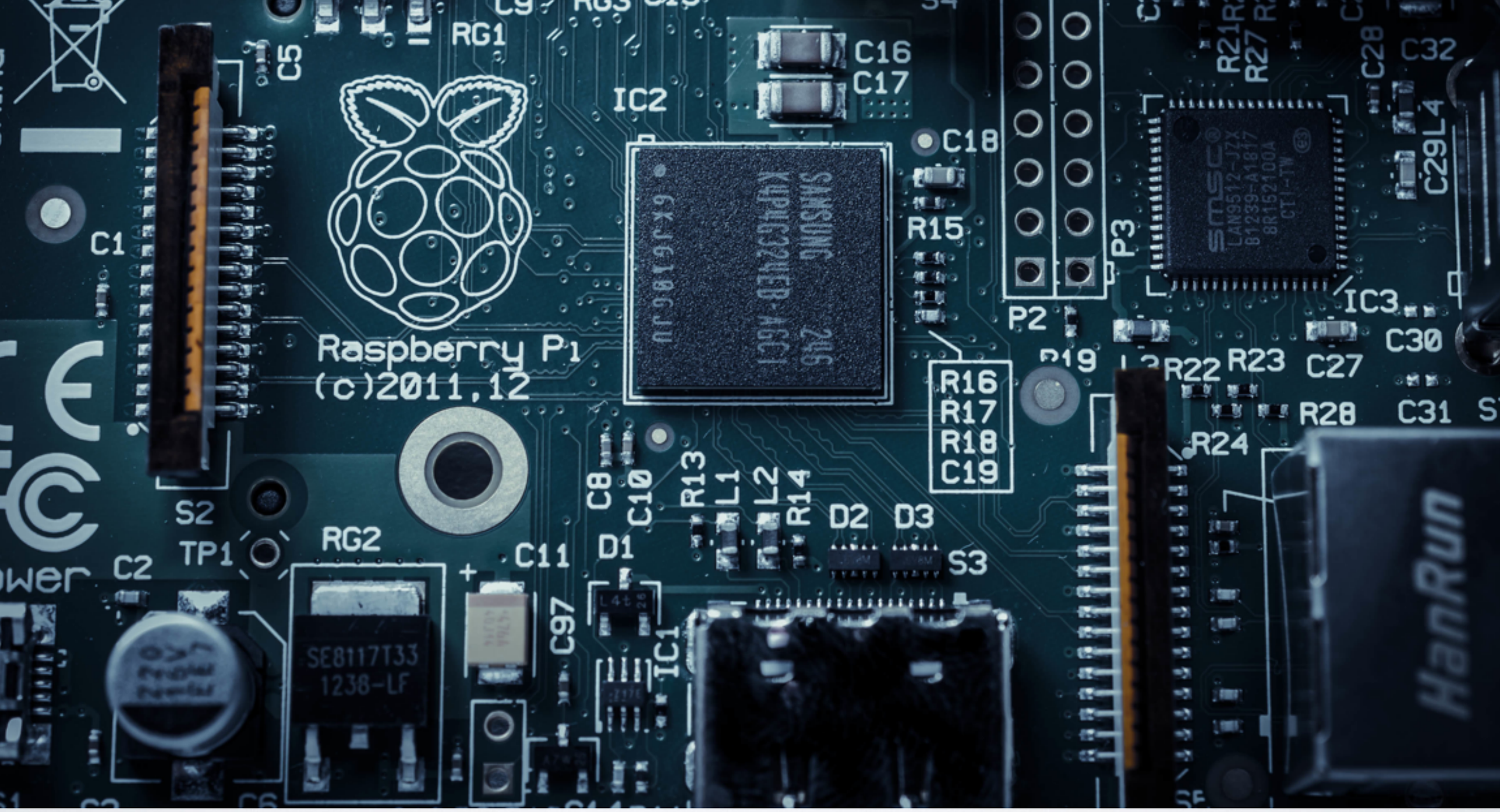


Intro to Robotics with Raspberry Pi!

Section 2. Physical Computing – Controlling Electronics

Outline

Controlling Hardware with Raspberry Pi

Embedded Systems

Microcontrollers

Popular Development Platforms

Raspberry Pi Tech Specs

Access To The Physical World

Understanding GPIO Pins

Analog vs. Digital Signals

Hardware Components

Using a breadboard to wire a circuit

Motor Control HAT

Controlling Motors With Code

How DC Motors Work

Understanding PWM

Making a Motor Move

Making Multiple Motors Move

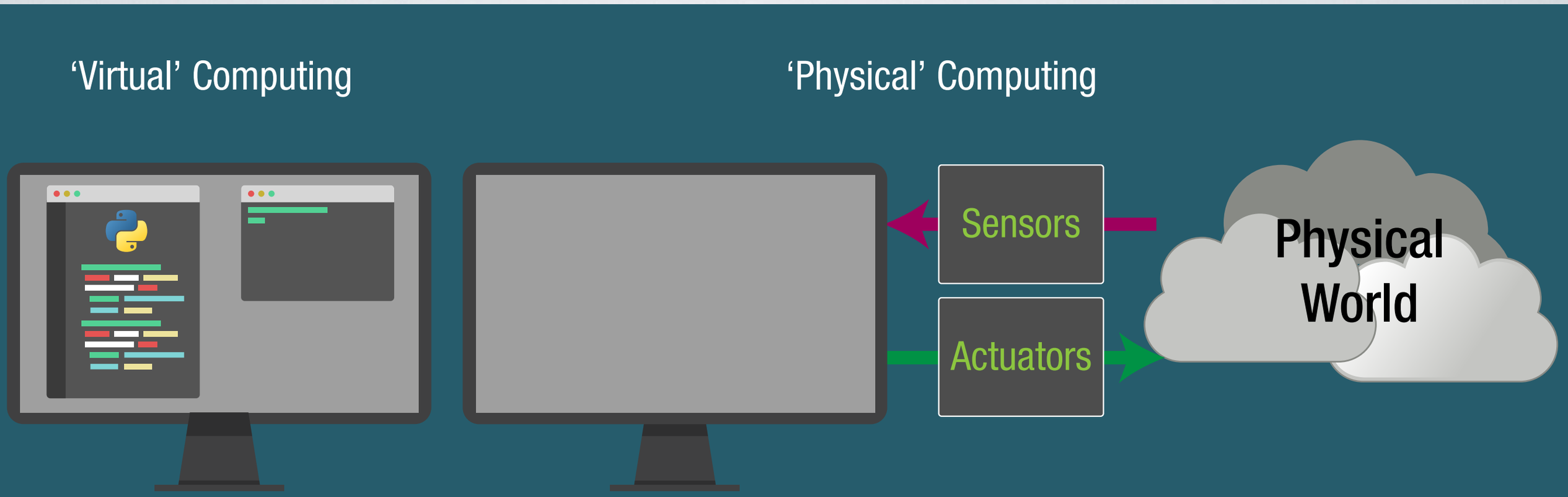
Creating a Closed-Loop System

Using a Sensor

Velocity Control Using Feedback

Introducing Physical Computing

What is Physical Computing?

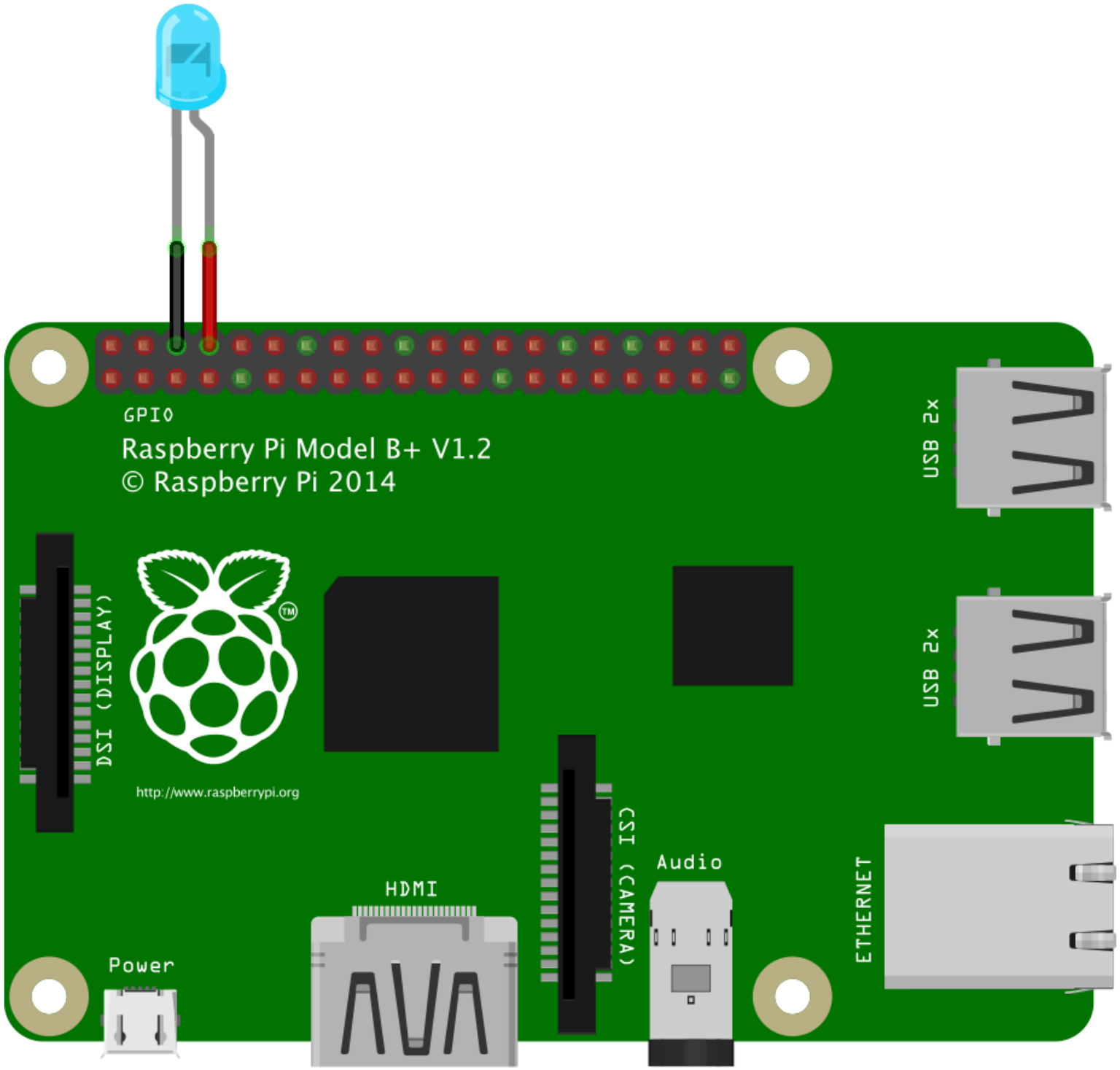


- **Computing** consists of the use of a computer device for any purpose.

- **Physical computing** consists of the use of computer devices for the specific purpose of interacting with the physical world.

Raspberry Pi for Physical Computing

- The Raspberry Pi is a (regular) computer with the ability to perform physical computing tasks using 40 input/output pins.

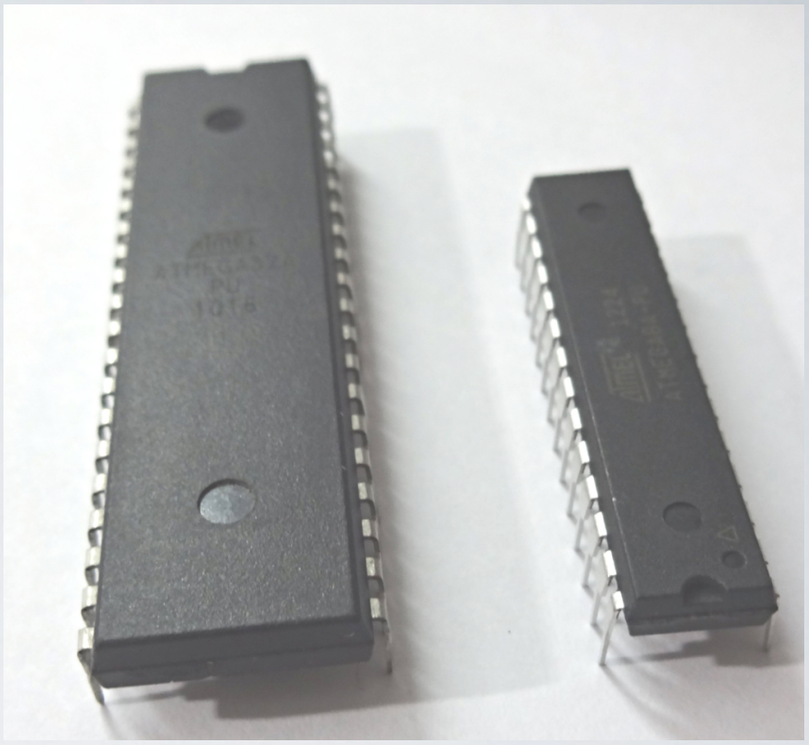


Embedded Systems

Embedded Systems - What They Do

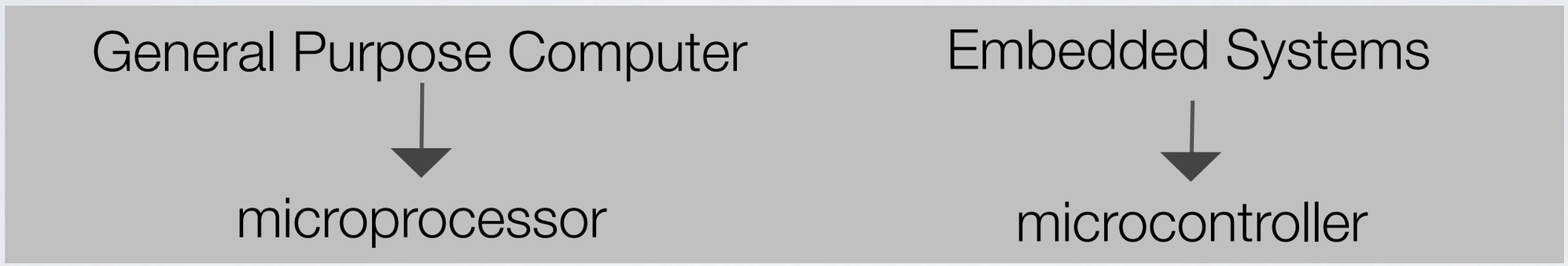
- Definition: small computer system with a **dedicated** function/program.
- Only carry out pre-defined task(s) within a device/machine (embedded).
- Fixed capability vs. general purpose computers (Macs, PCs).
- Software loaded onto Hardware = **firmware**.
- Hardware includes: memory; inputs & outputs (IO); user interface; display.

Microcontrollers



A simple computing device

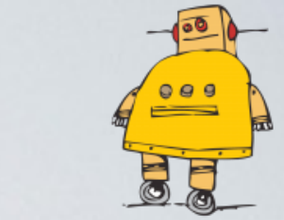
- Small computer on an IC (integrated circuit).
- Easy to program.
- Interacts with the physical world (I/O peripherals).
- Heart of **Embedded Systems**.



Programming a microcontroller

- Previously used assembly language/machine code (e.g., **at university labs**)
- Now we can use higher order languages: C, Python, custom.
- Vendors develop tools (interpreters, compilers, simulators).

Embedded Systems IRL

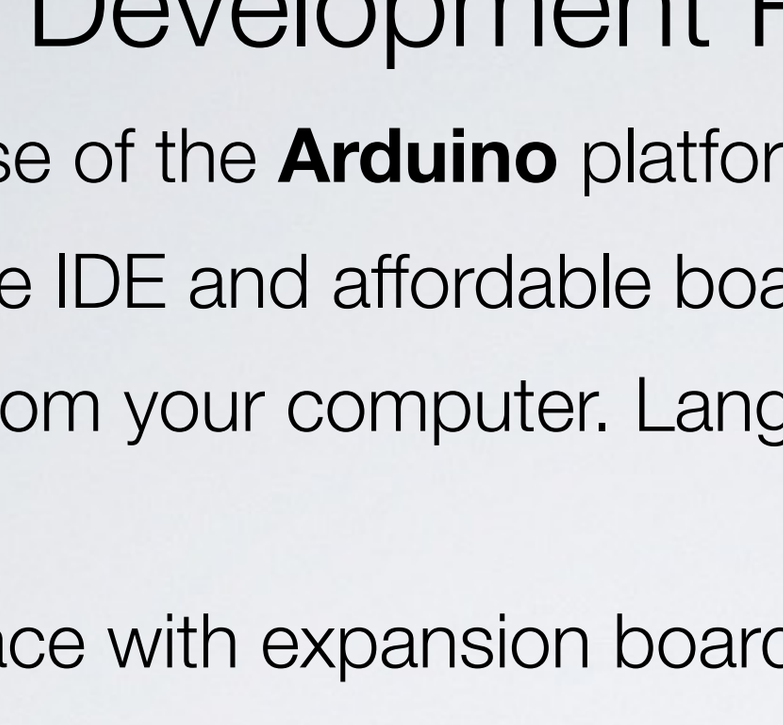


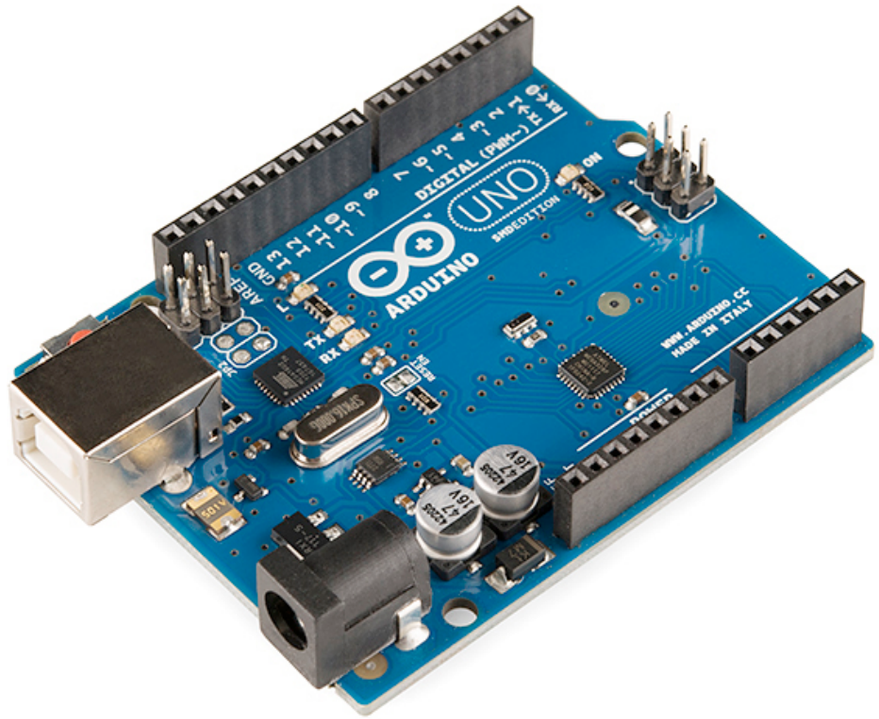
- Mid 2000's: Expansion out of academia into consumer market.
- **Open-Source** software and hardware (Creative Commons and General Public licenses) lets anyone take designs and change/add to them.
- Anyone can access and learn how to use.
- STEM/STEAM: Current national attention on increasing tech teaching in schools.
- Electronics tinkerers are subset of Maker community.
- Art community also making use of electronics Maker tools.

Popular Development Platforms

2005 saw rise of the **Arduino** platform

- Easy to use IDE and affordable board (Open-Source).
- Program from your computer. Language based on C++.
- Can interface with expansion boards (shields).
- Lots of copy-cat boards though “Arduino” name is protected.

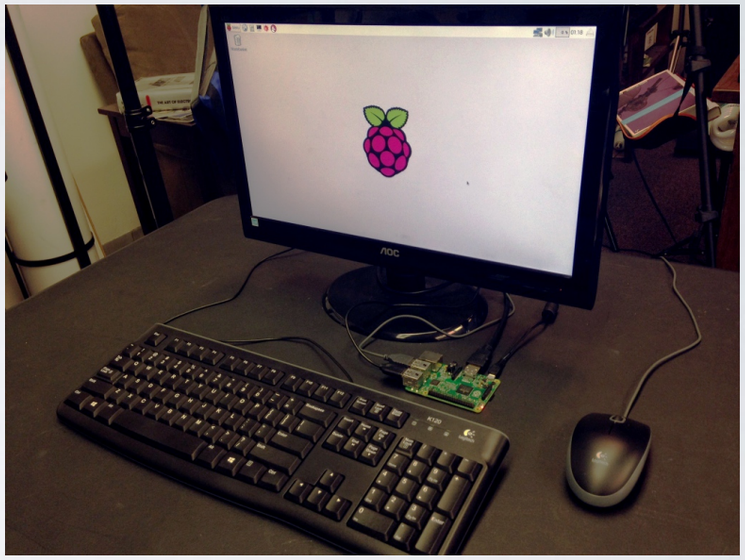




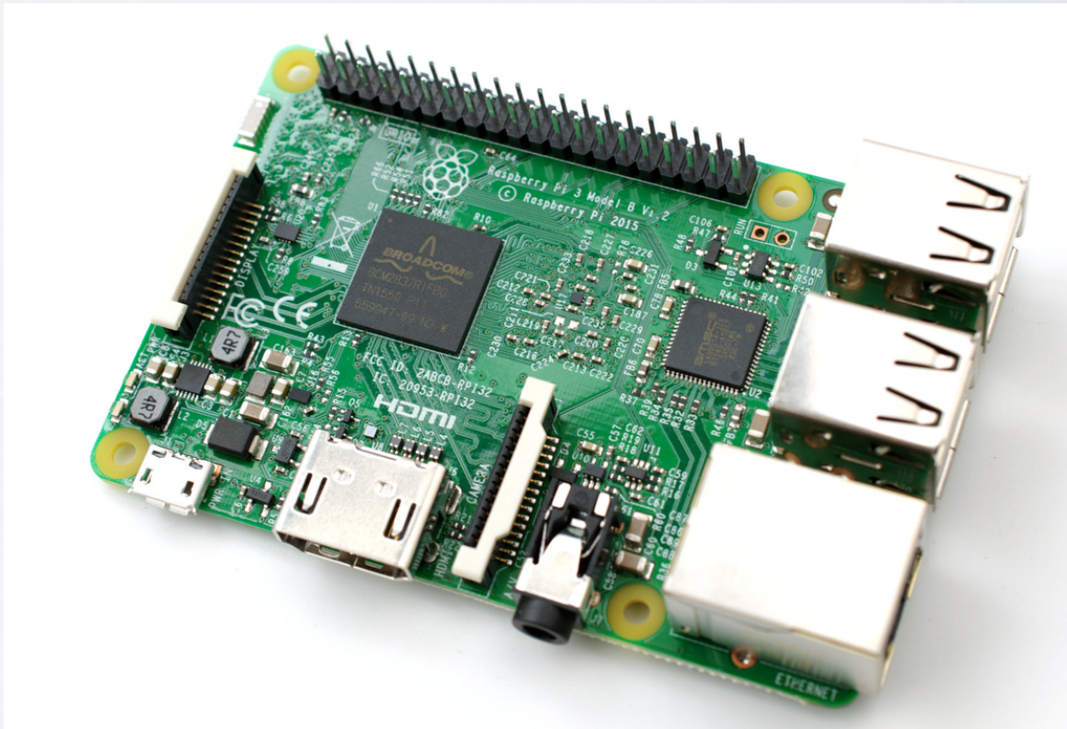
Popular Development Platforms

Raspberry Pi enters the scene in 2008

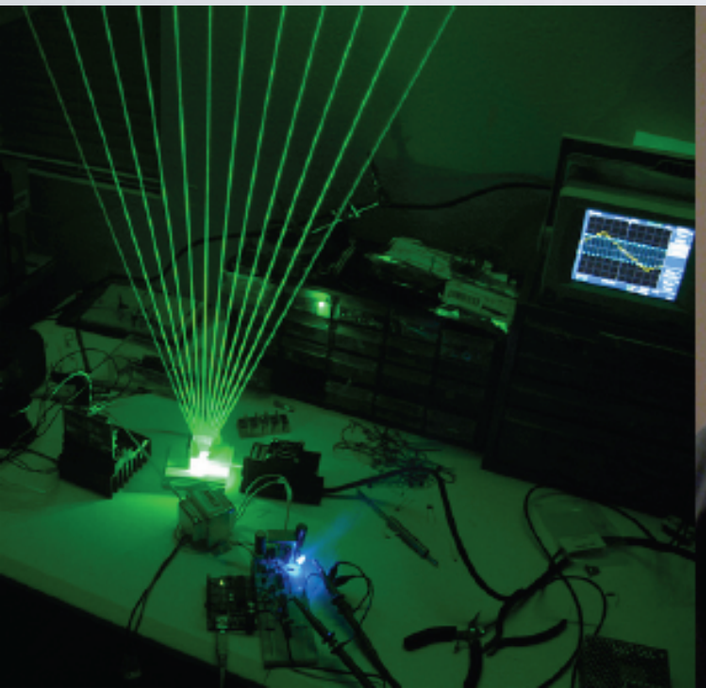
- Focus is on accessible computers for kids.
- Developed and made in the UK as answer to lack of CS interest.
- Some parts are **not** Open-Source.
- Expansion boards are HATs.
- Built-in WiFi connectivity.

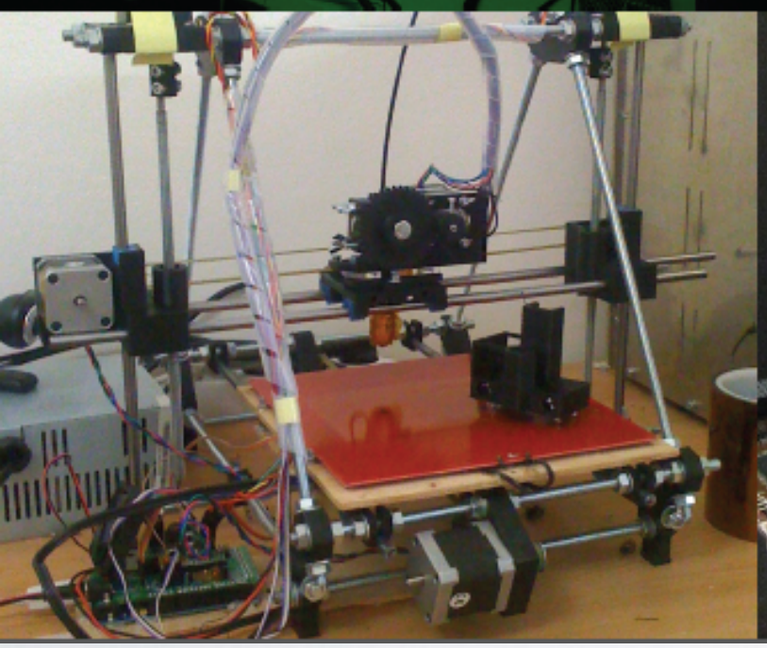


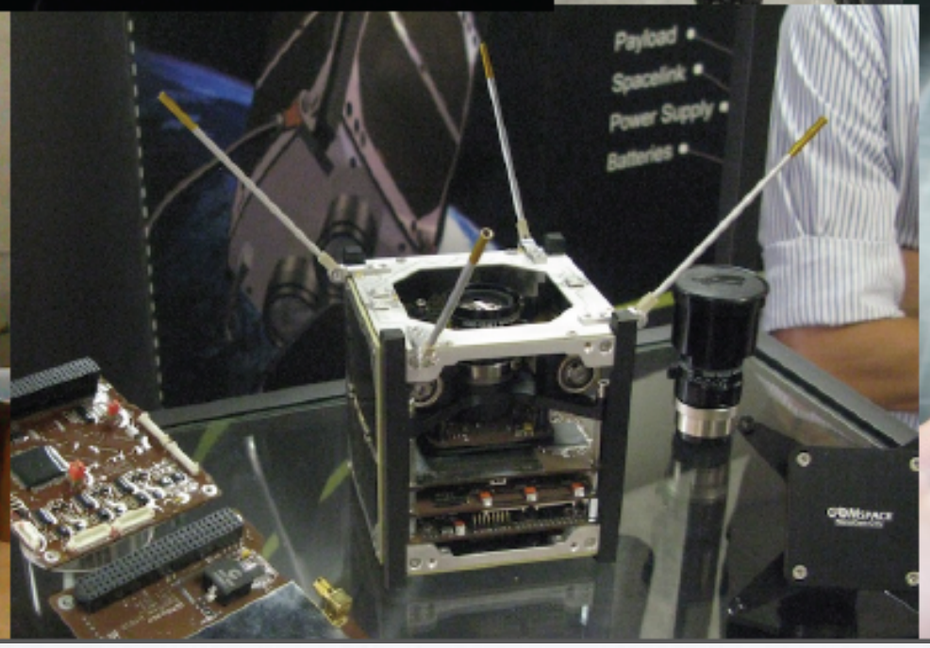
Now the most popular development board for tinkerers (~8M boards sold since ‘12).

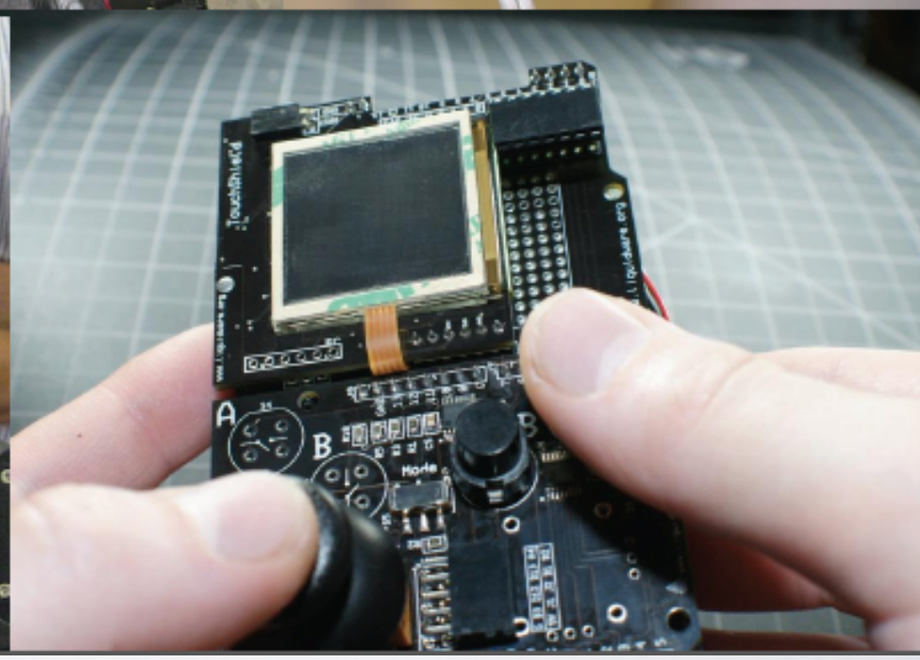


Uses and Applications







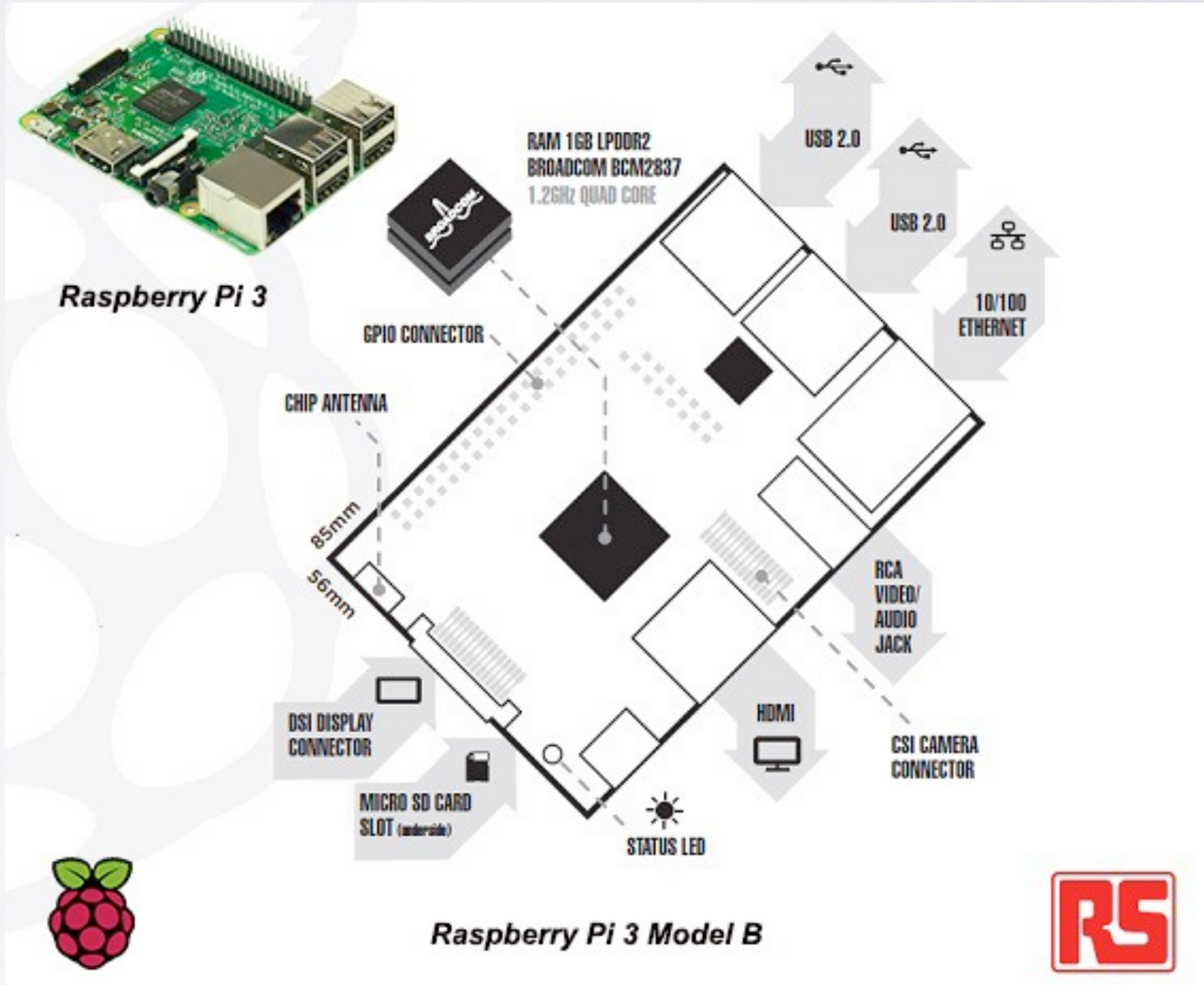


Raspberry Pi - Under the hood

- CPU: 1.2GHz 64-bit quad-core (ARM)
- GPU: Broadcom VideoCore IV @ 400 MHz
- Memory: 1 GB LPDDR2-900 SDRAM
- Network: 10/100 MBPS Ethernet, 802.11n Wireless LAN, Bluetooth 4.0
- **40 GPIO pins**

All the parts of a regular, general purpose computer with the bonus of built-in microcontroller.

How? **GPIO** pins that connect to the outside world.



Access To The Physical World

Understanding GPIO Pins

General Purpose Input & Output Pins (GPIO)

- They connect to components and/or other devices.
- Can read or send data as voltages.
- Power pins (5V, 3.3V) supply constant voltage.

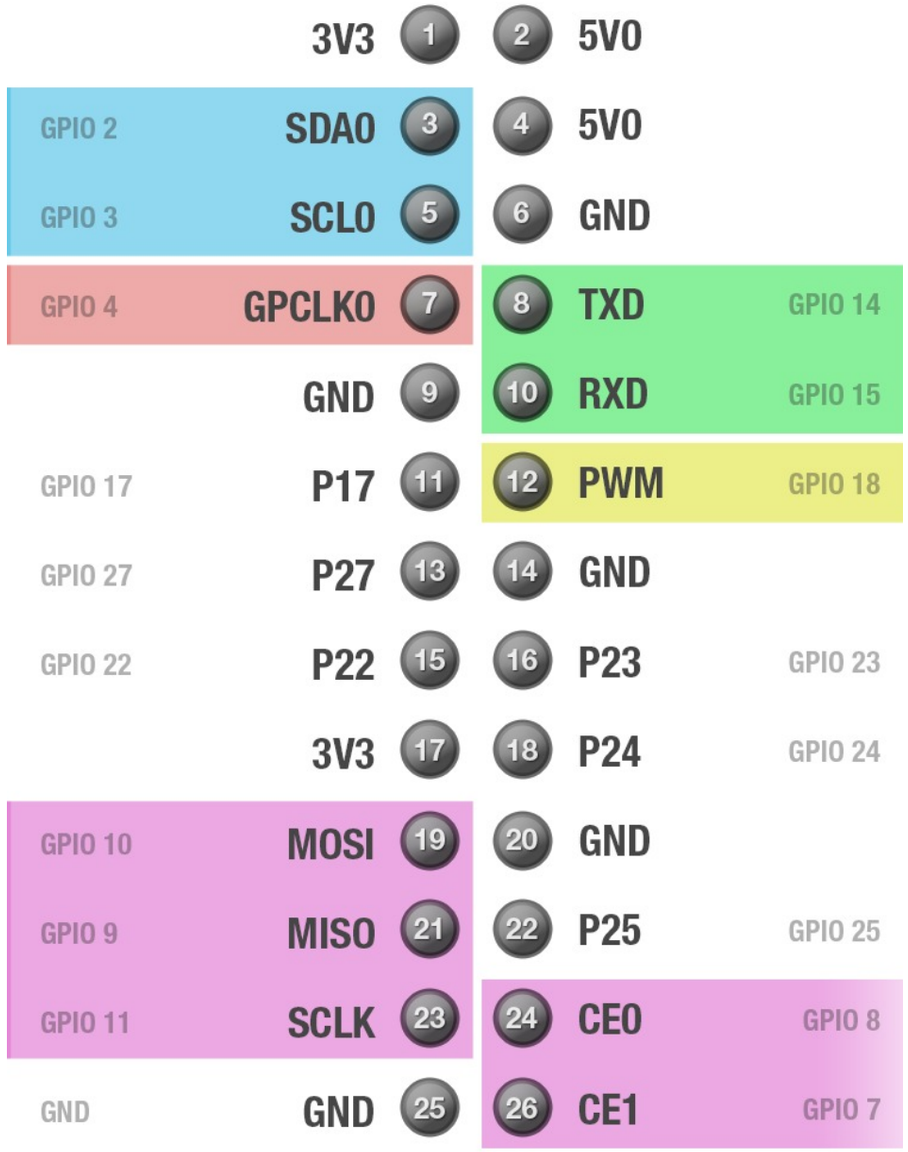
I2C- 2 wire comms interface that supports multiple devices.

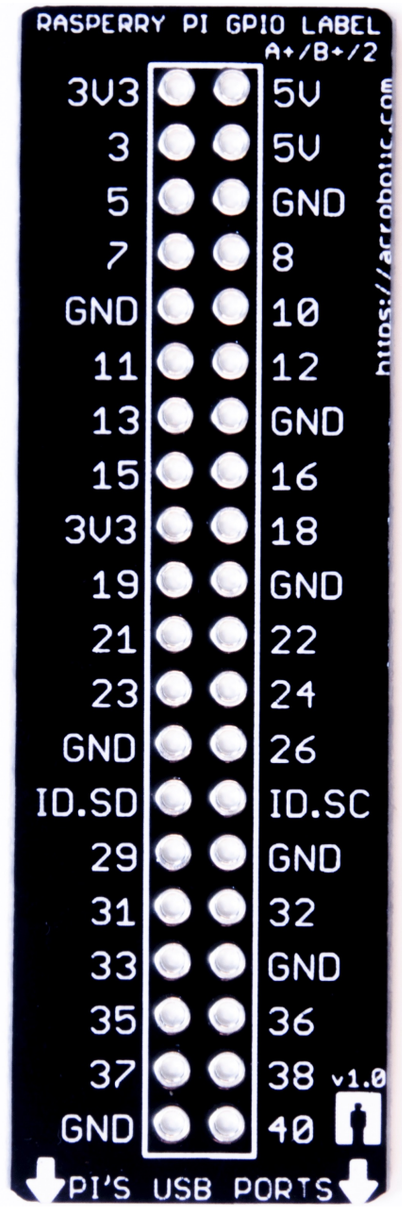
CLK- Synchronizes system actions that are time sensitive.

UART- Access to serial console for advanced stuff.

SPI- Older method of comms. Used for complicated devices.

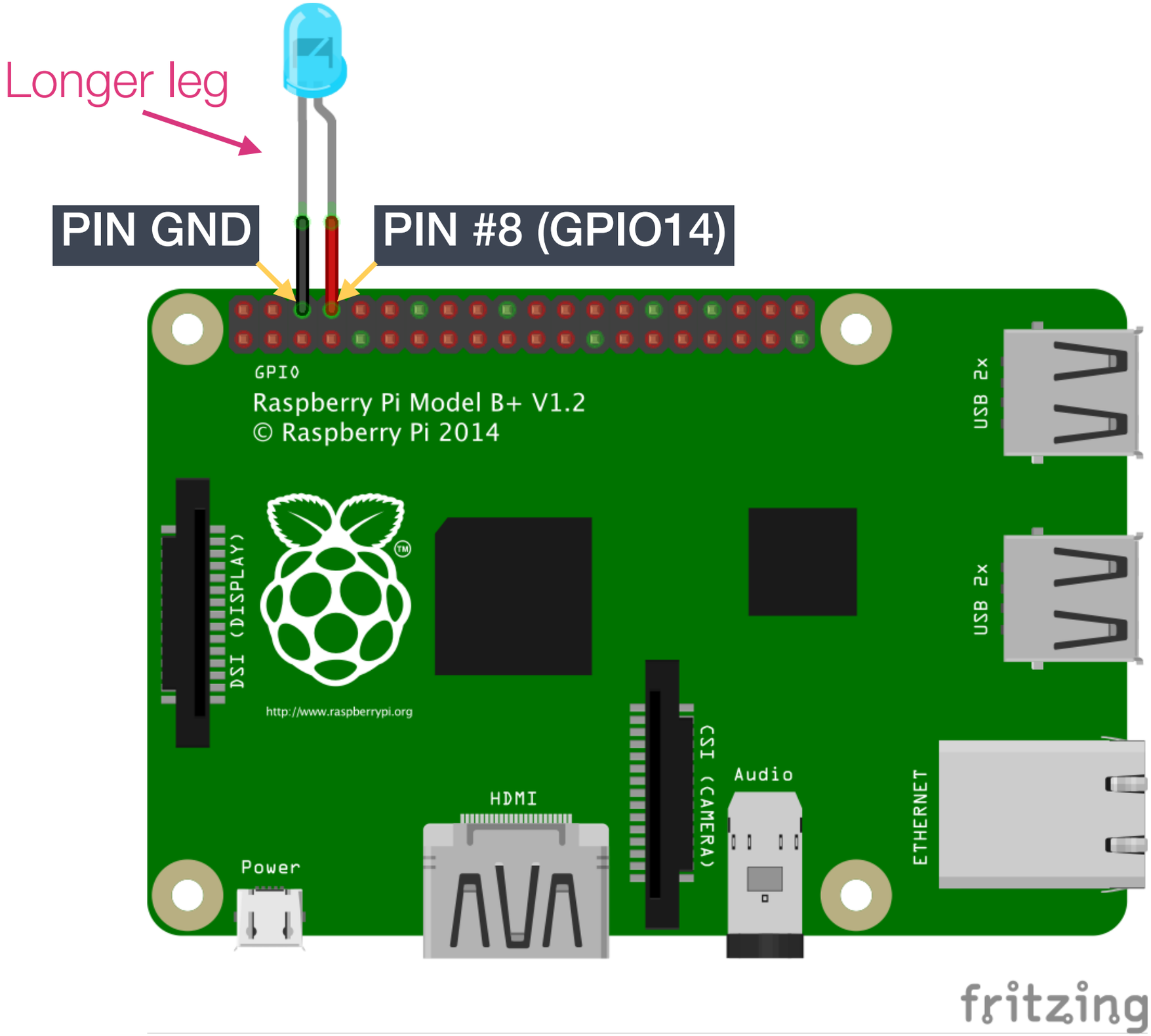
PWM- “Analog style” signals that provide variable power.





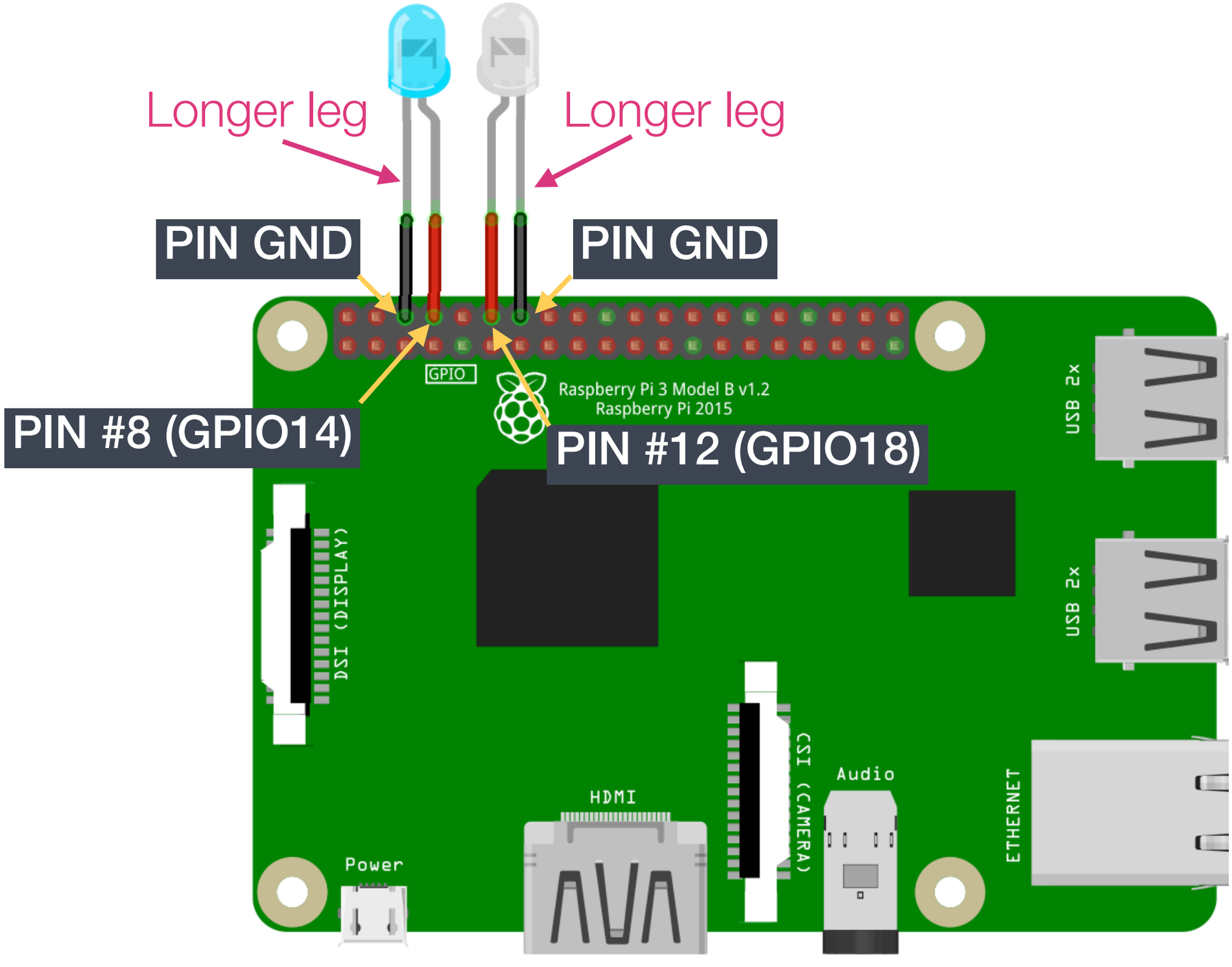
Activity 1

Challenge: Recreate LED blinking system



Activity 2

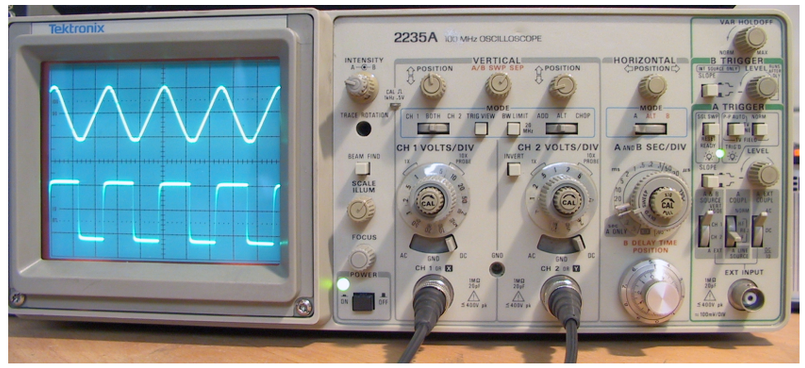
Add a 2nd LED and alternate blinking patterns



Analog vs. Digital Signals

Signals

- Information sent over a physical medium.
- Data is encoded and transmitted.
- Our signals are carried as voltages through wires.



Digital

- World of electronics (microcontrollers, computers) are digital.
- Signals are square-waves (steps) of limited resolution.
- We use system of two logic levels (0V, 5V).
- Digital signals must be converted for analog applications.

Analog vs. Digital Signals

Analog

- We live in an analog world.
- Sound, light, temperature, motion, etc. has infinite resolution.
- Signals are smooth, continuos.
- Analog signals must be converted for use in a digital system (ADC).

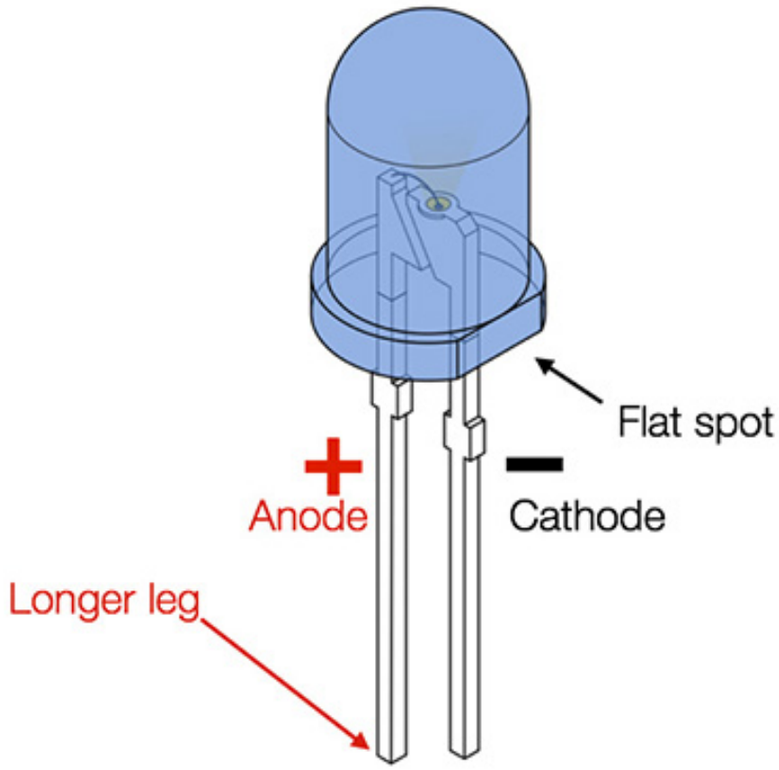
Hardware Components

Components

Resistors

- Passive components that reduce the flow of current.
- Color coded according to amount of resistance (ohms).
- No polarity. Current flows in any direction.

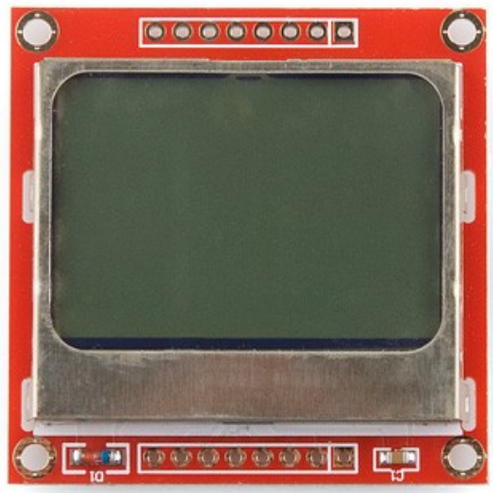
Light Emitting Diodes



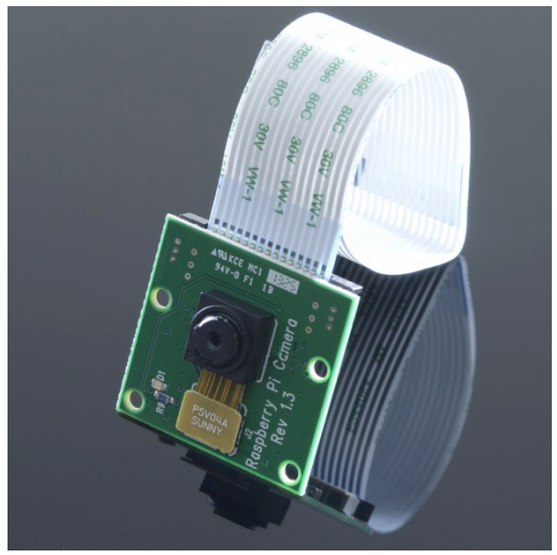
- Lights up when current passes through.
- Current flows in one direction (+ to -). Has polarity.
- Light color determines voltage rating. Colors lower on frequency spectrum need less voltage.

Other Components

LCD

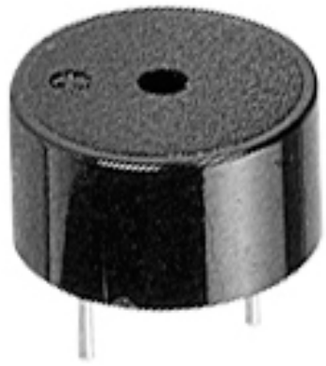


Camera



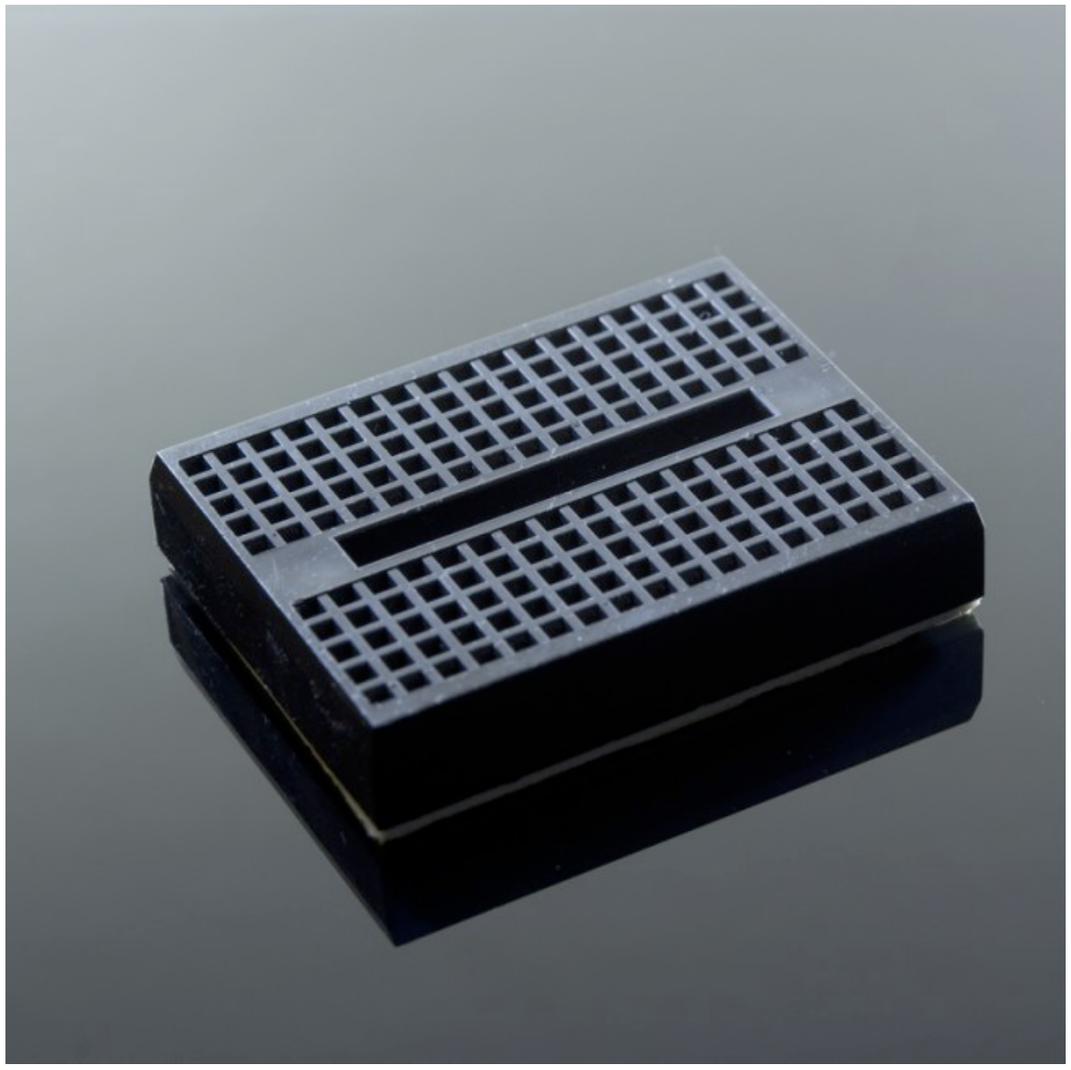
Motor

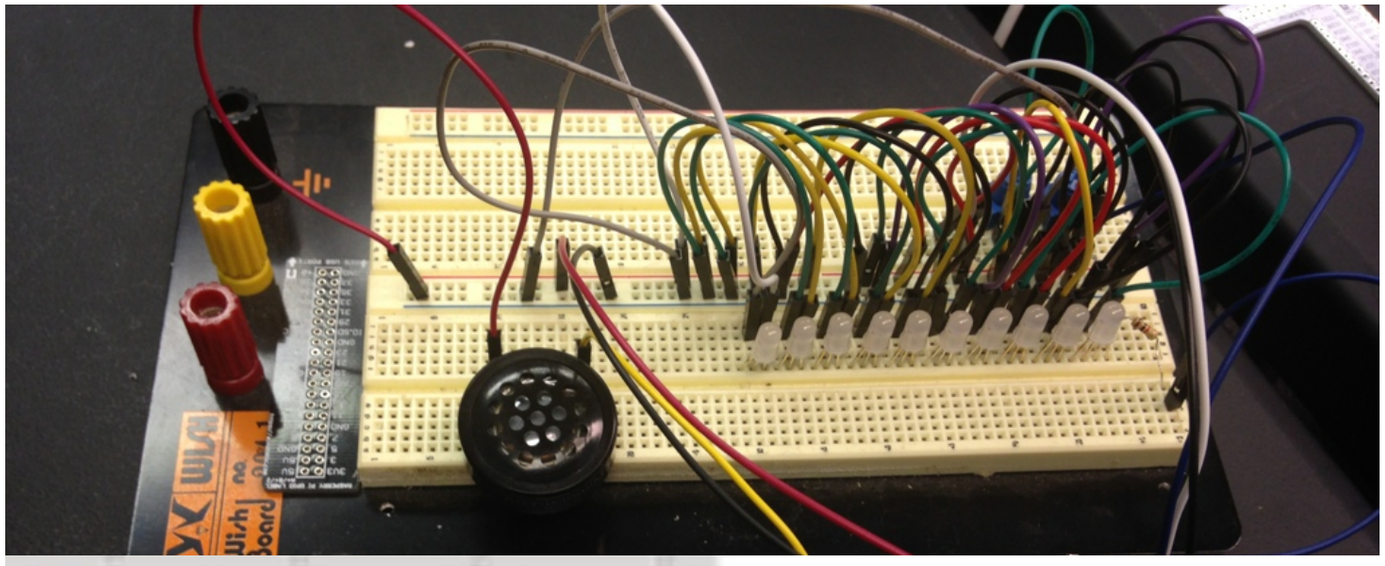
Speaker

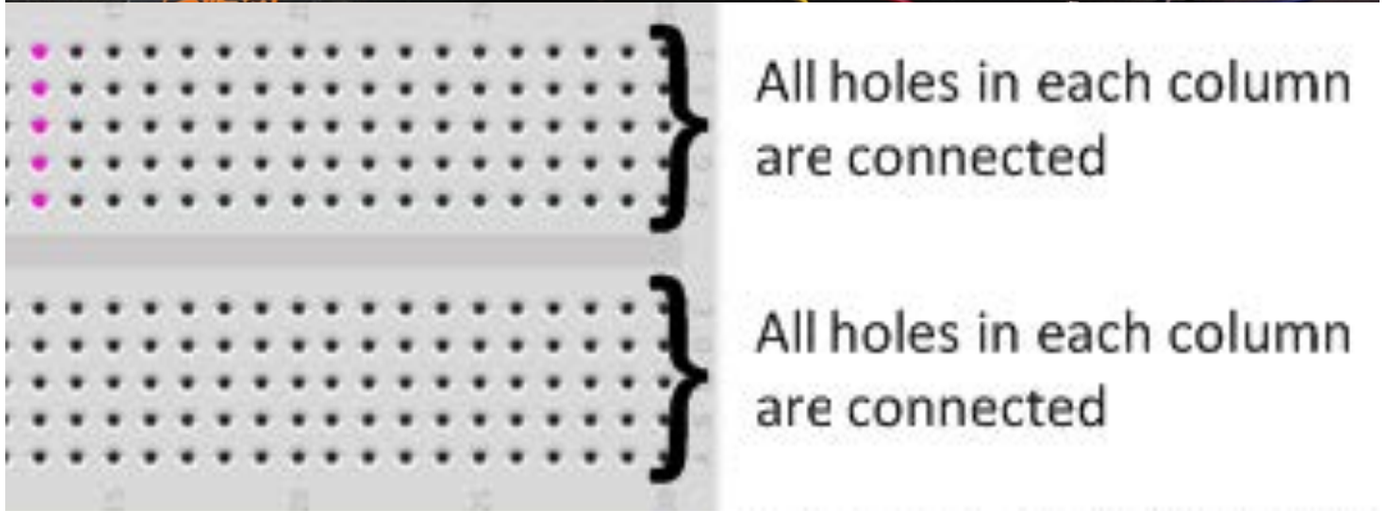


Using A Breadboard

- Solder-less breadboards are used for testing out circuit designs.
- Components can be connected/disconnected easily.
- Configurations are NOT permanent.

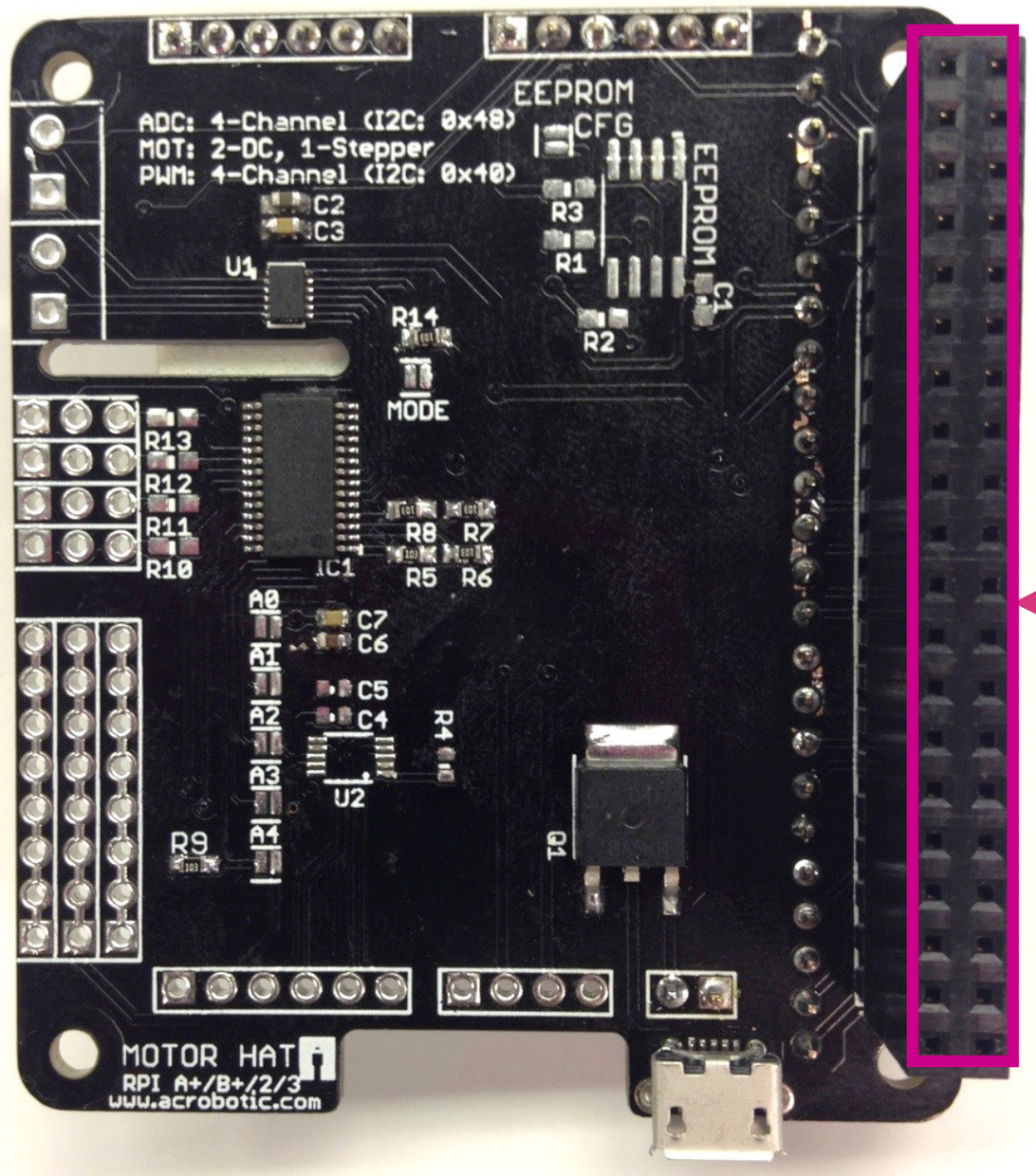




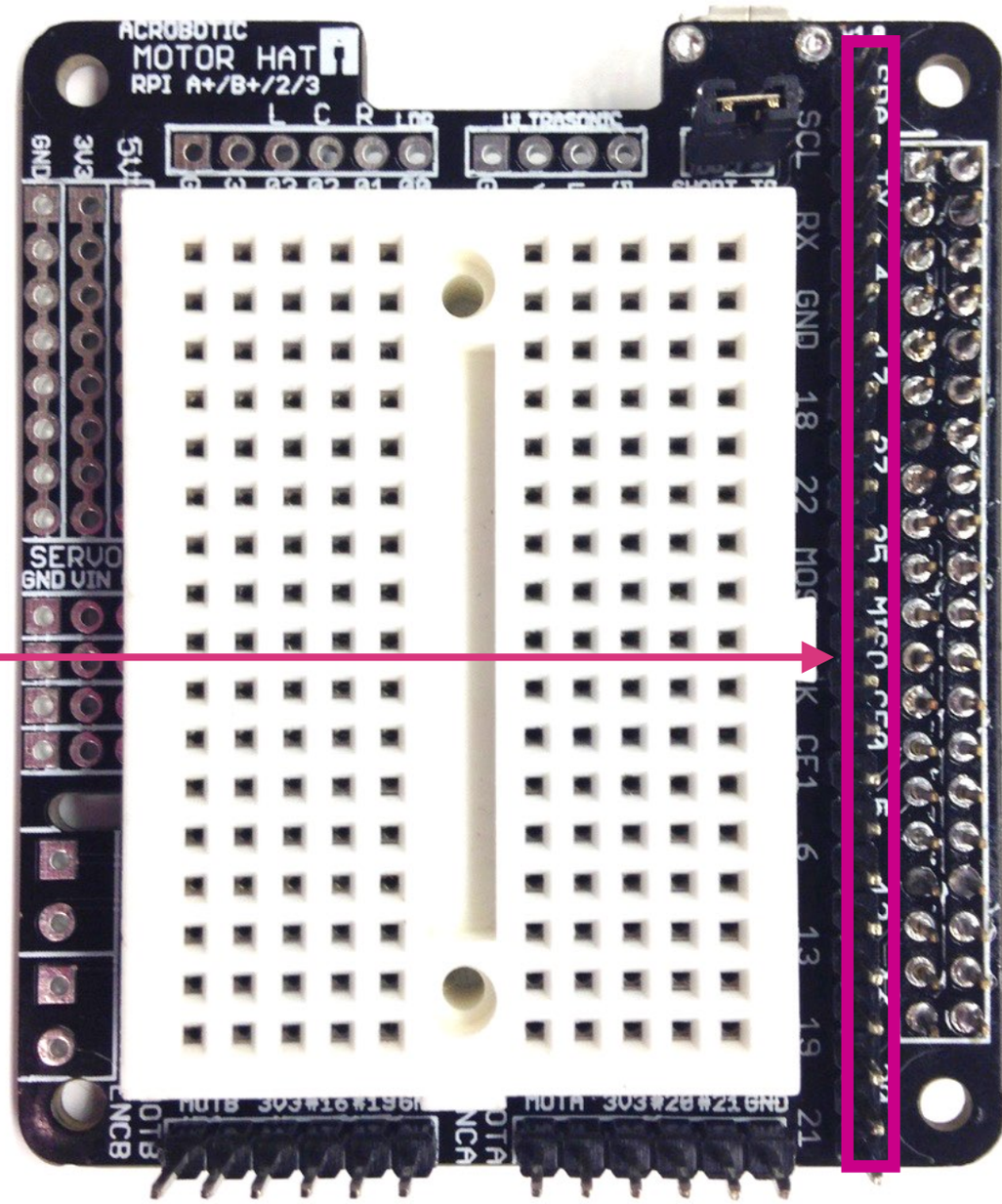


Expanding the Raspberry Pi

- Raspberry Pi functionality can be expanded by using daughterboards (aka HATs).
- **HAT**- **H**ardware **A**ttached on **T**op

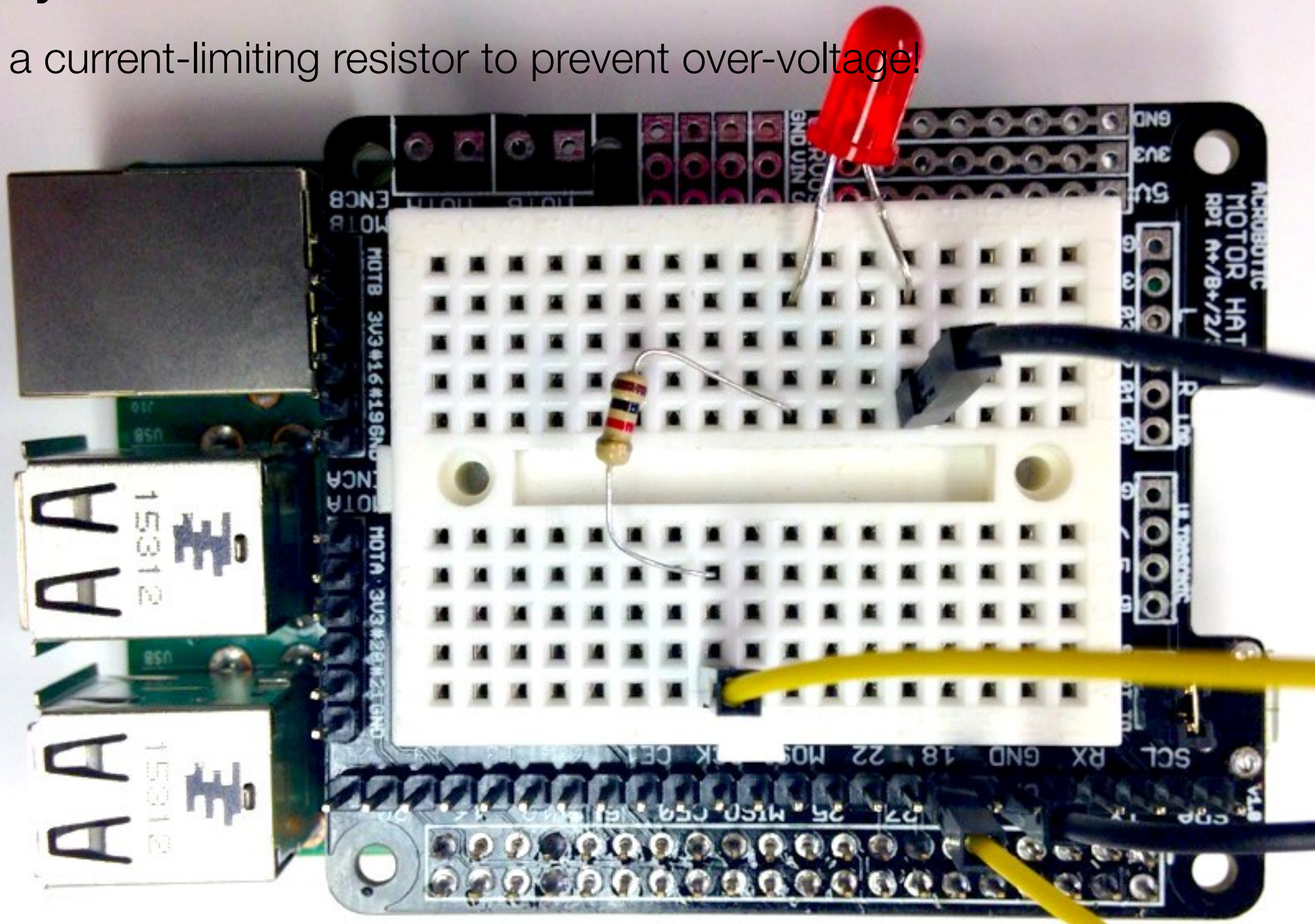


Most I/O pins are remapped



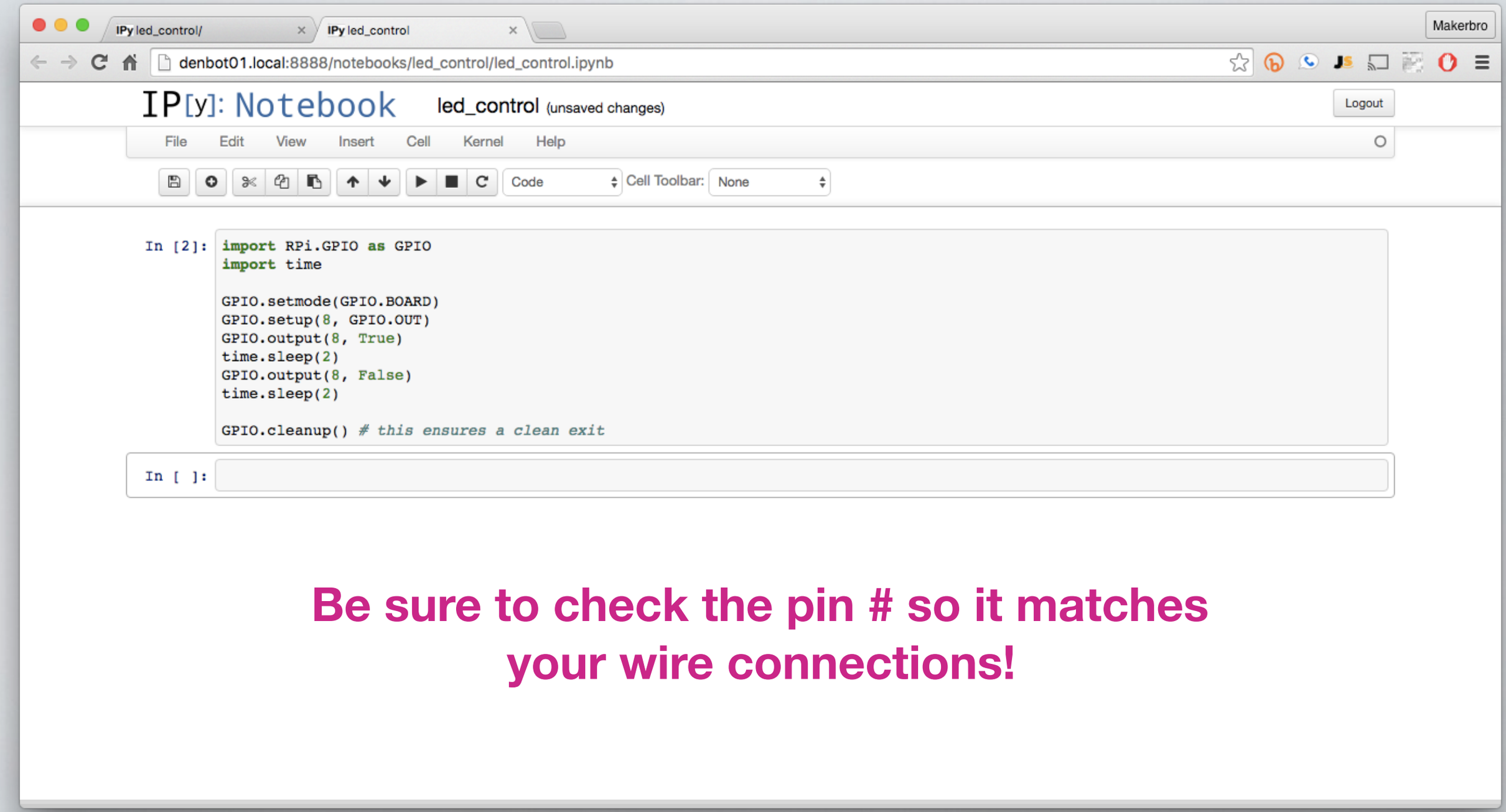
Activity 3

- Using a current-limiting resistor to prevent over-voltage!



Quiz 1

- Can you replicate the code written previously to blink an LED?



Be sure to check the pin # so it matches your wire connections!

Quiz 1

What happens if we make the `time.sleep()` delays very small?

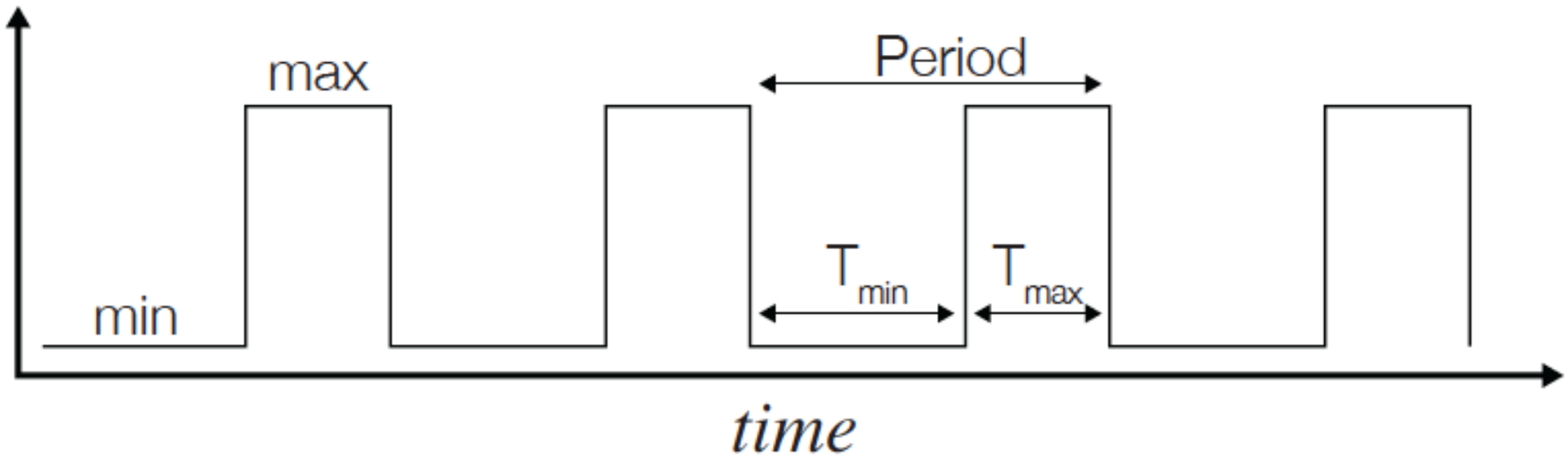
Understanding PWM – Pulse Width Modulation

- A PWM signal is a **digital signal** whose average value of voltage (and current) is controlled by turning the switch between supply and load on and off at a fast rate.

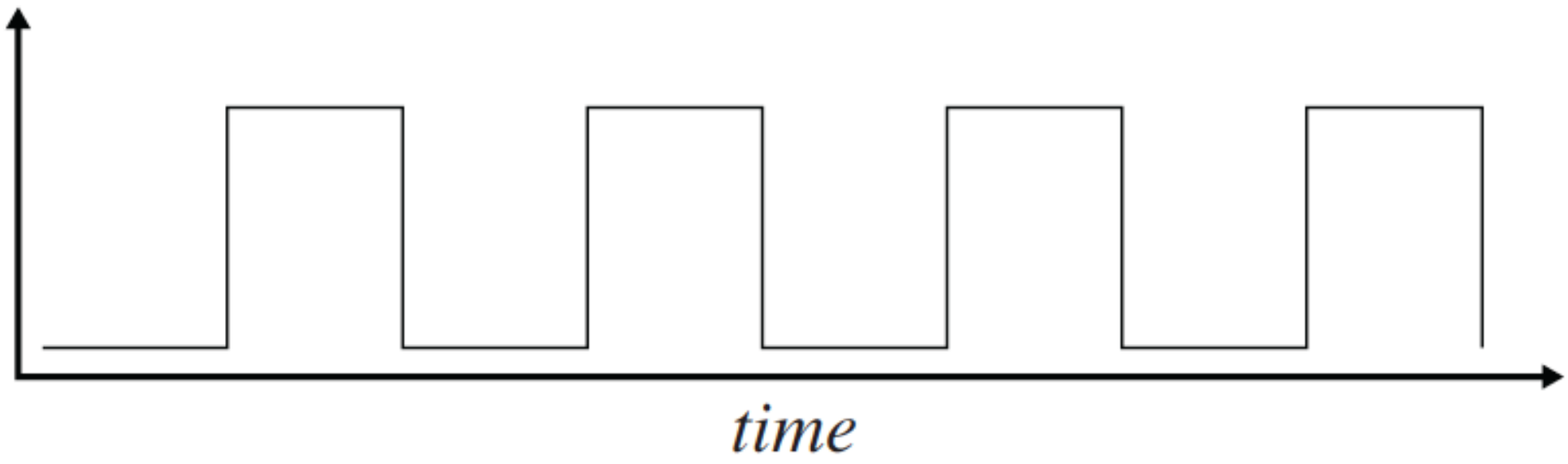
- The longer the switch is **on** compared to the **off** periods, the higher the total power supplied.

Understanding PWM-Pulse Width Modulation

Rectangular or Pulse Wave



Special case: Square Wave ($T_{min} = T_{max}$)



Frequency = 1/Period

Duty Cycle = 100% x $T_{max}/(T_{max}+T_{min})$

Understanding PWM-Pulse Width Modulation

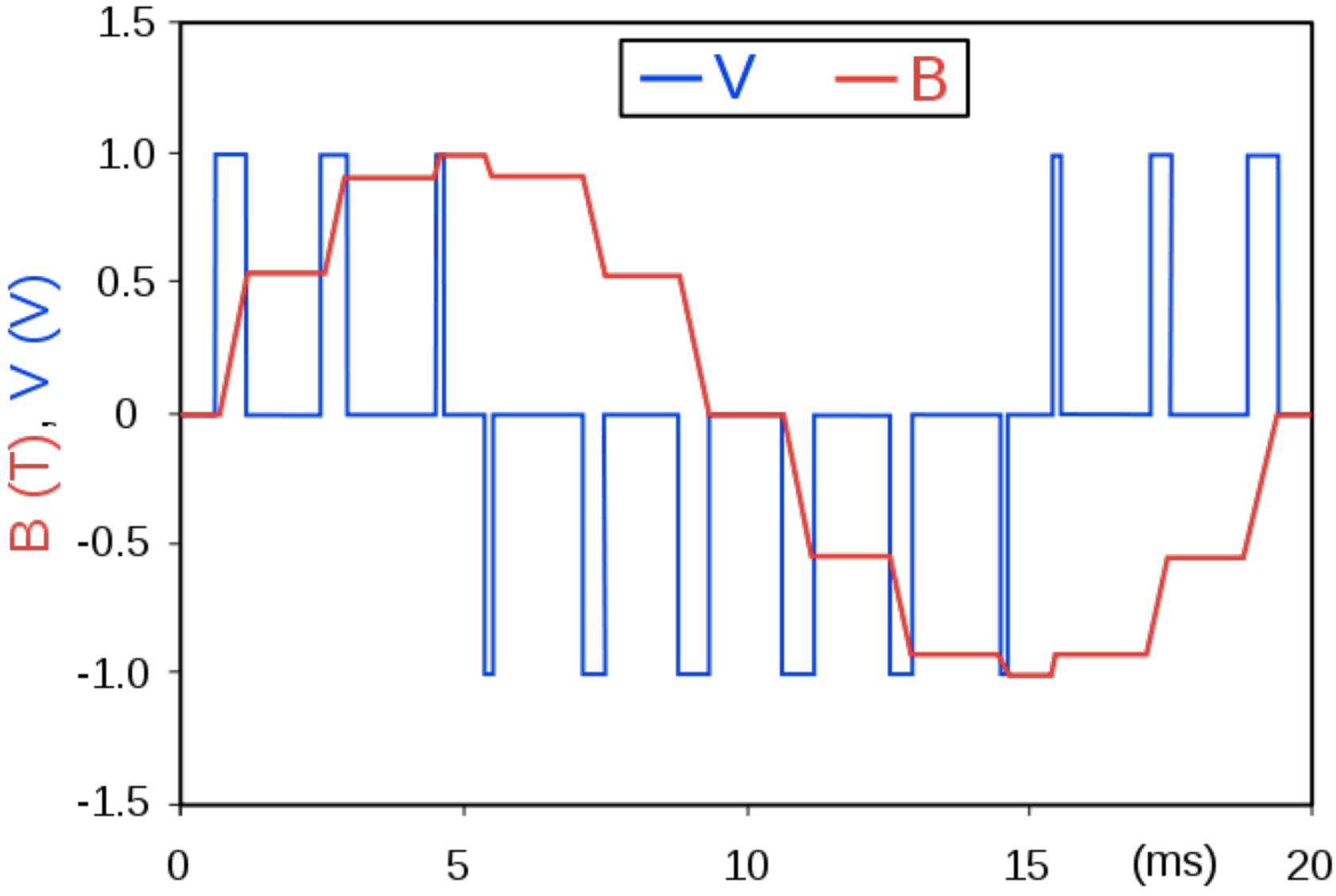
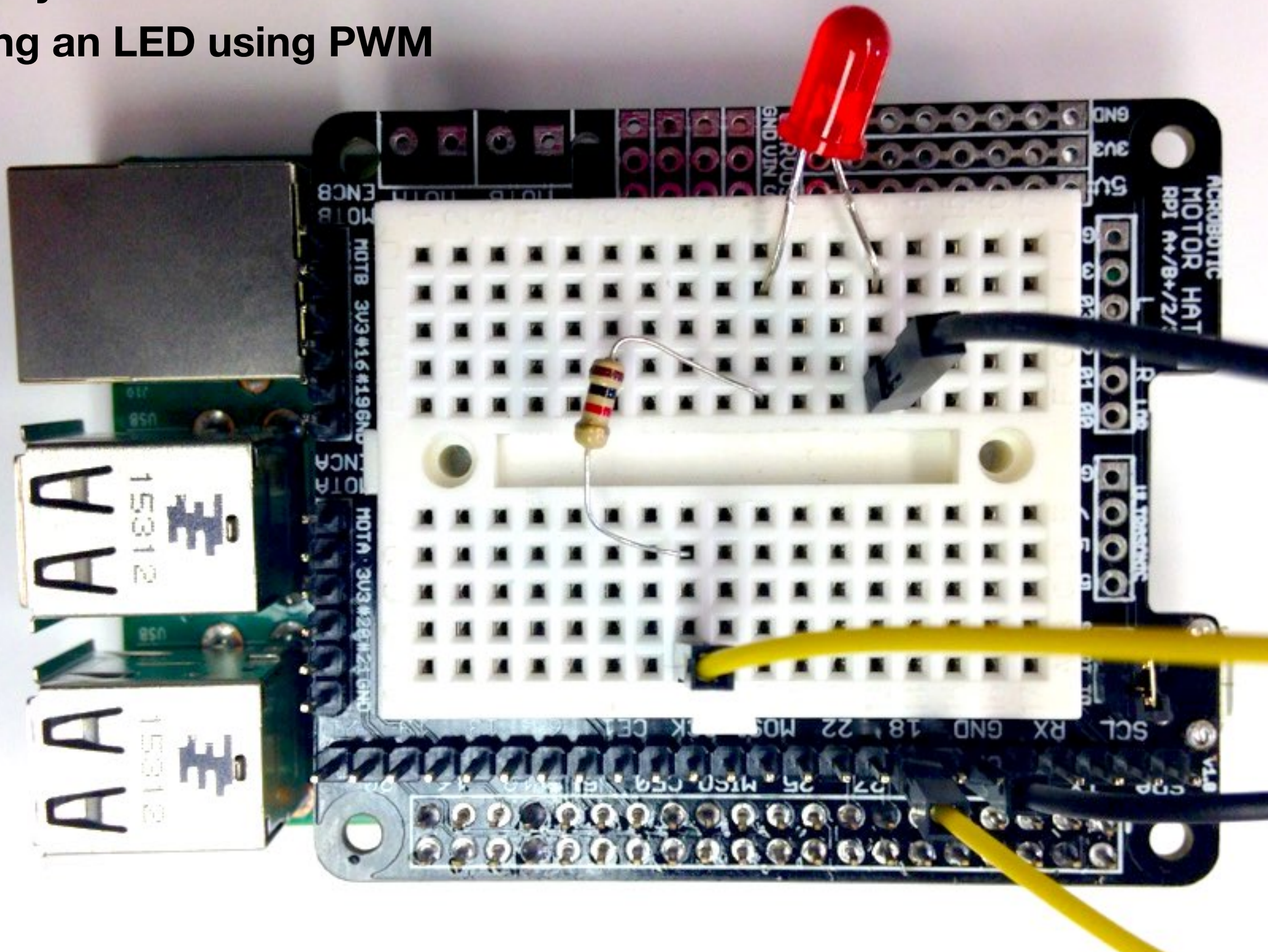


Image credit: Zureks - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1835361

Activity 3

Dimming an LED using PWM



Activity 3

- Let's use PWM for dimming an LED!

IPy led_control/ | IPy led_dimmer | Makerbro

denbot16.local:8888/notebooks/led_control/led_dimmer.ipynb

IP[y]: Notebook led_dimmer (unsaved changes) Logout

File Edit View Insert Cell Kernel Help

Code | Cell Toolbar: None

In [ ]:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

# Initialize PWM on pin 12 at 1KHz
pwm_obj = GPIO.PWM(12, 1000)

# Start PWM with duty cycle of 0
pwm_obj.start(0)
time.sleep(1)
# Change PWM to a 25% duty cycle
pwm_obj.ChangeDutyCycle(25)
time.sleep(1)
# Change PWM to a 50% duty cycle
pwm_obj.ChangeDutyCycle(50)
time.sleep(1)
# Change PWM to a 100% duty cycle
pwm_obj.ChangeDutyCycle(100)
time.sleep(1)

GPIO.cleanup()
```