# Disclaimer

- Any opinions or viewpoints expressed do not necessarily reflect those of IEEE, IEEE GOLD, UMIEEE, or any of our sponsors
- Any software provided or mentioned is for educational purposes only
- You break it, you buy it
  - Just kidding, you already bought it

# The Agenda (1/2)

- Introductions and parts hand-out
- SD Card flashing procedure, setup walk-through
- Raspberry Pi intro (while we flash SD cards)
  - Hardware overview
  - Software (OS) overview
  - Community Support
- Loading up the Desktop (GUI)
- Command line fun – installing software for our workshop

IEEE

# The Agenda (2/2)

- Programming example 1 – “Hello World”
- Break for Pie & demonstrations!
- Programming example 2 – Blinking Lights
- Programming example 3 - The MCP3001 SPI ADC
- Arduino interaction
- Various demos

# What is not covered

- In-depth linux material
  - We will guide you through examples, however to do this topic justice would require a series of workshops
- Programming techniques
  - Will give insights and explanations relevant to examples only, again in the interests of time
- Circuit theory
  - Again, insights relevant to the material covered today only
- Arduino use
  - We will get a basic example working, but that’s about it

# What is not covered

- As much as I would like to teach all of this extra material, this is a “Raspberry Pi” event
  - UMIEEE regularly throws workshops that cover these topics
- If questions stray too far from the prepared material, I will defer them to after the allotted time (9:30)
- Will stay beyond 9:30 if there is interest
- Feedback is appreciated!

# Introductions – get your parts!

- Presenters: Troy Denton, Ahmad Byagowi
- Honorable mentions:

- MTS
- Memory Express
- Visions Electronics
- UMIEEE
- IEEE GOLD
- The Raspberry Pi Foundation! ( http://raspberrypi.org)

# Setup

- Log on to the windows machine in front of you
  - Password is “ece”
- Go to the following website: http://troydenton.ca/rpi.html
- Unzip the UMRPImg_4G.zip that is on your desktop
- If it is not there, download it from http://troydenton.ca/UMRPImg_4G.zip

# Setup – flashing the card

## Windows

- Extract win32diskimager
- Right click win32diskimager, run as administrator
- “Browse” to the extracted rpi.img file

## *nix

- Figure out /dev/ path for your SD card
- dd bs=4M if=/path/to/rpi.img of=/dev/<sd card>

IEEE

# Setup – flashing the card

## Windows

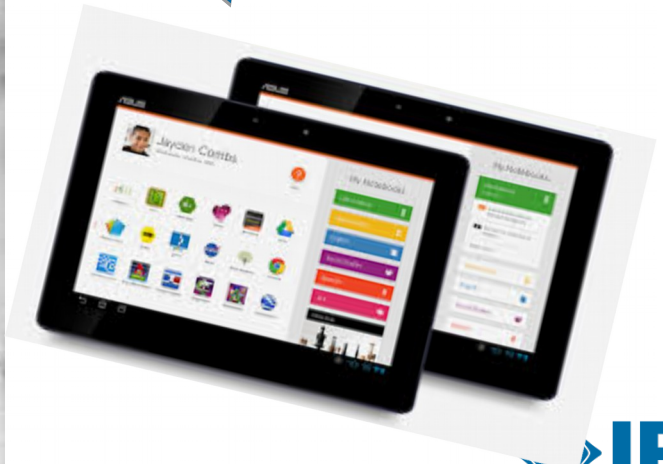
- Select the proper device in the “Device” drop down.
- On lab machines this should be device E:/ or G:/
- Click “Write”
- Be careful!!

## *nix

- Figure out /dev/ path for your SD card
- dd bs=4M if=/path/to/rpi.img of=/dev/<sd card>

IEEE

# Raspberry Pi - Philosophy



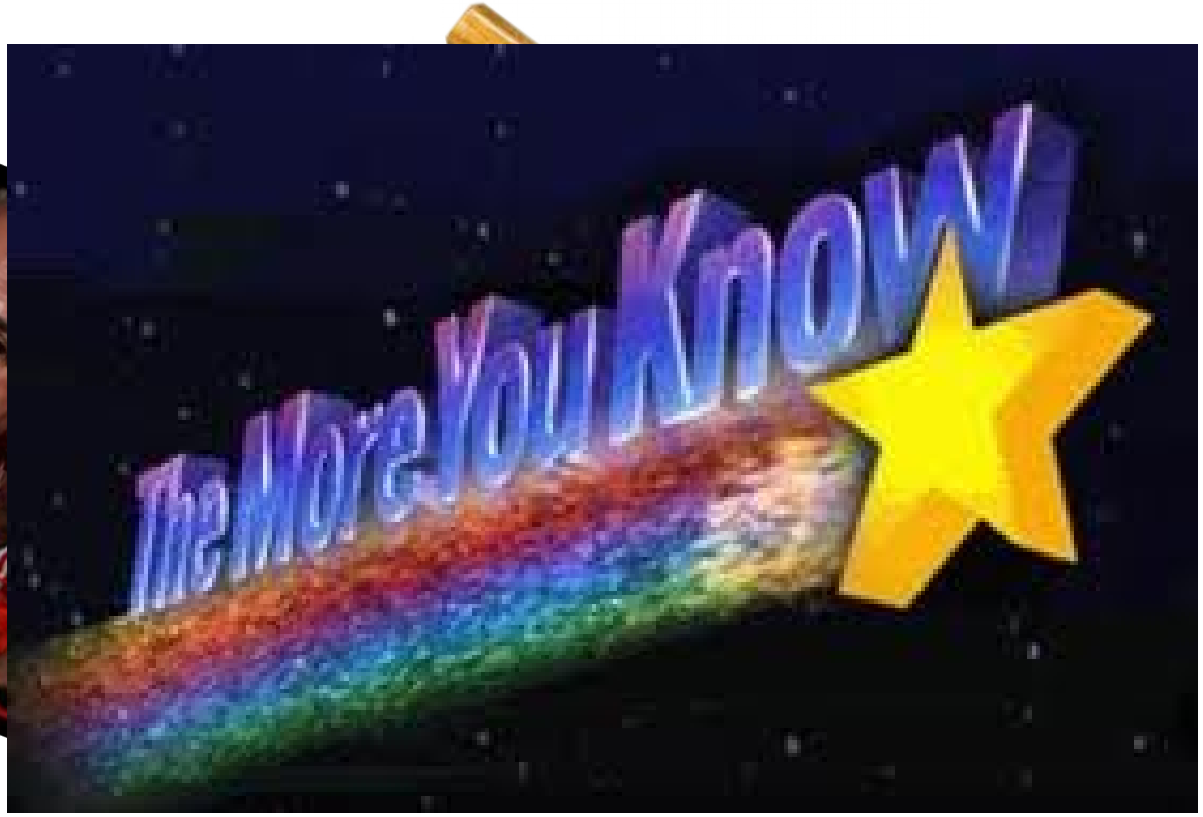

IEEE

# Raspberry Pi - Philosophy

IEEE

# Raspberry Pi - Philosophy

# Raspberry Pi - Philosophy



# Raspberry pi - Philosophy

Σ(

)=

BROADCOM

BCM2835 ARM Peripherals

IEEE

# Raspberry pi - Philosophy

- raspberrypi.org forums:
  - Just shy of 60,000 members
- http://www.raspberrypi.org/archives/tag/educational-manual
- http://www.adafruit.com/blog/category/raspberry-pi/
  - The latest and greatest raspberry pi projects and tutorials
- Amazing online presence (stats as of 2013-03-13)
  - 15,000,000 results for "raspberry pi" – launched 2012
  - 27,000,000 results for "arduino" – launched 2005

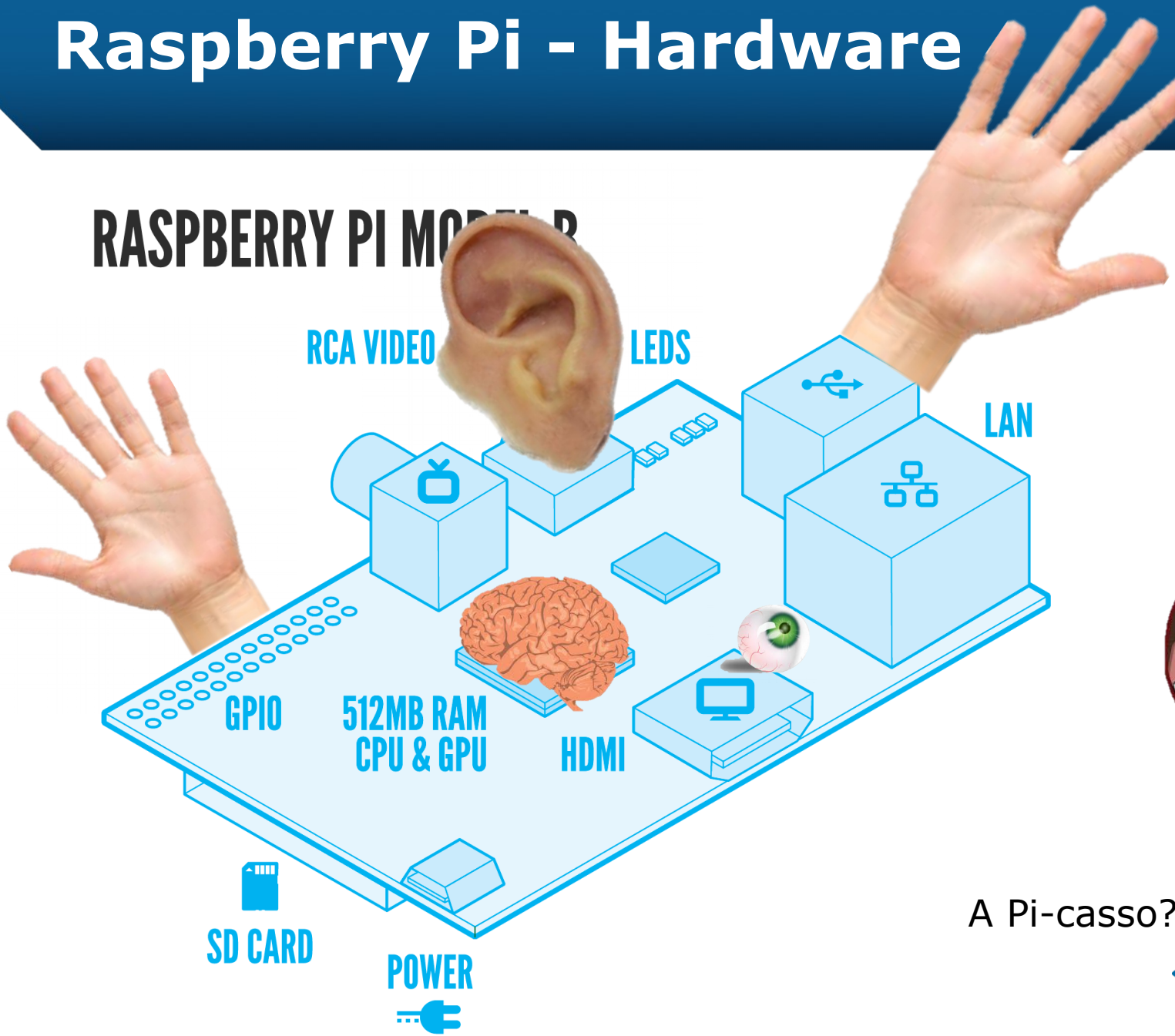
# Raspberry pi - Philosophy

UM IEEE

IEEE GOLD
Graduates Of The Last Decade

IEEE

# Raspberry Pi - Hardware




A Pi-casso?

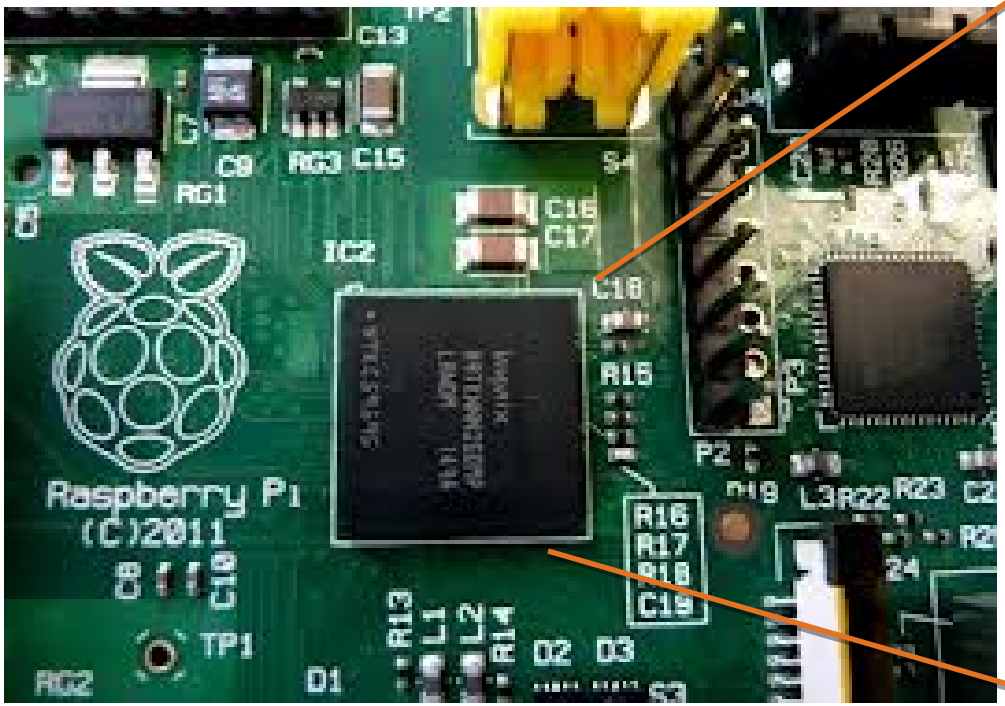
IEEE

# Raspberry Pi – Hardware

# Raspberry Pi – Hardware

Broadcom BCM2835

CPU core is an ARM11




GPU

+

CPU

+

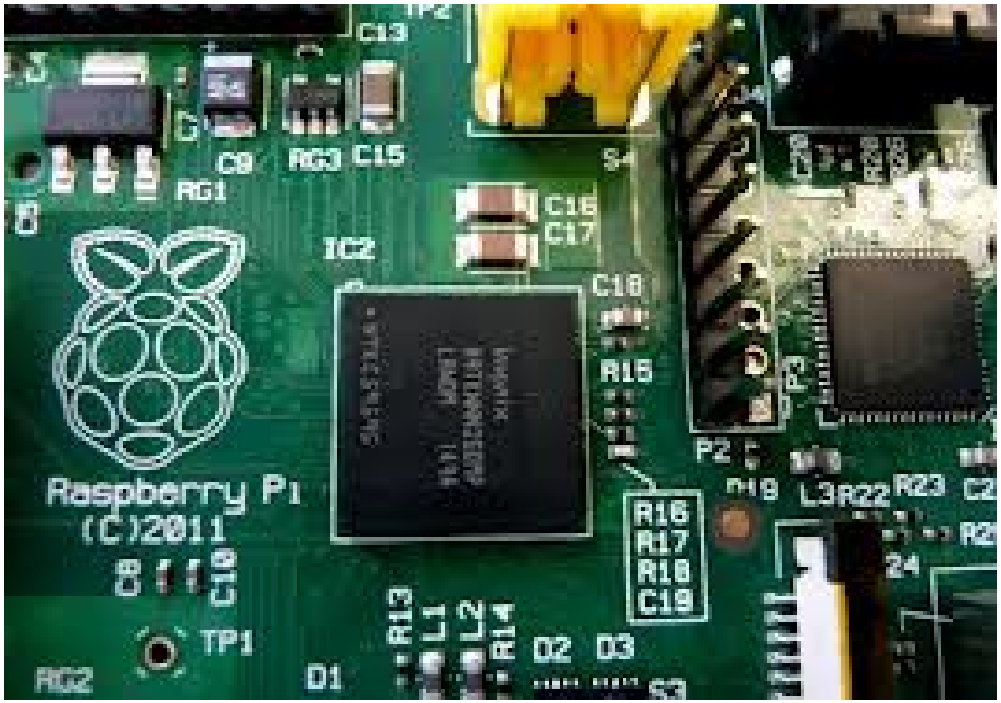
RAM


IEEE

# Raspberry Pi – Hardware

Broadcom BCM2835

CPU core is an ARM11




RAM: 256MB (Model B Rev1)
512 MB (Model B Rev2)

CPU Speed: can technically go up to 1GHz

Contains all logic for serial buses (USB, SPI, I2C)

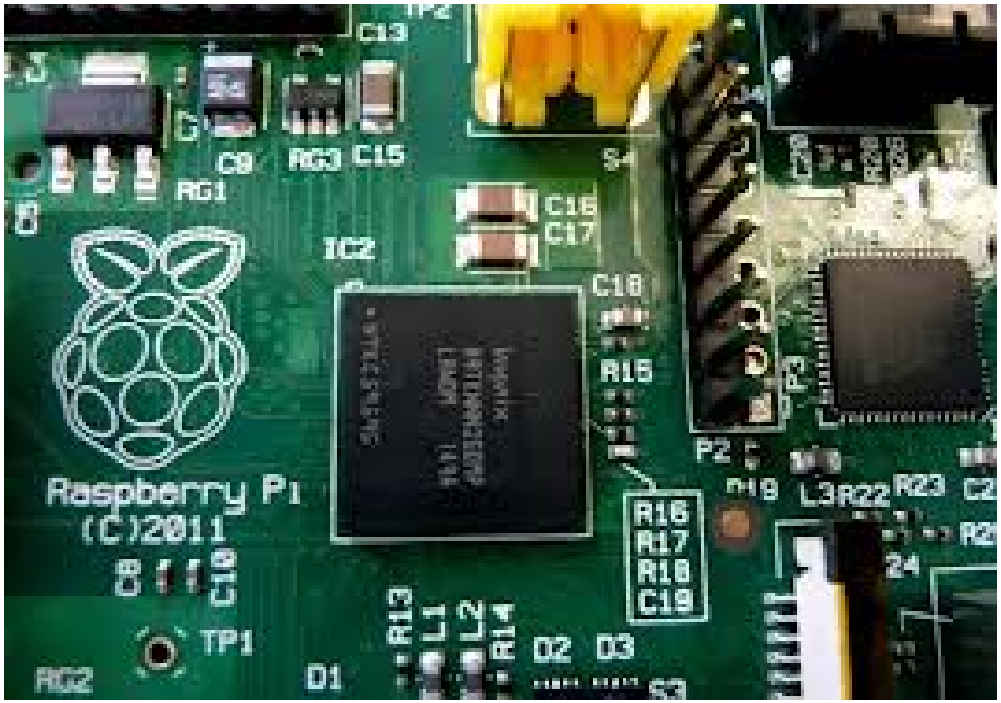
GPIO pins go _directly_ to the CPU

IEEE

# Raspberry Pi – Hardware

Broadcom BCM2835

CPU core is an ARM11

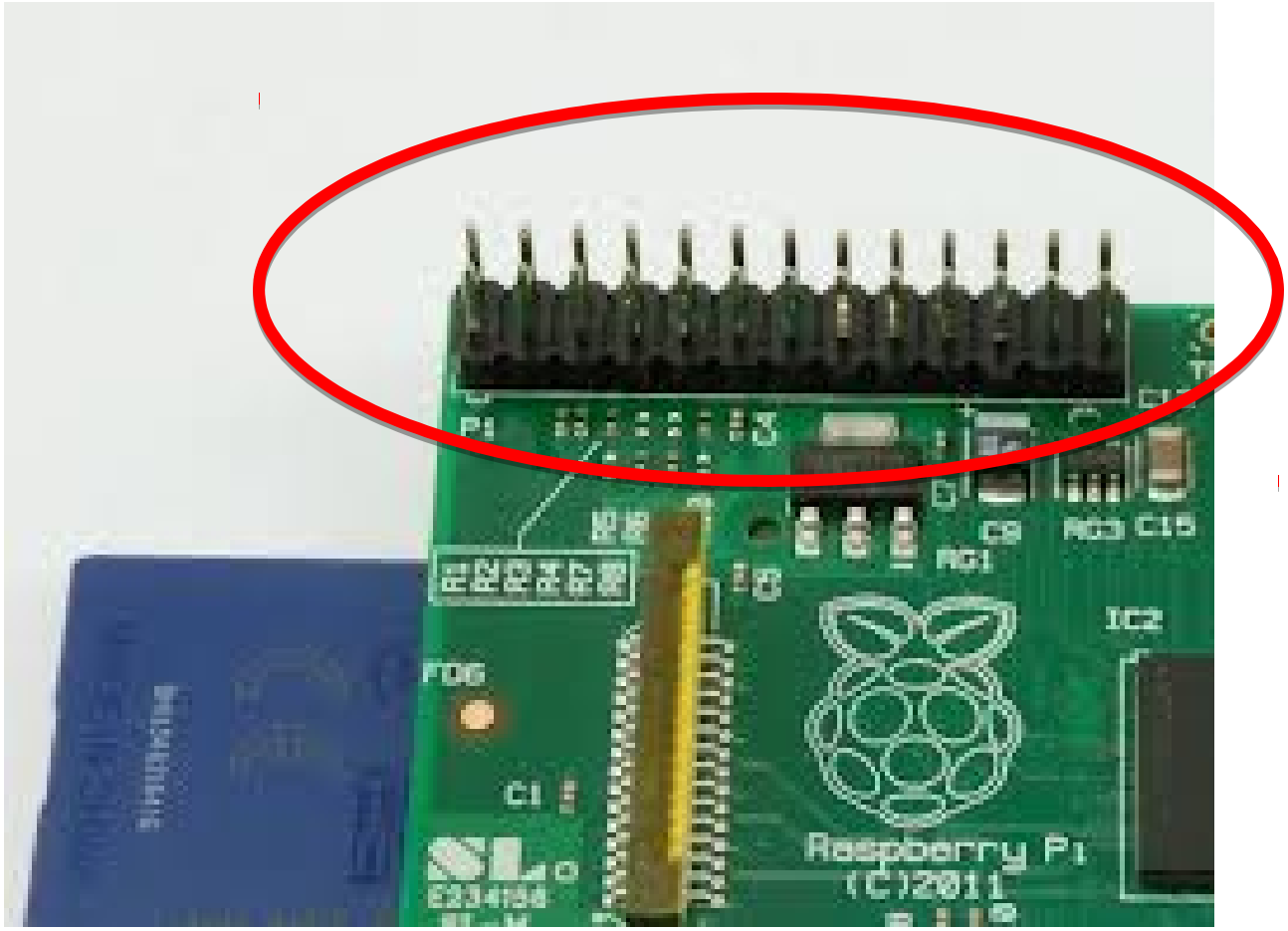



Can overclock it via software settings

CPU and GPU each have their own memory allocation – where the "memory split" occurs is configurable in software

GPU needs a minimum 16MB

# Raspberry Pi - Hardware



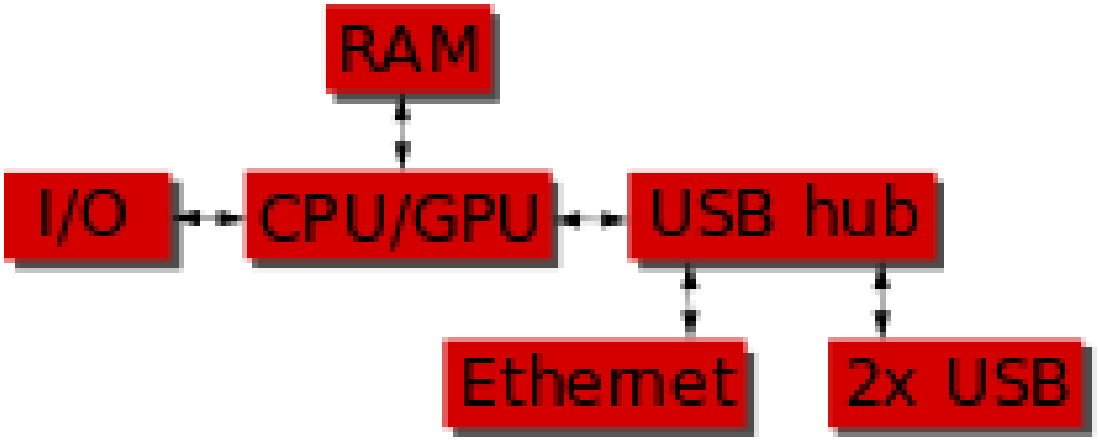

**Digital inputs and outputs only! No analog!**

# Raspberry Pi - Hardware

# Raspberry Pi - Hardware

USB to Ethernet Adapter

# Raspberry Pi - Hardware



# Raspberry Pi - Software

- Designed with linux systems in mind, not necessarily required
  - RISCOS
  - Arch linux
  - Fedora
  - Occidentalis
  - Raspbian (our distro of choice)
  - A slew of others….

# Raspberry Pi - Software

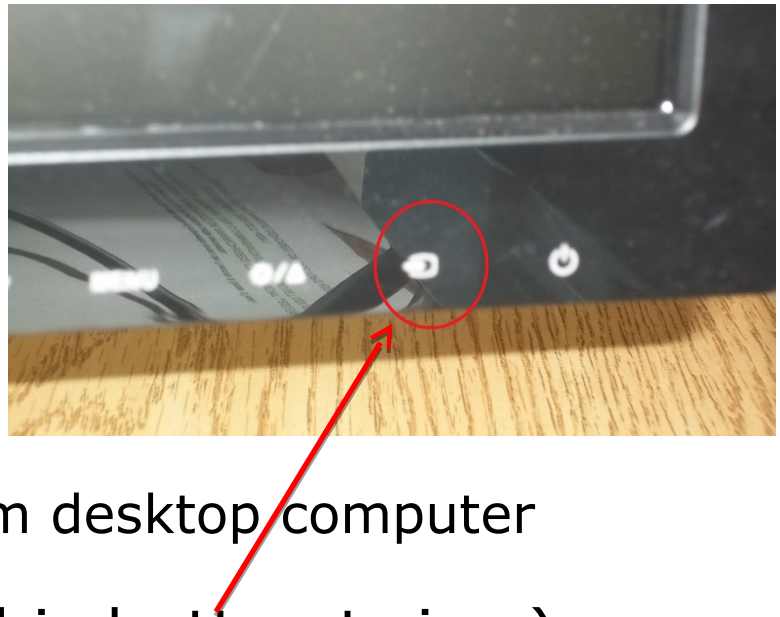
- Raspbian is what is known as the “Operating System” (OS)
  - Similar to windows running on your home PC, the raspberry pi requires an operating system to work
  - Raspbian is a special-purpose distribution based on Debian (a variant of linux)
- The OS is what lets the raspberry pi not only run programs, but run multiple programs concurrently
- It accomplishes this with a “scheduler” in the OS
  - Simply put, this makes programs “take turns”
  - Only one program is actually executing at any one time

# Raspberry Pi - Software

- The SD card contains:
  - The OS
  - All Programs on the Raspberry pi
  - All permanent (nonvolatile) storage, in general

# Loading up the OS

- Plug in the Network cable
- Plug in your SD card
- Plug in your HDMI cable
- Plug in the USB hub
  - Plug in mouse and keyboard from desktop computer
- Set monitor to hdmi (press this button twice )
- Plug in usb power (act fast)
- The Raspberry pi automatically boots off of the SD card



# Loading up the OS

- You should see the raspberry pi logo, and a bunch of computer jargon printing to the screen
- This is the linux kernel in action!
- When it is finished, you will be prompted to login.
  - Username: pi
  - Password: raspberry

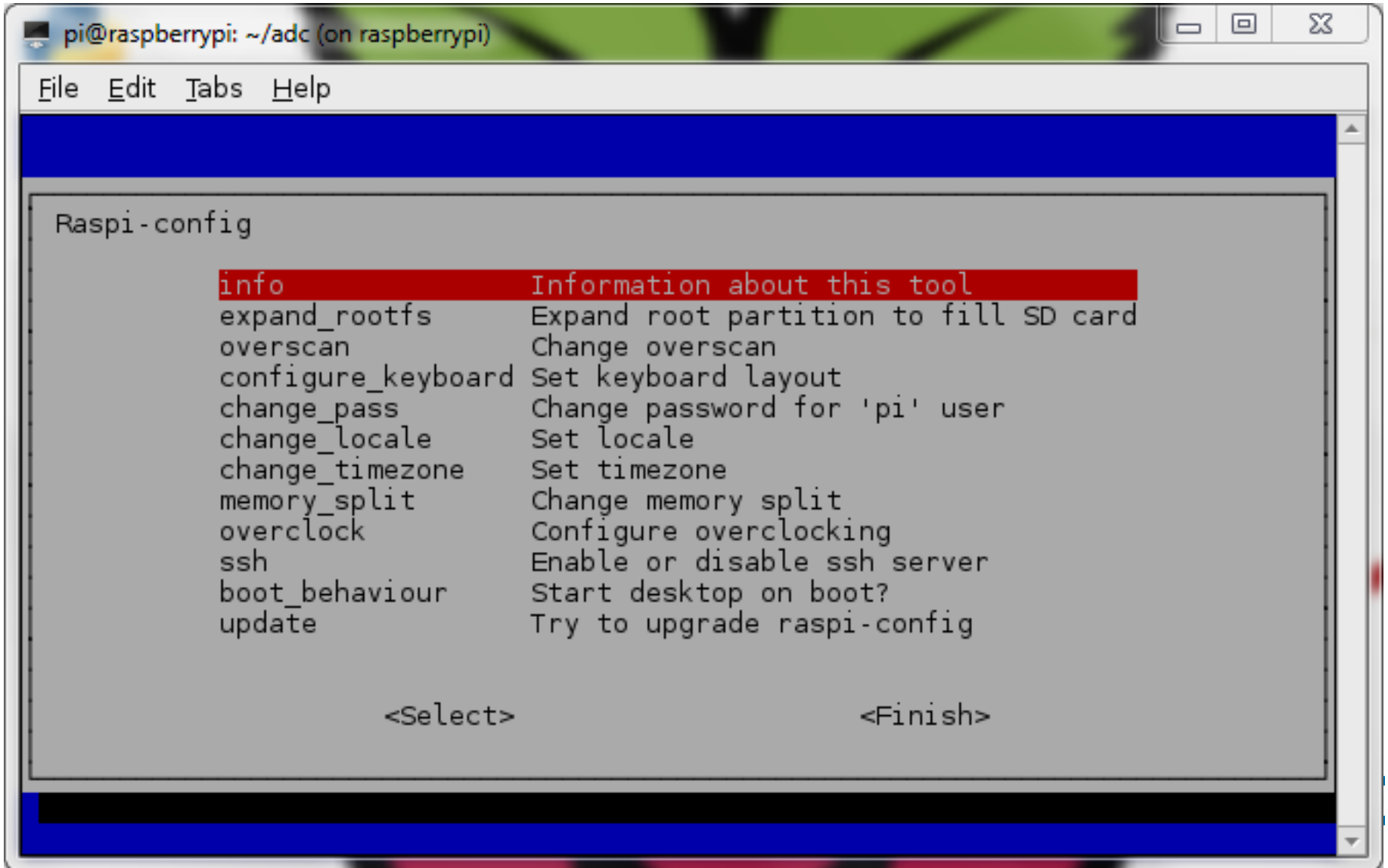
# Loading up the OS

- You are now at the system prompt
- Commands to try:
  - ls
    - ls –a,  ls –l,
    - Lists the files in the directory
  - cd <directory>
    - Move to directory <directory>
    - Enter “cd” without a directory to return “home”
  - top
    - This shows all of the programs running

# Loading up the OS

- Let’s take full advantage of our SD card: type *sudo raspi-config* and hit enter
  - This is a program you can use to configure the raspberry pi
  - Select “expand_rootfs”

# Loading up the OS



# Loading up the OS

- To load up the graphical desktop, type *startx* and hit enter
- LXDE is the "Lightweight X Desktop Environment"
  - In linux, you can use different programs to provide a graphical interface
    - GNOME
    - KDE
    - LXDE
    - XFCE
    - Etc…
- Many programs have been preinstalled, look around!

# Loading up the OS

- Midori – internet browser
- OpenOffice.org – Open source alternative to Microsoft Office, “compatible”
- XBMC – Xbox media Center
  - Demo on this later
- Roughly equivalent to a 300MHz *Pentium* 2

# Loading up the OS

- When we're done looking around, close your open programs (for performance purposes)
- Double click the "LXTerminal" icon on the desktop
  - This will open a small terminal window

# Command line fun

- In the console window, enter the following:
- cd
- git clone git://github.com/doceme/py-spidev.git
- sudo apt-get install python-dev
  - Will take a few minutes
- cd py-spidev
- make
- sudo make install

IEEE

# Command line fun

- What did we just do?
  - cd
    - Command used to change directories
  - git clone git://github.com/doceme/py-spidev.git
    - Download sourcecode from a git repository
  - sudo apt-get install python-dev
    - Download and install development libraries from our package manager
  - make
    - Compile the sourcecode that we got via git into usable programs
  - sudo make install
    - Install these programs into system directories

# Command line fun

- sudo nano /etc/modprobe.d/raspi-blacklist.conf

Change the file so it looks like:

#blacklist spi-bcm2708

#blacklist i2c-bcm2708

Use arrow keys to move around

Then hit CTRL+O to save, enter to confirm

CTRL+X to exit

# Command line fun

- To enable the SPI drivers, we need to restart the system
  - sudo shutdown –r now
  - More on SPI later
- We are now done installing software!

# Programming example 1 – "Hello World"

- Without further ado, let's fire up geany and write our first program on the raspberry pi!
- For our purposes today, we will be coding in Python.

# Programming example 1 – "Hello World"

- Log in, "startx" if you havn't already
- Open up geany (LXDE icon/start menu > programming > geany)
- Enter:

```
print "Hello, World!"

print "This is our first"

print "Python example!"
```

# Programming example 1 – "Hello World"

- Save this as helloworld.py in your "home" directory
- To run the program, we need to go back to our command window. (open if you havn't yet)
- Enter: cd
- Enter the command: python helloworld.py
- Congratulations, You're a nerd now

# Programming example 2 – blinking lights

- “Hello world” is the first step in programming traditional computers
- Similarly, blinking an LED is the first step in the microcontroller world
- This is accomplished through the GPIO pins!
- GPIO: General Purpose Input Output

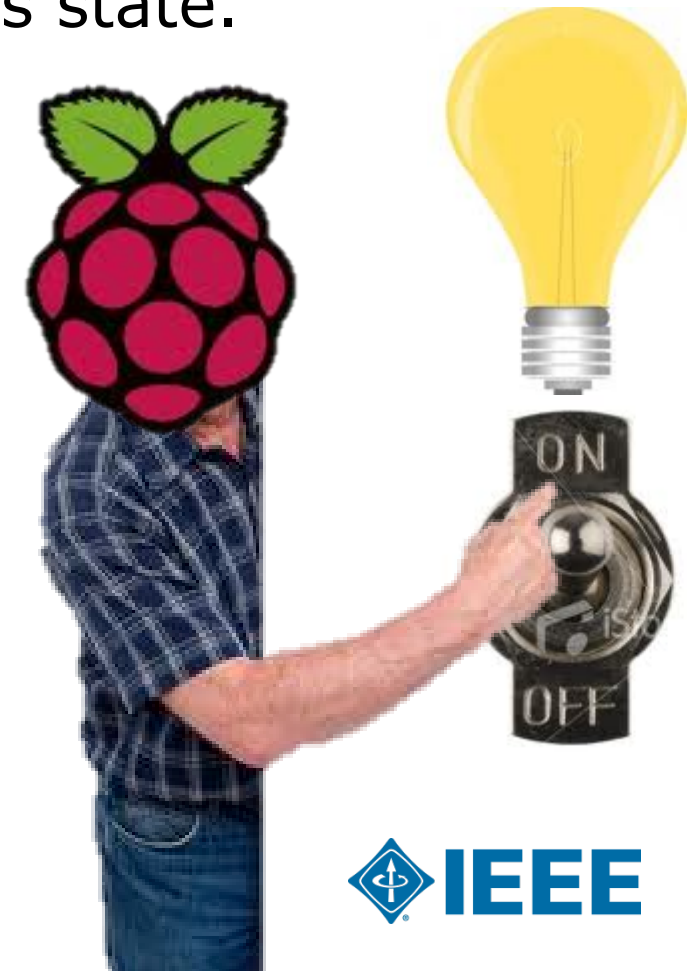
# Programming example 2 – blinking lights

- GPIO pins are special connections to the processor that can be controlled by code
- Instead of your program telling the raspberry pi to put words on the screen, you can tell the raspberry pi to turn a switch on or off

# Programming example 2 – blinking lights

- Each GPIO pin is either switched “on”, or “off”. We “call” a function to change its state.

*Hey! Turn it on!*




IEEE

# Programming example 2 – blinking lights

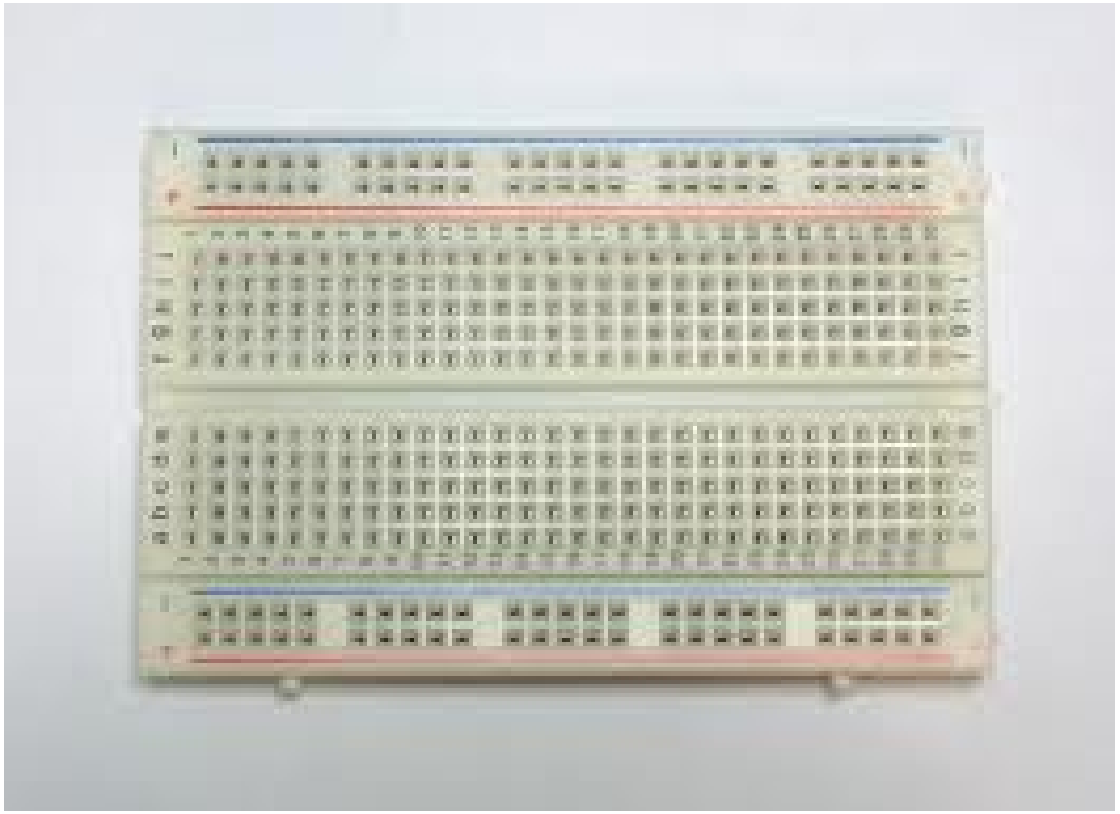
- Each GPIO pin is either switched “on”, or “off”. We “call” a function to change its state.

*Turn it off now!*


IEEE

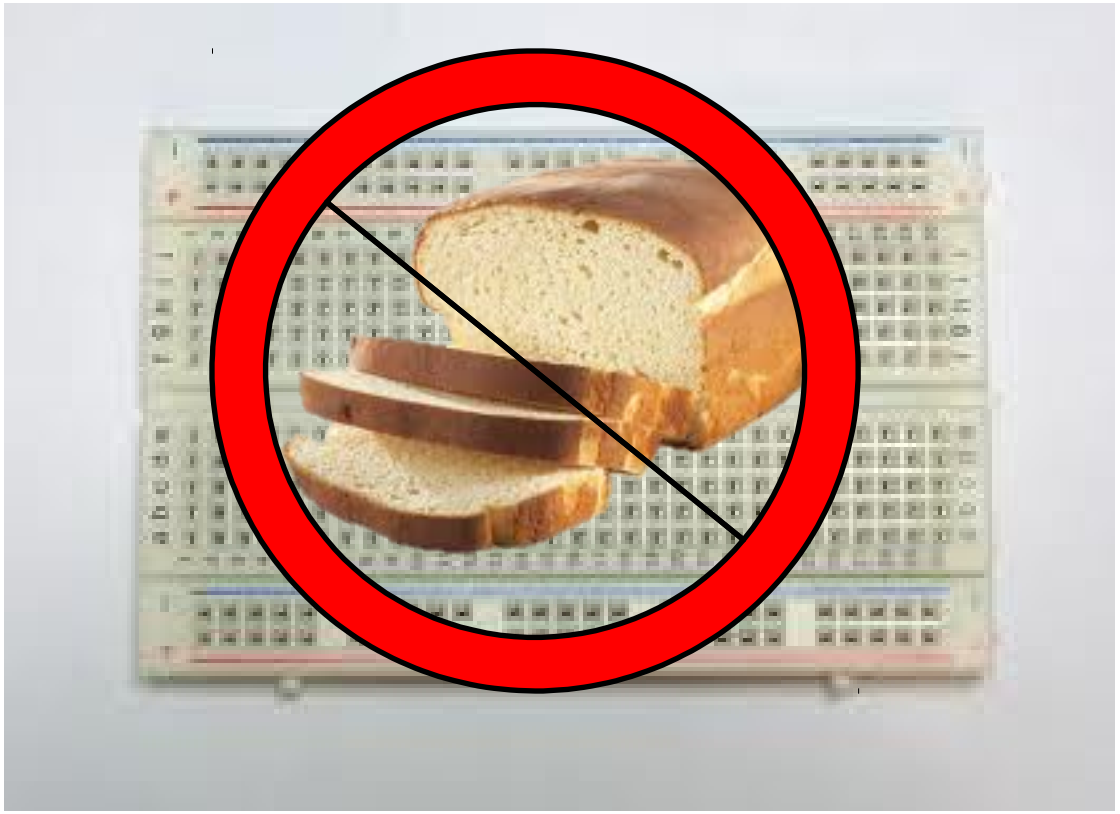
# Programming example 2 – blinking lights

- Enter, the breadboard:



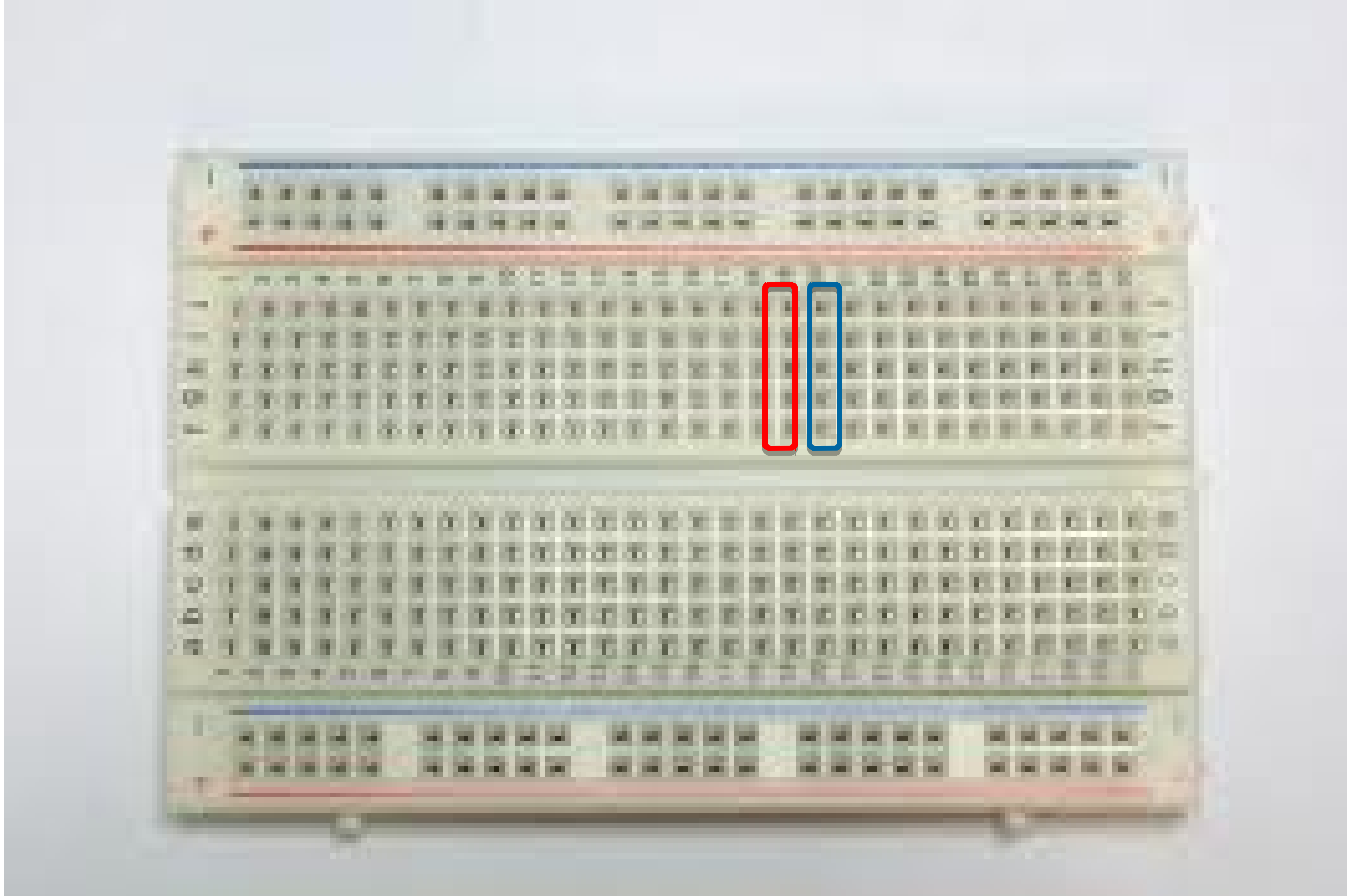
# Programming example 2 – blinking lights

- Little to do with bread



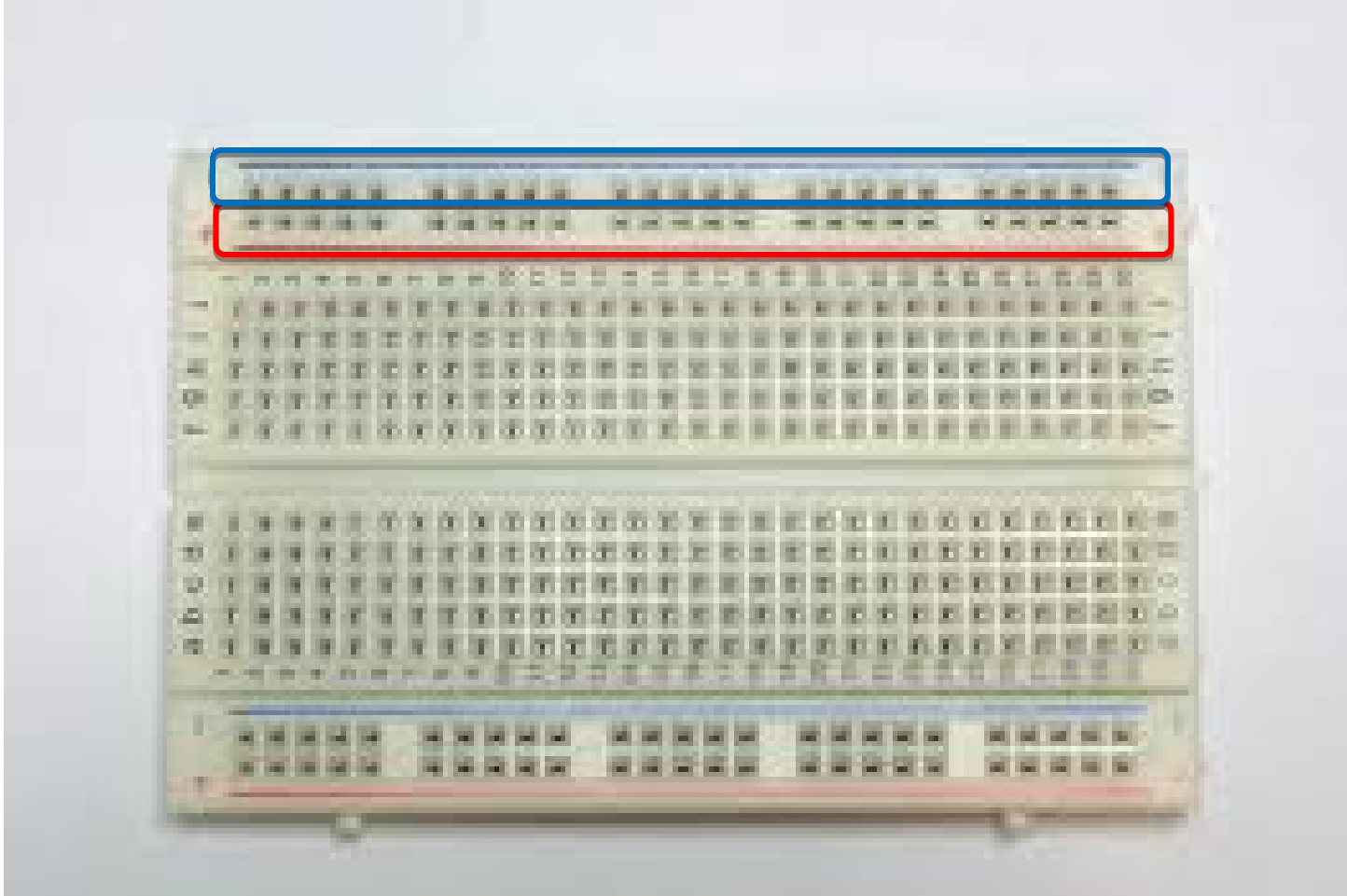
# Programming example 2 – blinking lights

- Rows of adjacent holes are connected together



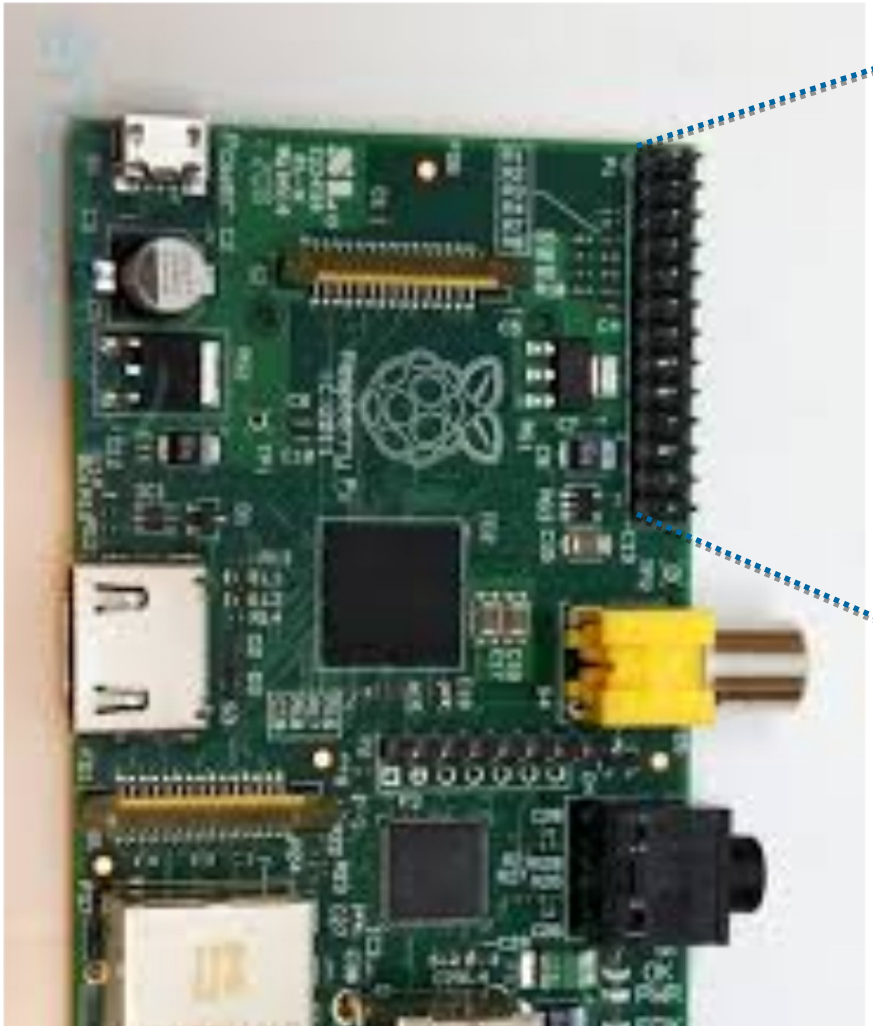
# Programming example 2 – blinking lights

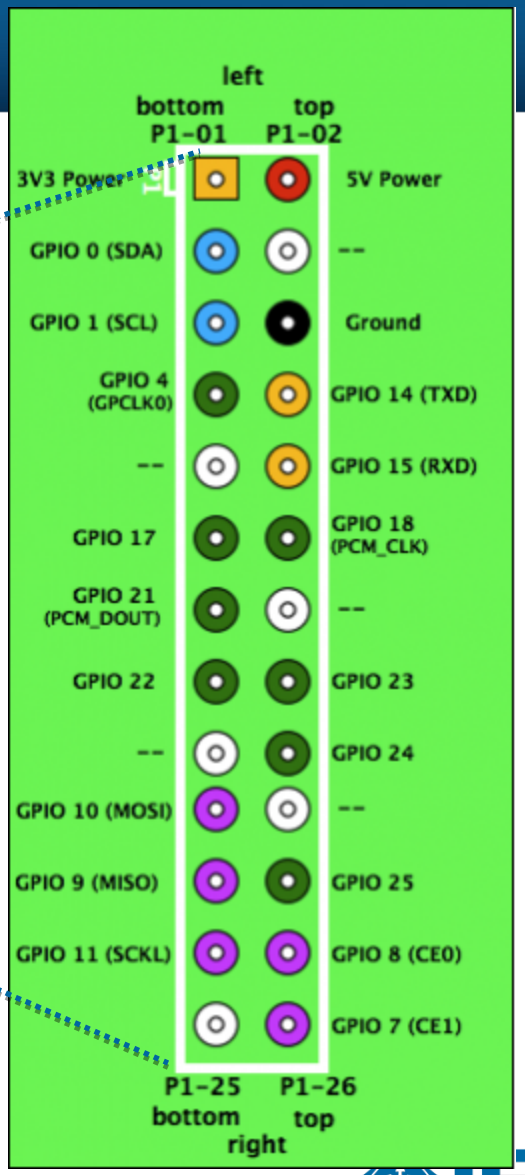
- The outer “strips” are connected as well



# Programming example 2 – blinking lights

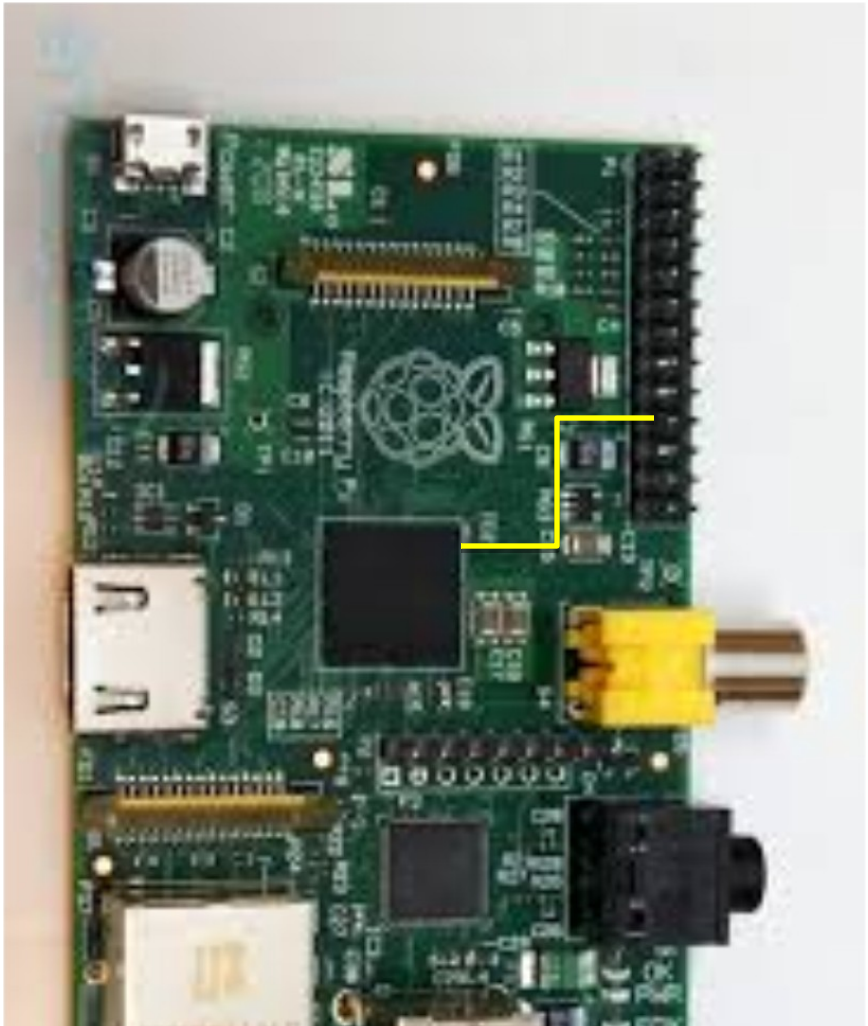
Pin numbering: BCM style

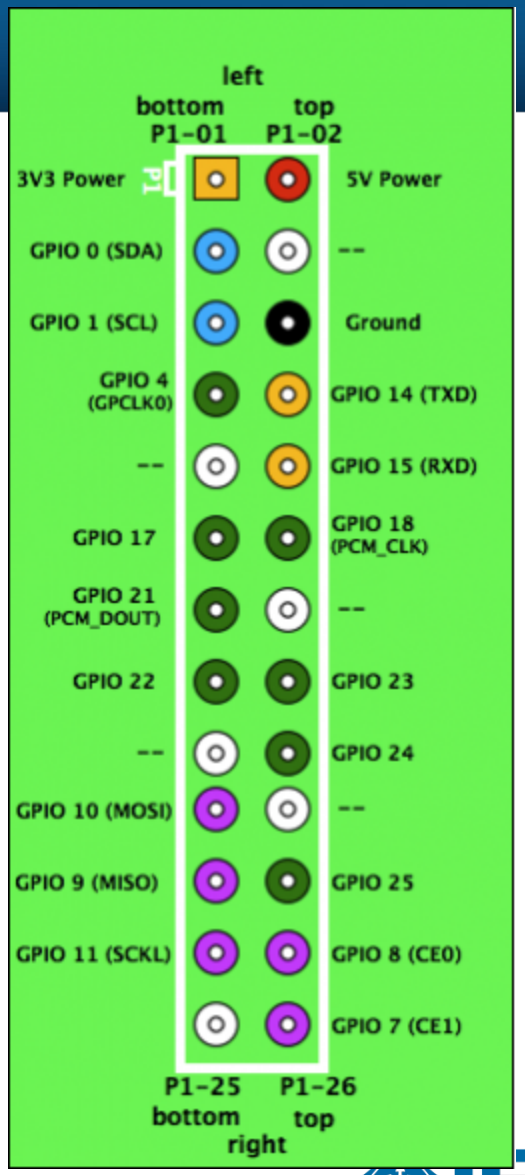




# Programming example 2 – blinking lights

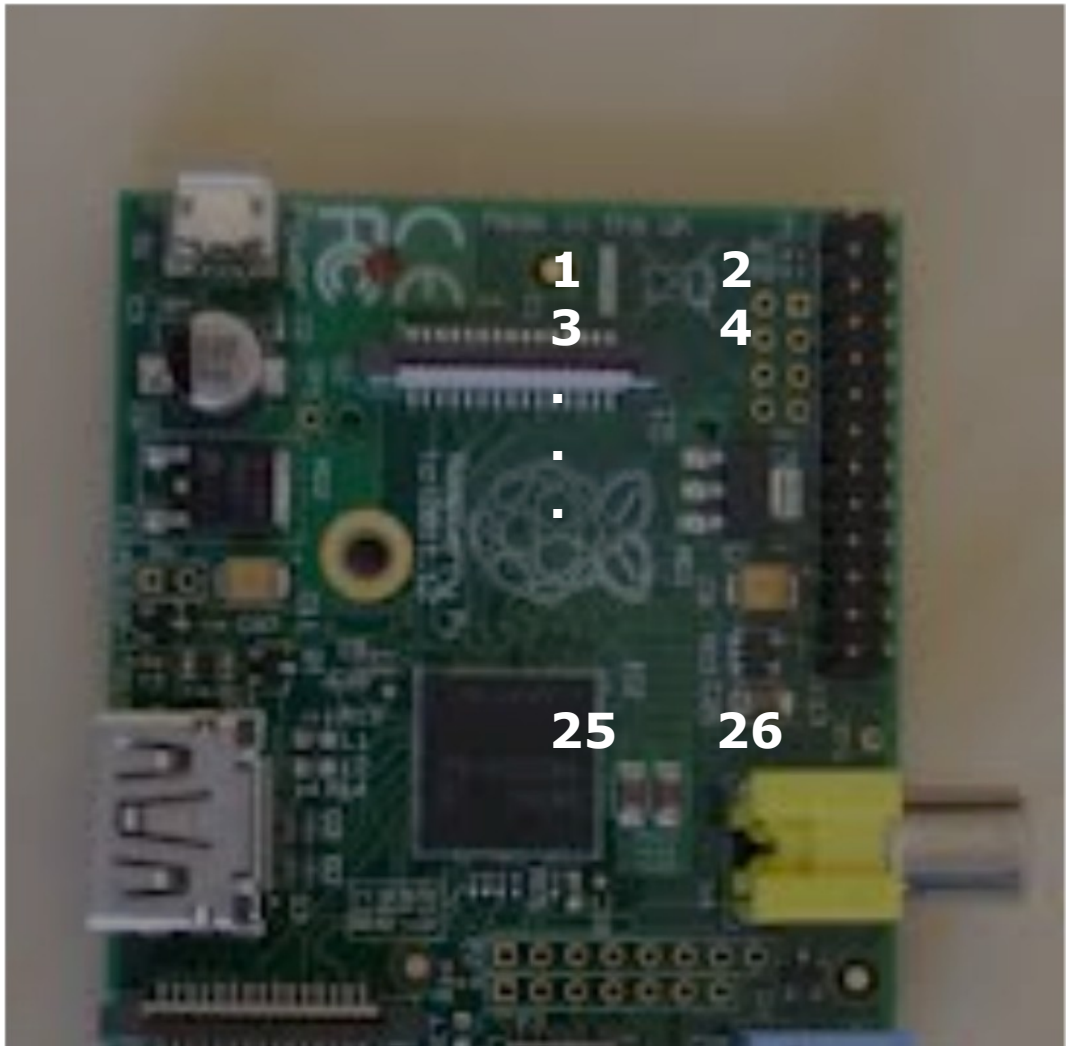
Pin numbering: BCM style



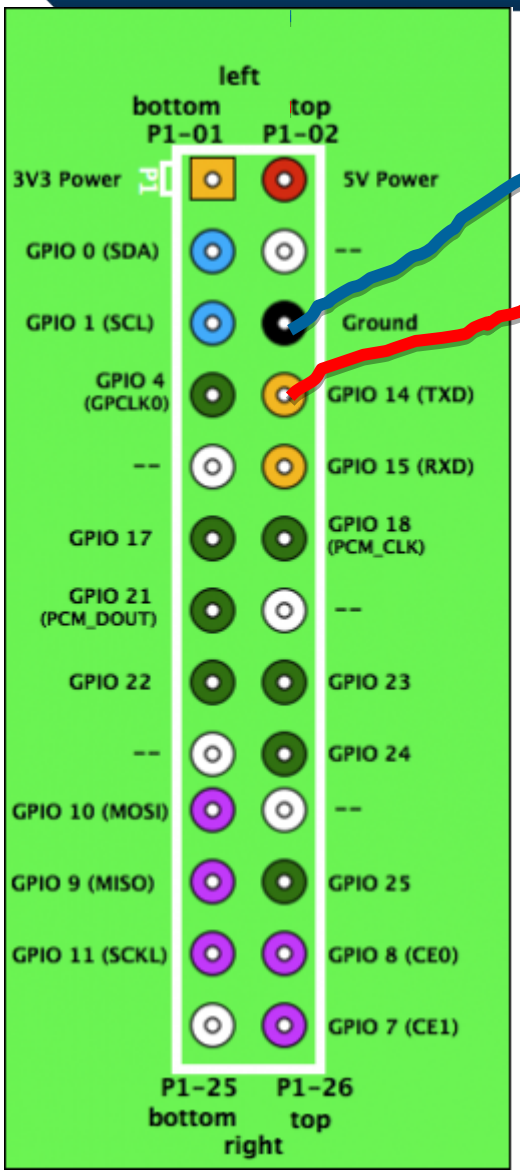


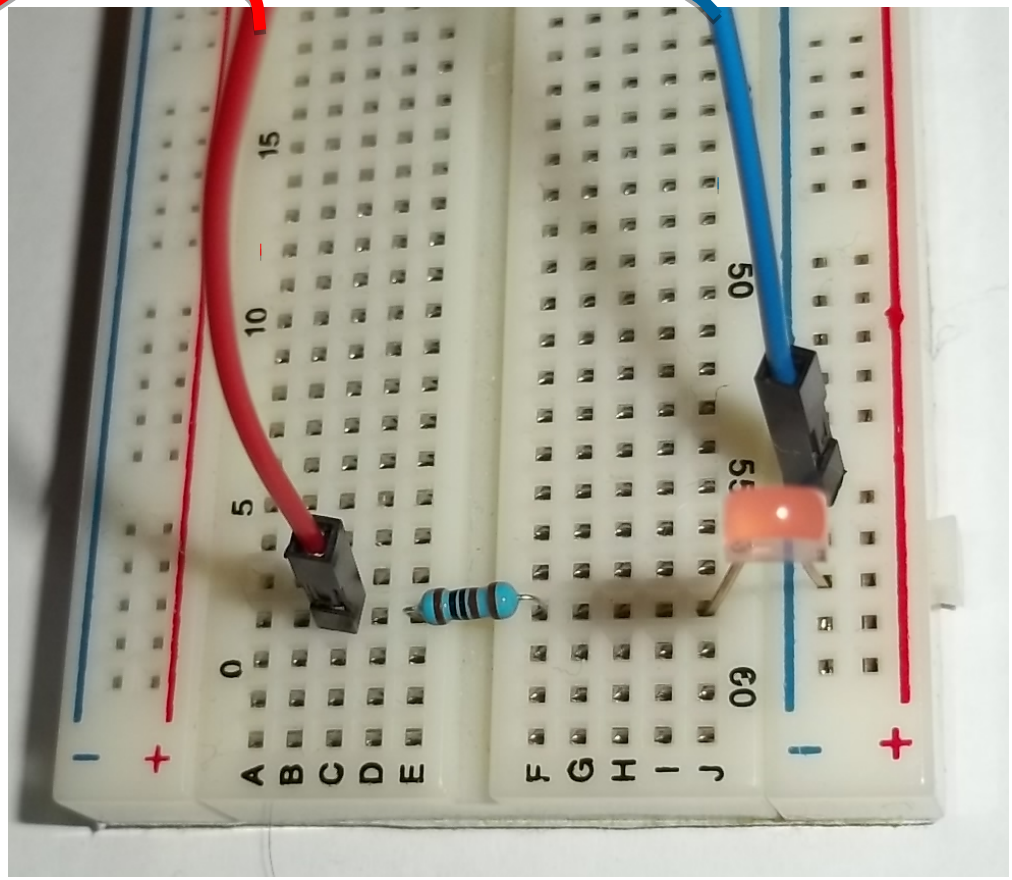
# Programming example 2 – blinking lights

Pin numbering: Raspberry Pi header - NOT USING THIS! Just FYI



# Programming example 2 – blinking lights



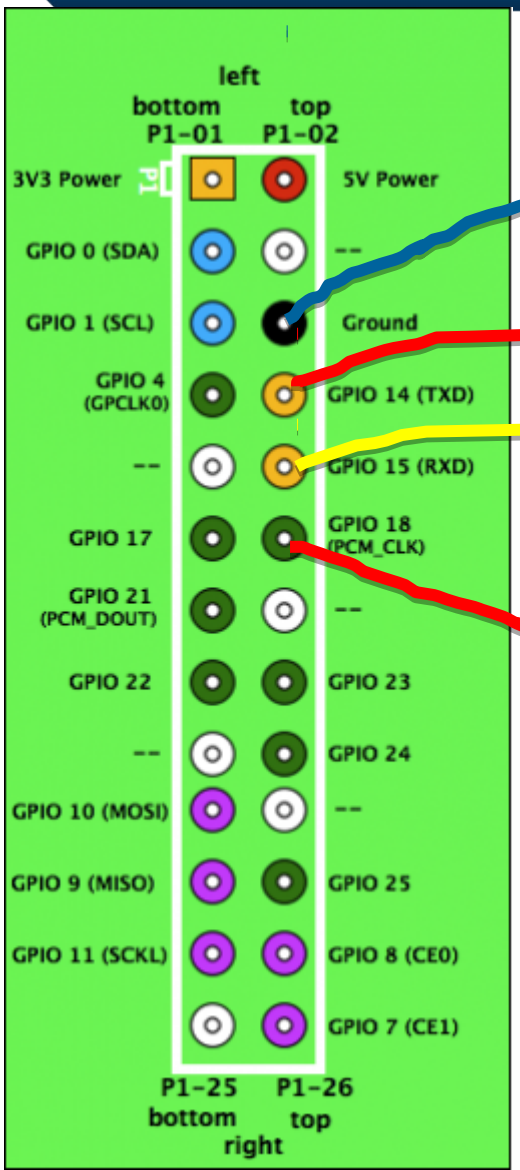
# Programming example 2 – blinking lights

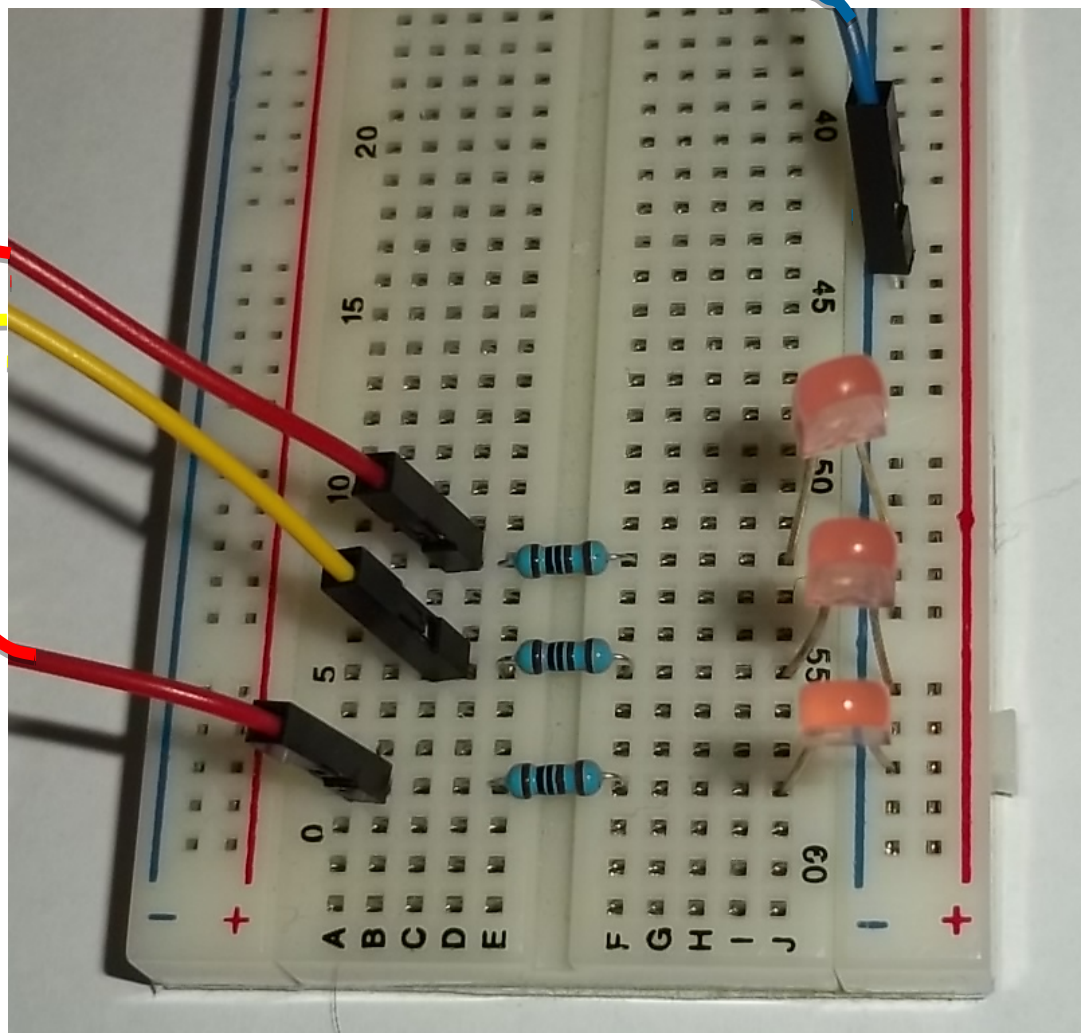
- Open up your terminal
- cd
- mkdir gpio
- cd gpio
- wget troydenton.ca/blink1.py
- sudo python blink1.py

# Programming example 2 – blinking lights

- While this is running, open the file in geany

# Programming example 2 – blinking lights



# Programming example 2 – blinking lights

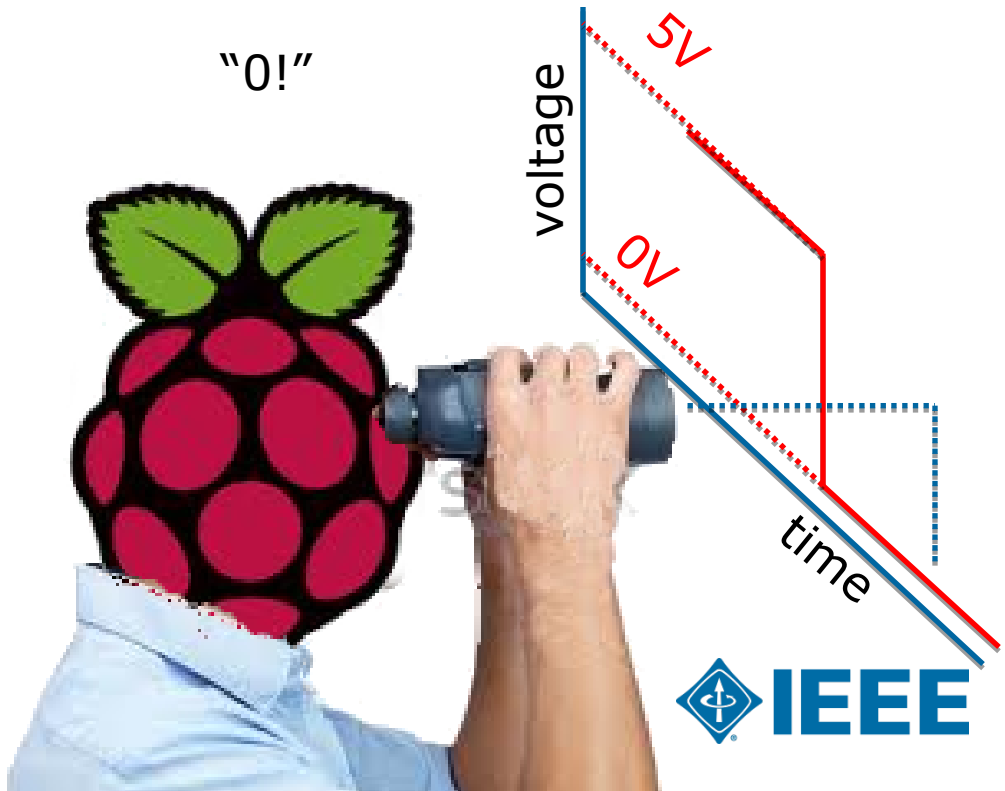
- wget troydenton.ca/blink2.py
- sudo python blink2.py

# Programming example 3 – reading an input

- Similar to how we can tell the CPU to output a value, we can have the CPU "read" a value

*"What do you see ??"*

"0!"

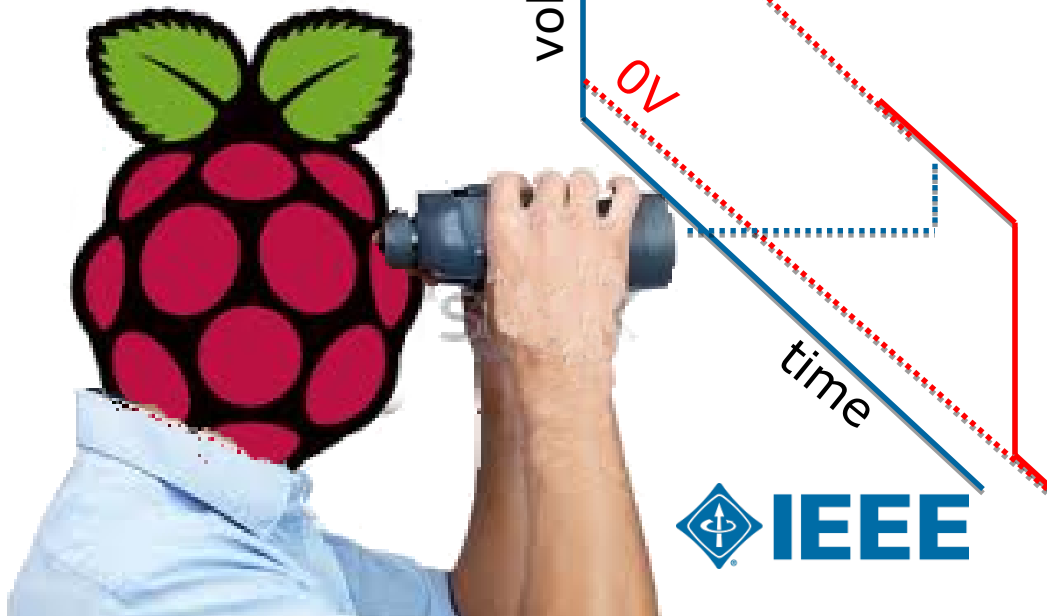


# Programming example 3 – reading an input

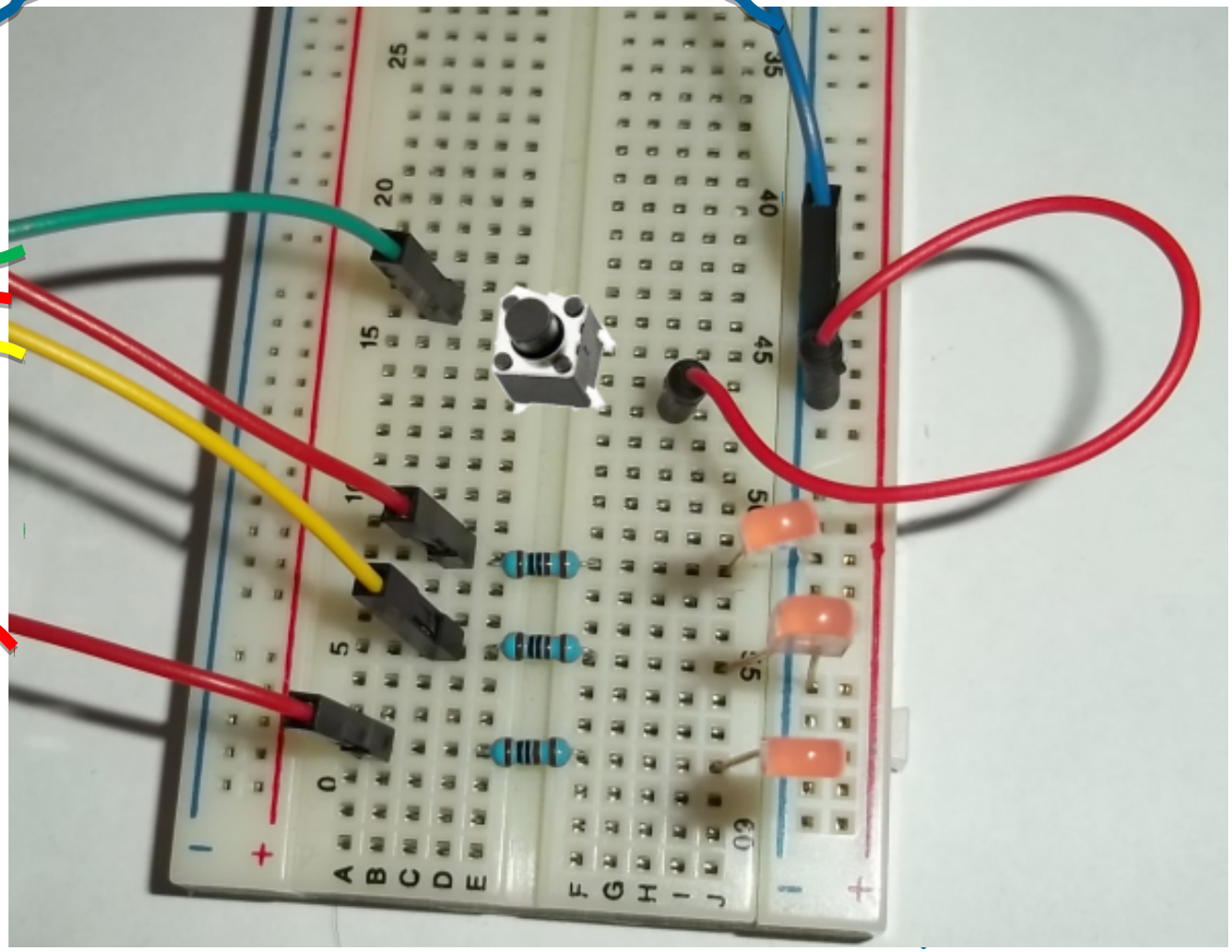
- Similar to how we can tell the CPU to output a value, we can have the CPU "read" a value

*"What do you see ??"*

"1!"

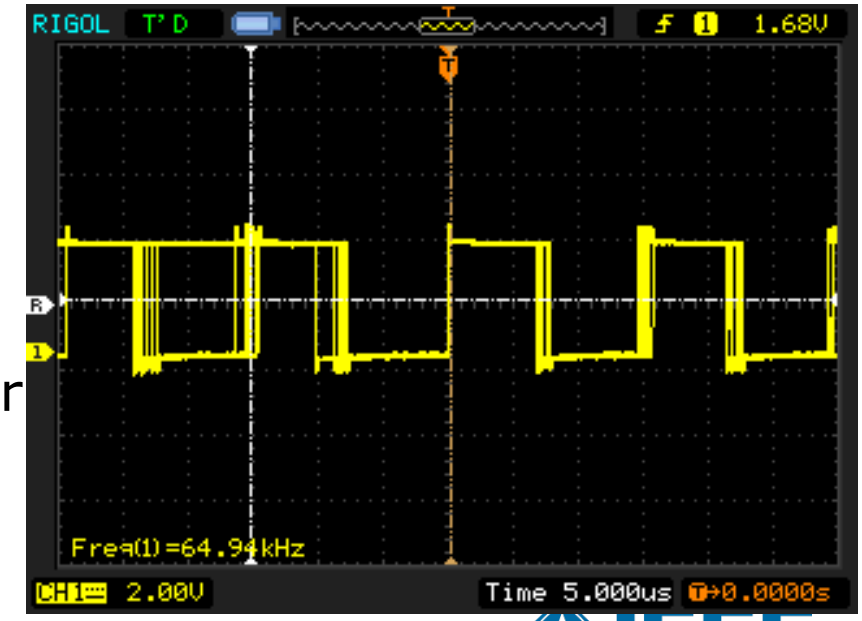


# Programming example 3 – reading an input

# Programming example 3 – reading an input

- Limitations of GPIO:
  - When accessing via python, not at its fastest
    - C is much faster… but, not as easy for a workshop
  - Max GPIO output speed (using python in raspbian)
    - ~60 kHz by this measurement
    - Upwards of 80 observed
    - Jitter issues
    - Acceptable for real-time?
      - Not likely!
  - Max input speed slightly better
    - ~111 kHz



# Programming example 3 – reading an input

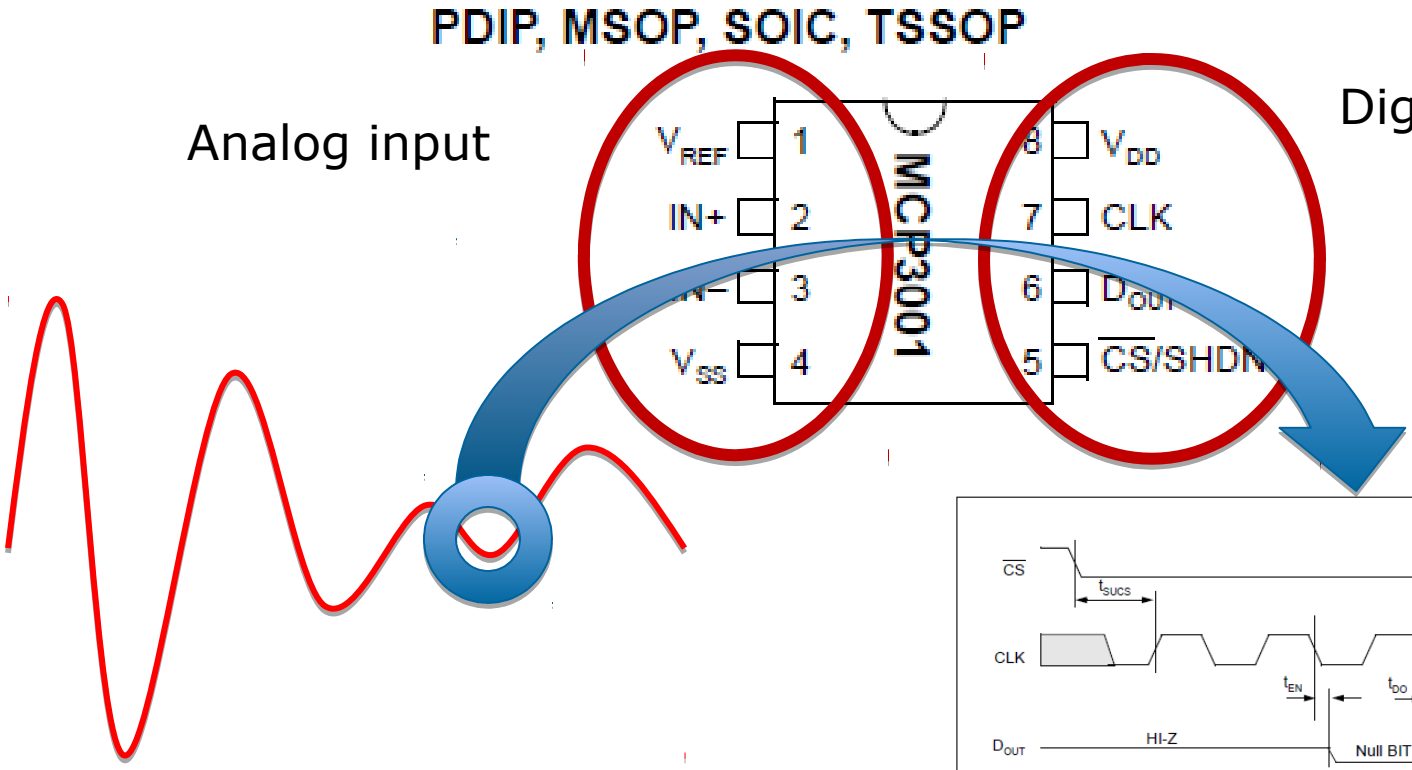
- wget http://troydenton.ca/blink3.py
- sudo python blink3.py

# Intro to SPI

- SPI: Serial Peripheral Interface bus
- A loose standard used to transmit data serially
  - Implementation is up to chip manufacturers
- Many chips out there speak SPI…
  - EEPROMs (memory)
  - Some Network interface IC's
  - Analog to digital converters (MCP3001)
- In this section, we will use the MCP3001 to read an analog voltage into the raspberry pi

# Intro to SPI

- Example from MCP3001 datasheet



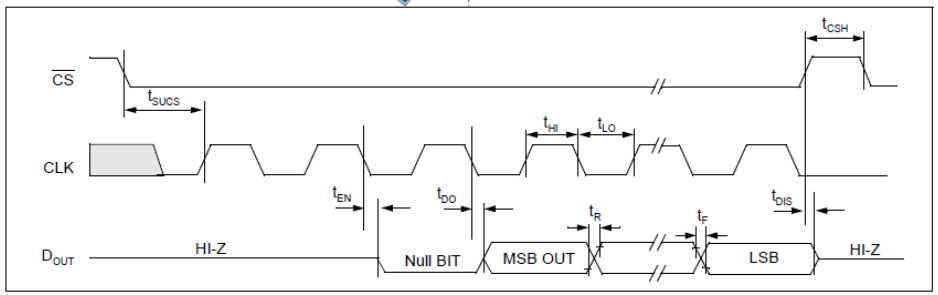
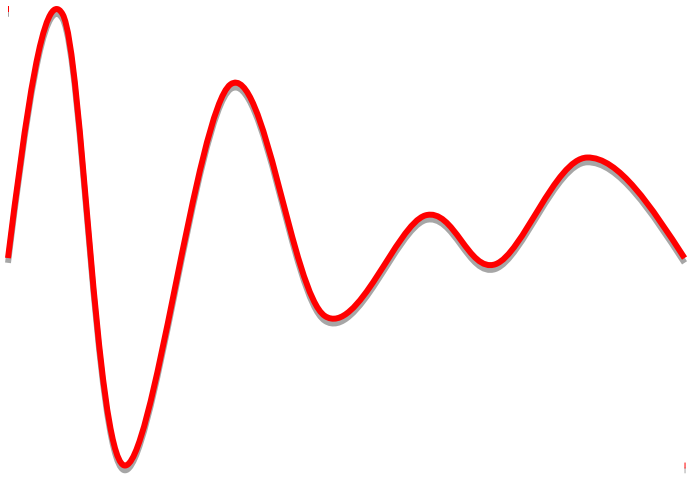




*FIGURE 1-1:* *Serial Timing.*

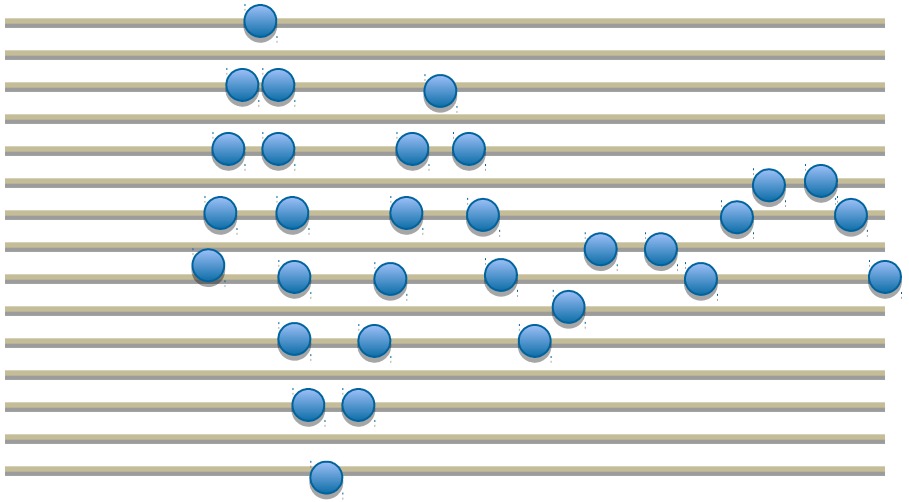
# Intro to SPI

- Example from MCP3001 datasheet

Analog input



Digital Output



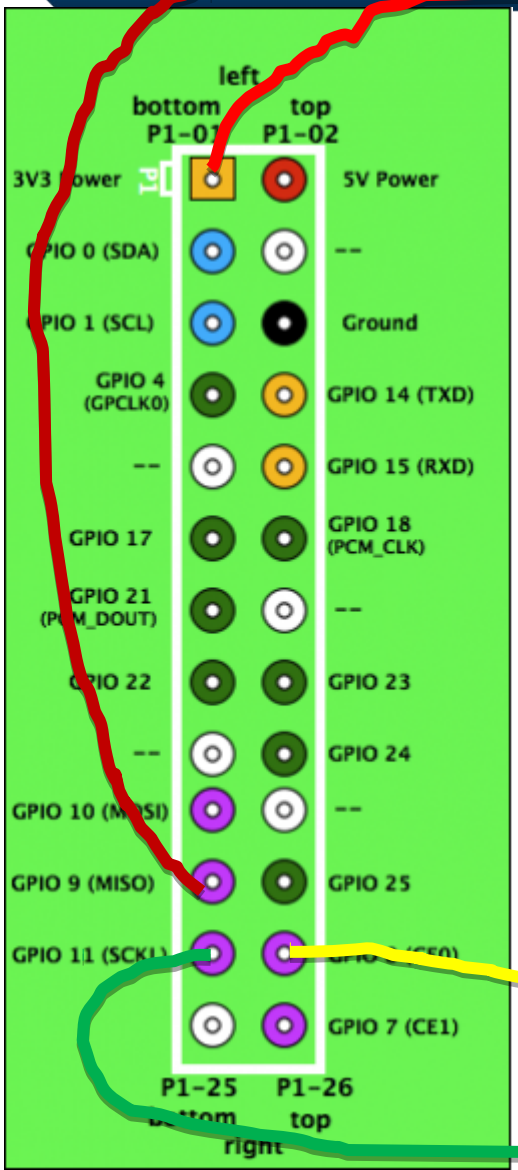
IEEE

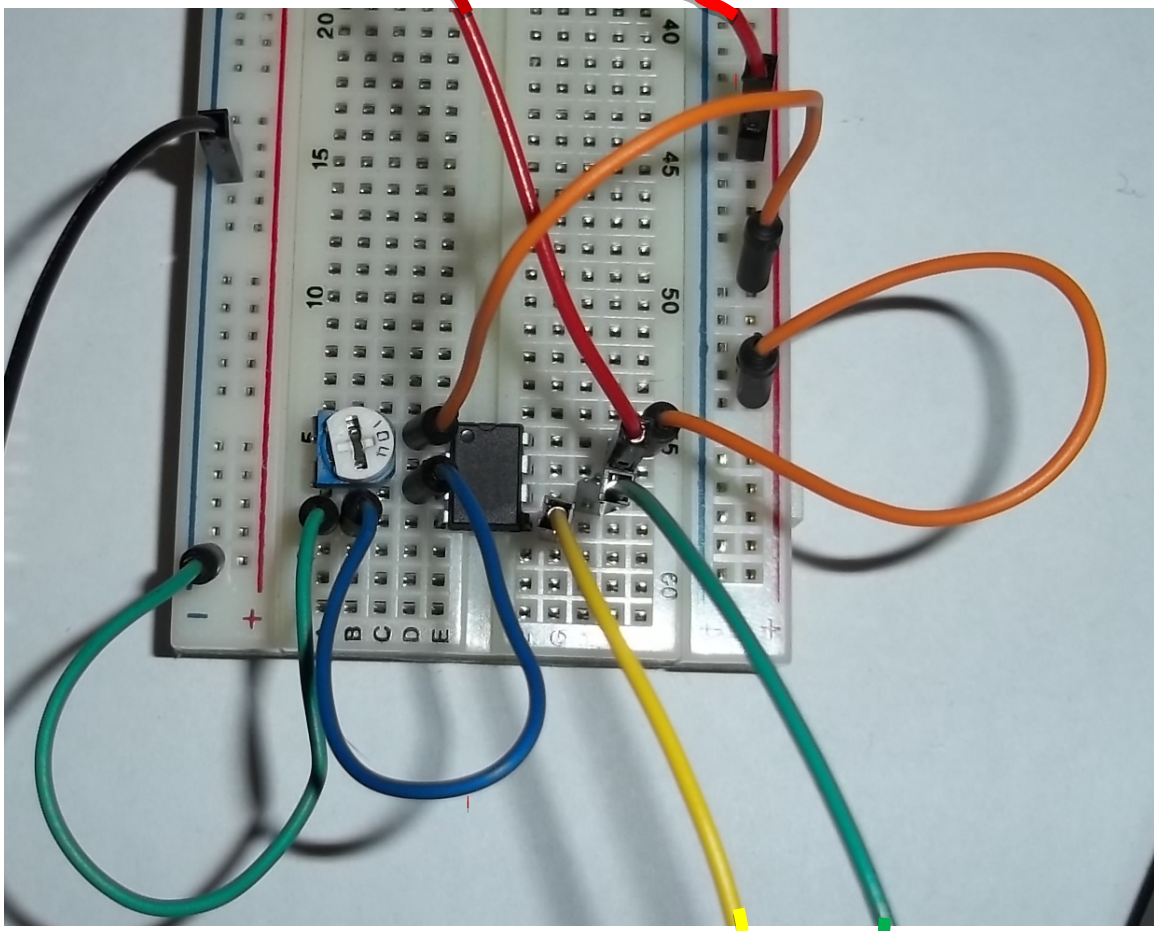
# Intro to SPI

- Example from MCP3001 datasheet


**FIGURE 1-1:** *Serial Timing.*

# Intro to SPI



IEEE

# Installing SPI Libraries

- Run the following commands:
- cd
- mkdir adc
- cd adc
- wget http://troydenton.ca/adc.py
- python adc.py
  - Turn the potentiometer and watch the display change

# Arduino interaction

- The arduino is another embedded platform/microcontroller
- Based on the Atmel AVR series of microprocessors
- Today we are going to use our embedded computer (Raspberry pi) to program our embedded computer (Arduino!)
  - Because we can.

IEEE

# Arduino interaction

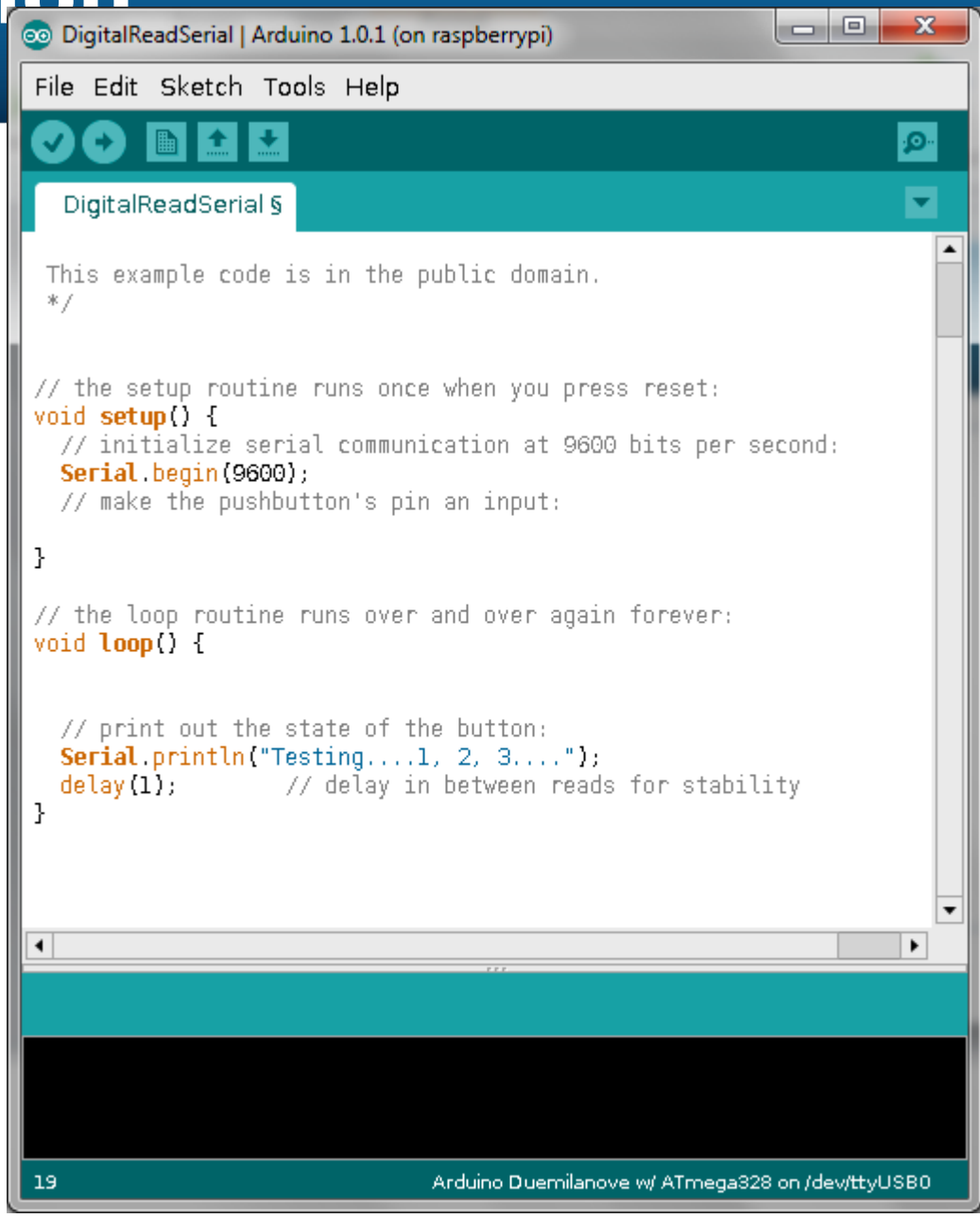
- Plug your arduino into an open USB port on the raspberry pi
- Open up the Arduino IDE by clicking “start” > Programming > Arduino IDE
- In the Arduino IDE, select Tools > Board, and select your model of Arduino
  - If you do not know which, ask one of us and we will get it going for you

# Arduino interaction

- Next, go to tools > Serial Port > /dev/ttyUSB0
- Load up some example code by going to File > Example > Basics > DigitalReadSerial

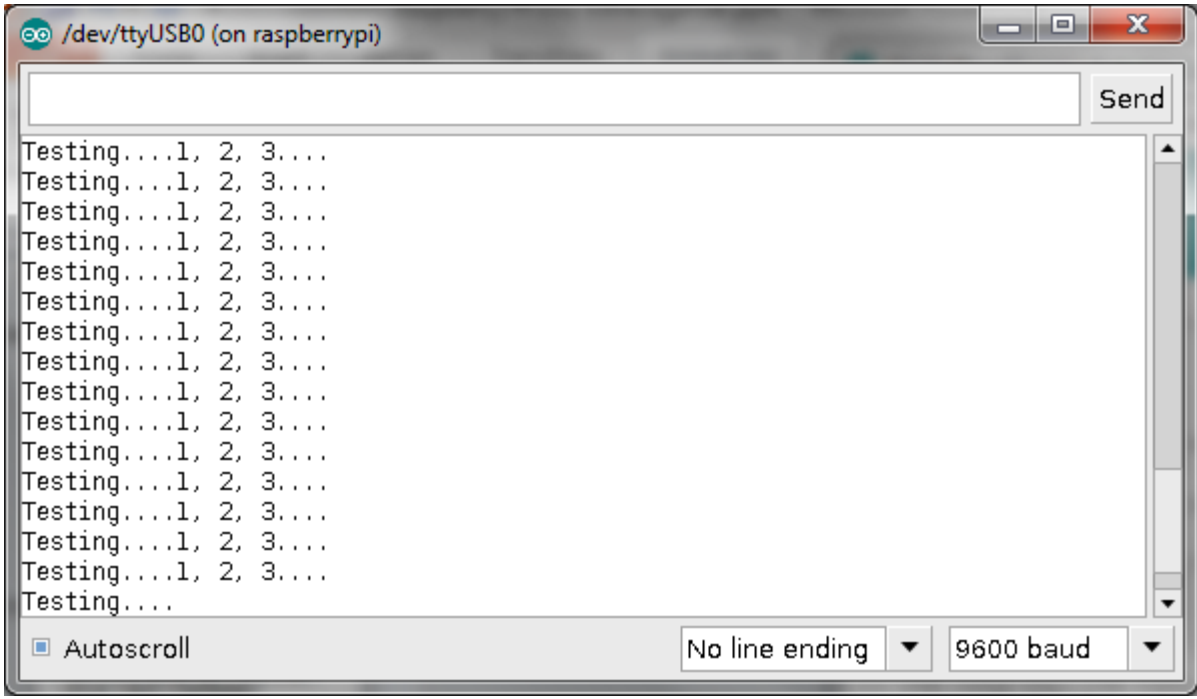
# Arduino interaction

- Modify the code to look like this:
- Then, click the  button to upload the test code to the arduino
- Click  to bring up the serial console



# Arduino interaction

- Make sure the “Baud Rate” is set to 9600 in the serial console:




- You should see your test message printing once a second

IEEE