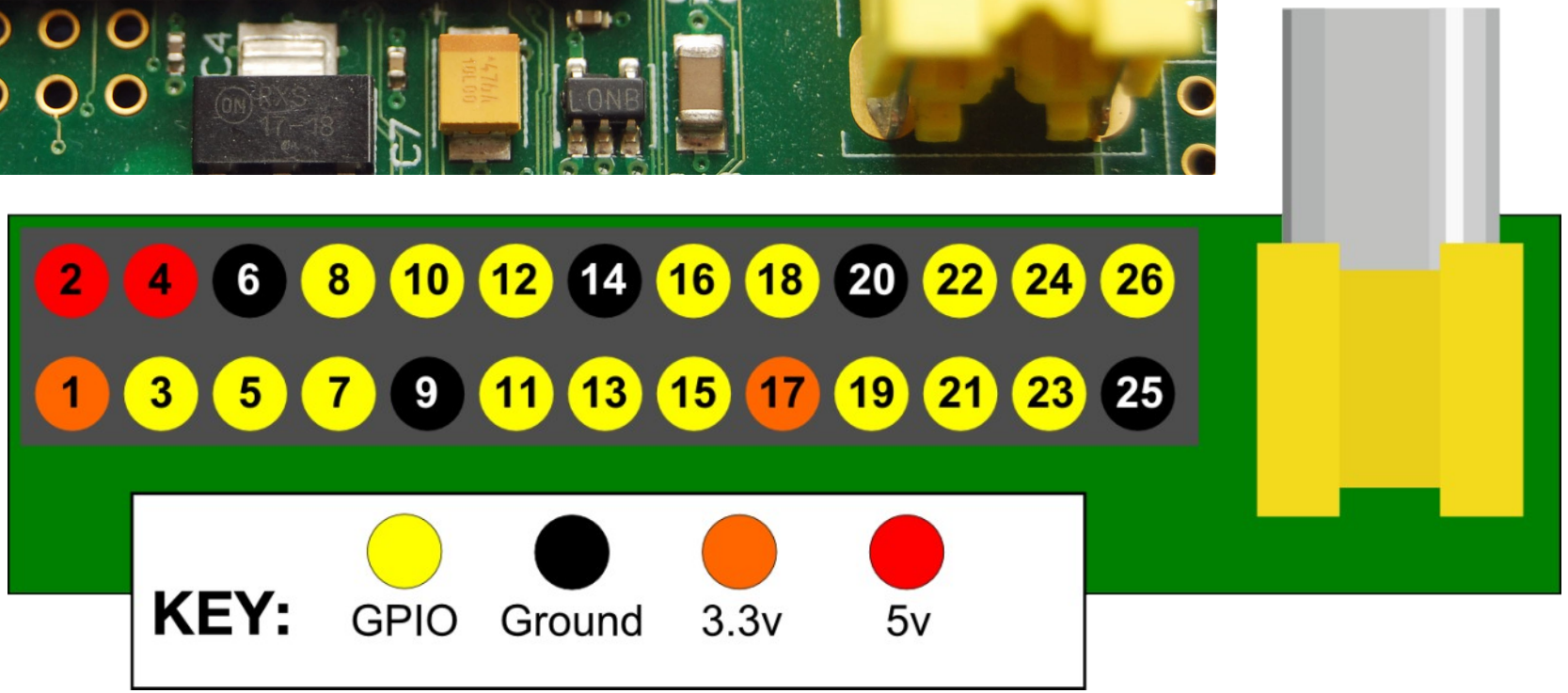


GPIO

- **Connection of a GPIO to a voltage higher than 3.3V will likely destroy the GPIO block within the SoC**
- To answer that question I/O ports have two parameters which deal with the output level:
- V_{IL} : The maximum low level voltage. (0.8V on the BCM2835)
  - 0.8V means that if the output is Low it will be <= 0.8V
- V_{IH} : The minimum high level voltage. (1.3V on the BCM2835)
  - 1.3V means that if the output is High it will be >= 1.3V

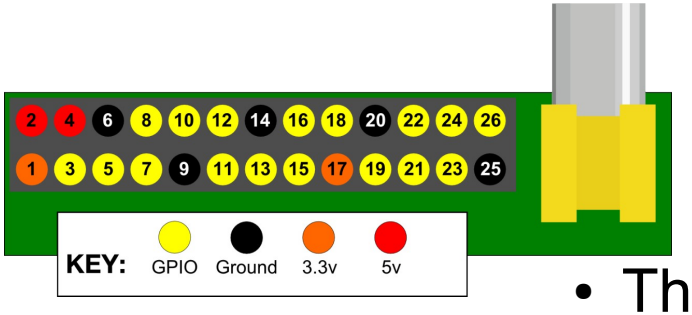
GPIO


P1
R1
R2
C4
C7
C13
TP2


2 4 6 8 10 12 14 16 18 20 22 24 26
1 3 5 7 9 11 13 15 17 19 21 23 25
KEY:
GPIO
Ground
3.3v
5v

Raspberry Pi Safe Current

- What is a safe current?
- All the electronics of the pads are designed for 16mA. That is a safe value under which you will not damage the device. Even if you set the drive strength to 2mA and then load it so16mA comes out that will not damage the device. **Other than that there is no guaranteed maximum safe current.**

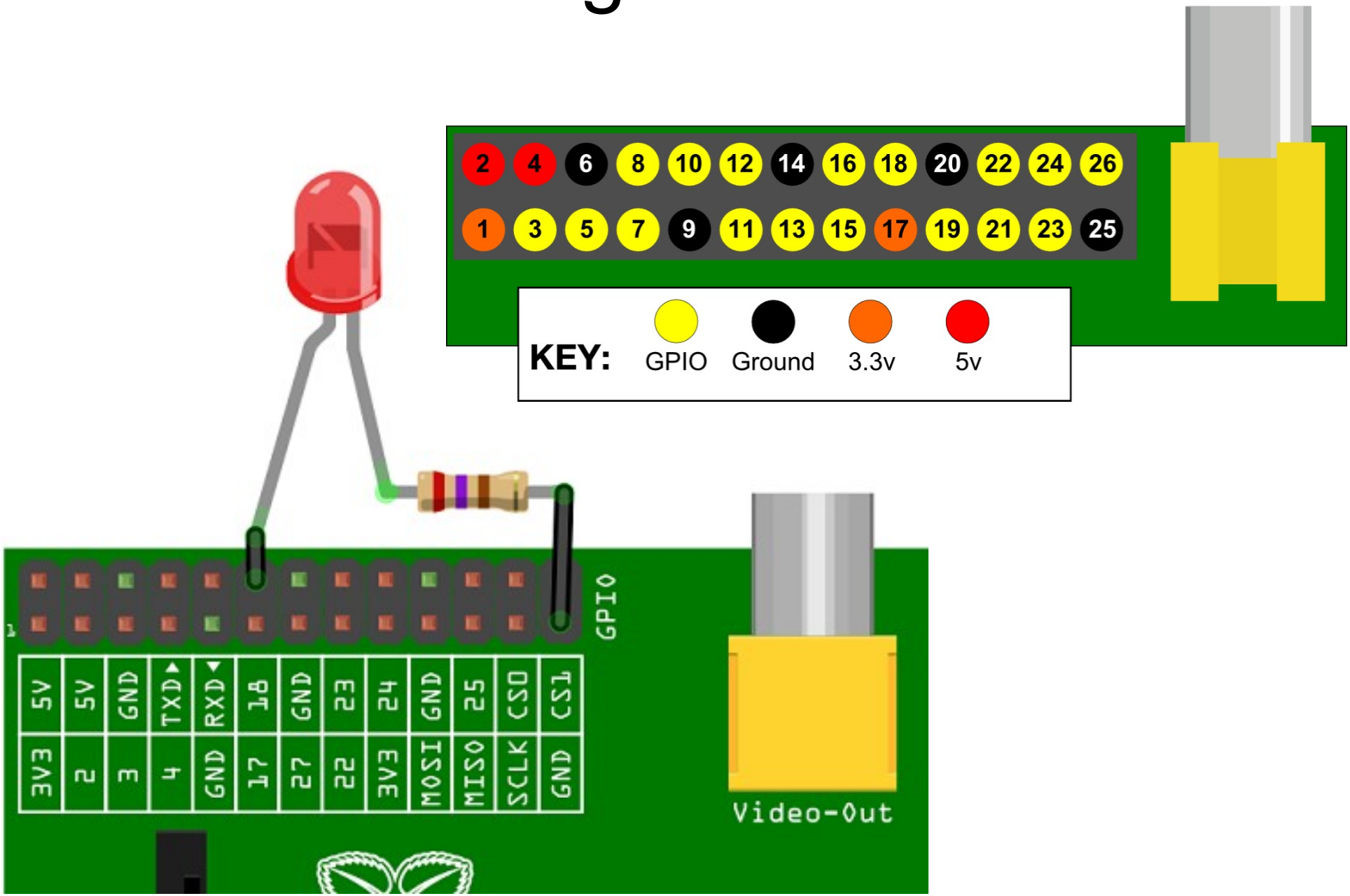


Power Pins

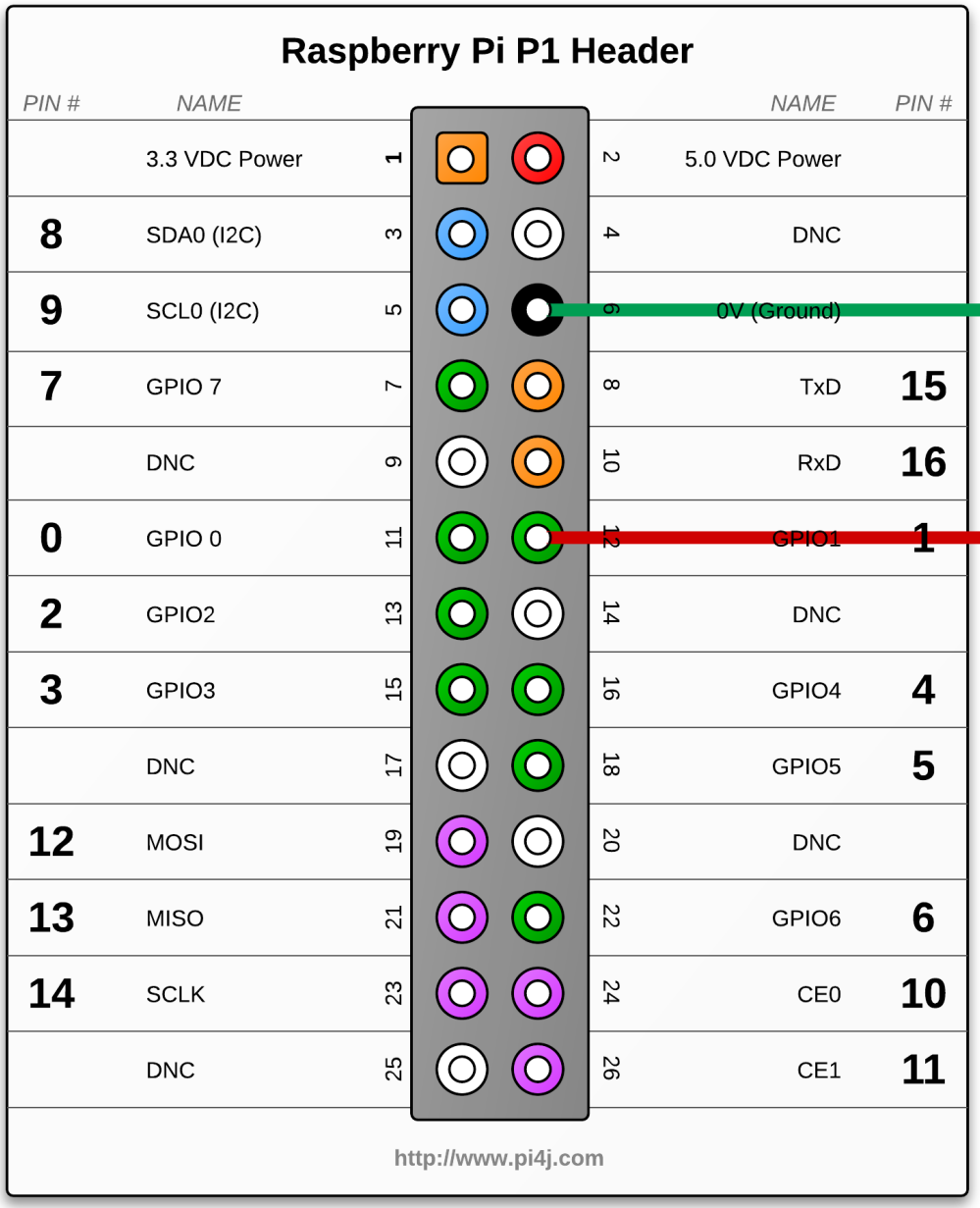
- The maximum permitted current draw from the 3.3 V pins is 50 mA.
- Maximum permitted current draw from the 5 V pin is the USB input current (usually 1 A) minus any current draw from the rest of the board.
- Model A: 1000 mA - 500 mA -> max current draw: 500 mA
- Model B: 1000 mA - 700 mA -> max current draw: 300 mA
- **Be very careful with the 5 V pins P1-02 and P1-04, because if you short 5 V to any other P1 pin you may permanently damage your RasP**i. Before probing P1, it's a good idea to strip short pieces of insulation off a wire and push them over the 5 V pins so you don't accidentally short them with a probe.

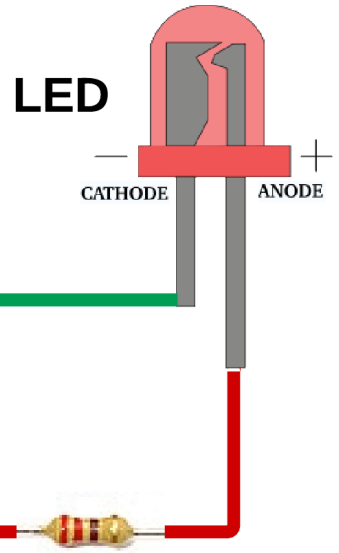
http://elinux.org/RPi_Low-level_peripherals

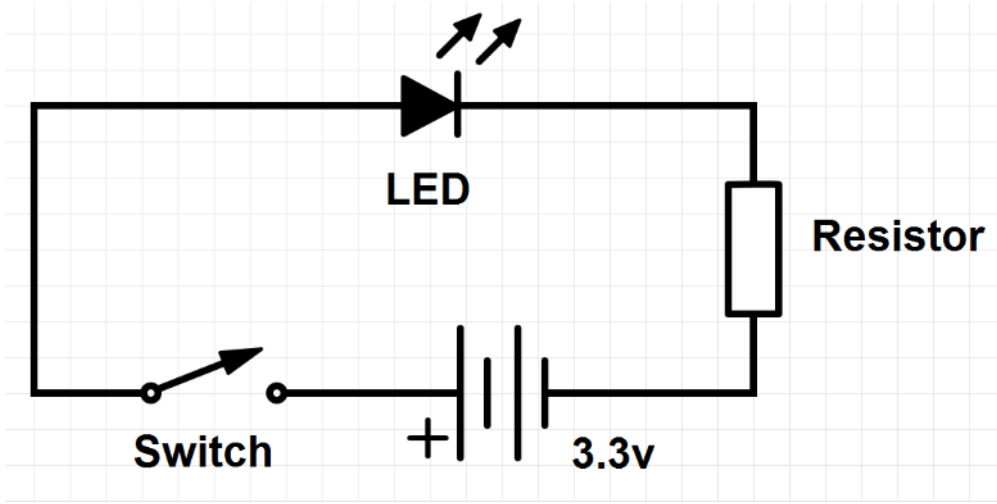
Wiring a LED


2 4 6 8 10 12 14 16 18 20 22 24 26
1 3 5 7 9 11 13 15 17 19 21 23 25
KEY:
GPIO
Ground
3.3v
5v
GPIO
3V3 5V GND TXD RXD 18 GND 23 24 GND 25 CS0 CS1
3V3 2 3 4 GND 17 27 22 3V3 MOSI MISO SCLK GND
Video-Out

Connection


Raspberry Pi P1 Header
PIN # NAME NAME PIN #
3.3 VDC Power 1 2 5.0 VDC Power
8 SDA0 (I2C) 3 4 DNC
9 SCL0 (I2C) 5 6 0V (Ground)
7 GPIO 7 7 8 TxD 15
DNC 9 10 RxD 16
0 GPIO 0 11 12 GPIO1 1
2 GPIO2 13 14 DNC
3 GPIO3 15 16 GPIO4 4
DNC 17 18 GPIO5 5
12 MOSI 19 20 DNC
13 MISO 21 22 GPIO6 6
14 SCLK 23 24 CE0 10
DNC 25 26 CE1 11
http://www.pi4j.com


LED
CATHODE
ANODE
220 Ohm
Resistor


LED
Resistor
Switch
3.3v

Installing GPIO Module

$sudo apt-get update

$sudo apt-get install python-dev

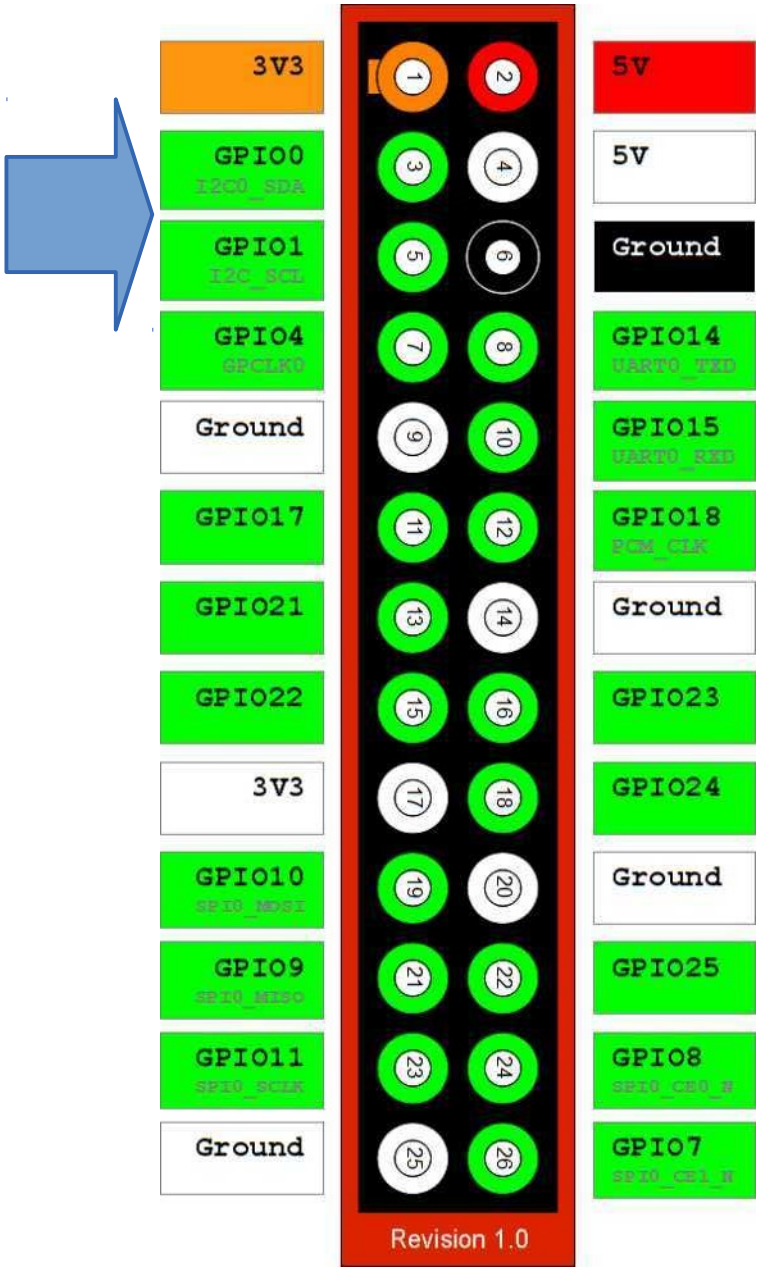
$sudo apt-get install python-rpi.gpio

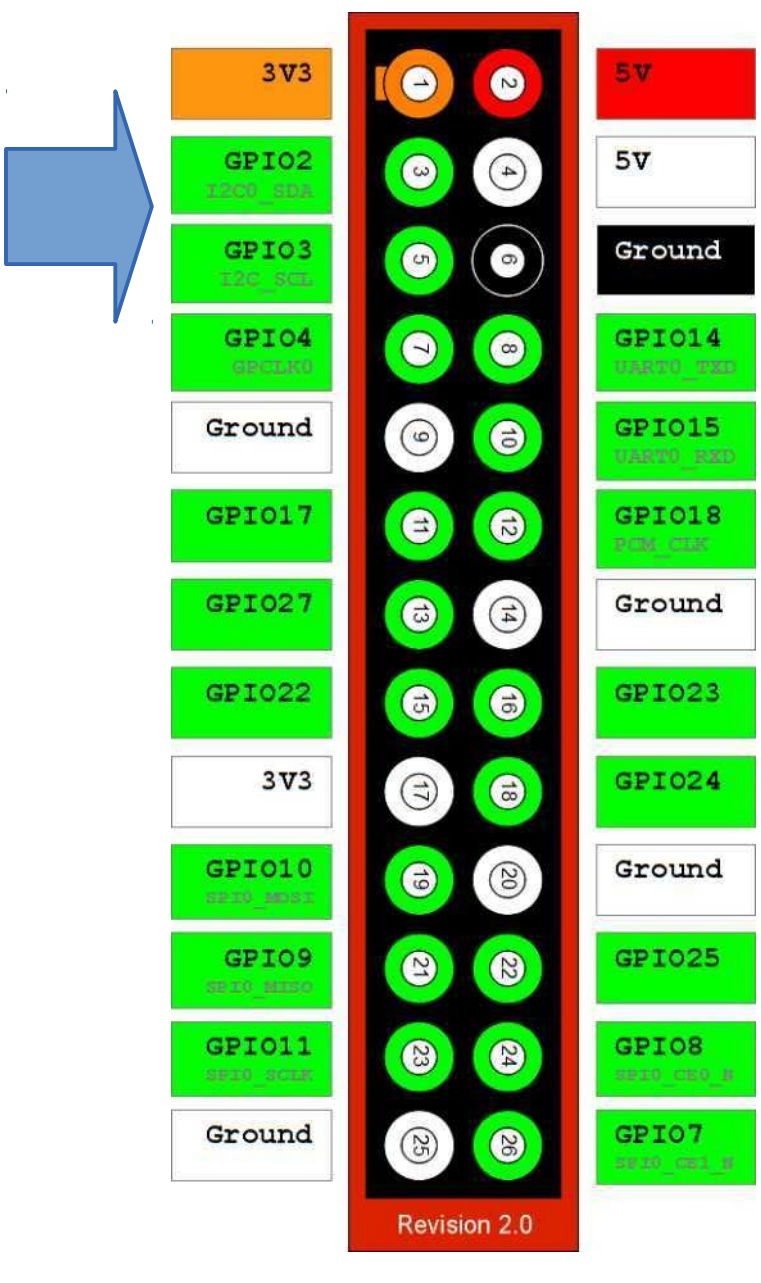
GPIO.setmode(GPIO.BCM)
GPIO.setmode(GPIO.BOARD)

- The GPIO.BOARD option specifies that you are referring to the pins by the number of the pin the the plug - i.e the numbers printed on the board (e.g. P1) and in the middle of the diagrams below.
- The GPIO.BCM option means that you are referring to the pins by the "Broadcom SOC channel" number, these are the numbers after "GPIO" in the green rectangles around the outside of the below diagrams:
- Unfortunately the BCM numbers changed between versions of the Model B, and you'll need to work out which one you have guide here. So it may be safer to use the BOARD numbers if you are going to use more than one pi in a project.

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

GPIO Change Rev 1 to Rev 2





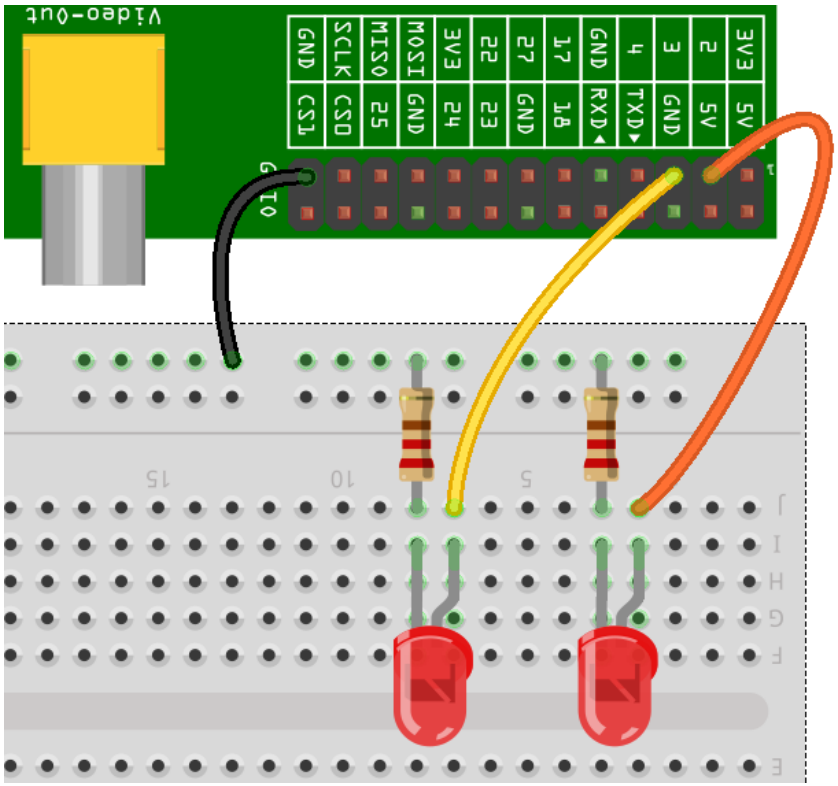
Identify Pi Revision

```
pi@raspberrypi ~ $ cat /proc/cpuinfo
processor       : 0
model name      : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS        : 2.00
Features        : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xb76
CPU revision    : 7

Hardware        : BCM2708
Revision        : 000e
Serial          : 0000000092ca1038
```

```
'0002' => 'Model B Revision 1.0',
'0003' => 'Model B Revision 1.0 + Fuses mod and D14 removed',
'0004' => 'Model B Revision 2.0 256MB',
'0005' => 'Model B Revision 2.0 256MB',
'0006' => 'Model B Revision 2.0 256MB',
'0008' => 'Model A 256MB',
'000e' => 'Model B, Revision 2.0, 512MB'
'000f' => 'Model B Revision 2.0 512MB',
```

Simple IO testing

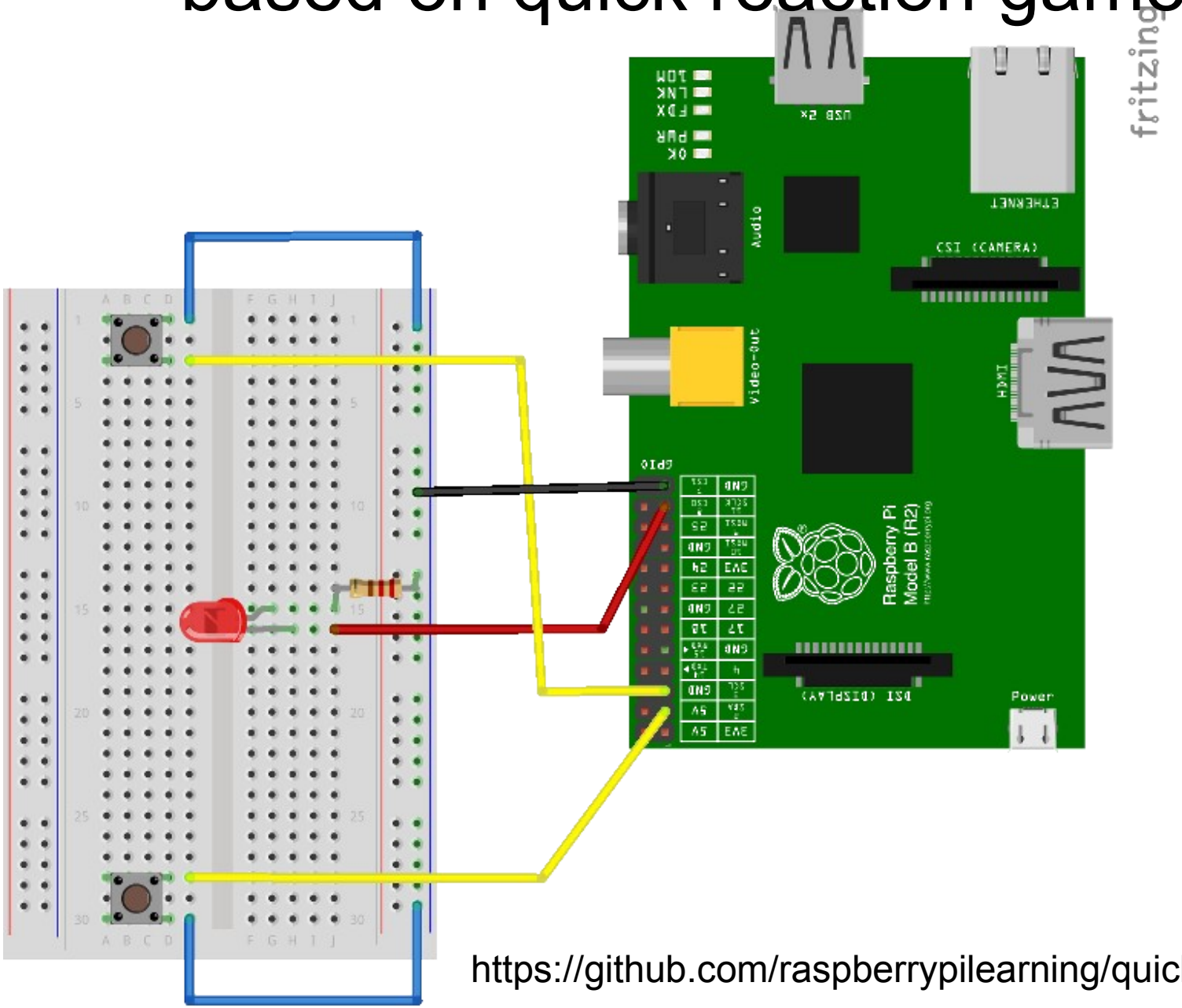
```
sudo python
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
>>> GPIO.setup(2, GPIO.OUT, initial=GPIO.LOW )
>>> GPIO.output(2, True )
>>> GPIO.output(2, False )
>>> GPIO.output(2, True )
>>> GPIO.output(2, False )
```



Resistors (typically about 400 ohm)

Button + LED

based on quick reaction game

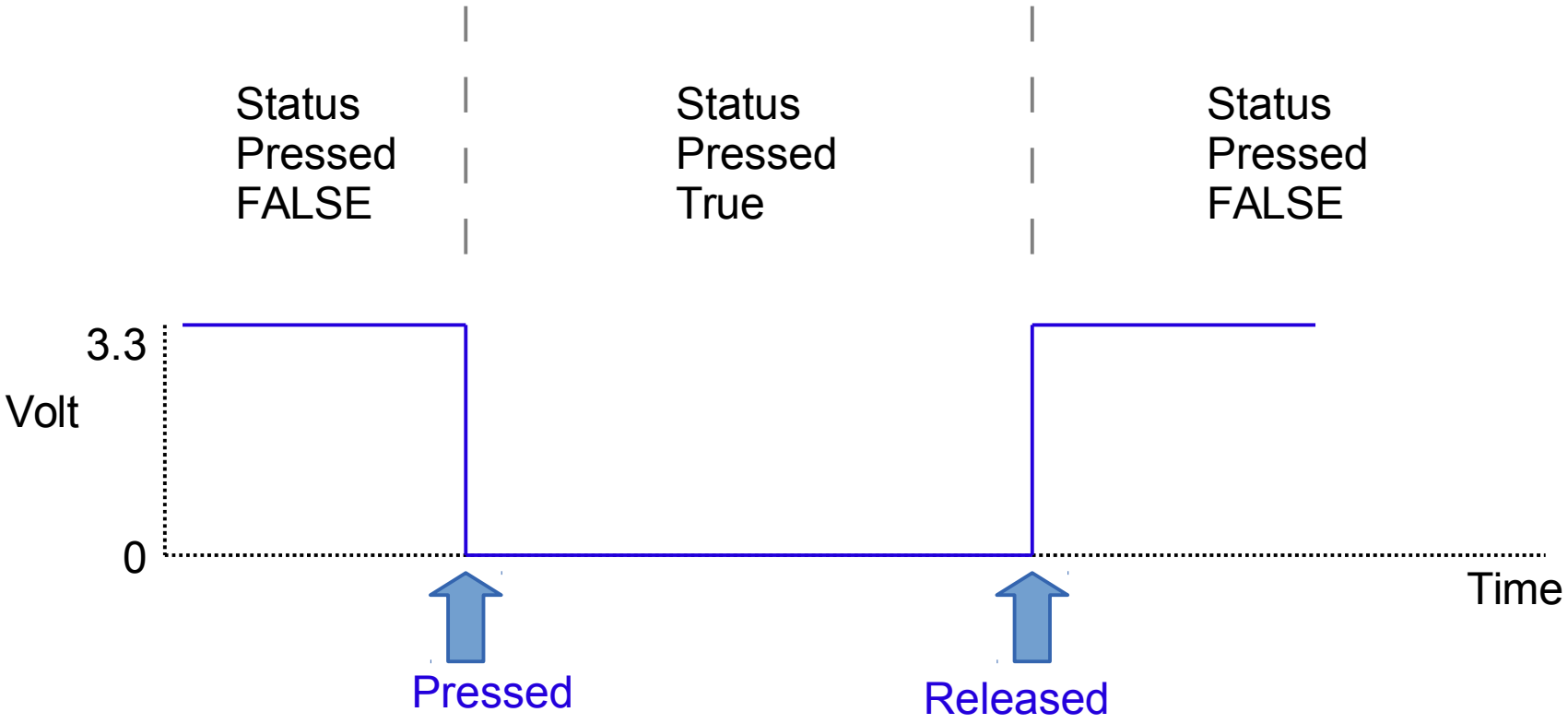


https://github.com/raspberrypilearning/quick-reaction-game

Detect button pressed

```
#!/usr/bin/env python

from time import sleep
import os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN)
GPIO.setup(24, GPIO.IN)
GPIO.setup(25, GPIO.IN)

while True:
	if ( GPIO.input(23) == False ):
		os.system('mpg321 binary-language-moisture-evaporators.mp3 &')
	if ( GPIO.input(24) == False ):
		os.system('mpg321 power-converters.mp3 &')
	if ( GPIO.input(25)== False ):
		os.system('mpg321 vader.mp3 &')
	sleep(0.1);
```

Voltage Status

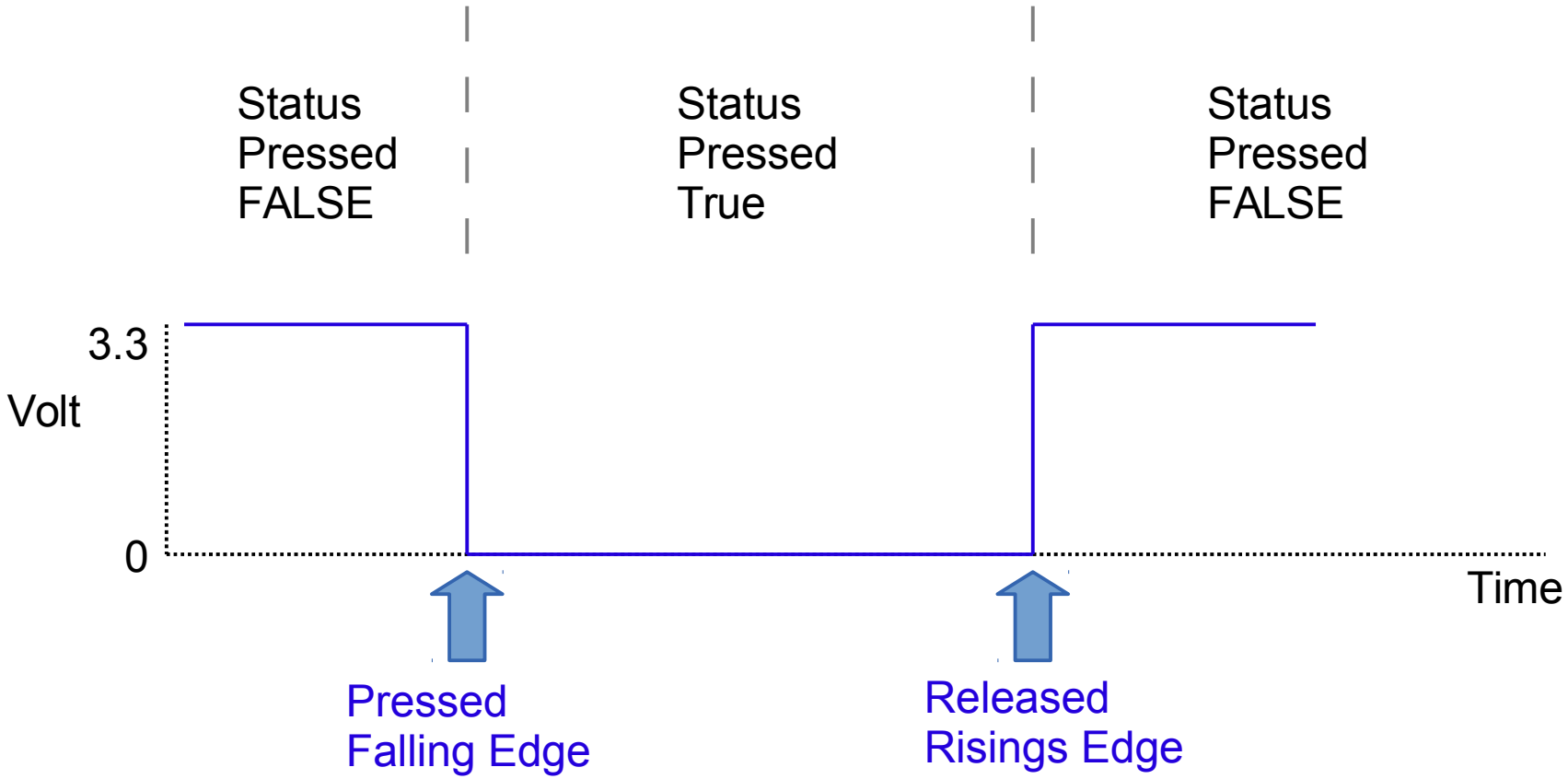


THE PROBLEM WITH POLLING

- This code works, but prints a line for each frame that the button is pressed.
- Luckily, the GPIO library has built in a rising-edge and falling-edge function.
- A rising-edge is defined by the time the pin changes from low to high, but it only detects the change. Similarly, the falling-edge is the moment the pin changes from high to low.

http://makezine.com/projects/tutorial-raspberry-pi-gpio-pins-and-python/

Voltage Status



Edge trigger event

```
#!/usr/bin/env python

from time import sleep
import os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup( 2, GPIO.IN)
GPIO.setup(11, GPIO.OUT, initial=GPIO.LOW )

led_on = False

while True:
    #GPIO.input( 2 ) == False when pressed
    GPIO.wait_for_edge(2, GPIO.FALLING)
    led_on = not led_on
    GPIO.output( 11, led_on )
```

Non blocking - callback

```
#!/usr/bin/env python

from time import sleep
import os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setup( 2, GPIO.IN)
GPIO.setup( 3, GPIO.IN)
GPIO.setup(11, GPIO.OUT, initial=GPIO.LOW )

led_on = False

def postAndroid(channel):
    print 'Push called'

#GPIO.RISING
GPIO.add_event_detect(3, GPIO.FALLING, callback=postAndroid, bouncetime=300)

while True:
    #GPIO.input( 2 ) == False when pressed
    GPIO.wait_for_edge(2, GPIO.FALLING)
    led_on = not led_on
    GPIO.output( 11, led_on )
```

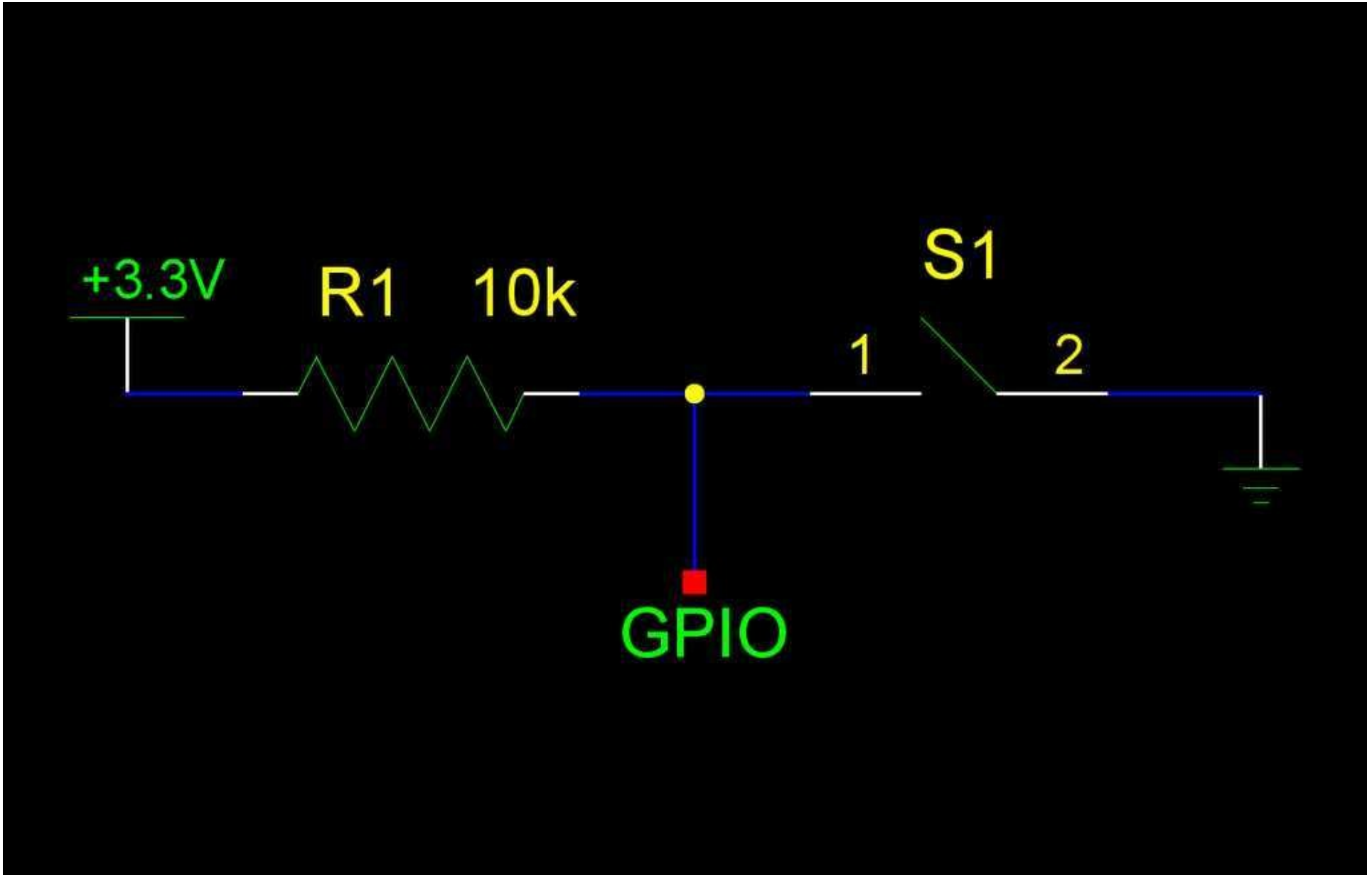
P1 Header Pinout, top row:

Pin Number	Pin Name Rev1	Pin Name Rev2	Hardware Notes	Alt 0 Function	Other Alternative Functions
P1-02	5V0		Supply through input poly fuse		
P1-04	5V0		Supply through input poly fuse		
P1-06	GND				
P1-08	GPIO 14		Boot to Alt 0 ->	UART0_TXD	ALT5 = UART1_TXD
P1-10	GPIO 15		Boot to Alt 0 ->	UART0_RXD	ALT5 = UART1_RXD
P1-12	GPIO 18			PCM_CLK	ALT4 = SPI1_CE0_N ALT5 = PWM0
P1-14	GND				
P1-16	GPIO23				ALT3 = SD1_CMD ALT4 = ARM_RTCK
P1-18	GPIO24				ALT3 = SD1_DAT0 ALT4 = ARM_TDO
P1-20	GND				
P1-22	GPIO25				ALT3 = SD1_DAT1 ALT4 = ARM_TCK
P1-24	GPIO08			SPI0_CE0_N	
P1-26	GPIO07			SPI0_CE1_N	

P1 Header Pinout, bottom row:

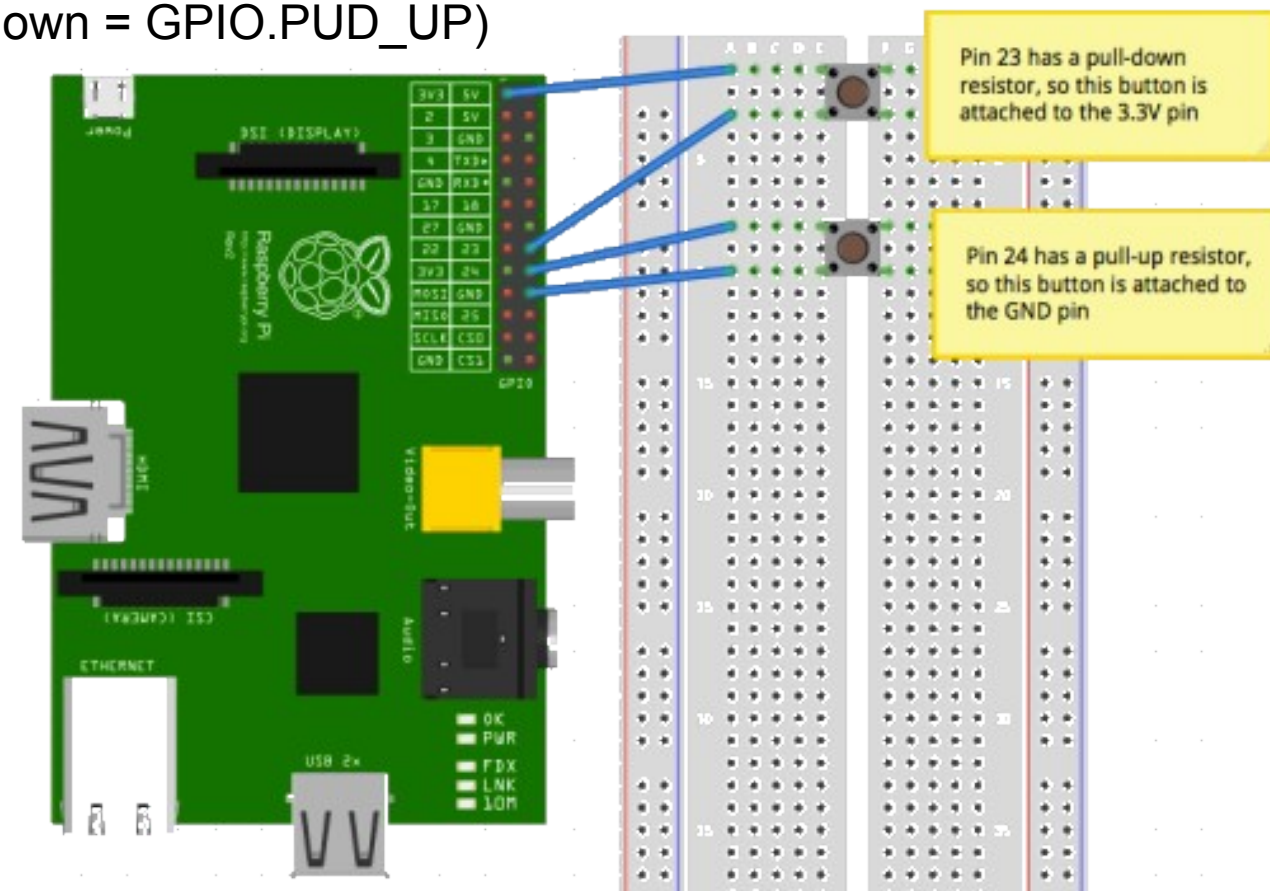
Pin Number	Pin Name Rev1	Pin Name Rev2	Hardware Notes	Alt 0 Function	Other Alternative Functions
P1-01	3.3 V		50 mA max (01 & 17)		
P1-03	GPIO 0	**GPIO 2**	1K8 pull up resistor	I2C0_SDA / I2C1_SDA	
P1-05	GPIO 1	**GPIO 3**	1K8 pull up resistor	I2C0_SCL / I2C1_SCL	
P1-07	GPIO 4			GPCLK0	ALT5 = ARM_TDI
P1-09	GND				
P1-11	GPIO17				ALT3 = UART0_RTS ALT4 = SPI1_CE1_N ALT5 = UART1_RTS
P1-13	GPIO21	**GPIO27**		PCM_DOUT / reserved	ALT4 = SPI1_SCLK ALT5 = GPCLK1 / ALT3 = SD1_DAT3 ALT4 = ARM_TMS
P1-15	GPIO22				ALT3 = SD1_CLK ALT4 = ARM_TRST
P1-17	3.3 V		50 mA max (01 & 17)		
P1-19	GPIO10			SPI0_MOSI	
P1-21	GPIO9			SPI0_MISO	
P1-23	GPIO11			SPI0_SCLK	
P1-25	GND				

Button with Pull-up resistor

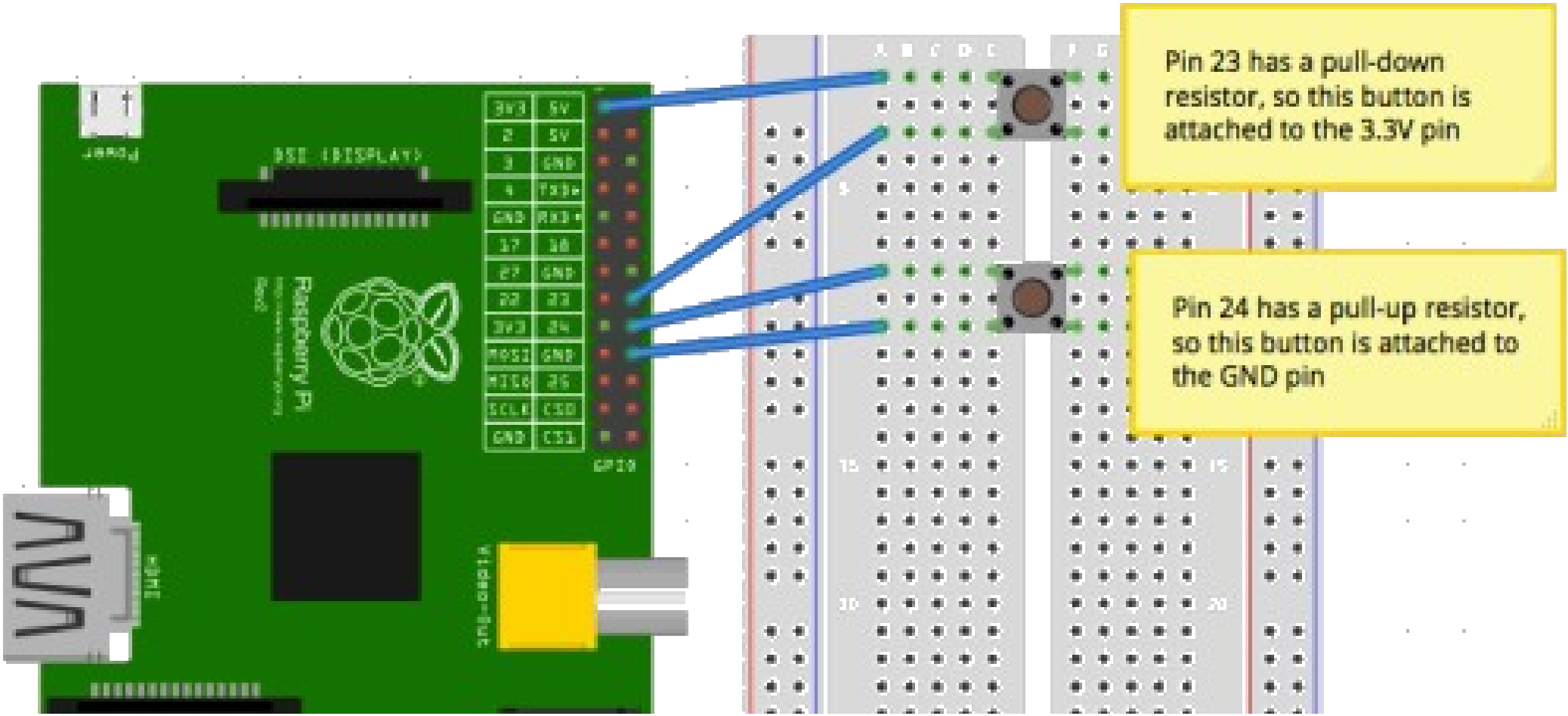


Input 2 buttons

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

GPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)

while True:

    if(GPIO.input(23) ==1):

        print("Button 1 pressed")

    if(GPIO.input(24) == 0):

        print("Button 2 pressed")

GPIO.cleanup()
```



Input 2 buttons



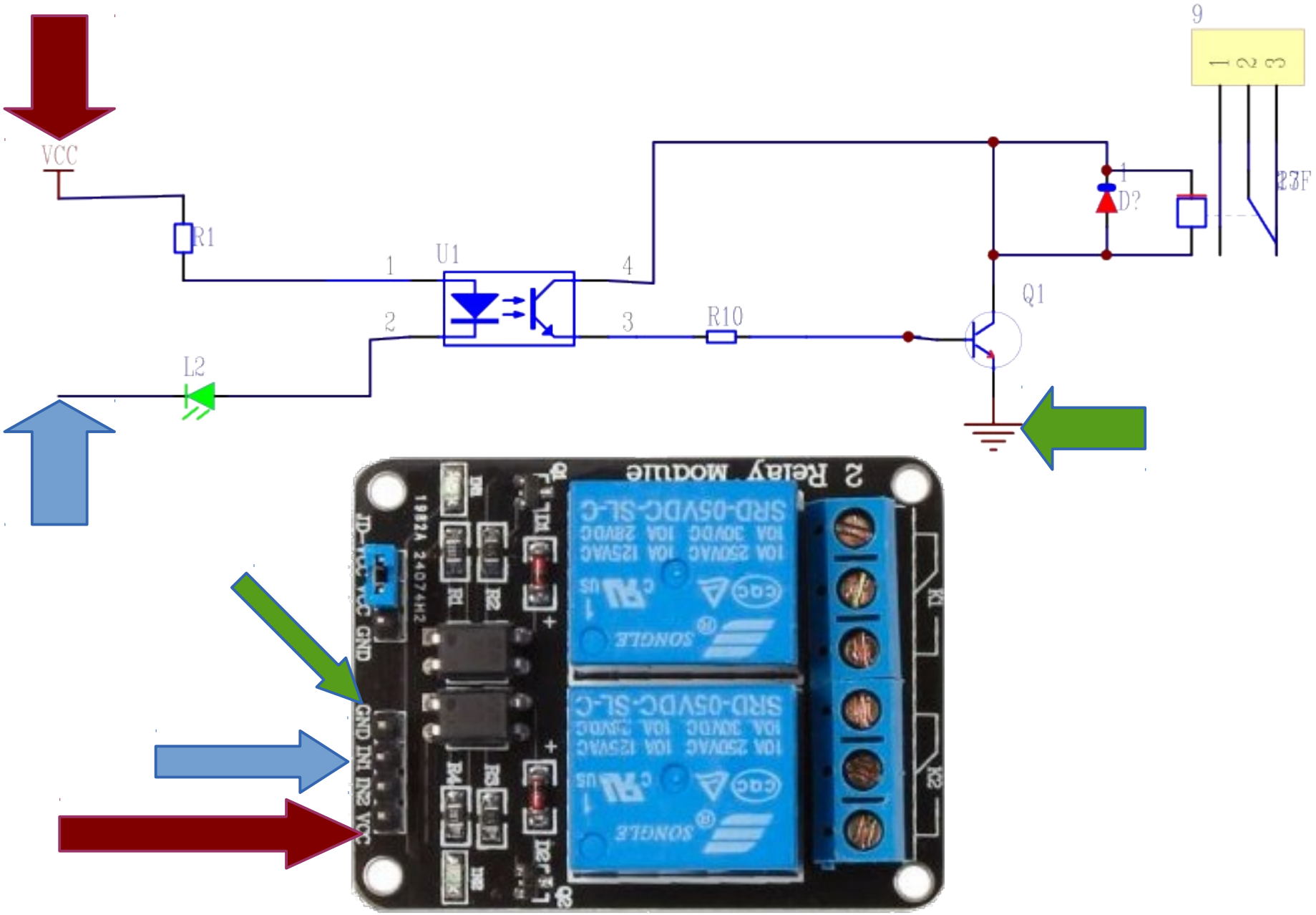
http://makezine.com/projects/tutorial-raspberry-pi-gpio-pins-and-python/

Input 2 buttons

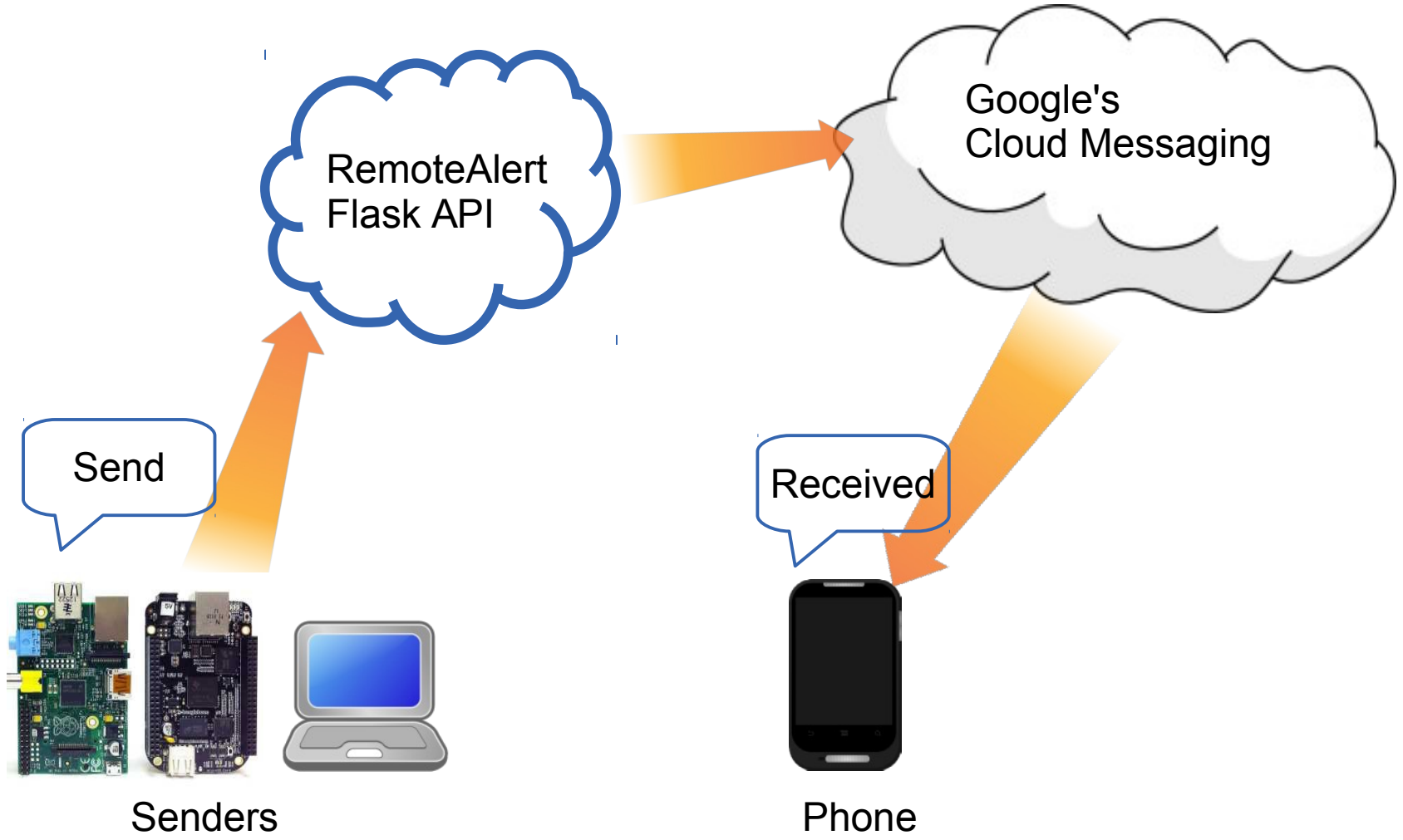
Events

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
GPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)
def printFunction(channel):
    print("Button 1 pressed!")
    print("Note how the bouncetime affects the button press")
GPIO.add_event_detect(23, GPIO.RISING, callback=printFunction, bouncetime=300)
while True:
    GPIO.wait_for_edge(24, GPIO.FALLING)
    print("Button 2 Pressed")
    GPIO.wait_for_edge(24, GPIO.RISING)
    print("Button 2 Released")
GPIO.cleanup()
```

Relay Board Circuit



Remote Alert



Non blocking - callback

```
#!/usr/bin/env python

from time import sleep
import os
import RPi.GPIO as GPIO
from remotealert import RemoteAlert

GPIO.setmode(GPIO.BCM)
GPIO.setup( 2, GPIO.IN)
GPIO.setup( 3, GPIO.IN)
GPIO.setup(11, GPIO.OUT, initial=GPIO.LOW )

led_on = False
ra = RemoteAlert()
dev_id = 'a32b831c-7623-4c43-99cf-b614ff54e902'

def postAndroid(channel):
    ra.send( dev_id, 'Message from pi' )
    print 'Push called'
#GPIO.RISING
GPIO.add_event_detect(3, GPIO.FALLING, callback=postAndroid, bouncetime=300)

while True:
    #GPIO.input( 2 ) == False when pressed
    GPIO.wait_for_edge(2, GPIO.FALLING)
    led_on = not led_on
    GPIO.output( 11, led_on )

#Turn it off to leave it off
GPIO.output( 11, False )
GPIO.cleanup()
```

Adam Savage's 10 Commandments for Makers

1. Make something. Anything
2. Make stuff that improves your life, either mechanically or aesthetically
3. Don't wait
4. Use a project to learn a skill
5. ASK. Ask for help
6. Share your methods and knowledge and don't make them a secret
7. Discouragement and failure are intrinsic to the process
8. Measure carefully
9. Make things for other people
10. Use more cooling fluid