

# 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

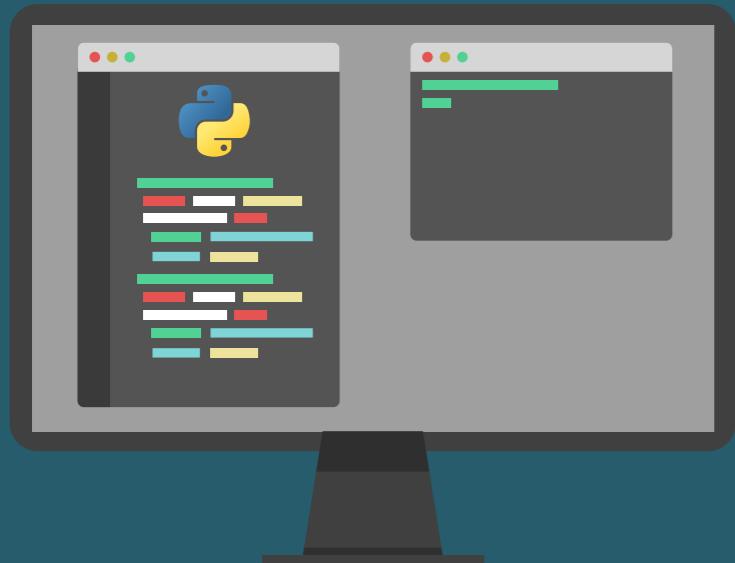
Using a Sensor

Velocity Control Using Feedback

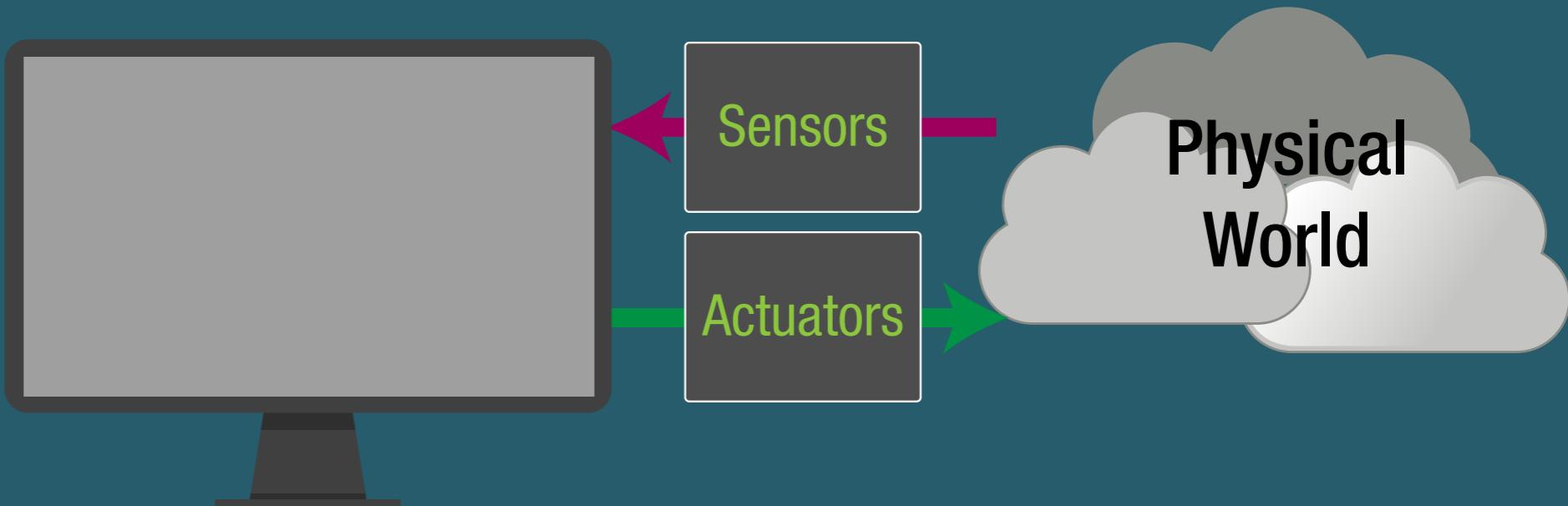
# Introducing Physical Computing

# What is Physical Computing?

‘Virtual’ Computing



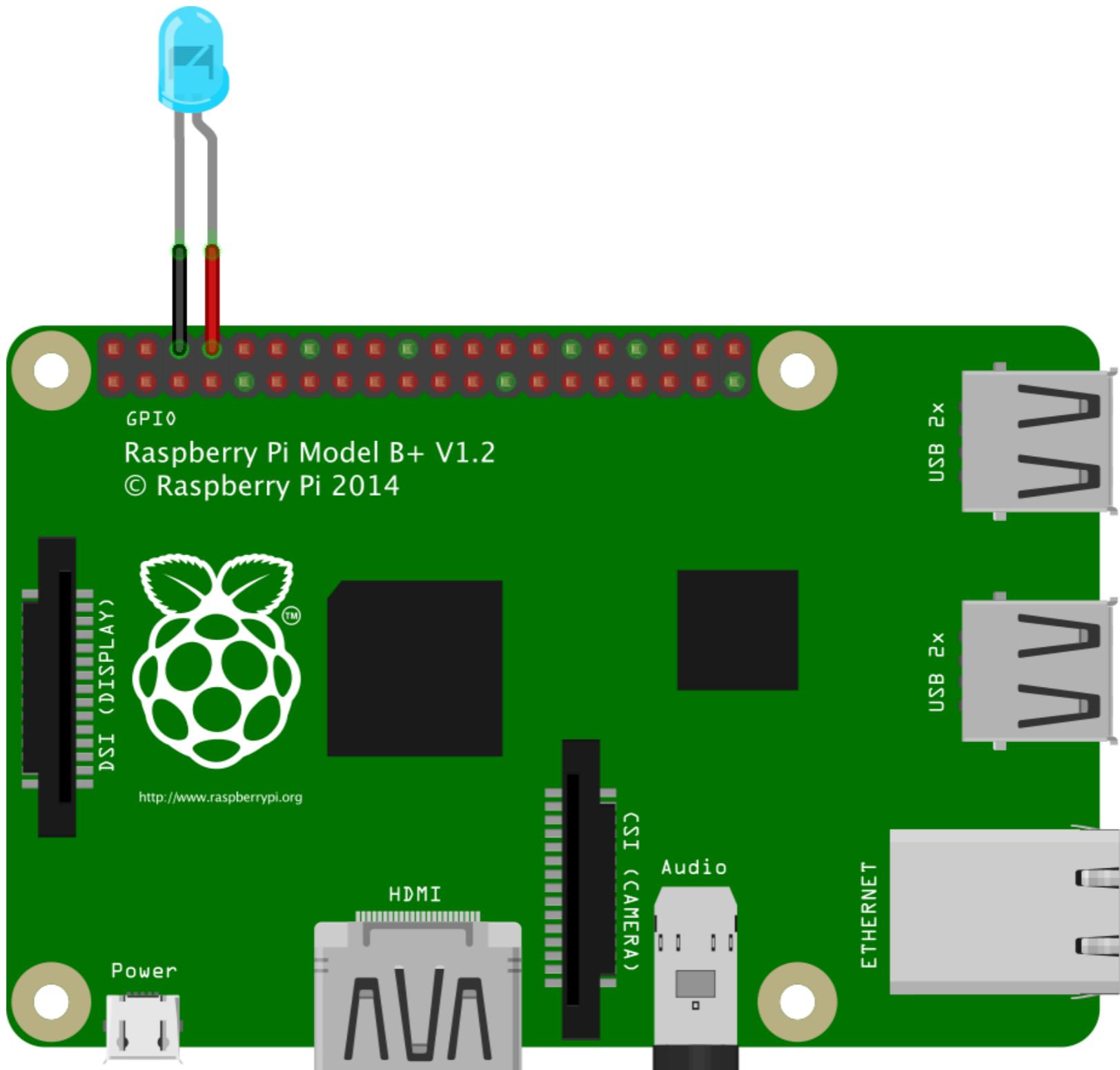
‘Physical’ Computing



- **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

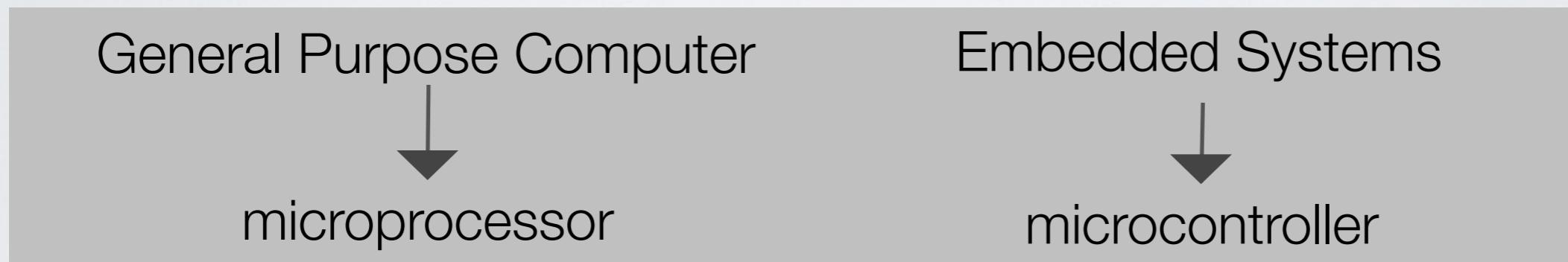
- Definition: small computer system with a **dedicated** function/program.
- Only carry out pre-defined task(s) within a device/machine (embedded).
- Fixed capability vs. general purpose computers (Macs, PCs).
- Software loaded onto Hardware = **firmware**.
- Hardware includes: memory; inputs & outputs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



# Microcontrollers

## A simple computing device

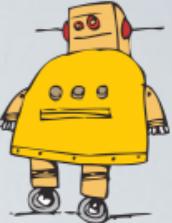
- Small computer on an IC (integrated circuit).
- Easy to program.
- Interacts with the physical world (I/O peripherals).
- Heart of **Embedded Systems**.



## 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



instructables

- 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.



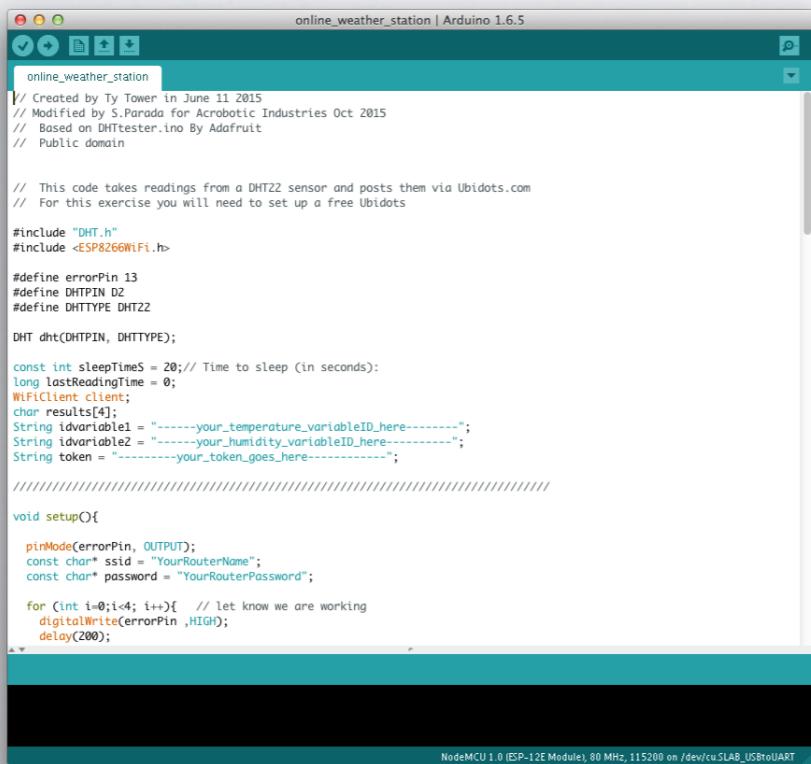
O'REILLY  
0 74470 25728 8  
5 79

kits.makezine.com

# 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.



The screenshot shows the Arduino IDE interface with the title bar "online\_weather\_station | Arduino 1.6.5". The code is for an "online\_weather\_station" project. It includes comments about the creation date (June 11 2015), modification date (Oct 2015), and source (DHTtester.ino by Adafruit). It defines pins for a DHT22 sensor and an ESP8266 WiFi module. The code sets up a WiFi client and defines variables for temperature, humidity, and token. In the setup() function, it initializes pins and prints a message to the serial port. The code uses the DHT and WiFi libraries.

```
// Created by Ty Tower in June 11 2015
// Modified by S.Parada for Acrobotic Industries Oct 2015
// Based on DHTtester.ino By Adafruit
// Public domain

// This code takes readings from a DHT22 sensor and posts them via Ubidots.com
// For this exercise you will need to set up a free Ubidots

#include "DHT.h"
#include <ESP8266WiFi.h>

#define errorPin 13
#define DHTPIN D2
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

const int sleepTime5 = 20;// Time to sleep (in seconds);
long lastReadingTime = 0;
WiFiClient client;
char results[4];
String idvariable1 = "-----your_temperature_variableID_here-----";
String idvariable2 = "-----your_humidity_variableID_here-----";
String token = "-----your_token_goes_here-----";

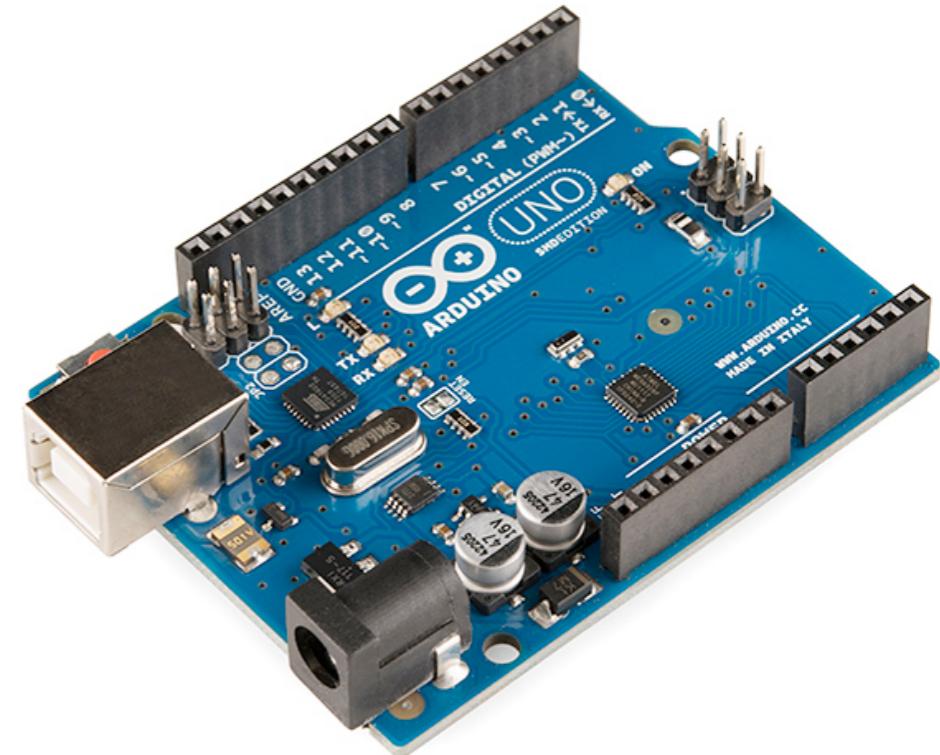
////////////////////////////

void setup(){
  pinMode(errorPin, OUTPUT);
  const char* ssid = "YourRouterName";
  const char* password = "YourRouterPassword";

  for (int i=0;i<4; i++){ // let know we are working
    digitalWrite(errorPin ,HIGH);
    delay(200);
  }
}

void loop(){
  if (millis() - lastReadingTime > sleepTime5 * 1000) {
    lastReadingTime = millis();
    dht.read();
    float temp = dht.temperature();
    float hum = dht.humidity();

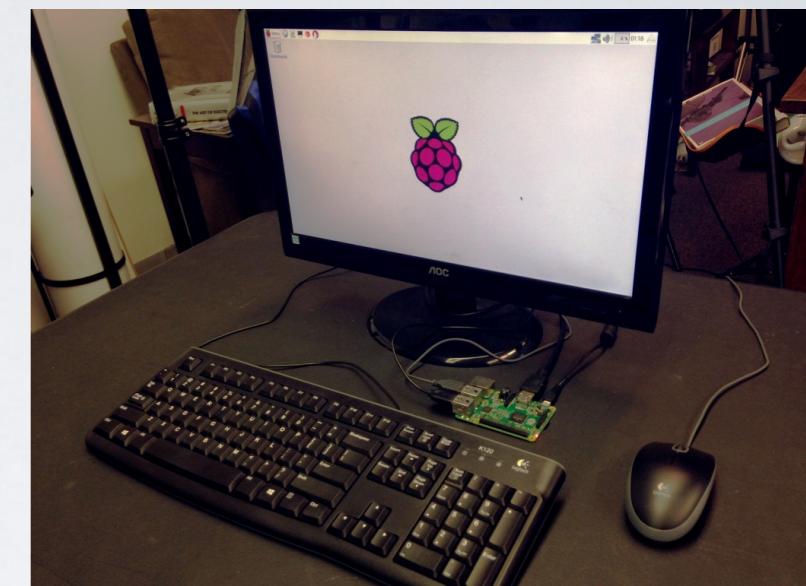
    String result = "Temperature: " + String(temp) + "C\nHumidity: " + String(hum) + "%\n";
    client.print(result);
    Serial.println(result);
  }
}
```



# Popular Development Platforms

**Raspberry Pi** enters the scene in 2008

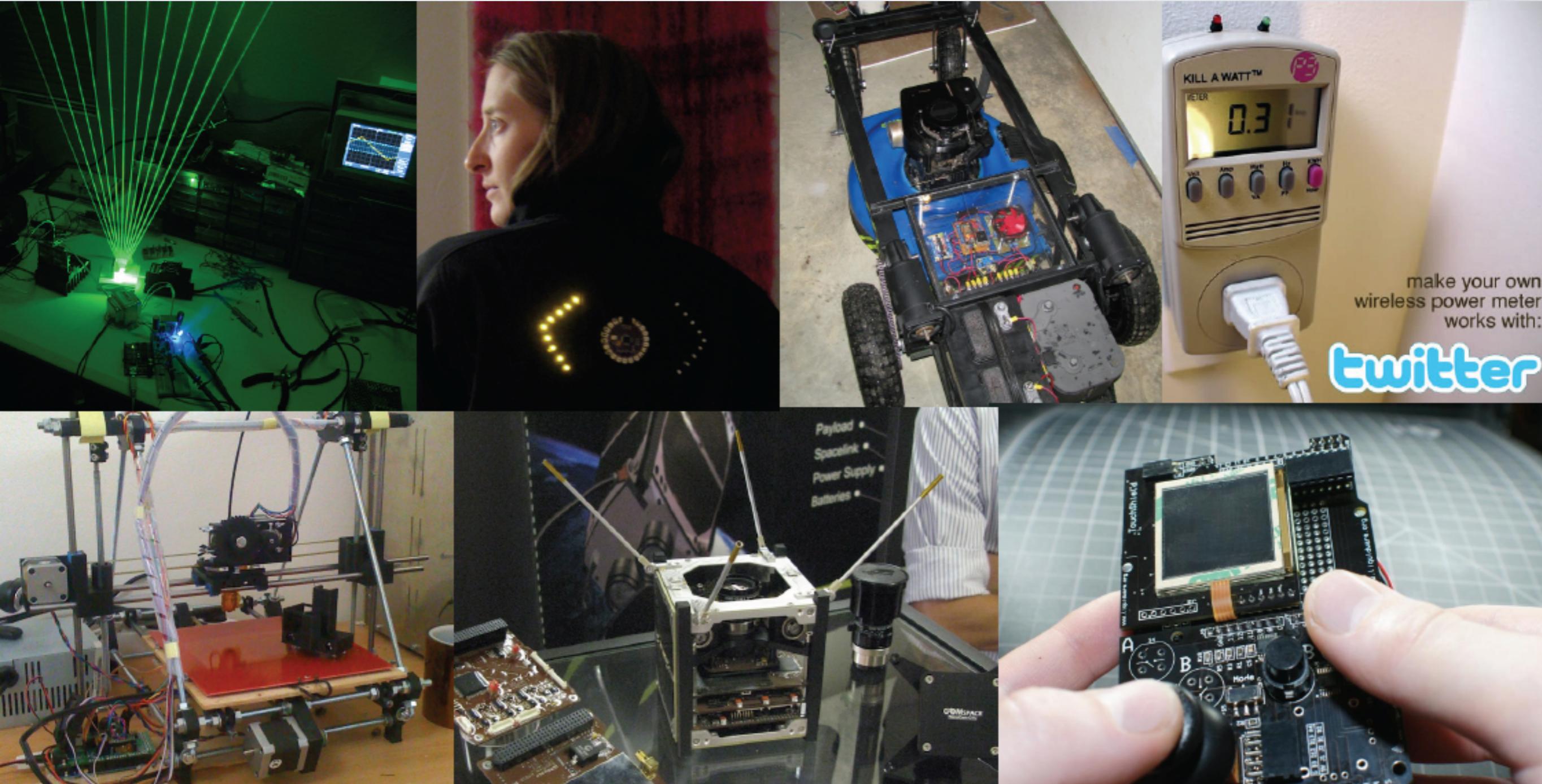
- Focus is on accessible computers for kids.
- Developed and made in the UK as answer to lack of CS interest.
- Some parts are not Open-Source.
- Expansion boards are HATs.
- Built-in WiFi connectivity.



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



# Uses and Applications

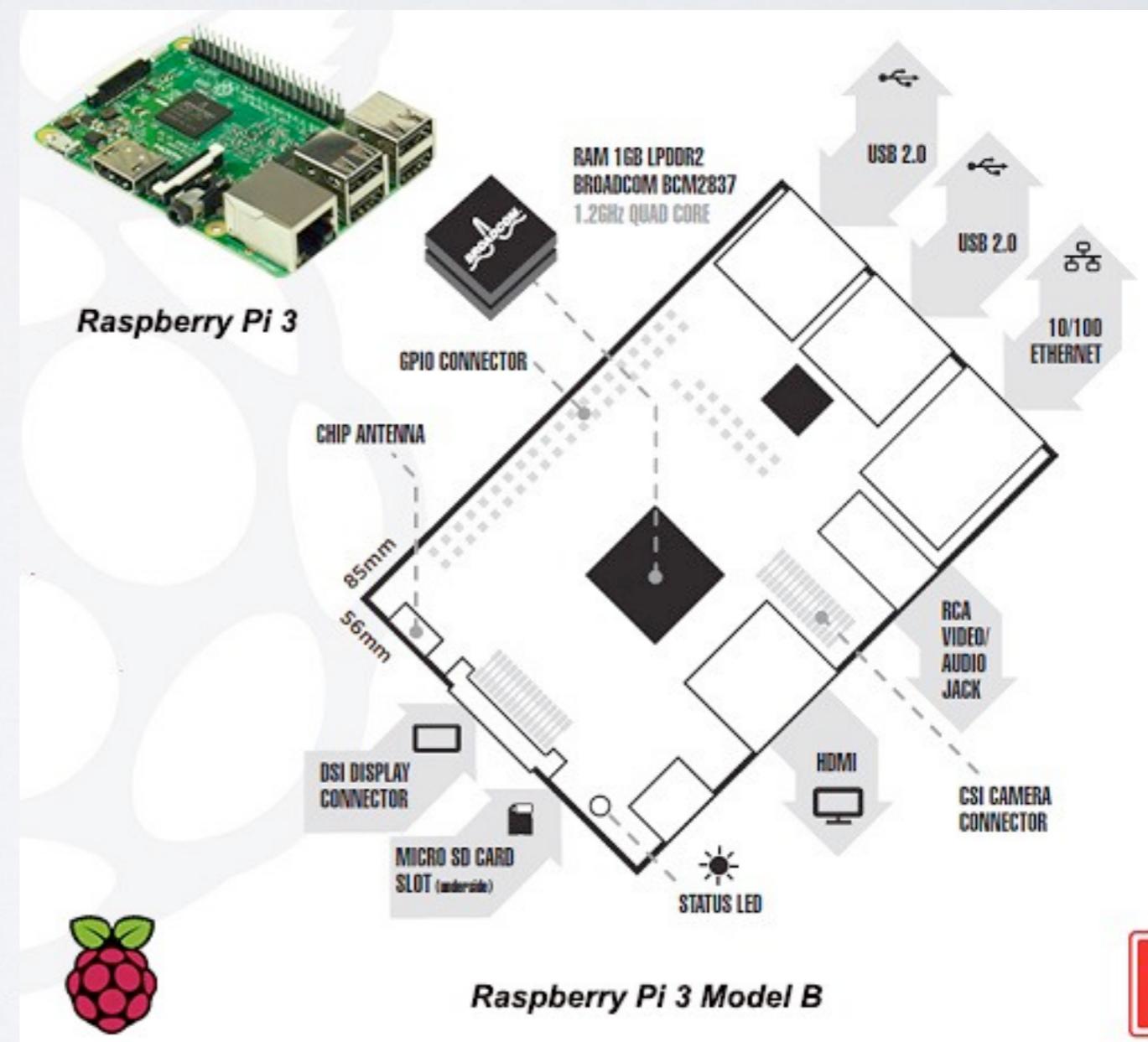


# Raspberry Pi - Under the hood

- 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.



Raspberry Pi 3 Model B



# **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.

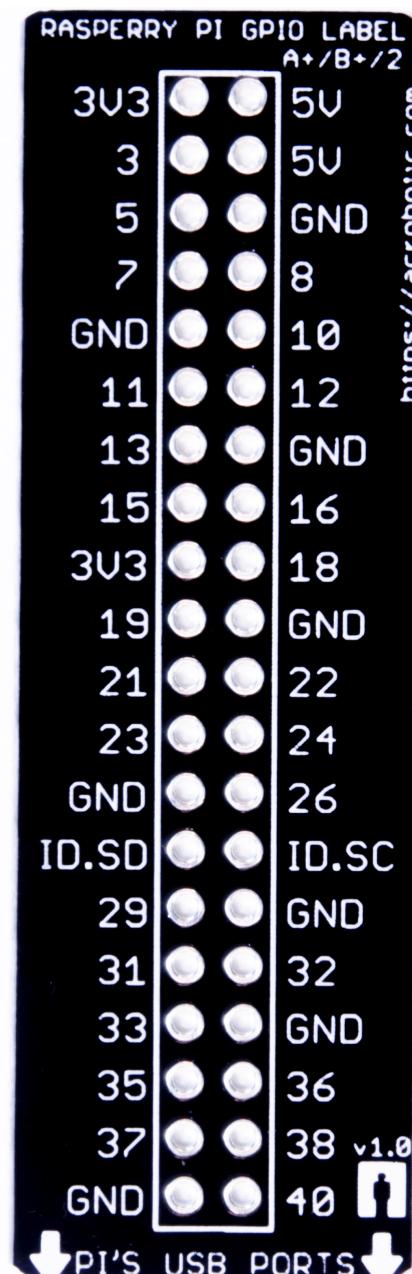
**CLK**- Synchronizes system actions that are time sensitive.

**UART**- Access to serial console for advanced stuff.

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

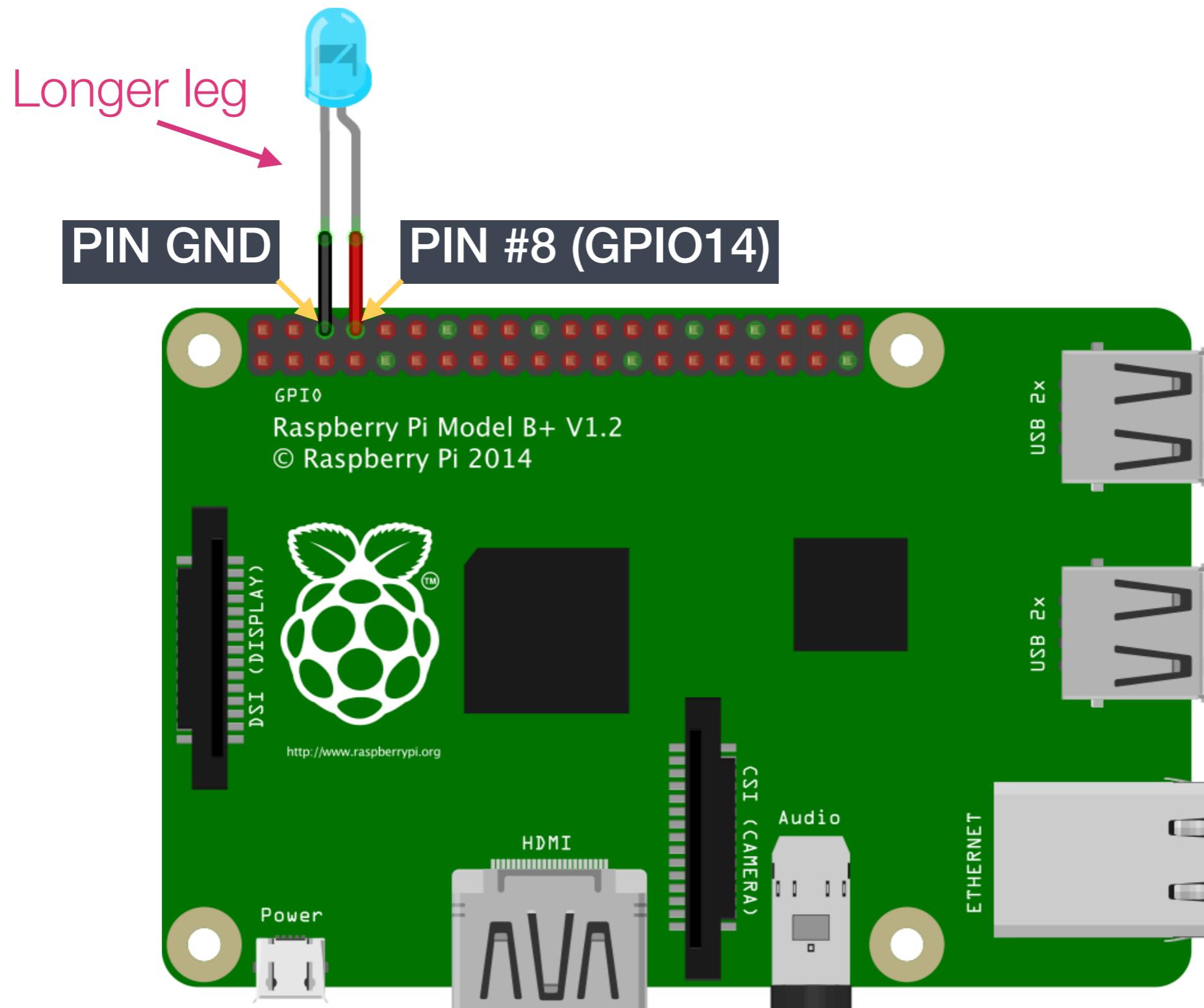
**PWM**- “Analog style” signals that provide variable power.

	3V3	1	2	5V0
GPIO 2	SDA0	3	4	5V0
GPIO 3	SCL0	5	6	GND
GPIO 4	GPCLK0	7	8	TxD
	GND	9	10	RxD
GPIO 17	P17	11	12	PWM
GPIO 27	P27	13	14	GND
GPIO 22	P22	15	16	P23
	3V3	17	18	P24
GPIO 10	MOSI	19	20	GND
GPIO 9	MISO	21	22	P25
GPIO 11	SCLK	23	24	Ce0
	GND	25	26	Ce1



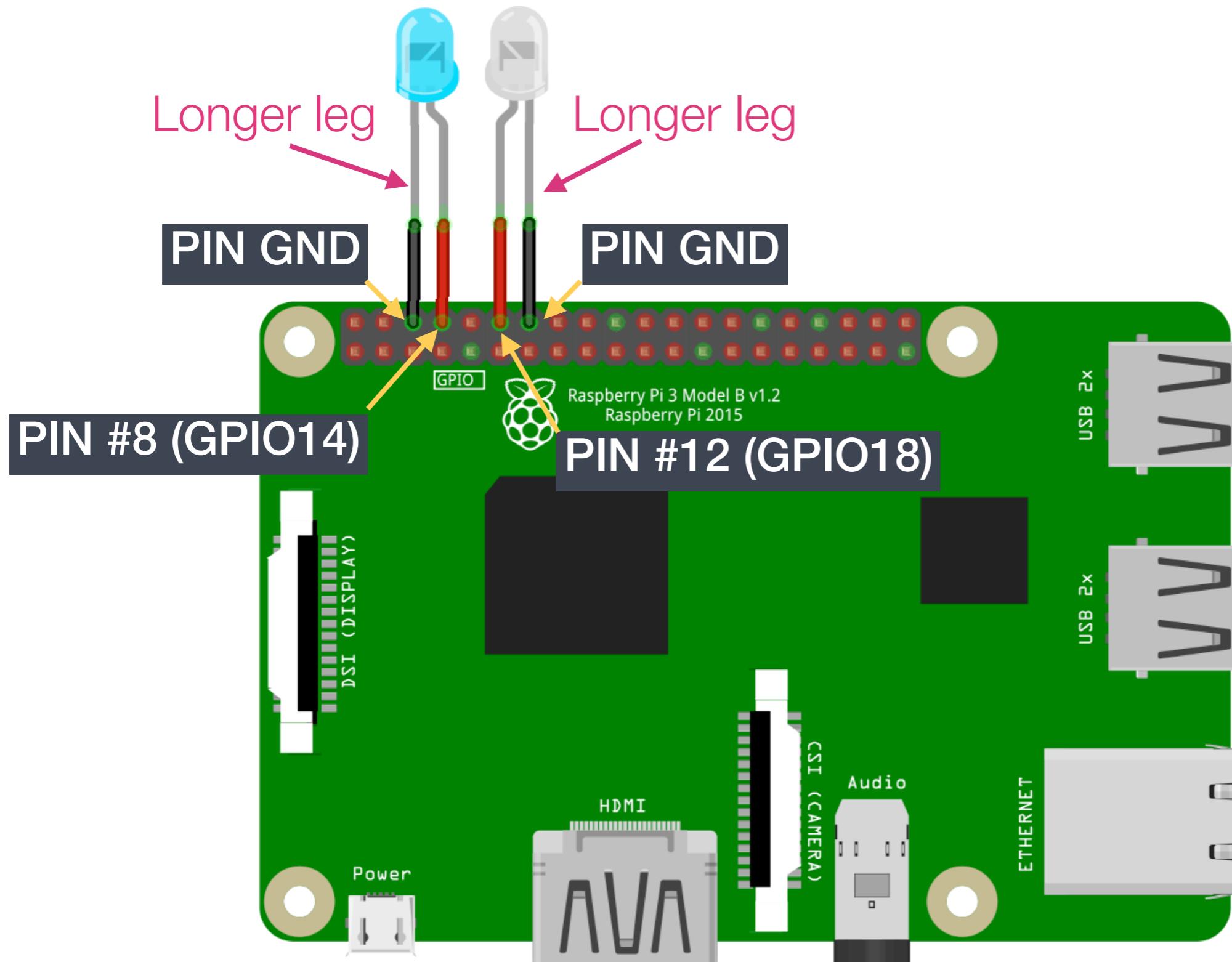
# Activity 1

## Challenge: Recreate LED blinking system



# Activity 2

## Add a 2nd LED and alternate blinking patterns



# Analog vs. Digital Signals

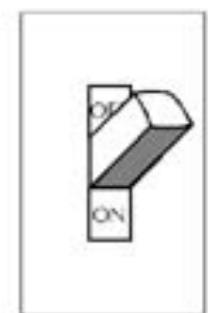
## Signals

- 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.

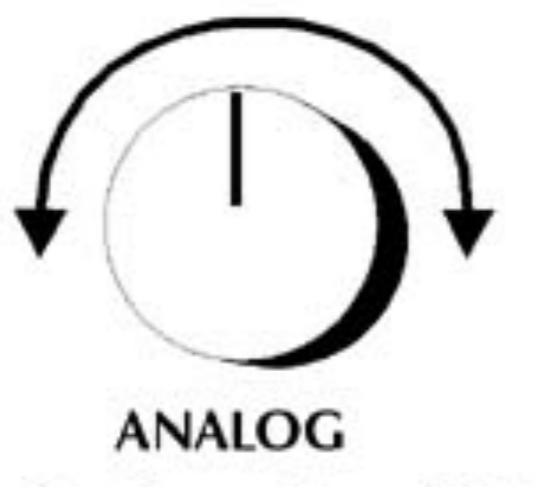


DIGITAL  
On or off

# Analog vs. Digital Signals

## Analog

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



# **Hardware Components**

# Components

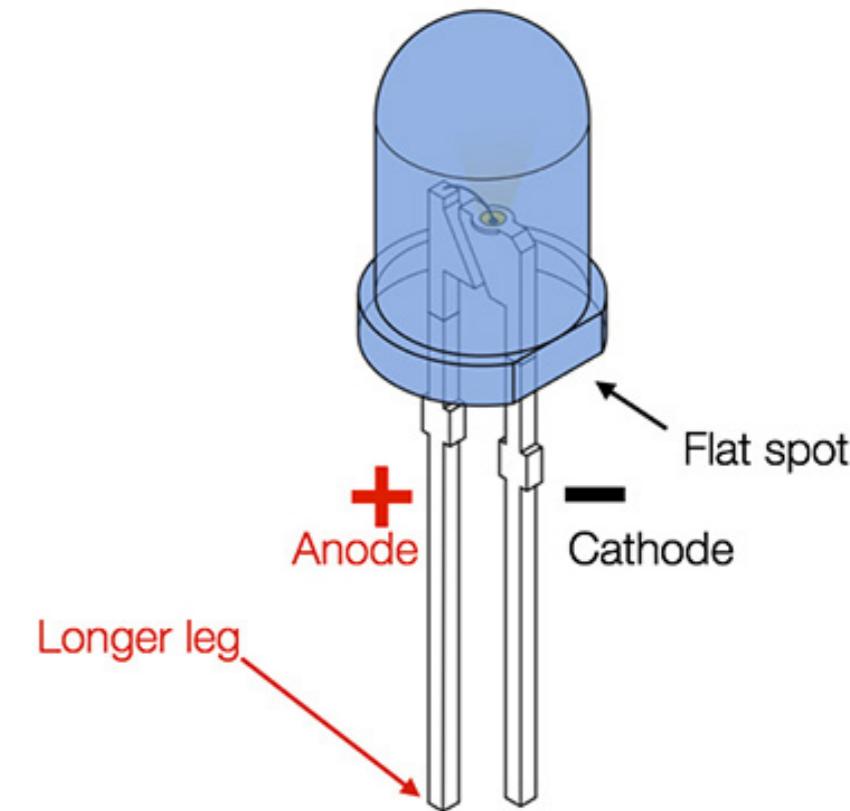
## Resistors

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



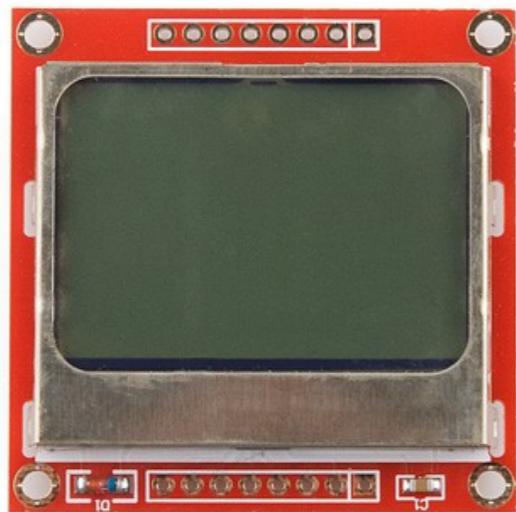
## Light Emitting Diodes

- 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.

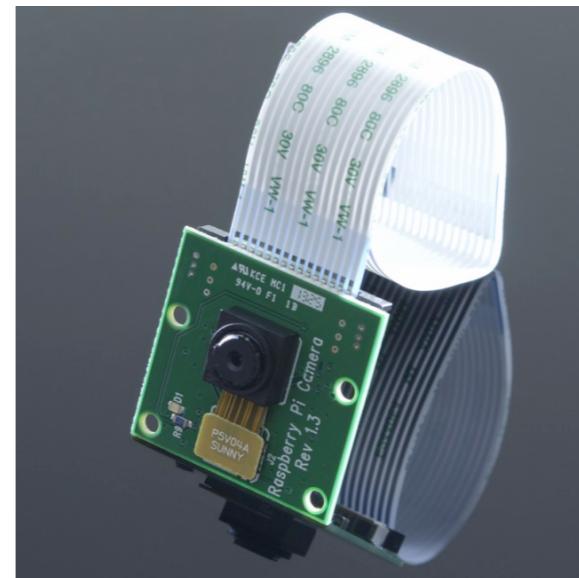


# Other Components

**LCD**



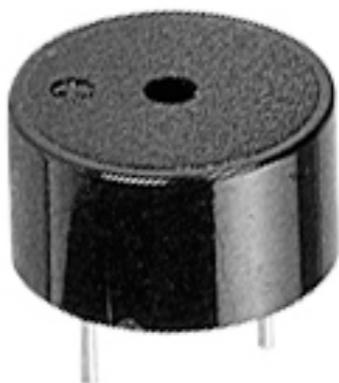
**Camera**



**Motor**

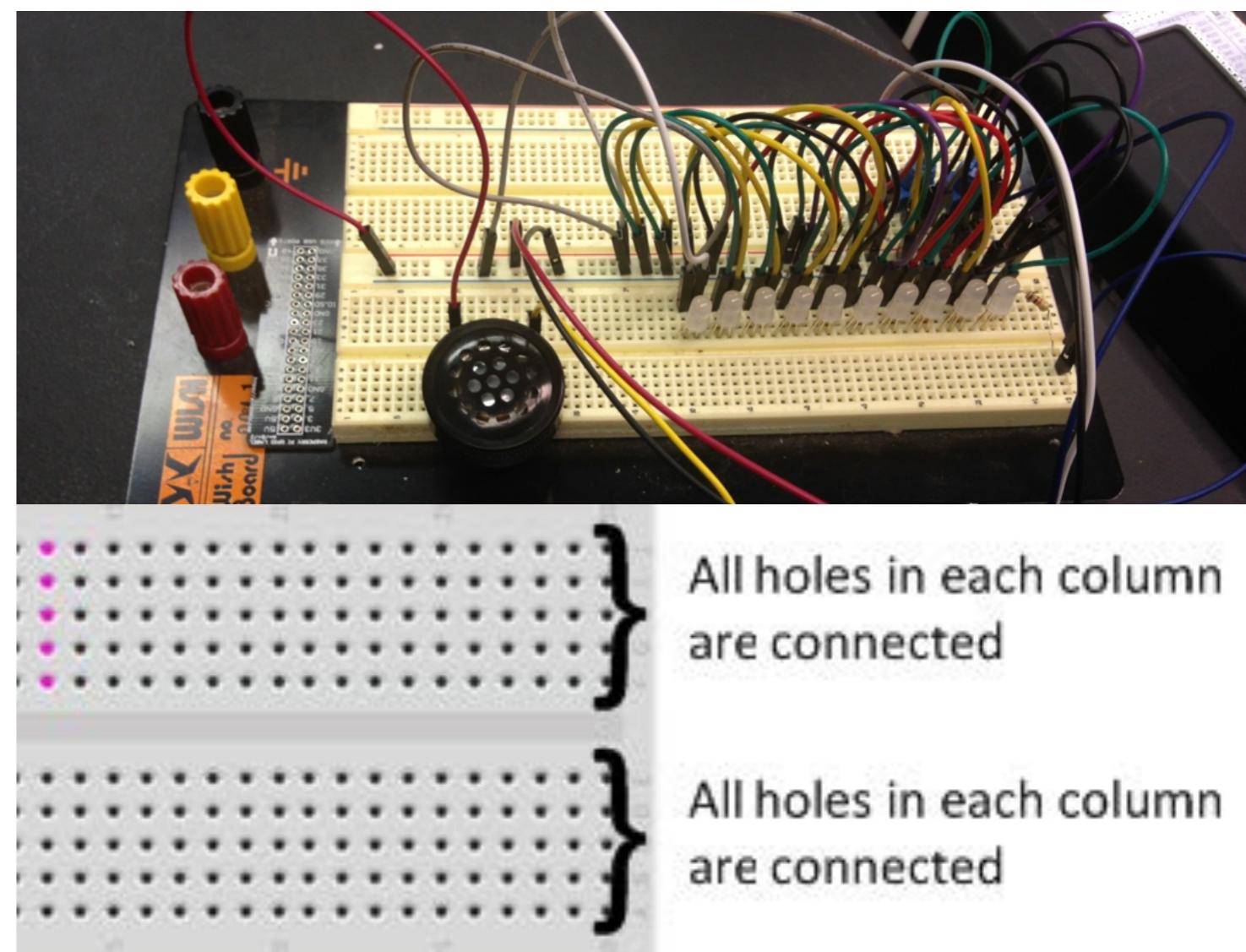
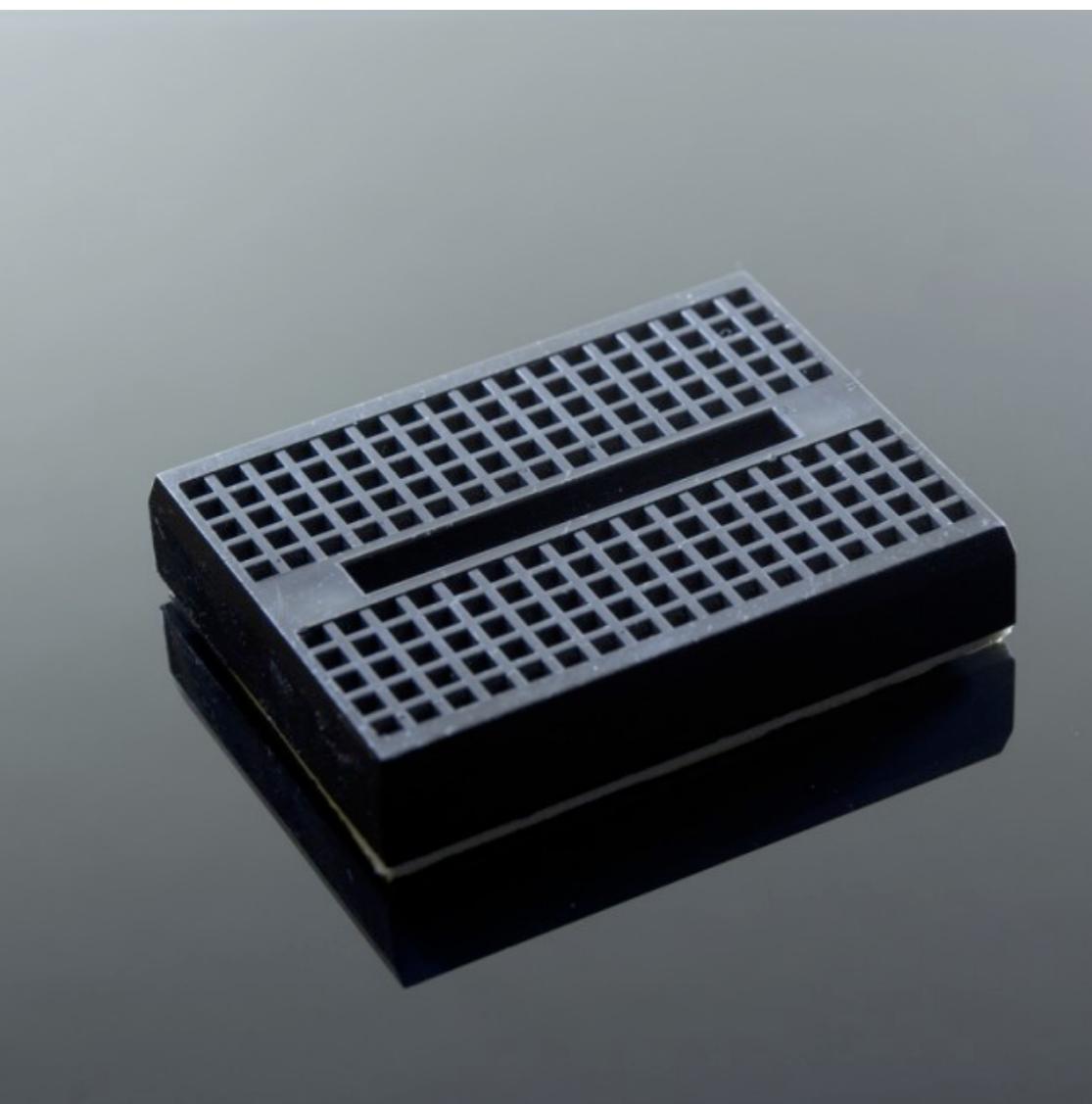


**Speaker**



# Using A Breadboard

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

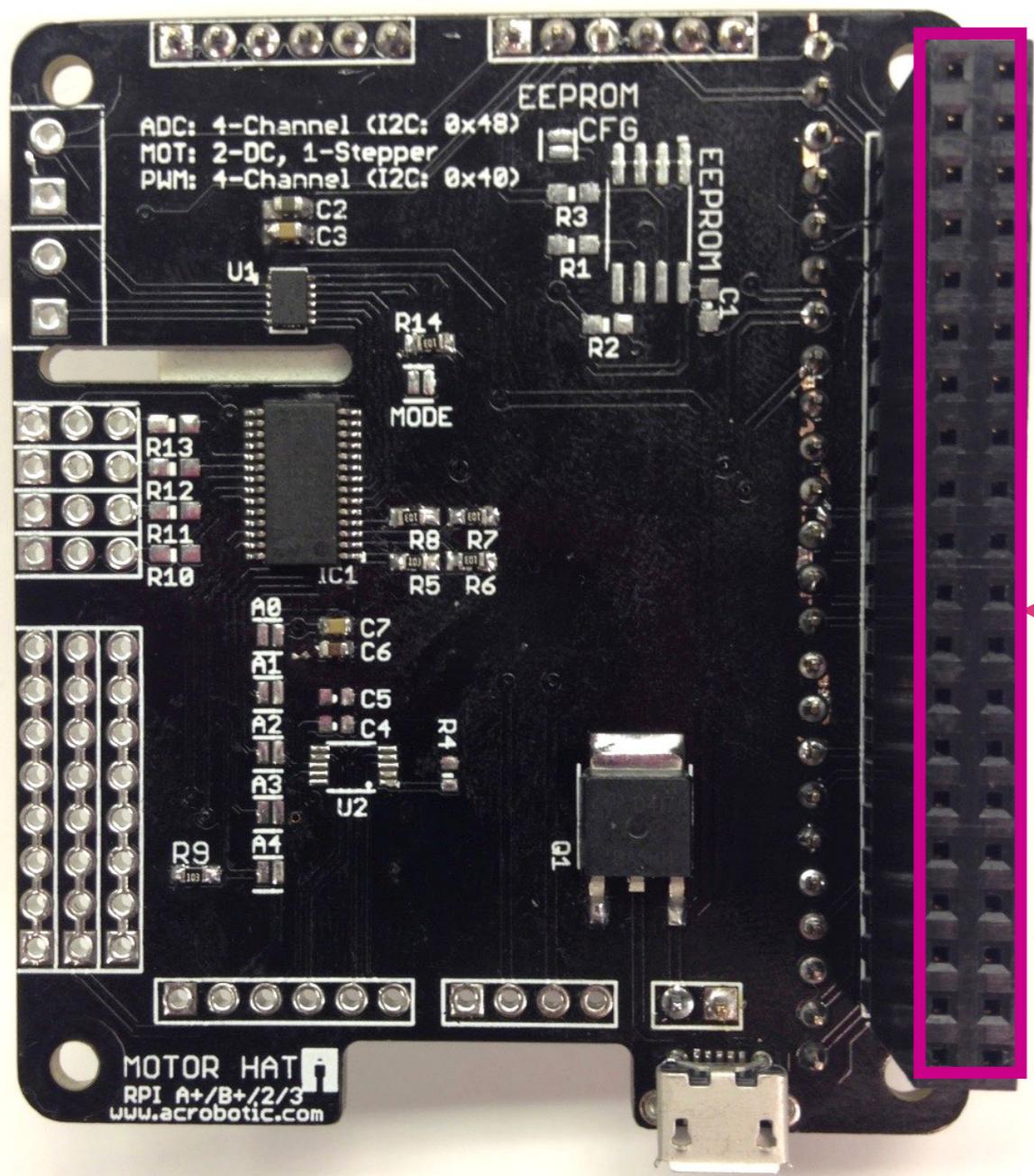


All holes in each column  
are connected

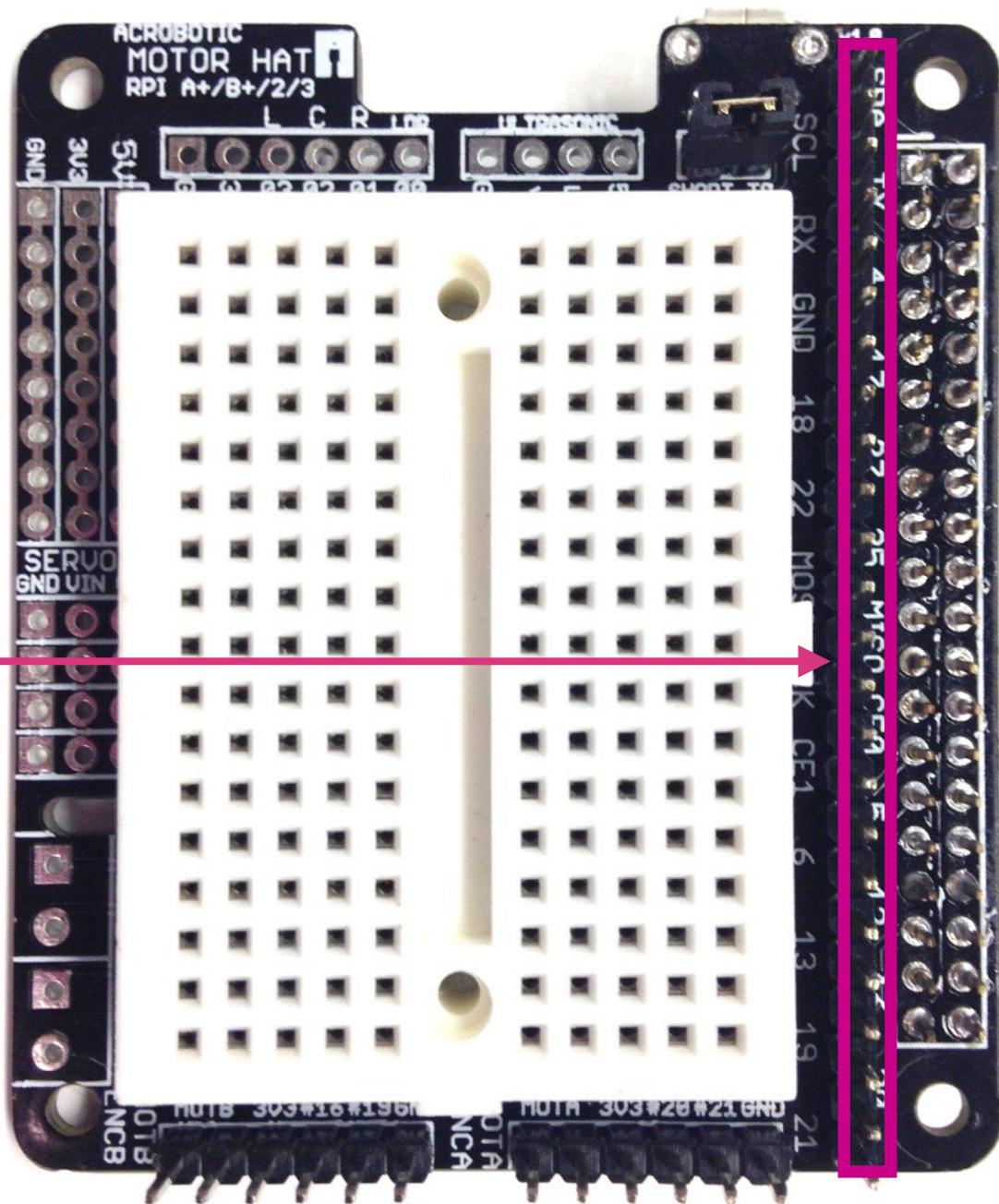
All holes in each column  
are connected

# Expanding the Raspberry Pi

- Raspberry Pi functionality can be expanded by using daughterboards (aka HATs).
- **HAT- Hardware Attached on Top**

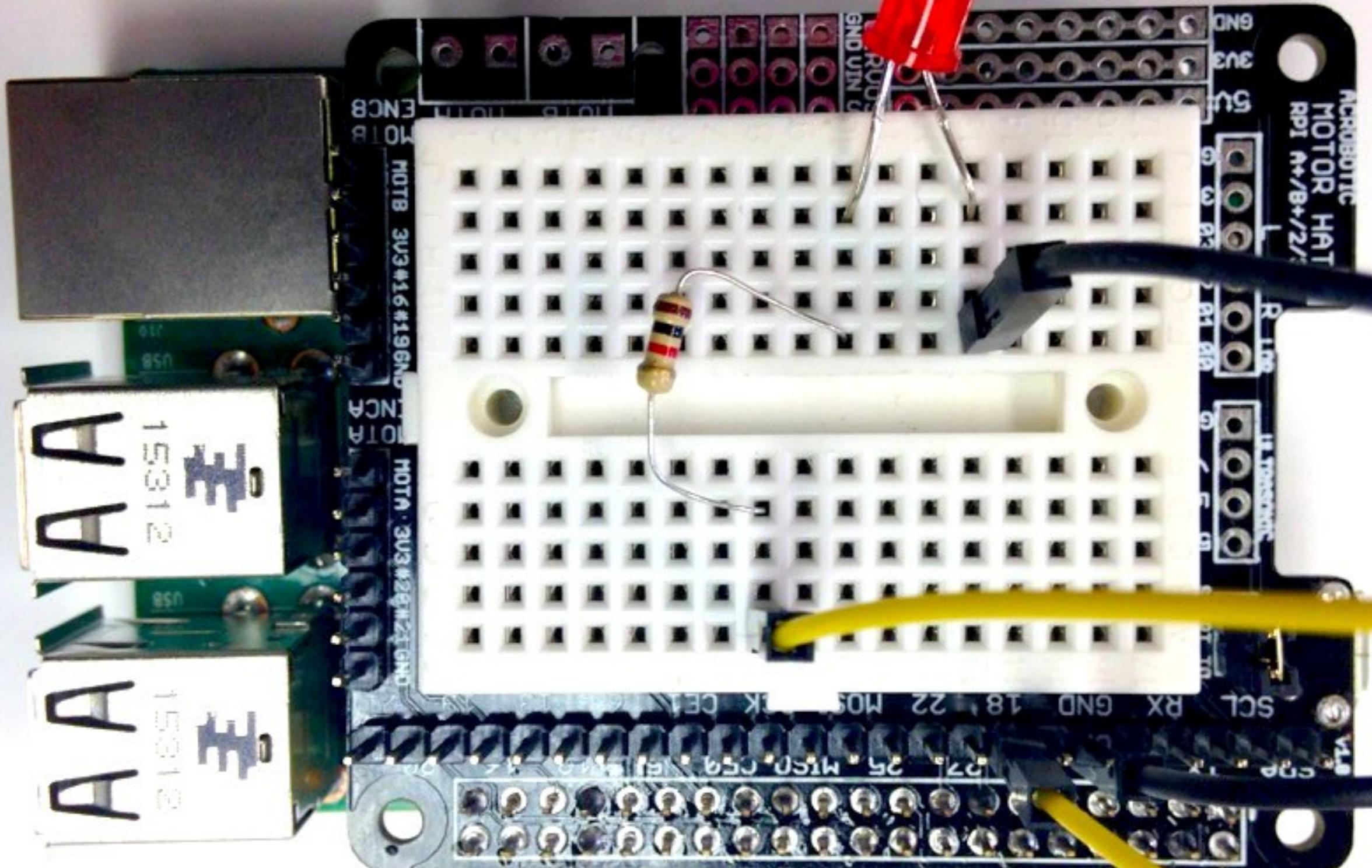


Most I/O pins  
are remapped



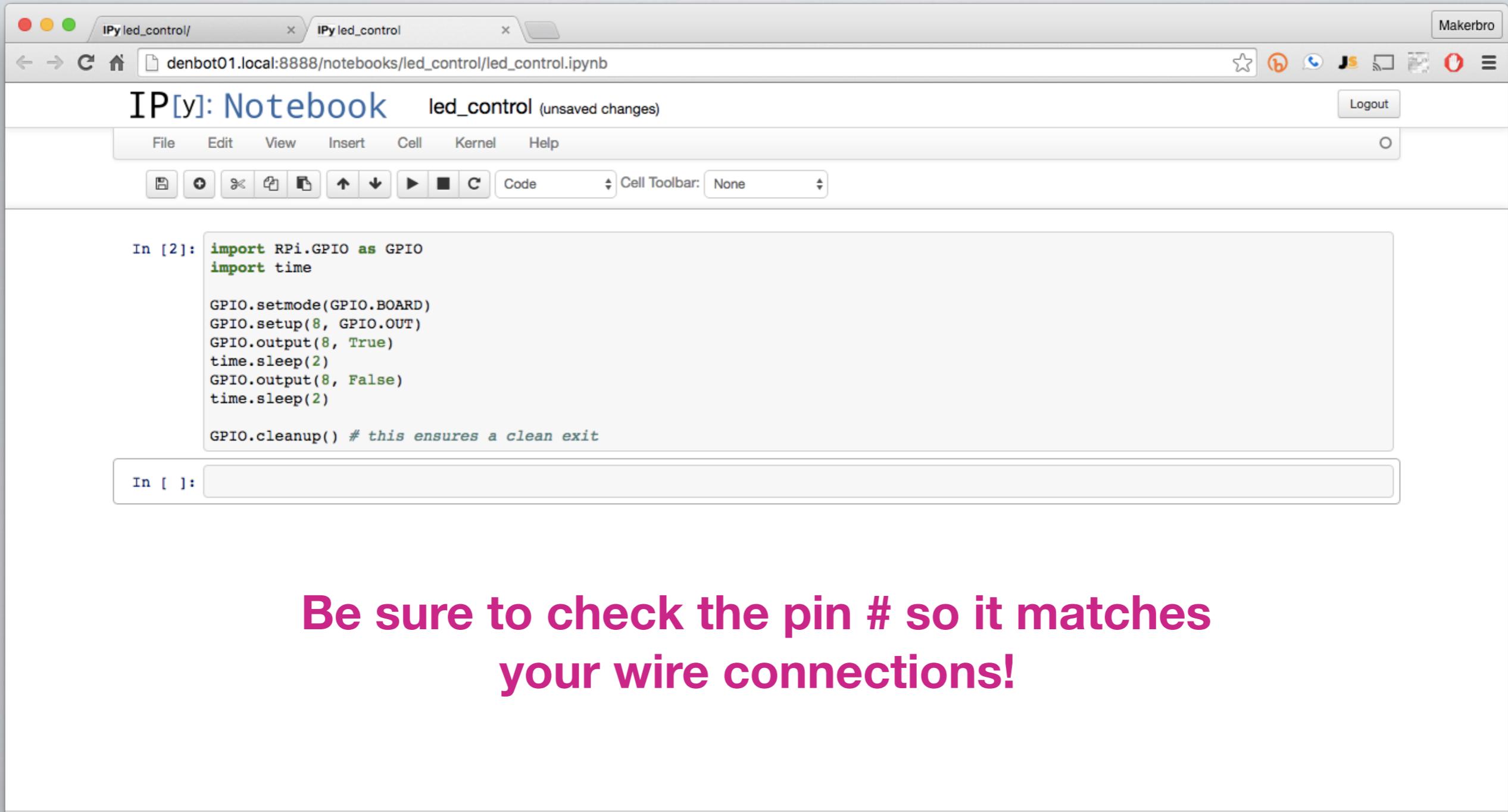
# Activity 3

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



# Quiz 1

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



The screenshot shows an IPython Notebook interface with the title bar "IPy led\_control" and the URL "denbot01.local:8888/notebooks/led\_control/led\_control.ipynb". The notebook is titled "IP[y]: Notebook" and shows an unsaved changes indicator. The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. The toolbar below the menu includes icons for file operations like New, Open, Save, and Cell execution. A "Cell Toolbar" dropdown is set to "None". The main area contains a code cell labeled "In [2]:" with the following Python code:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
GPIO.output(8, True)
time.sleep(2)
GPIO.output(8, False)
time.sleep(2)

GPIO.cleanup() # this ensures a clean exit
```

Below the code cell is another input field labeled "In [ ]:". A large pink text overlay at the bottom of the slide reads:

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

# Quiz 1

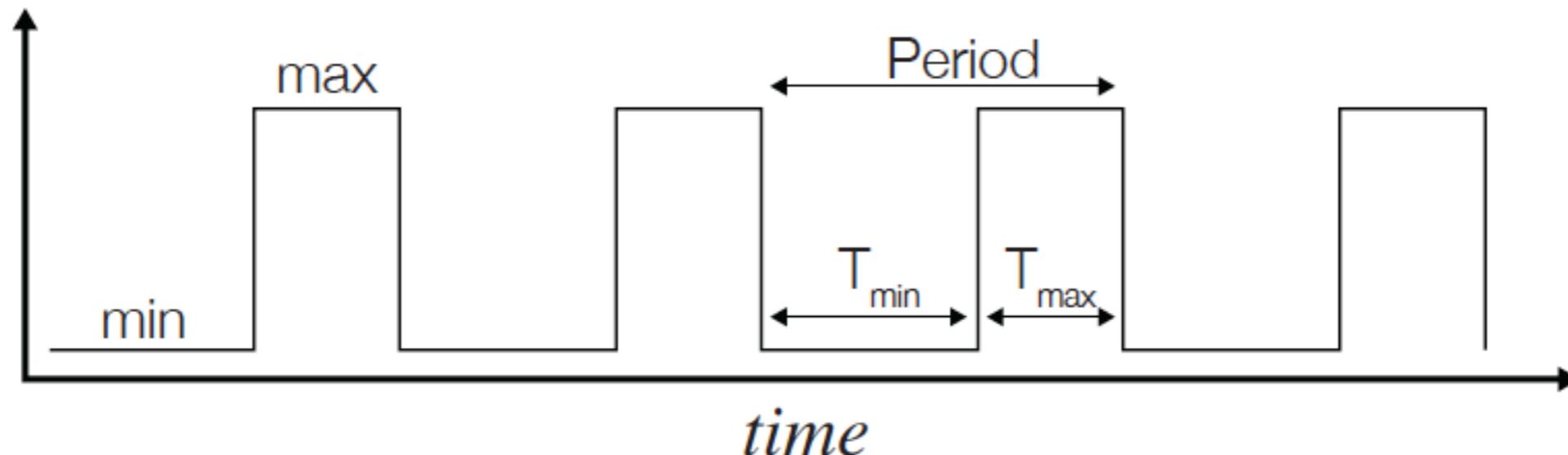
**What happens if we make the  
time.sleep () delays very small?**

# Understanding PWM – Pulse Width Modulation

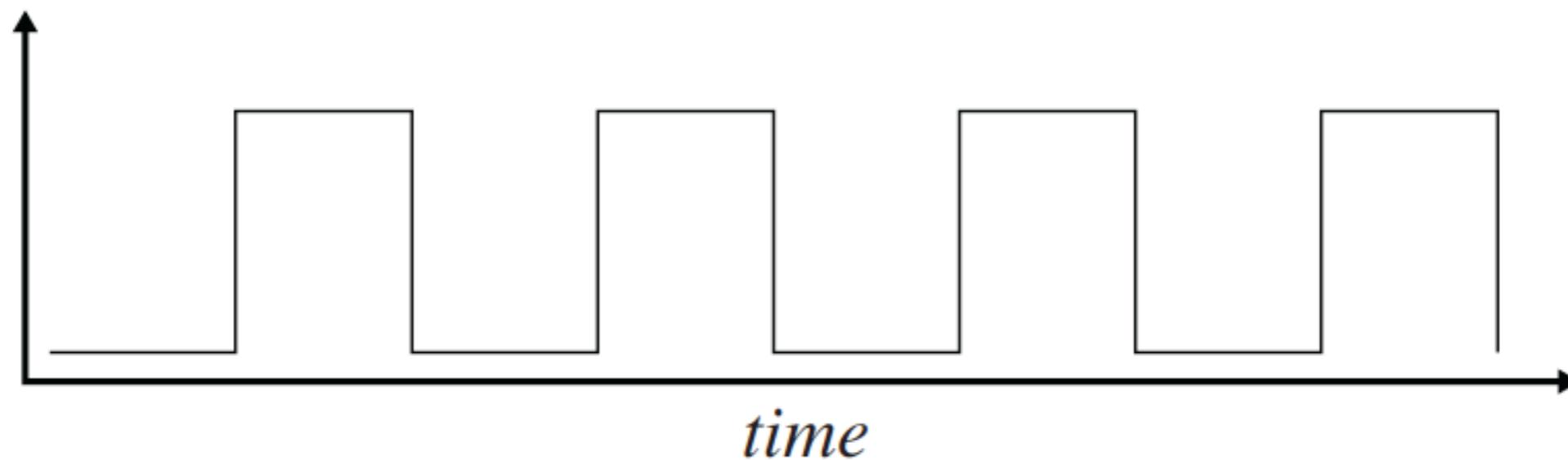
- A PWM signal is a **digital signal** whose average value of voltage (and current) is controlled by turning the switch between supply and load on and off at a fast rate.
- The longer the switch is **on** compared to the **off** periods, the higher the total power supplied.

# Understanding PWM-Pulse Width Modulation

Rectangular or Pulse Wave



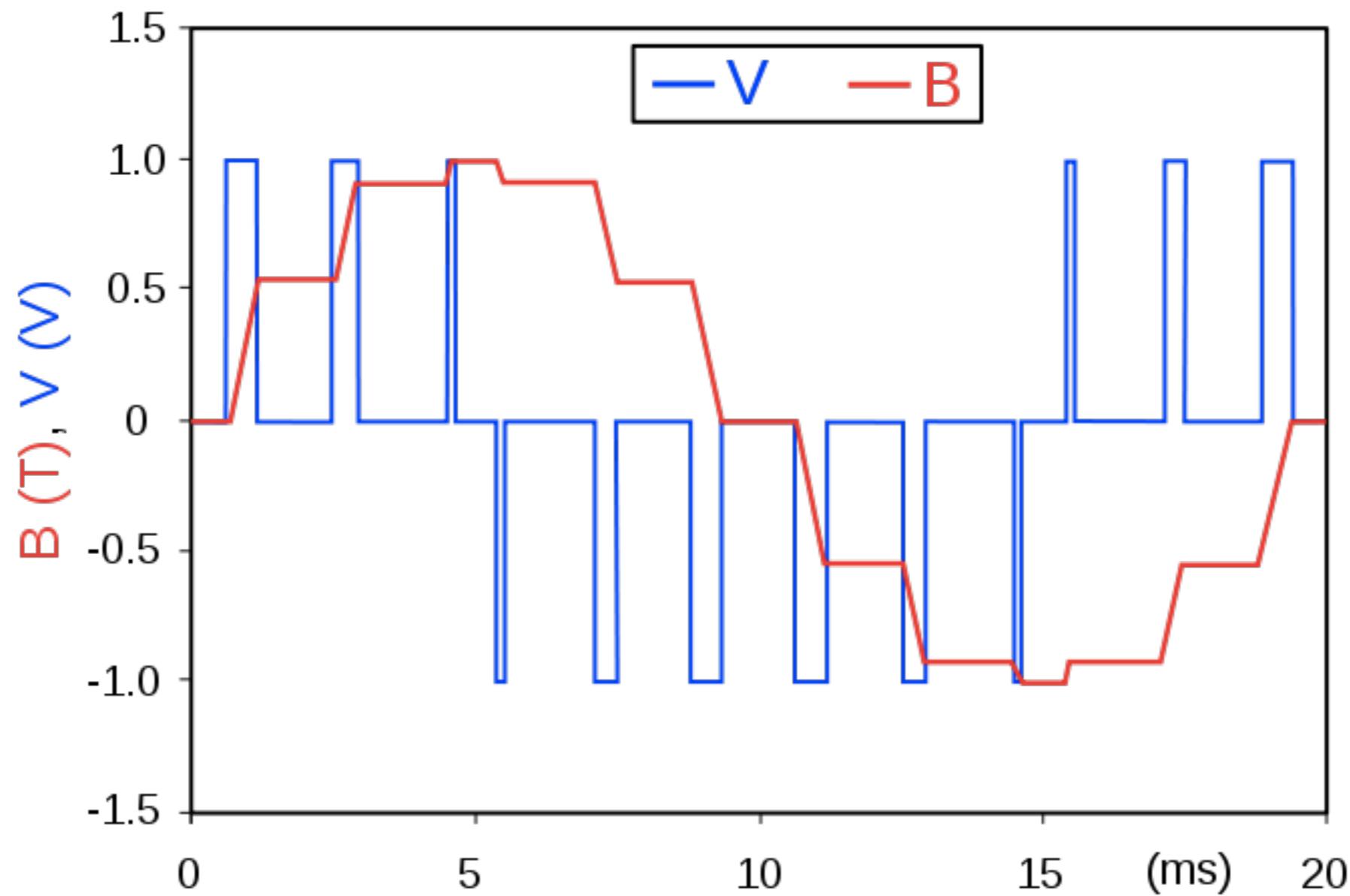
Special case: Square Wave ( $T_{\text{min}} = T_{\text{max}}$ )



$$\text{Frequency} = 1/\text{Period}$$

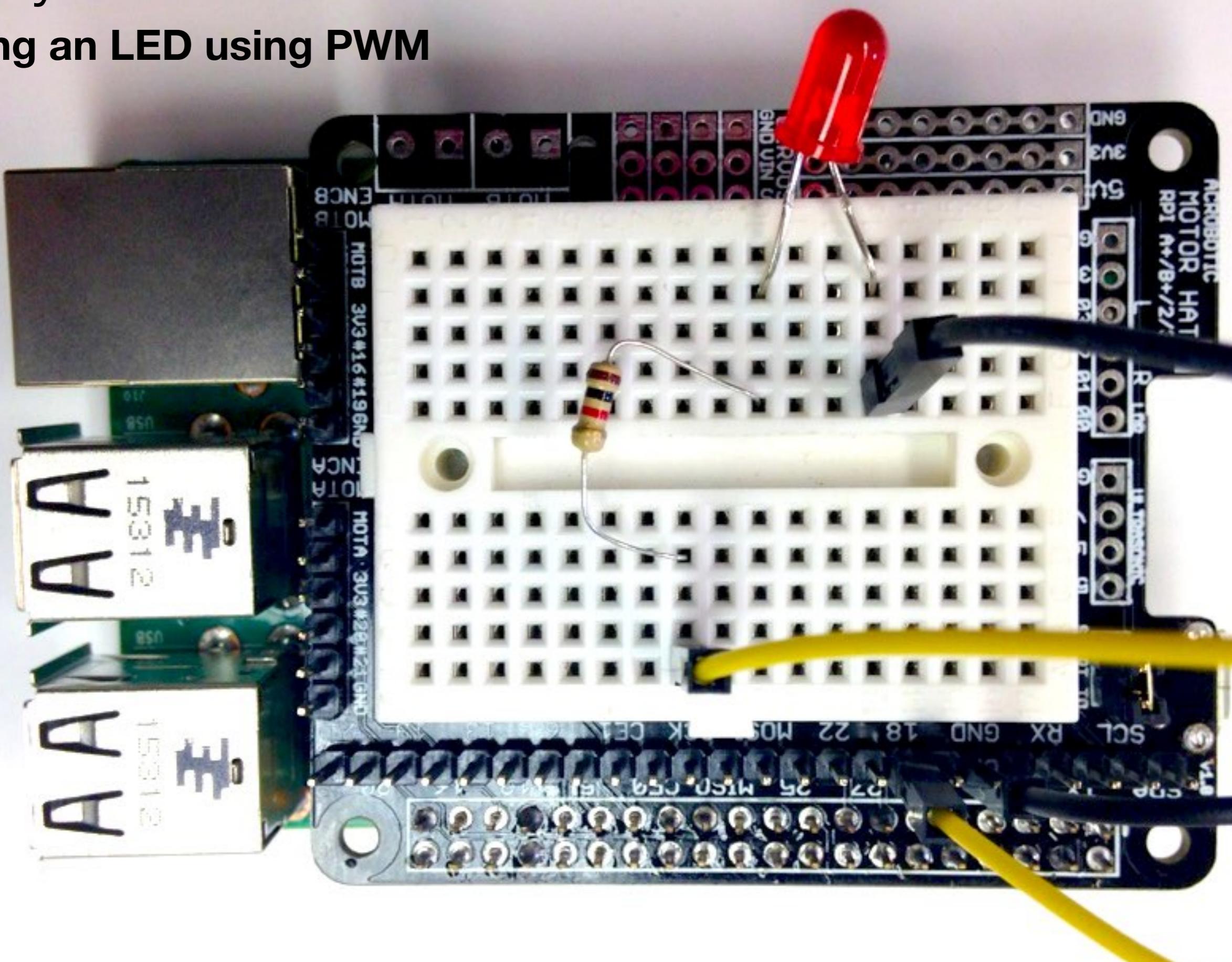
$$\text{Duty Cycle} = 100\% \times T_{\text{max}} / (T_{\text{max}} + T_{\text{min}})$$

# Understanding PWM-Pulse Width Modulation



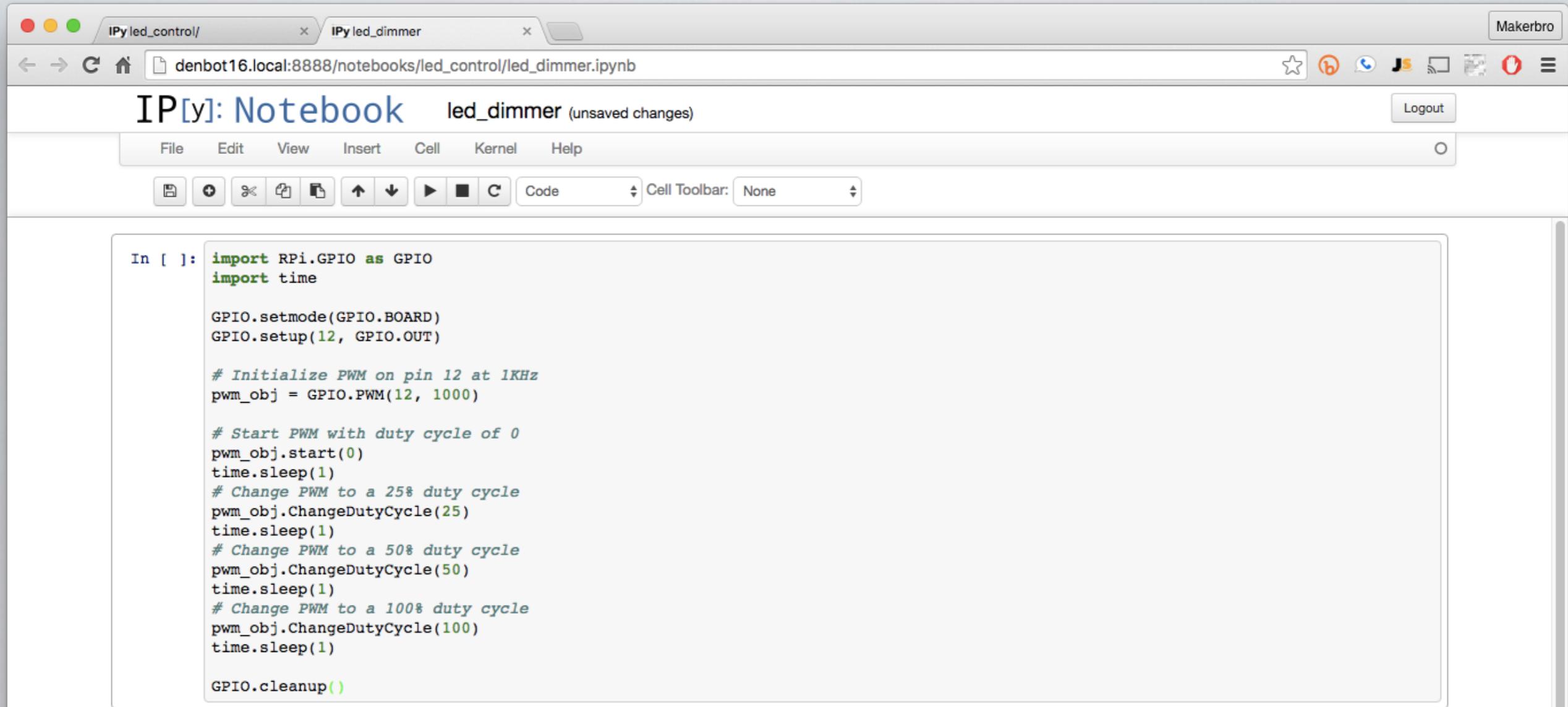
# Activity 3

## Dimming an LED using PWM



# Activity 3

- Let's use PWM for dimming an LED!



The screenshot shows an IPython Notebook interface with the title bar "IPy led\_control/" and "IPy led\_dimmer". The main area displays the following Python code:

```
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()
```