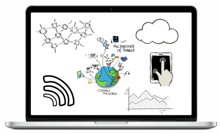
CSC 498R: Internet of Things

Lecture 03: Using the Pi for the First Time

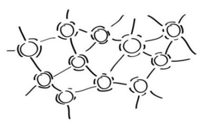
Instructor: Haidar M. Harmanani

Fall 2017

IoT Components

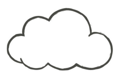


- Things we connect: **Hardware, sensors *and* actuators**

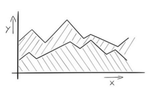


- Connectivity
  - Medium we use to connect things

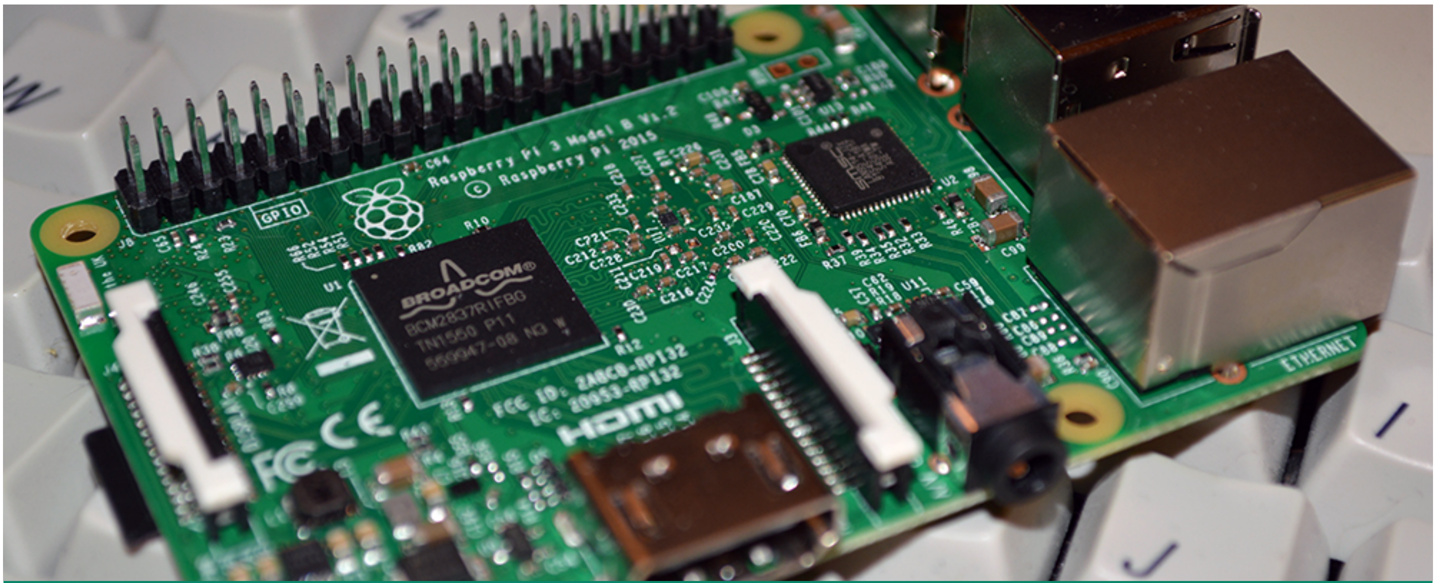
- Platform
  - Processing and storing collected data
    - Receive and send data via standardized interfaces or API
    - Store the data
    - Process the data



- Analytics
  - Get insights from gathered data



- User Interface

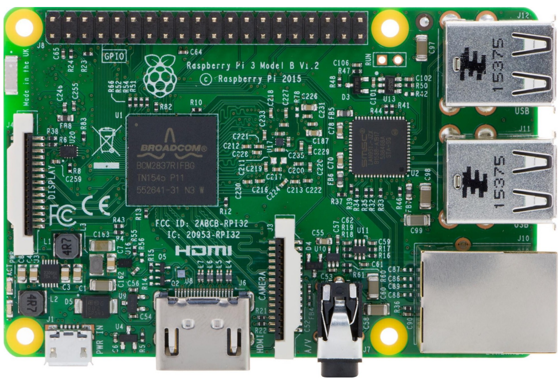


What Will we Use in this Course?

Raspberry Pi 3 Model B

Raspberry Pi 3 Model B (Pi 3)

- Introduced in February 2016
- SoC: Broadcom BCM2837 (roughly 50% faster than the Pi 2)
- CPU: 1.2 GHZ quad-core ARM Cortex A53 (ARMv8 Instruction Set)
- GPU: Broadcom VideoCore IV @ 400 MHz
- Memory: 1 GB LPDDR2-900 SDRAM
- USB ports: 4
- Network: 10/100 MBPS Ethernet, 802.11n Wireless LAN, Bluetooth 4.0



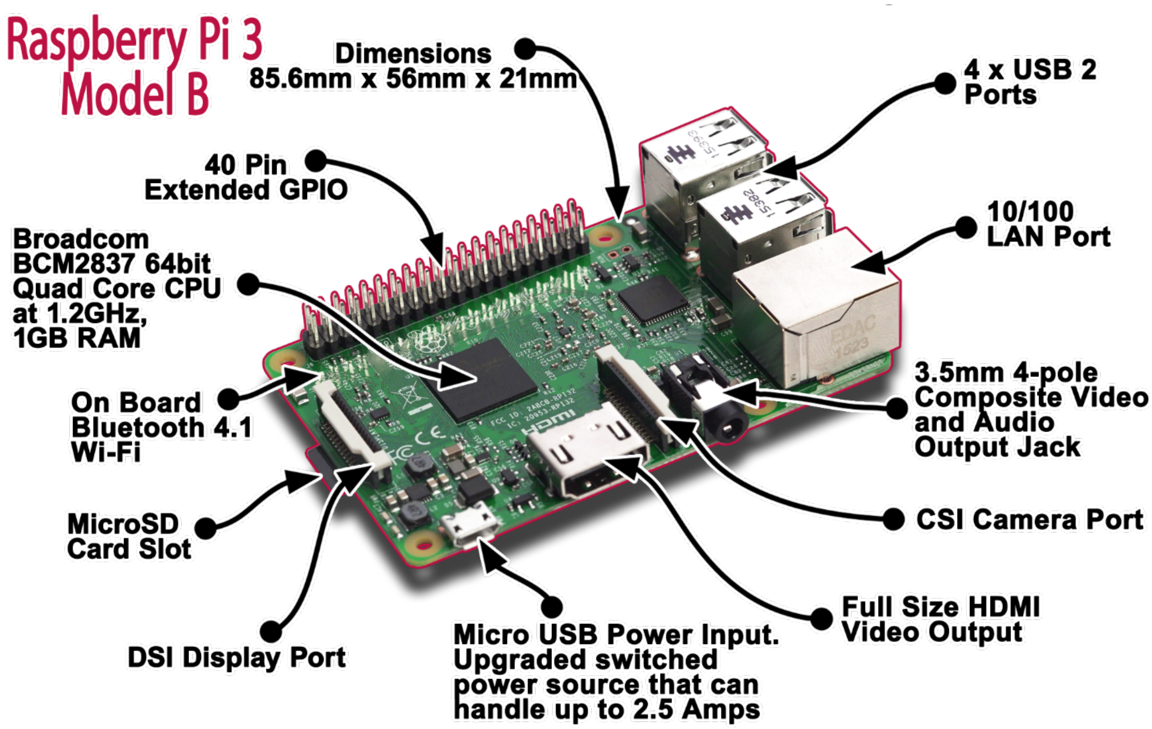
Useful Tools for Modeling and Implementation

- Fritzing for hardware modeling
- PubNub for cloud solution

	Raspberry Pi 3 Model B	Raspberry Pi 2 Model B
Processor Chipset	Broadcom BCM2837 64 Bit Quad Core Processor 1.2GHz	Broadcom BCM2836 32 Bit Quad Core Processor 900MHz
Processor Speed	QUAD Core @1.2 GHz	QUAD Core @900 MHz
RAM	1GB SDRAM @ 400 MHz	1GB SDRAM @ 400 MHz
Storage	MicroSD	MicroSD
USB 2.0	4x USB Ports	4x USB Ports
Max Power Draw/voltage	2.5A @ 5V	1.8A @ 5V
GPIO	40 pin	40 pin
Ethernet Port	Yes	Yes
WiFi	Built in	No
Bluetooth LE	Built in	No

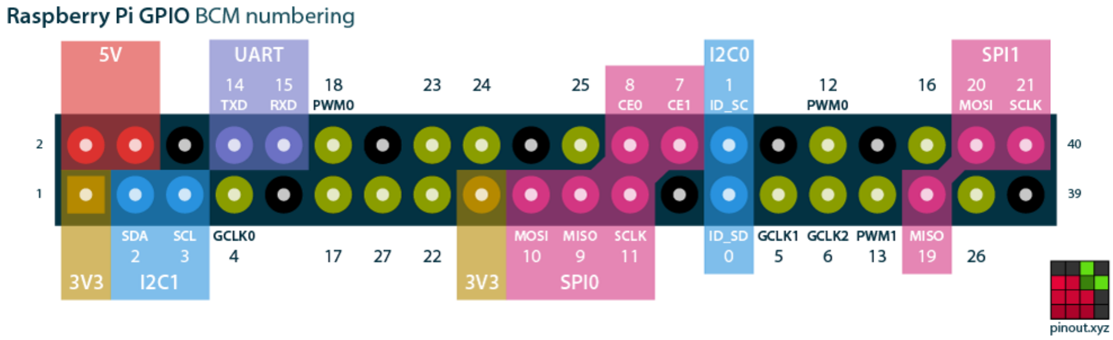
Raspberry Pi 3 Model B (Pi 3)

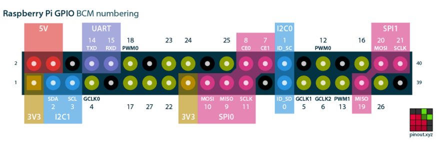
- We will:
  - Use the Raspberry Pi 3 Model B (Pi 3)
  - Perform communication with the embedded OS
  - Perform GPIO commands
  - Use a variety of sensors for Raspberry Pi
  - Control hardware using the Pi
    - Need jumper wires and a bread board
- We are open for any other development board that you would like to use, although we will not provide it!



Pi 3 General Purpose IO (GPIO) Pins

- Physical Computing on the Raspberry Pi is done using the GPIO pins





17 GPIO pins
- most have alternated functions
- two pins for UART; two for I2C; six for SPI

All 17 pins can be GPIO
- All support interrupts
- Internal pull-ups & pull-downs for each pin
- I2C pins have onboard pull-ups
  - using them for GPIO may not work

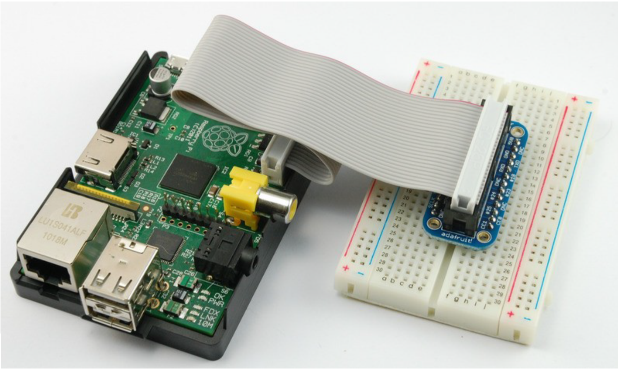
Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

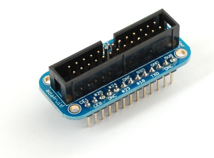
Pi 3 General Purpose IO (GPIO) Pins

- Connecting a 5 V supply to any pin on the Raspberry Pi's GPIO port, or directly shorting either of the power supply pins (Pin 1 and Pin 2) to any other pin will result in damage to the Pi.

Pi 3 General Purpose IO (GPIO) Pins

- You could use a cobbler connector rather than working directly with the Pi



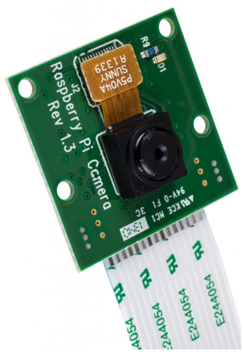


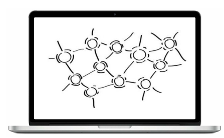
Using the GPIO Pins

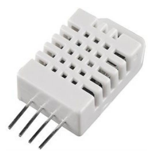
- There are two different methods to read or write these pins using Linux
  - Method 1: Creating a file-type access in the file system using /sys
    - /sys is a way the kernel provides information about (physical and virtual) devices
  - Method 2: Write/read memory addresses allocated to the GPIO peripheral of the SoC using pointers
    - Memory locations can be found in the datasheet for the BCM2835
    - RPi.GPIO
      - Comes with Raspbian
    - RPIO
      - Supports PWM & servos
    - WiringPi-python
    - Quick2Wire
  - The best way to access the GPIOs?
    - http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/
- More about the GPIO later!

PIR Motion Sensor

LED

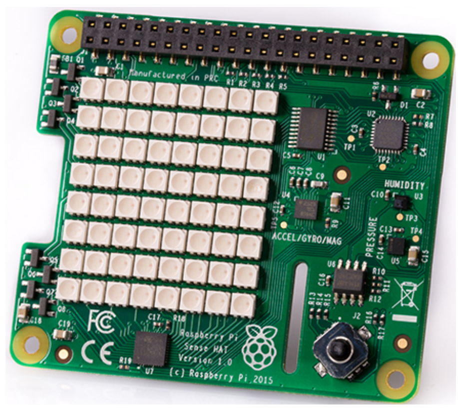


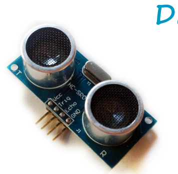




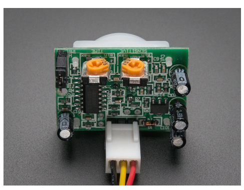
DHT22 sensor

Camera Modules





HC-SR04 ultrasonic sensor

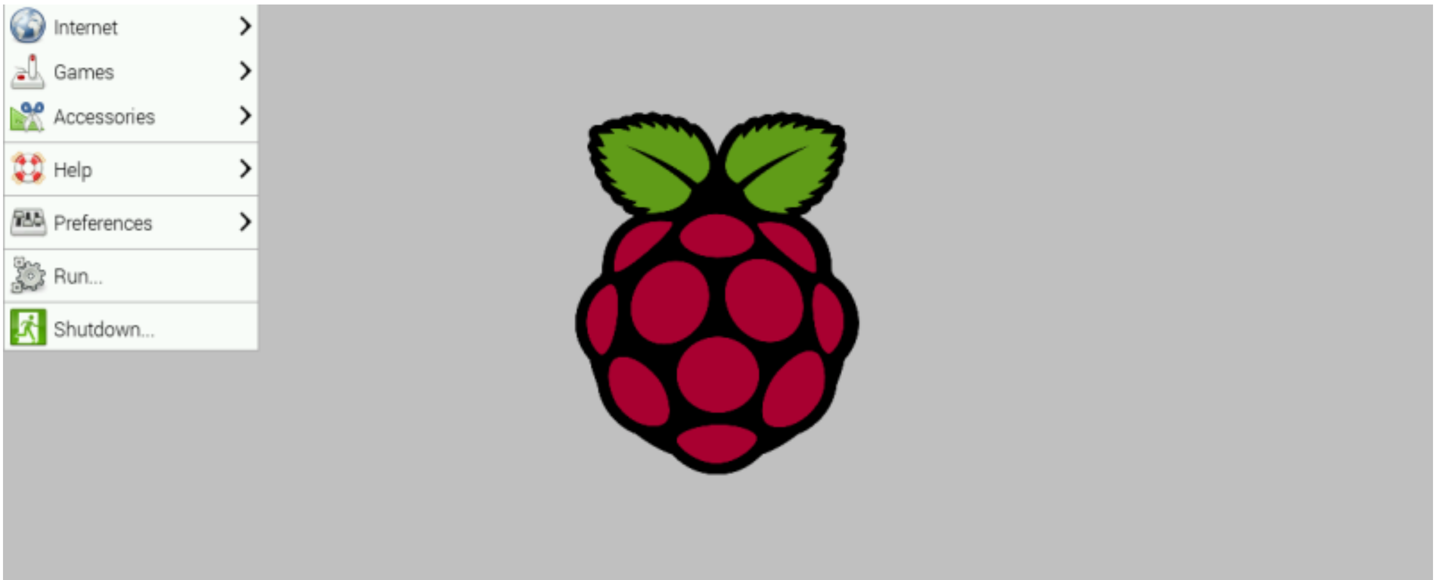


PIR Motion Sensor

Sense HAT

Getting Started

- What do you need?
  - Suitable Power Adapter
    - 5.1 volts, 2.5 Amps output
  - Keyboard with a mouse or Track Pad
    - Can use a usb or wireless
    - Specialized wireless keyboards are recommended if power consumption is an issue
  - Display, Pi 3 compatible
    - A regular display would do
    - Touch screen or a display for raspberry would be nice
  - Micro SD Card with SD Adapter loaded with the operating system



Operating Systems

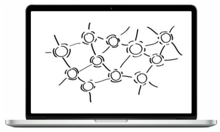
NOOBS

- New Out Of the Box Software
- Easy to download, install, and set up
- When you first boot up NOOBS, you'll get a selection of OSes to choose from
- Which operating systems are available depends on which model of Raspberry Pi you are using
  – Raspbian, OSMC, OpenELEC, Windows IoT Core, and RISC OS.

Raspbian

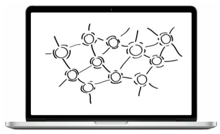
- The Raspberry Pi itself doesn't come with an operating system
- Raspbian is the "official" operating system of the Raspberry Pi
  – A version of Linux built specifically for the Raspberry Pi
  – Comes packed with all the software you'll need
    - LibreOffice, a web browser, email program, and some tools to teach programming to kids and adults alike, and a special version of Minecraft

OSMC



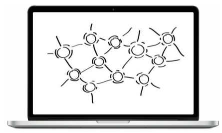
- OSMC (Open Source Media Center) is media center software based on Kodi (formerly XBMC)
  - Easy set up and use.
- If you're new to media centers or you're trying to set one up for non-techy people, OSMC is the one you want to use.

Windows 10 IoT [God forbids!]



- Windows 10 IoT is a special version of Windows built for the Raspberry Pi
  - Meant as a development platform for coders and programmers to prototype internet connected devices using the Raspberry Pi and Windows 10.
- *Only* compatible with Windows 10
  - Cannot do anything with it unless you have another computer with Windows 10 installed.
- You can't control or do anything on the Pi by itself.
  - For that, you'll need to download and install Visual Studio on your Windows PC
  - Once you do, you can program and control your Raspberry Pi from Visual Studio in Windows 10.
  - This means you can trigger blinking lights, connect to push buttons, control motors, and countless other things.

Other OSes

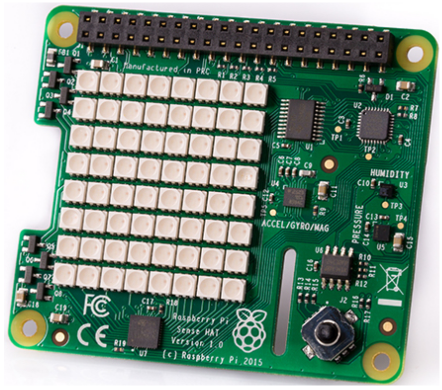


- OpenELEC
  - Open Embedded Linux Entertainment Center)
  - Built for one thing: playing media
- RISC OS
  - Not Linux, it is an operating system all its own
  - Rather weird

Sensors

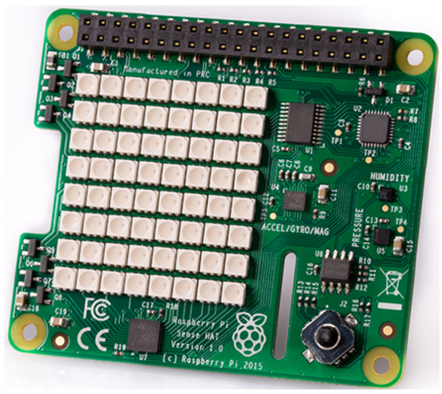
Sense HAT

- We will mostly use Sense Hat
  - Add-on board for Raspberry Pi
  - Made especially for the Astro Pi mission
  - Launched to the International Space Station in December 2015
- Pros:
  - Variety of sensors.
  - Easy to use, just plug them.
  - Good enough to start or for prototyping.
- Cons:
  - Not flexible and scalable



Sense HAT

- Has an 8×8 RGB LED matrix, a five-button joystick
- Includes the following sensors:
  - Gyroscope
  - Accelerometer
  - Magnetometer
  - Temperature
  - Barometric pressure
  - Humidity



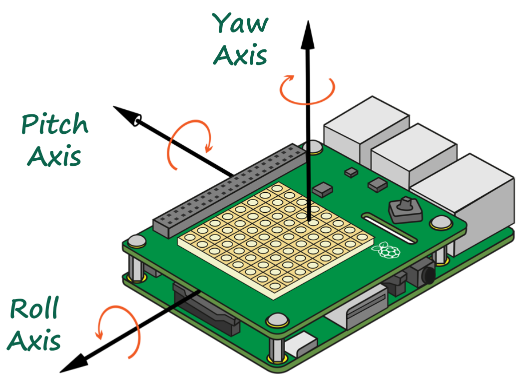
Sense HAT: Gyroscope

A very tiny gyroscope is built into the Sense HAT

The pitch, roll, and yaw values are returned as angles between 0 and 360 degrees.

Allows your program to react to changes in orientation

Can also read the gyroscope's X, Y, and Z axis values in radians per second



Sense HAT: Temperature Sensor

- Measure hot and cold
- Built into the Sense HAT and reports the temperature as a number in Celsius
- Care should be taken as the temperature sensor may be measuring some heat coming from the Raspberry Pi itself

Sense HAT: Barometric Pressure Sensor

- A pressure sensor measures the force exerted by tiny molecules of the air we breathe.
- The Sense HAT has an air pressure sensor built in and will report air pressure to you in millibars in your code.

Sense HAT: Humidity Sensor

- A humidity sensor measures the amount of water vapor in the air
- One of the main properties of air is that the hotter it is, the more water vapor can be suspended within it
- Relative humidity is a ratio, usually a percentage, between the actual amount of suspended water vapor to the maximum amount that could be suspended for the current temperature.
  - If there was 100% relative humidity, it would mean that the air is totally saturated with water vapor and cannot hold anymore.
- The Sense HAT has a humidity sensor that will report relative humidity as a percentage to you in your code
  - Uses data from the temperature sensor to give you the correct value
  - The sensor will be good enough to detect the water vapor in human breath

Sense HAT: 8×8 RGB LED Matrix Display

- Designed to be the only real form of visual output that the Astro Pis have up on the International Space Station
  - Was not allowed to plug the HDMI or composite outputs of the Raspberry Pi into anything on the ISS
- Consists of 64 LEDs arranged in an eight-by-eight grid, and each individual LED has a red, green, and blue component that you can control in code.
- For a single LED you can specify how much red, green, and blue you want, using numbers between 0 and 255
- Using various combinations of red, green, and blue for each LED, you can create any color or shade that you want
- It should allow you to create a basic display or status monitor, or even play animations showing what your program is doing.

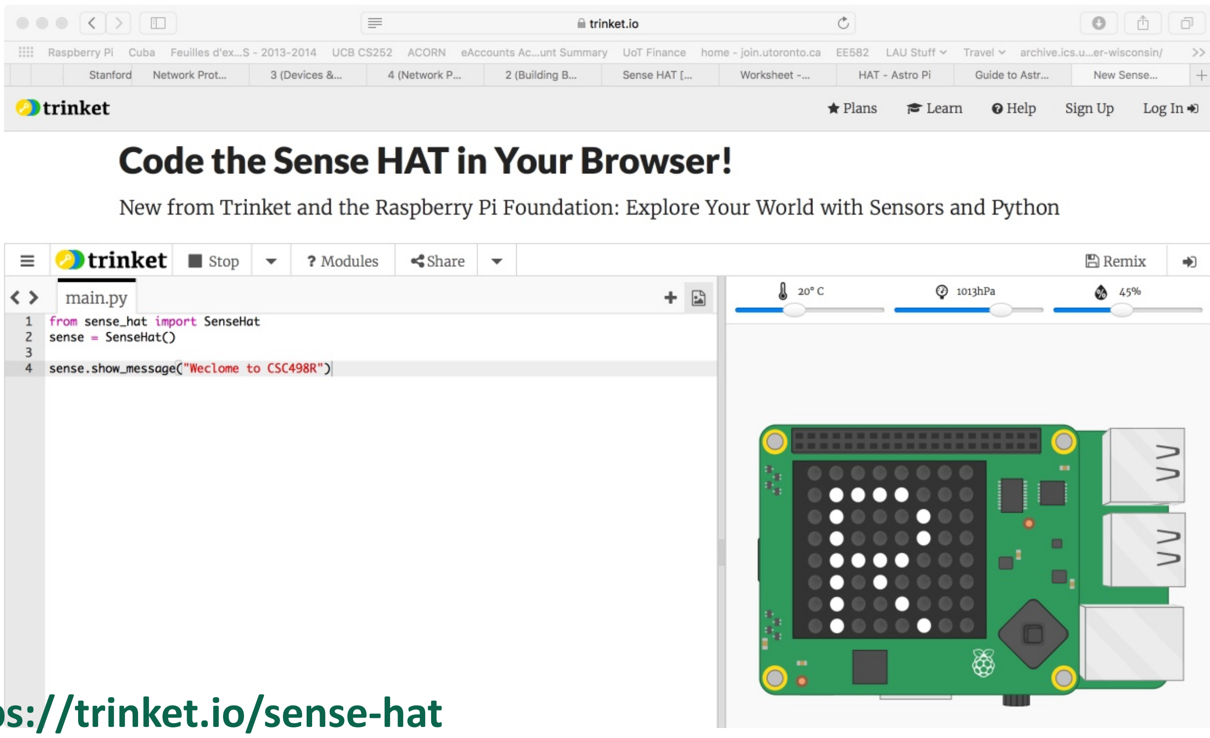
Sense HAT: Mini 5-button Joystick

- Made up of five buttons for up, down, left, right, and center
- Combine this with the LED matrix and you have the ability to create games
- Mapped to the four keyboard cursor keys, with the joystick middle-click being mapped to the Return key

Sense HAT

- Python provides a module to control the Raspberry Pi HAT
  - https://github.com/RPi-Distro/python-sense-hat
- Install
  - `sudo apt-get update`
  - `sudo apt-get install sense-hat`
  - `sudo reboot`
- Usage

```
from sense_hat import SenseHat
sense = SenseHat()
sense.show_message(”Welcome to CSC498R")
```

https://trinket.io/sense-hat

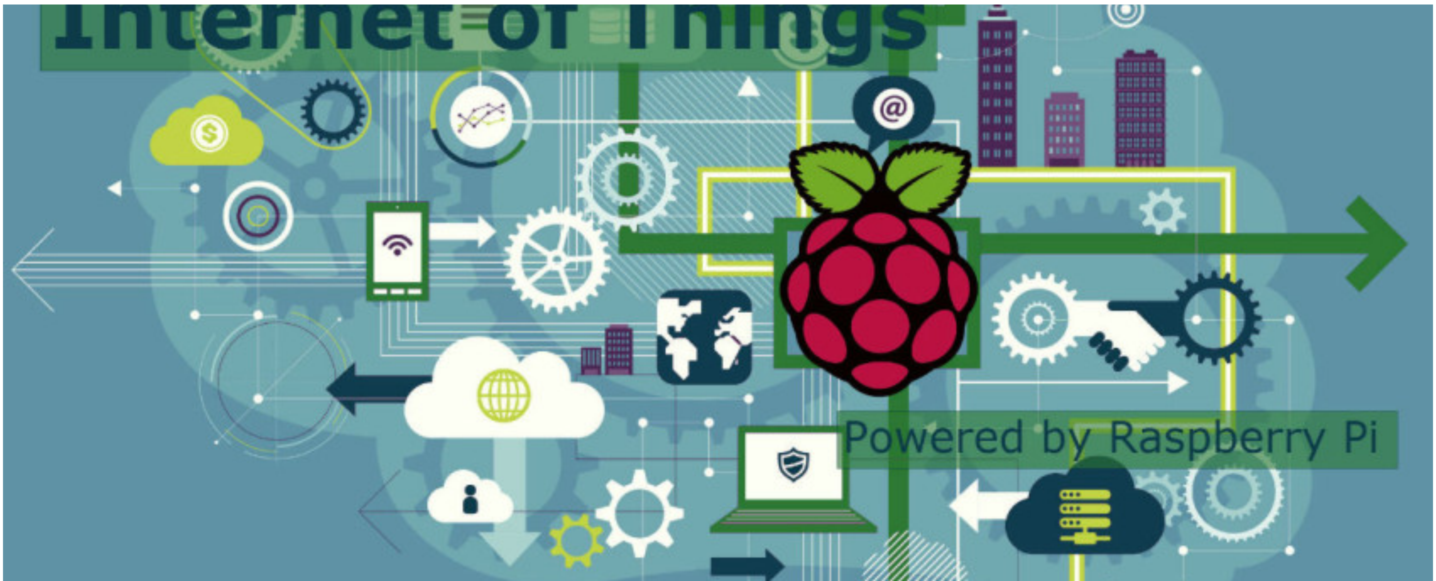
```
sense.get_temperature(): Return the temperature in Celsius.
sense.get_pressure(): Return the pressure in millibars.
sense.get_humidity(): Return the humidity as a percentage.
```


Interacting with the Sense HAT

```python
from sense_hat import SenseHat
sense = SenseHat()
while True:
    t = sense.get_temperature()
    p = sense.get_pressure()
    h = sense.get_humidity()
    t = round(t, 1)
    p = round(p, 1)
    h = round(h, 1)

    msg = "Temperature: {0}, Pressure: {1}, Humidity: {2}".format(t,p,h)
    sense.show_message(msg, scroll_speed=0.05)
```

Reference

- https://www.raspberrypi.org/learning/getting-started-with-the-sense-hat/worksheet/



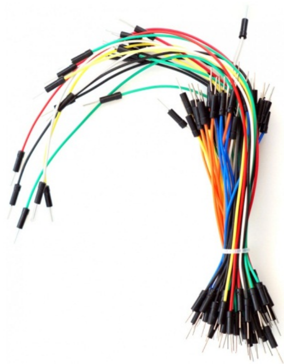
Powering the Raspberry Pi

What do We Need?

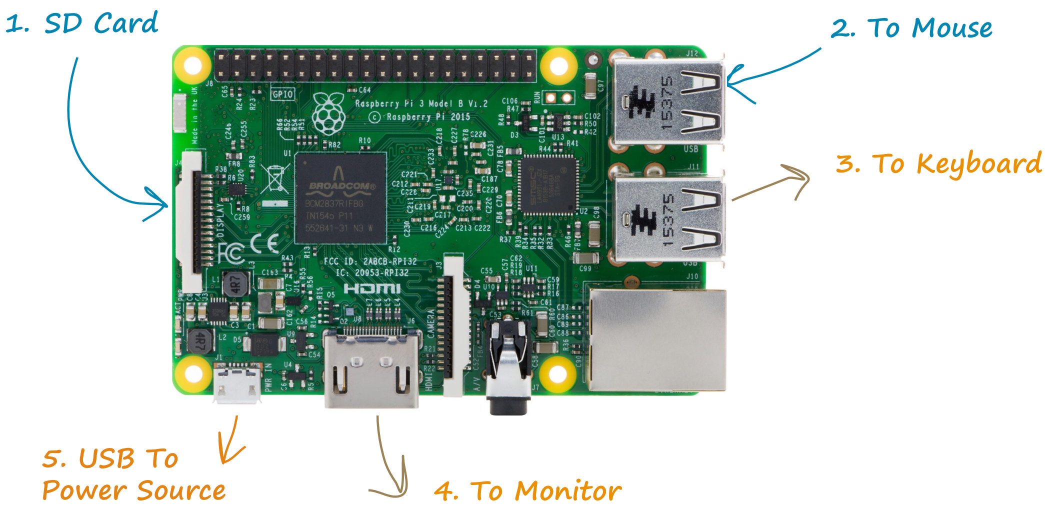
- Raspberry Pi 3
- Micro SD card (preloaded w/ Raspbian)
- Micro USB power supply
- HDMI Cable
- Wires
- Breadboard
- LED
- Resistors
- PIR sensor
- DHT22 sensor

What do We Need?

- Jumper wires female/male and male/male are useful to connect various components to the Pi 3.
- Prototyping board keeps things tidy!
- This is the minimum for prototyping
- Some sensors such as camera



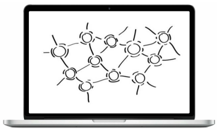
Setting up Your Pi



Starting Raspbian

- Username: pi
- Password: raspberry
- `pi@raspberrypi ~$ startx`

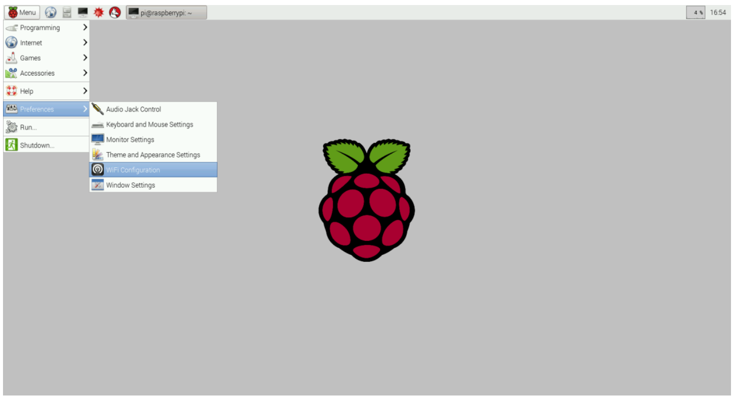
A first circuit



- Our goal is to get familiar with simple sensors, wires and the board connectivity.
- Important facts:
  - We'll use the 5.1V pin as input voltage for all sensors and accessories in this course.
  - Next to the 5V pin is the ground pin we will connect to all our sensors too.
  - Don't connect directly these two pins together.
  - It would damage the board

Wi-Fi Configuration

- Menu > Preference > WiFi Configuration



Remote Connect Pi

- By default, the Raspberry Pi hostname is raspberrypi.
- Getting your Pi's IP address
  - pi@raspberrypi ~$ hostname -i
- You can remote connect to your Pi using ssh

You'll need the IP address when you connect the Pi from your computer!!!

SSH into your Rasp Pi

SSH to Pi from your laptop (Terminal on Mac/Linux, PuTTY on Windows):

Your Pi's username

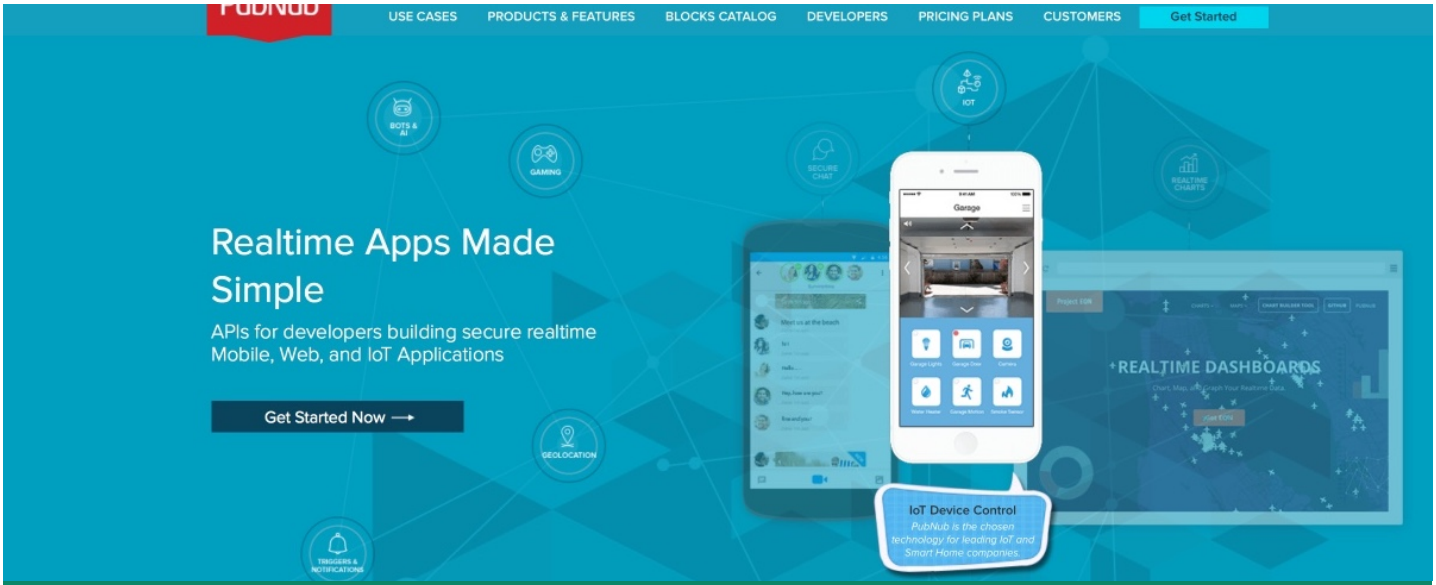
```
yoda:~ haidar$ ssh pi@10.96.70.1
```

Use your Pi's IP!

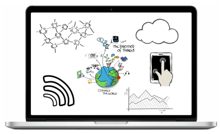
If SSH-ing fails, try:

```
$ sudo raspi-config
```

on your Pi

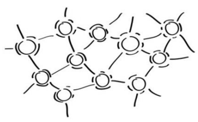


Get Started w/ PubNub Python SDK



IoT Components

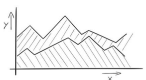
- Things we connect: **Hardware, sensors *and* actuators**



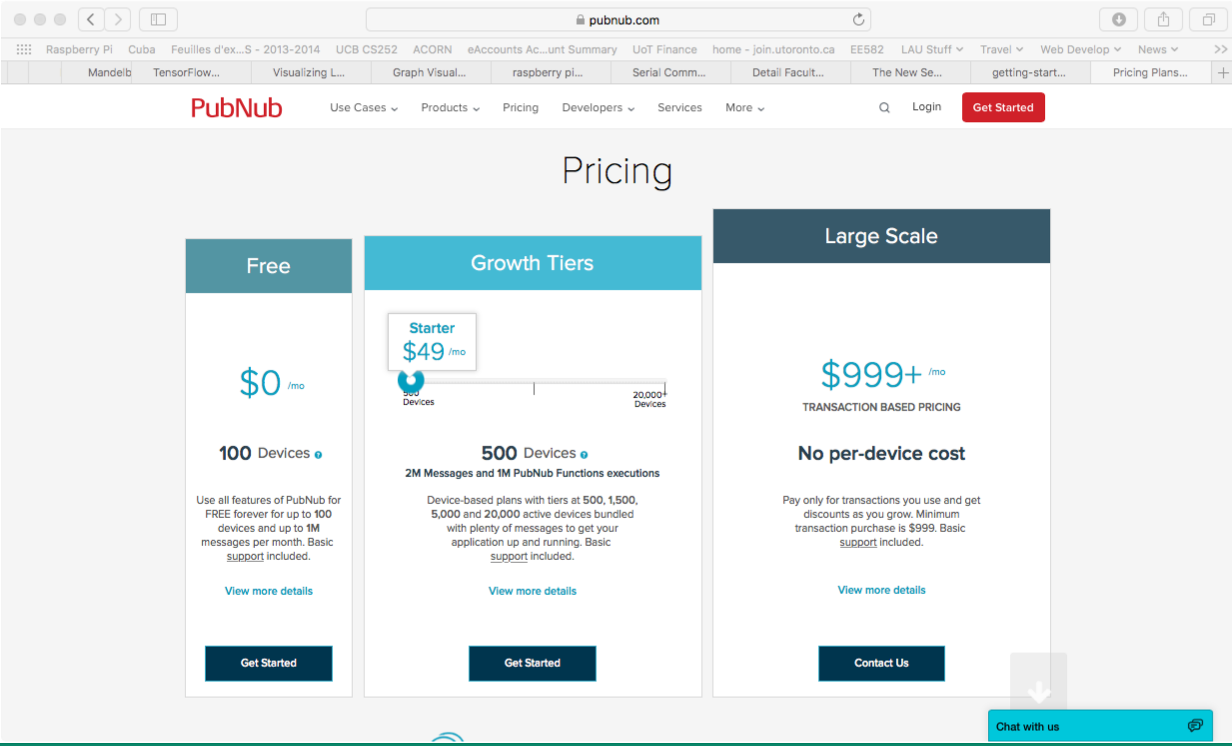
- Connectivity
  - Medium we use to connect things

- Platform
  - Processing and storing collected data
    - Receive and send data via standardized interfaces or API
    - Store the data
    - Process the data.

- Analytics
  - Get insights from gathered data



- User Interface

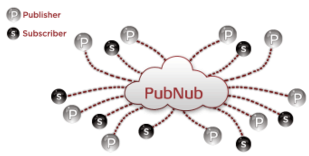


Four Easy Steps

- Update the System's package list
  - `~$ sudo apt-get update`
- Upgrade the installed packages to the latest versions
  - `~$ sudo apt-get upgrade`
- Install python and pip
  - `~$ sudo apt-get install python-dev`
  - `~$ sudo apt-get install python-pip`
- Install pubnub libs
  - `~$ sudo pip install pubnub`

Test it: Hello World w/ PubNub

```
## Hello World, PubNub

import sys
from pubnub import Pubnub

# Initiate Pubnub State - Get your own keys at admin.pubnub.com
pubnub = Pubnub(publish_key='pub-c-156a6d5f-22bd-4a13-848d-b5b4d4b36695', subscribe_key='sub-c-f762fb78-
2724-11e4-a4df-02ee2ddab7fe')

channel = 'hello-pi'

username = 'Your name'
message = 'Hello World from Pi!'

data = {
    'username': username,
    'message': message
}

# Asynchronous usage
def callback(m):
    print(m)

pubnub.publish(channel, data, callback=callback, error=callback)
```



Hello World w/ PubNub

- Import & init (hello.py)

```
import sys

from pubnub import Pubnub


pubnub = Pubnub(publish_key='pub-c-123...',
subscribe_key='sub-c-456...')
```

Hello World w/ PubNub

- Publish (Sending data)

```
channel = 'hello-pi'
data = { 'username': 'SpongeBob',
         'message': 'Hello world from Pi!'}

def callback(m):
   print(m)

pubnub.publish(channel, data, callback=callback, error=callback)
```

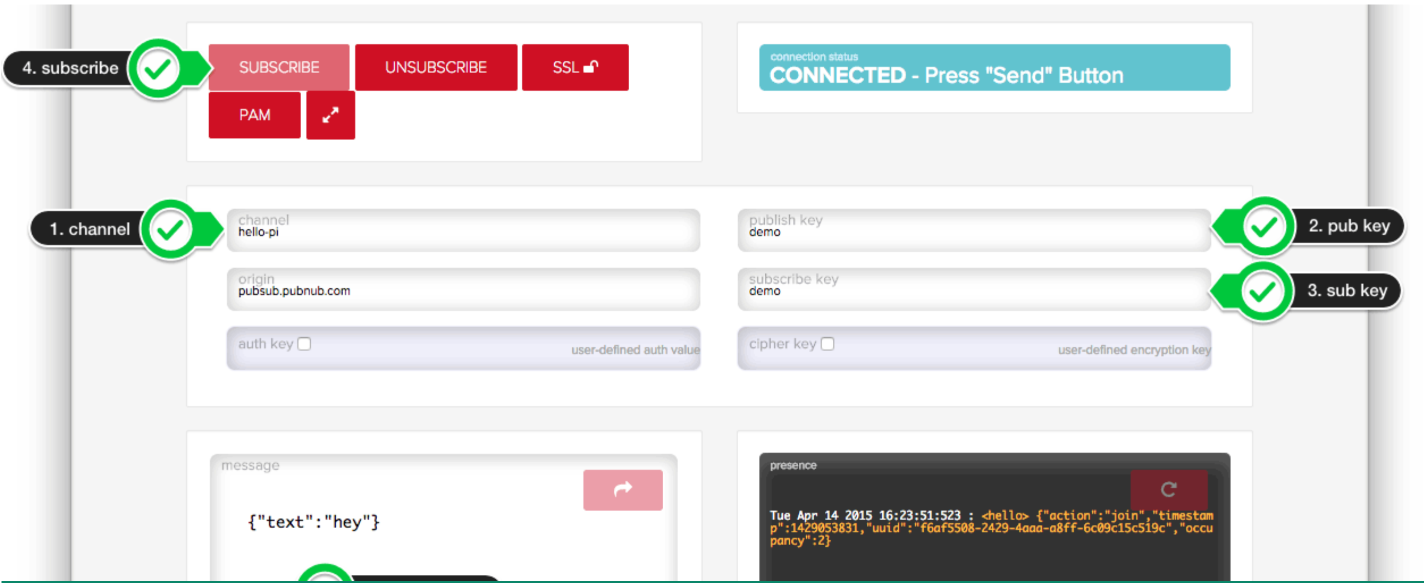
Use your own name & message!

Subscribe all other messages coming to the channel

```
def _callback(message, channel):
    print(message)

def _error(message):
    print(message)

pubnub.subscribe(channels='hello-pi', callback=_callback, error=_error)
```

Hello World w/ PubNub

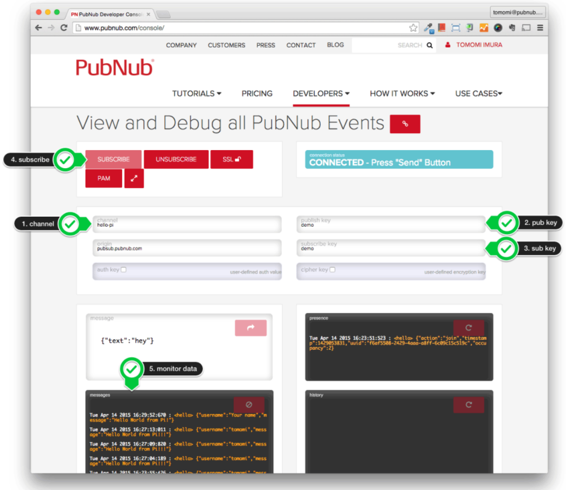
- Run your program
  - ~$ `sudo python hello.py`
- Subscribing data you are publishing
  - http://pubnub.github.io/workshop-raspberrypi/web/hello.html

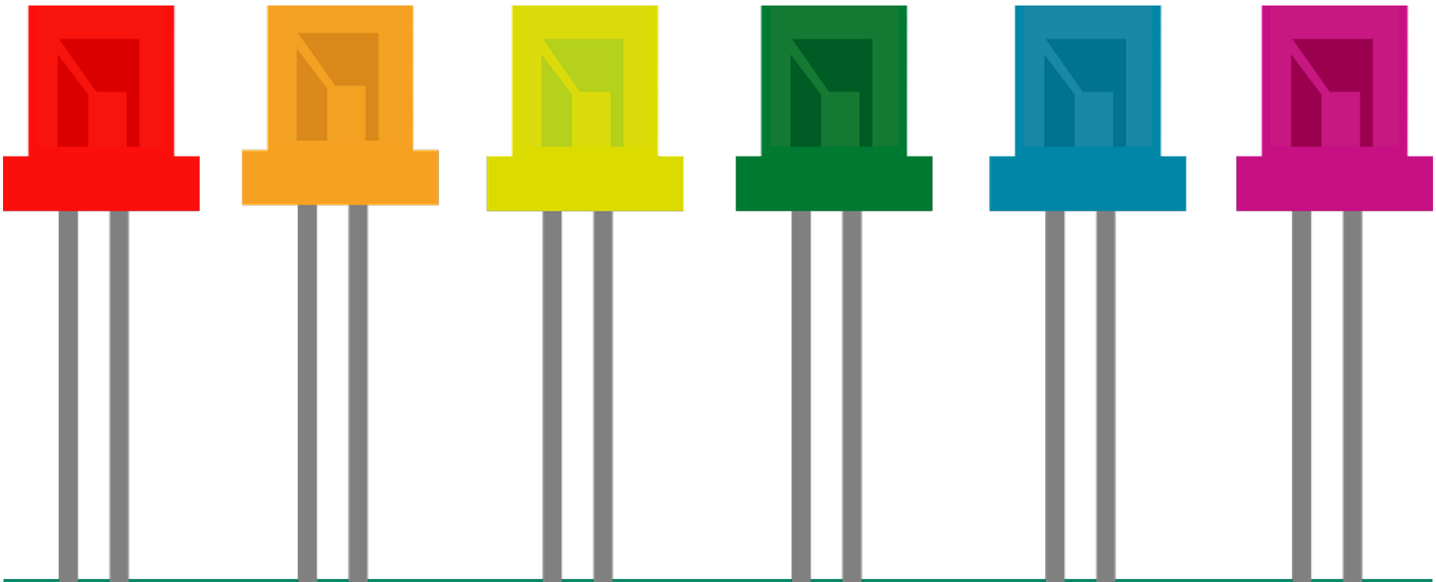


Using the Debug Console

Debug Console

- http://pubnub.com/console/
  - channel: hello-pi
  - pub key: demo
  - sub key: demo

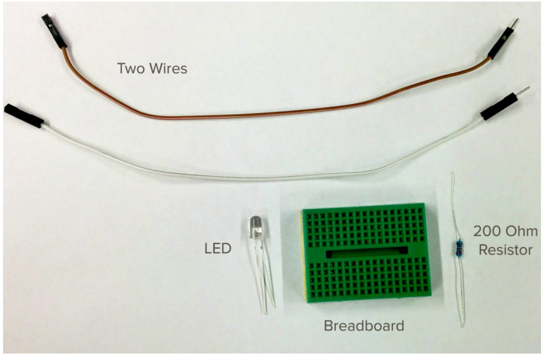




Example 1: Blinking LED

Blinking LED

- Raspberry Pi 3
- 1 LED (1.9 - 3.2V)
- 1 Resistor (200Ω)
- 1 Breadboard
- 2 M-to-F jumper wires, 2 colors

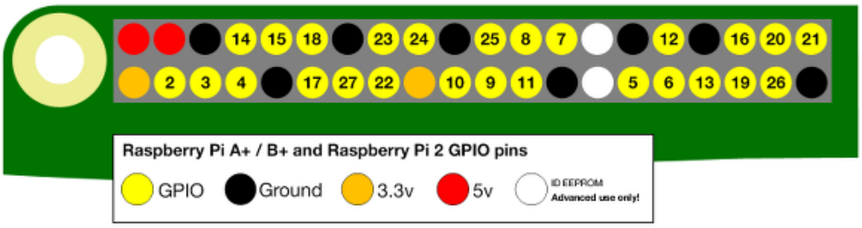


General Purpose Input Output (GPIO)

- The Raspberry Pi has 40 pins which can be used to control and monitor the outside world
- The Pi can:
  - Control LEDs, turning them on or off, or motors, or many other things
  - Can detect the pressing of a switch, change in temperature, or light etc.

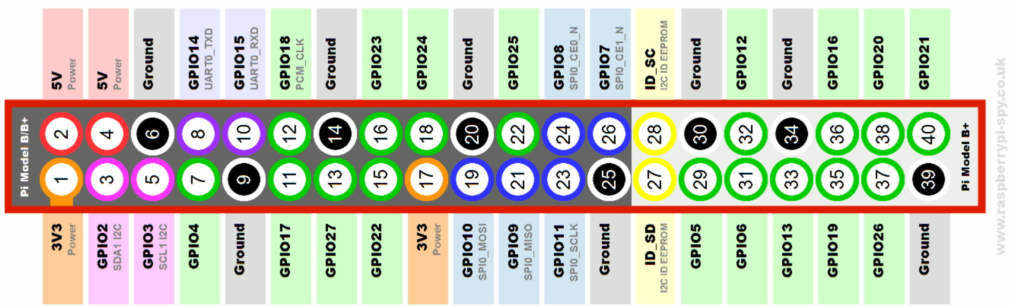
General Purpose Input Output (GPIO)

- The GPIO pins provide a physical interface between the Raspberry Pi and the outside world
  - Of the 40 pins, 26 are GPIO pins and the others are power or ground pins (plus two ID EEPROM pins)

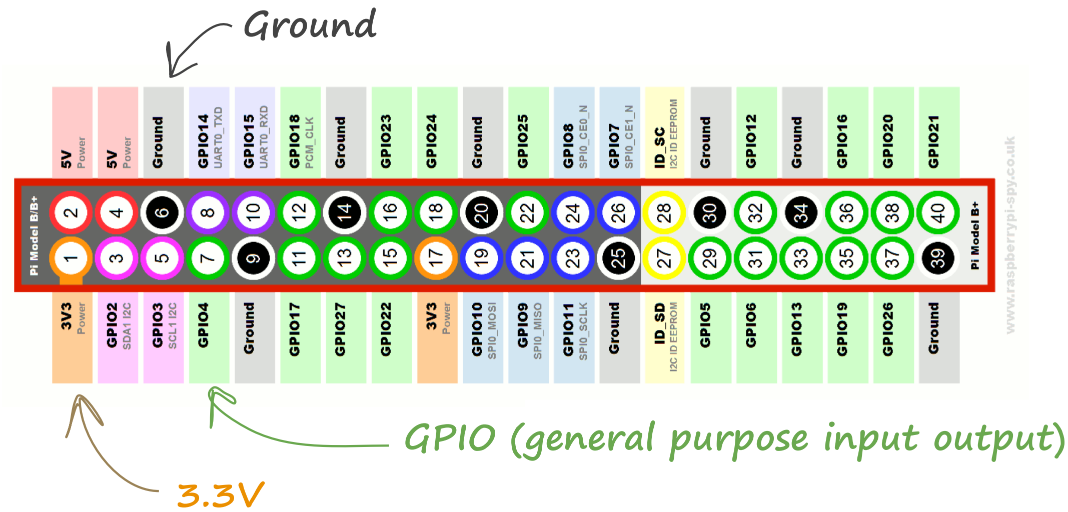


GPIO Pin Numbering

- Refer to the pins using the GPIO numbering scheme



GPIO Pin Numbering



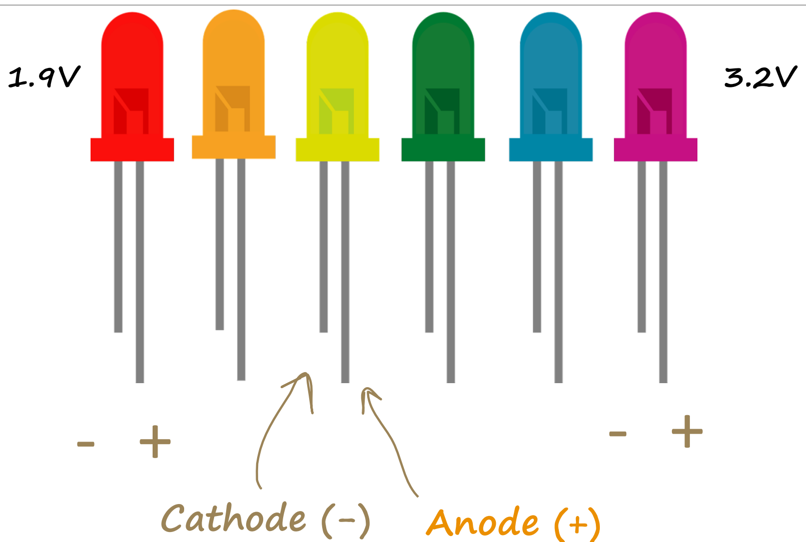
Raspberry Pi 3 GPIO Pins

- The output can be anything, from turning on an LED to sending a signal or data to another device.
- Inputs could be anything from a simple button to a sensor or a signal from another computer or device.
- Over the network, the *Raspberry Pi* can control devices that are attached to it from almost anywhere, and those devices can send data back to the network
- Connectivity and control of physical devices over the internet is the basis of the Internet Of Things (IoT) lab

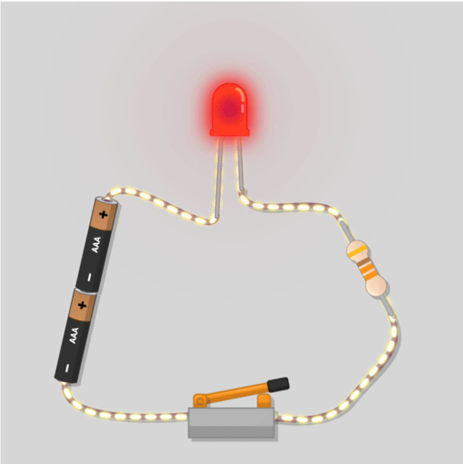
Some Circuit Theory

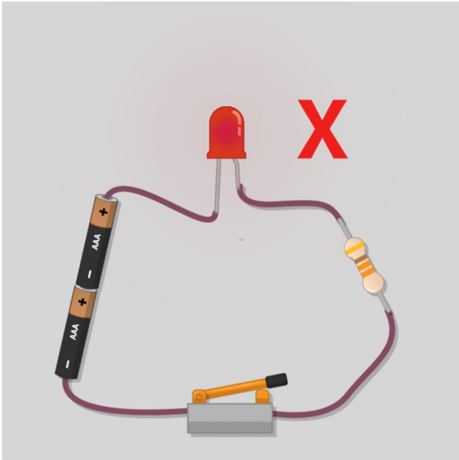
- A Light Emitting Diode (LED) is a type of Output Component. When current flows through the LED, it emits light
- A diode is a component that only lets current flow through it in one direction through it
- LEDs should be protected using resistors by reducing the amount of energy that reaches the LED
  – Without the resistor, the LED could burn out

LEDs

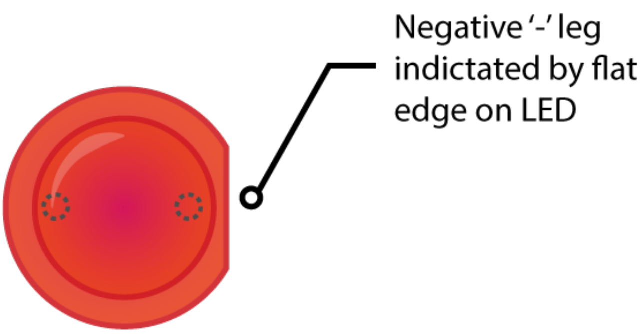


LEDs





Negative LEDs



OMG Physics!

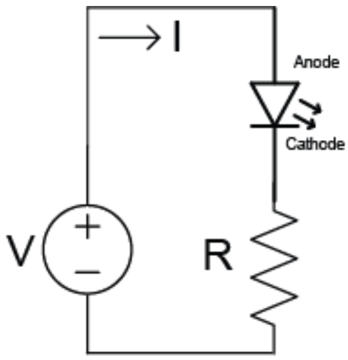
source voltage (V)

forward voltage (V)
(LED voltage drop)

$$R = \frac{V_s - V_f}{I}$$

resistance (Ω)

current thru
the LED (A)



OMG Physics!

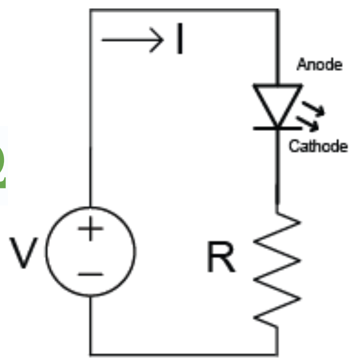
source voltage (V)

forward voltage (V)
(LED voltage drop)

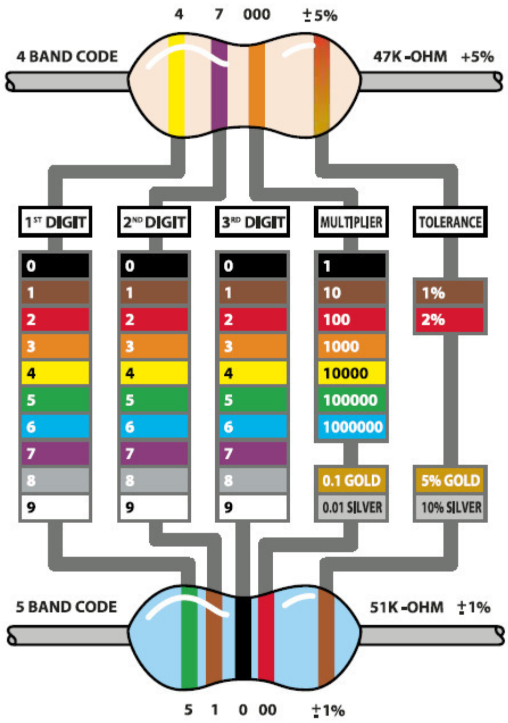
$$R = \frac{3.3\ \text{v} - 1.9\ \text{v}}{0.02\ \text{A}} = 70\ \Omega$$

resistance (Ω)

current thru
the LED (A)



4-Band Resistor Color Code



http://makezine.com

4-Band Resistor Color Code

4 7 10^2 +/- 5%

multiplier tolerance

47 x 100 = **4.7 k Ohms**

Learn more at: `https://learn.adafruit.com/multimeters/resistance`

Graphical Resistor Calculator

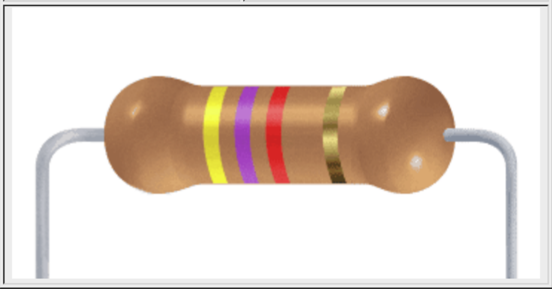
G+1 584

This JavaScript-based web app comes from my **JavaScript Bible** books (dating back to the very first edition with a few upgrades a long time ago). Although I have removed other book examples from this web site, this page remains the most popular destination within dannyg.com — presumably as a resource for students of electricity/electronics and my fellow radio geeks. Enjoy!

Calculate Resistor Values from Color Codes

Yellow | Violet | Red | Gold

Resistance Value:	4.7 Kohms, +/-5%



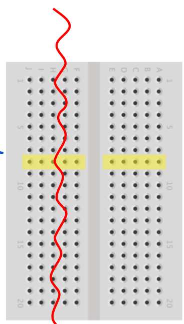
Graphical Resistor Calculator at: http://www.dannyg.com/examples/res2/resistor.htm

Breadboard

- An electronics breadboard is a fundamental tool to build circuits. It is solderless, and great tool for prototyping.

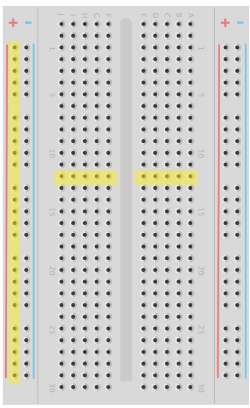
Connected!

conductive metal strips goes horizontally



Mini

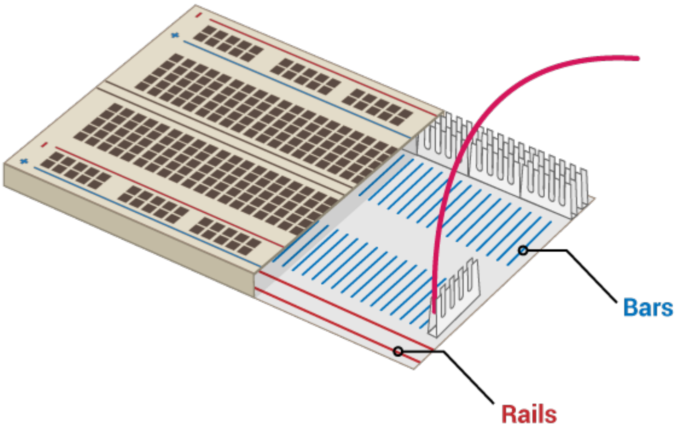
not connected !



You may find this type of breadboard when googling circuits. They have power rails that goes vertical!

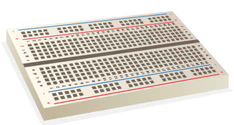
400-pin

Breadboard

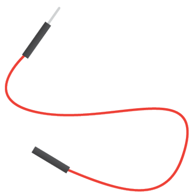


Experiment 1: Turning LED on

- What do we need?

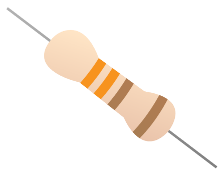


Breadboard



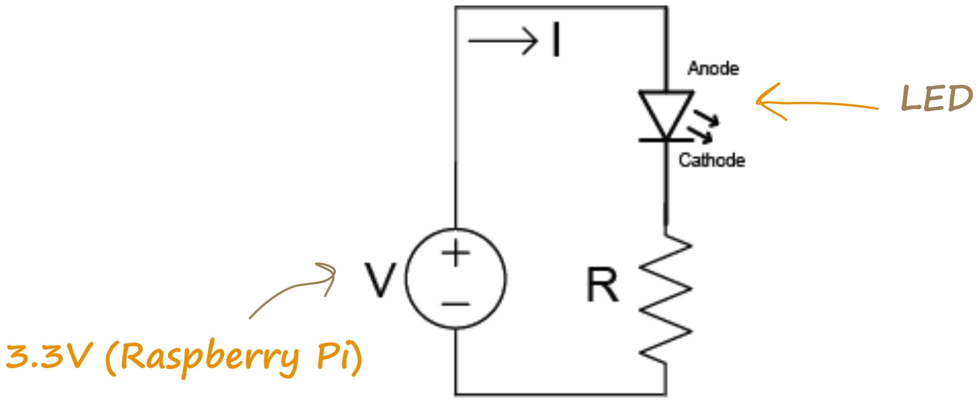
Male to female jumper wires

LED

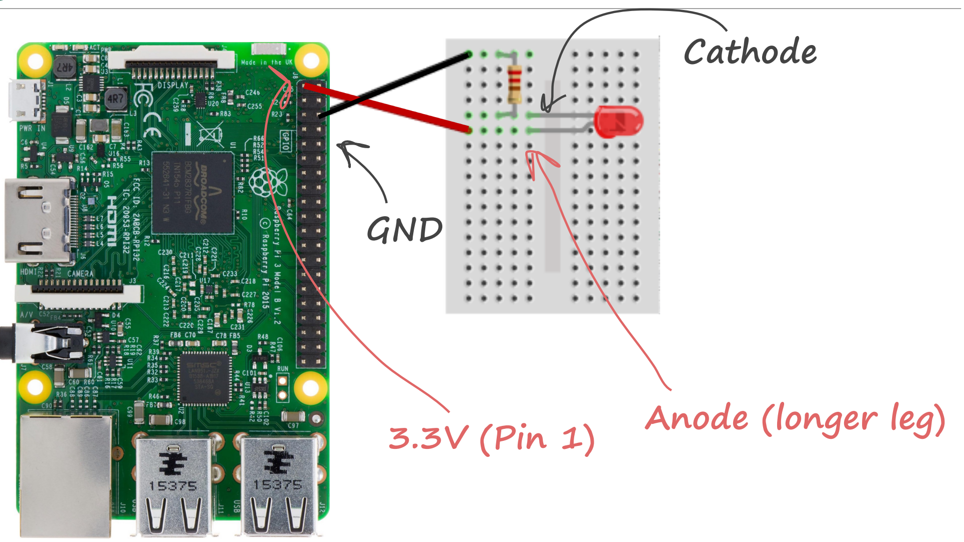


330 ohm Resistor

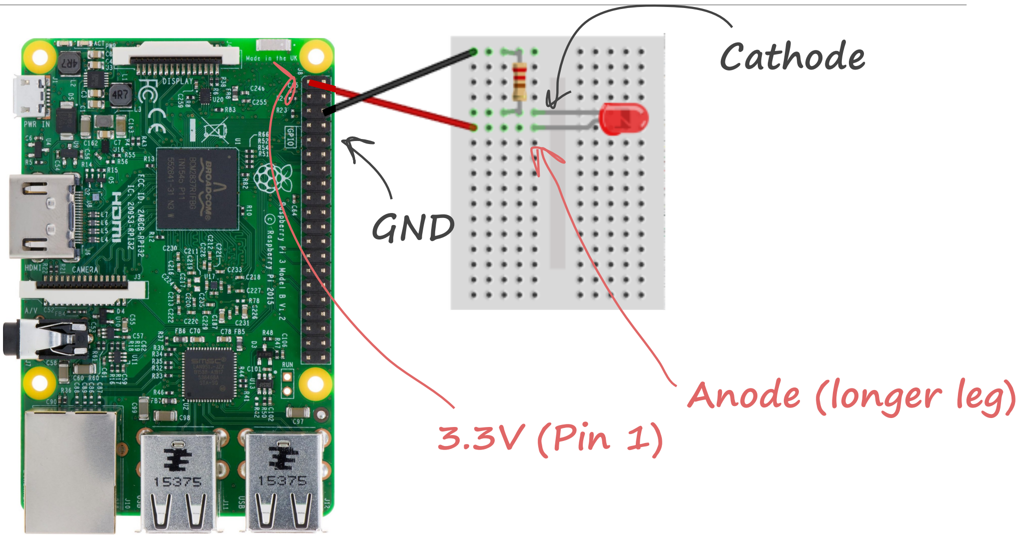
Turning LED on

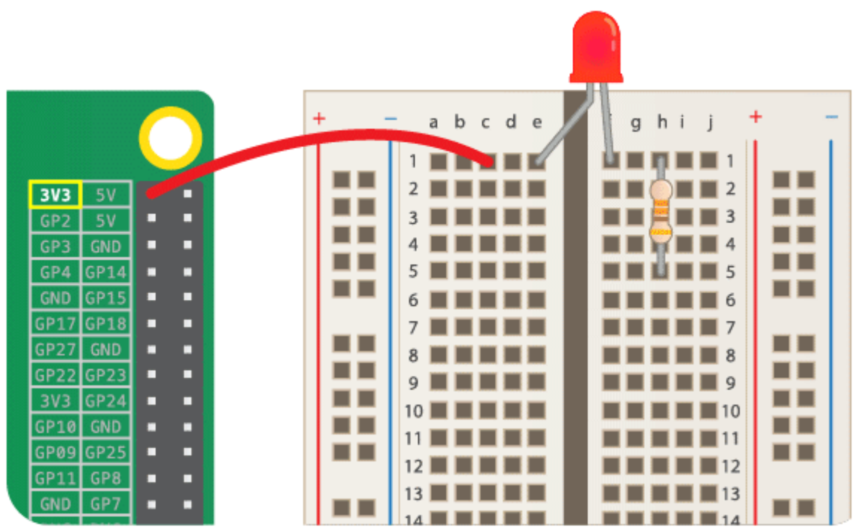


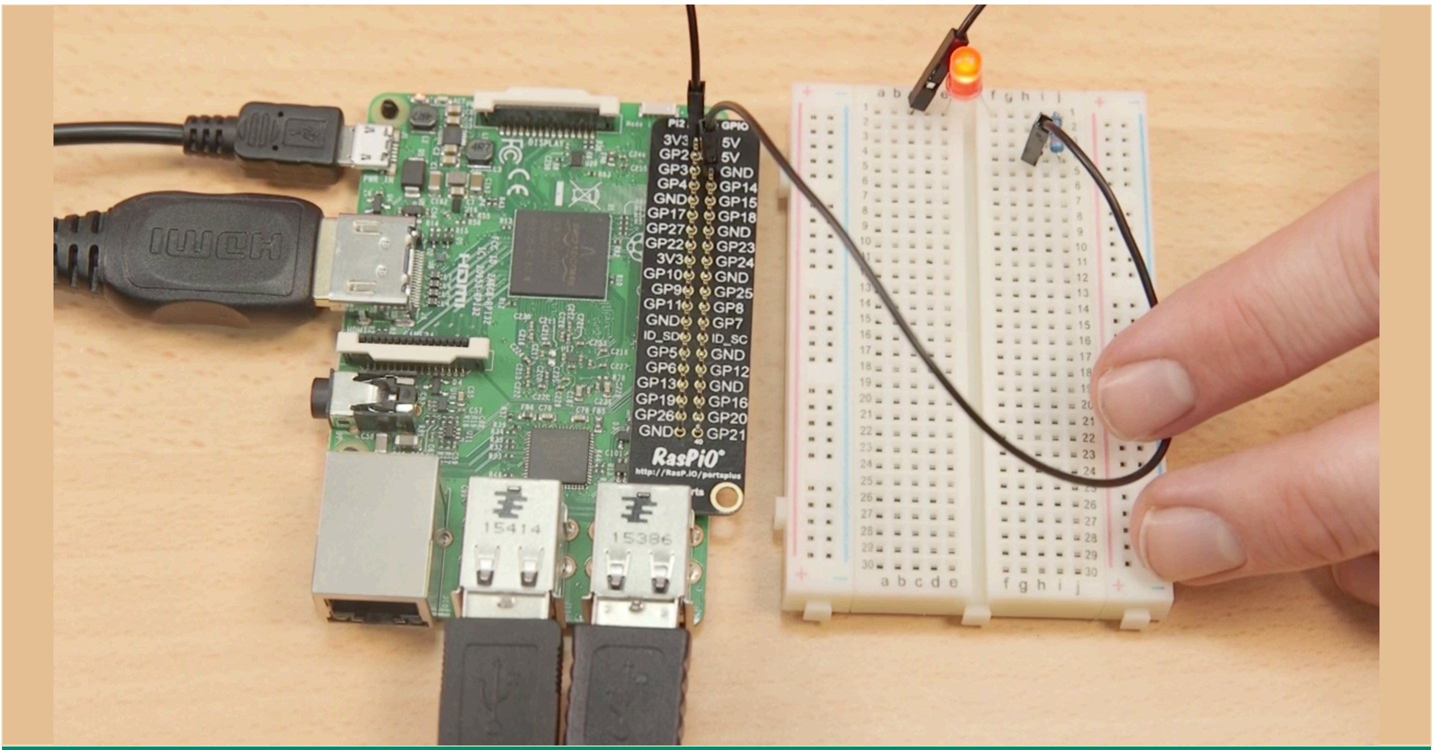
Turning LED on



Turning LED on







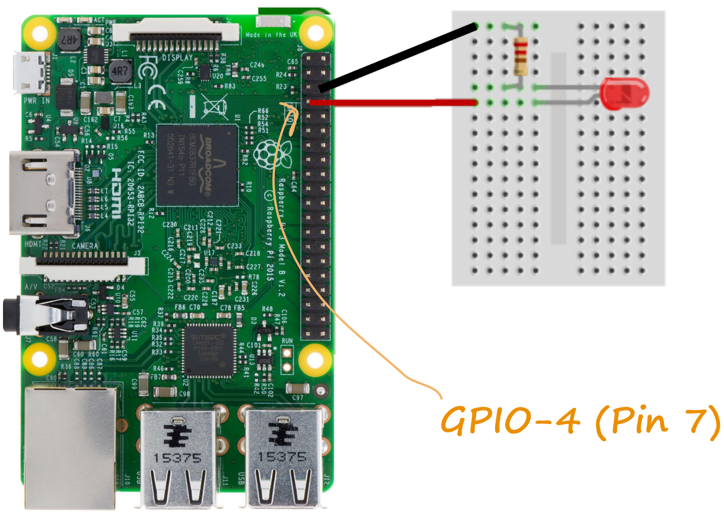
Observations

- Circuit is always on
  – Interesting but useless
- Can program the LED to turn on and off by using a GPIO pin
  – GPIO pins can be switched on and off
- Write some code that can control the behavior of the LED while making some subtle structural changes to the circuit

Programming the LED

- Remove the female-to-male jumper lead from the 3.3 V pin
- Connect the female-to-male jumper lead to GP4
  - Any other GPIO pin can do

Programming the LED



Programming LED Using RPi

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
LED = 4
GPIO.setup(LED,GPIO.OUT)

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

import RPi.GPIO libs

set pin type. use BCM, not pin number

GPIO 4 pin (Pin 7)

set LED pin as output

toggle light pin signal to low/high to make it blink.

```
# Import the GPIO and time libraries
import RPi.GPIO as GPIO
import time

# Set the pin designation type.
# In this case, we use BCM- the GPIO number- rather than the pin number itself.
GPIO.setmode (GPIO.BCM)

# So that you don't need to manage non-descriptive numbers,
# set "LIGHT" to 4 so that our code can easily reference the correct pin.
LIGHT = 4

# Because GPIO pins can act as either digital inputs or outputs,
# we need to designate which way we want to use a given pin.
# This allows us to use functions in the GPIO library in order to properly send and receive signals.
GPIO.setup(LIGHT,GPIO.OUT)


# Cause the light to blink until the keyboatrd is pressed, and print a message each time.
# To blink the light, we call GPIO.output and pass as parameters the pin number (LIGHT) and the state we want.
# True sets the pin to HIGH (sending a signal), False sets it to LOW.
# To achieve a blink, we set the pin to High, wait for a fraction of a second, then set it to Low.
# Adding keyboard interrupt with try and except so that program terminates when user presses Ctrl+C.
try:
    while True:
        GPIO.output(LIGHT,True)
        time.sleep(0.5)
        GPIO.output(LIGHT,False)
        time.sleep(0.5)
        print("blink")
except KeyboardInterrupt:
    GPIO.cleanup()
```

https://pypi.python.org/pypi/RPi.GPIO

Programming LED Using GPIOzero

- GPIOzero is another easy to use Raspberry PI Python

https://gpiozero.readthedocs.io/en/stable/

Iteration 1

```
# Import the gpiozero and time libraries
from gpiozero import LED
from time import sleep

# set "LIGHT" to 4 so that our code can easily reference the correct pin.
red_led = LED(4)

# Cause the light to blink and print a message each time.
# Adding keyboard interrupt with try and except so that program terminates when
user presses Ctrl+C.
try:
    while True:
      red_led.on()
      sleep(0.5)
      red_led.off()
      sleep(1)
      print("blink")
except KeyboardInterrupt:
    print("Bye")
```

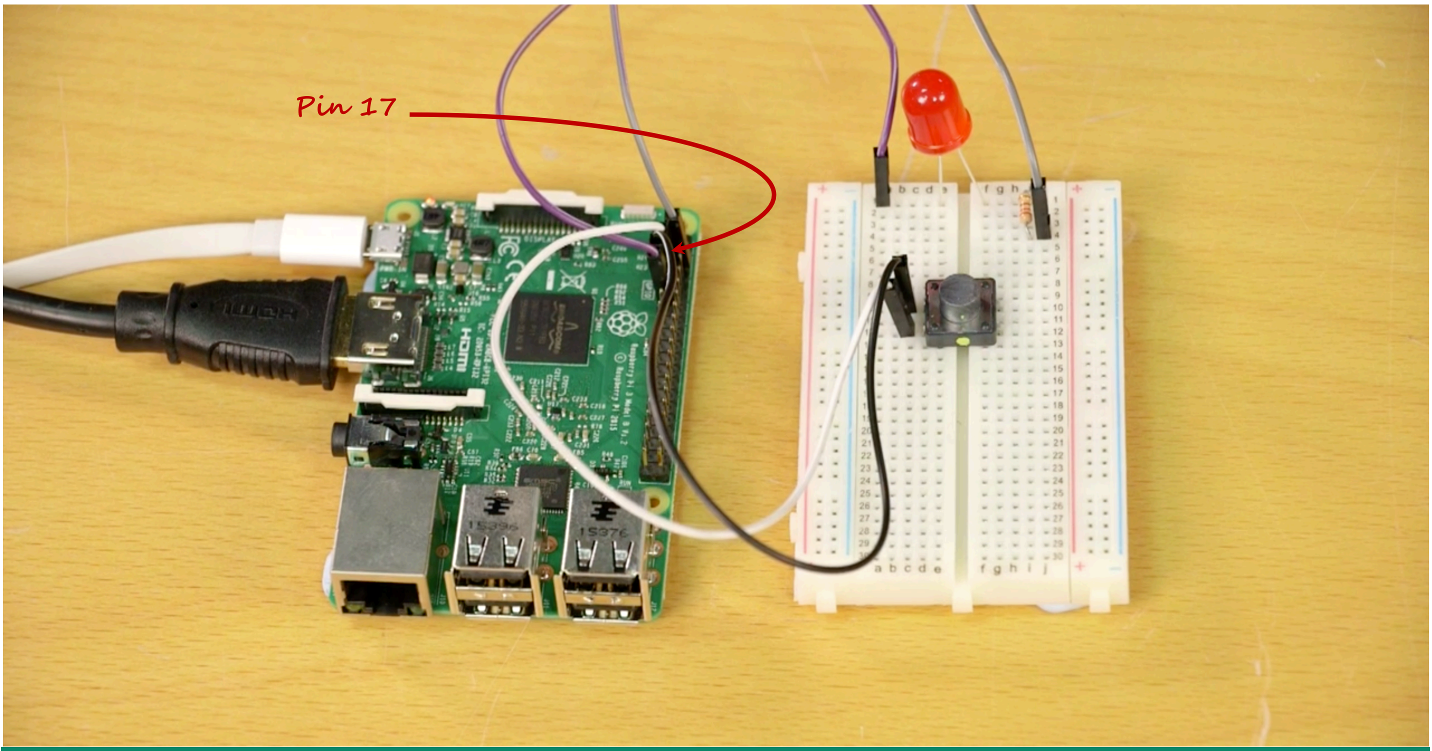
Iteration 2

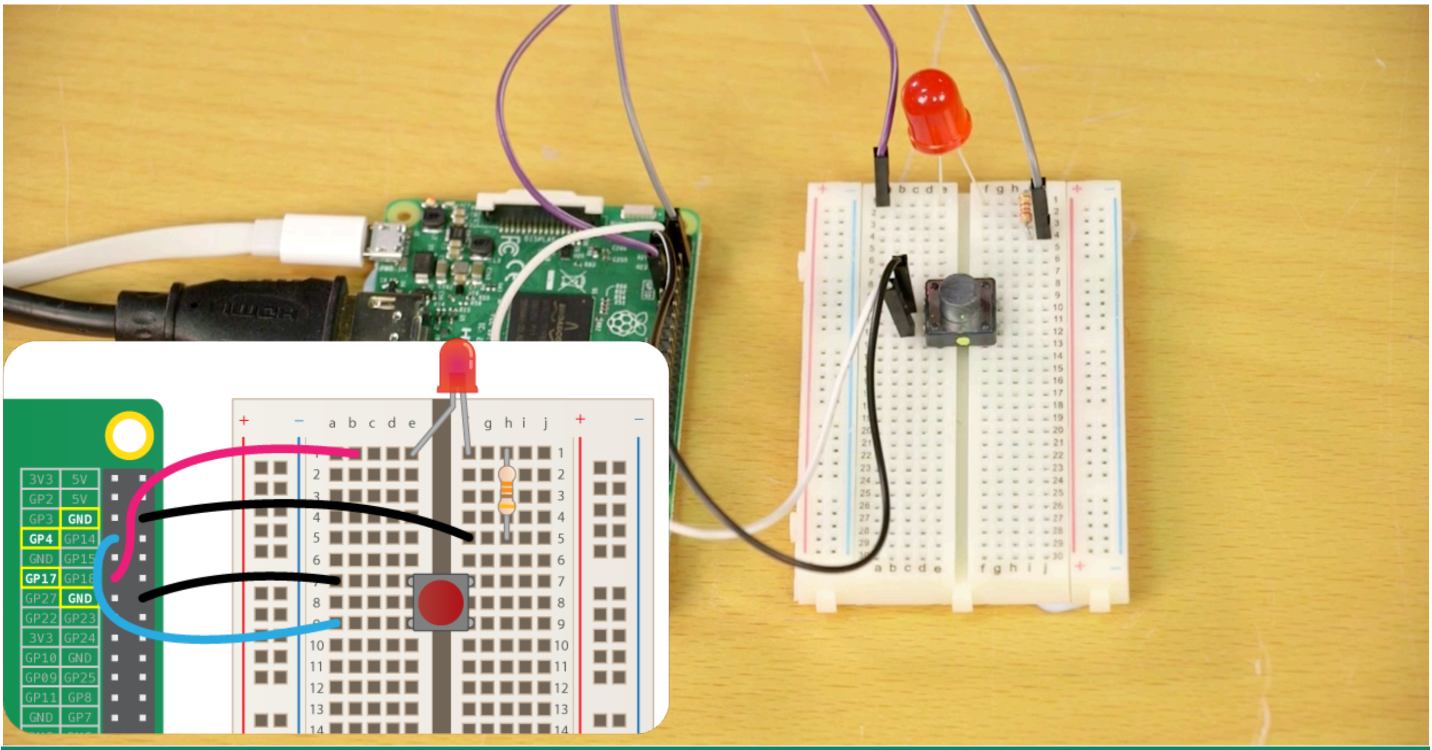
```
from gpiozero import LED
from time import sleep

red_led = LED(17)

# You may modify your code by adding the following function calls

red_led.blink(0.1, 0.2)             # red_led.blink(on_time, off_time, n, background=True)
red_led.blink(0.2, 0.1, 5)          # default: (1, 1, Infinite time, True)
red_led.toggle()                    # No need to keep track of whether the LED is on or off

sleep(1)

print(red_led.is_lit)               # Check if LED is on or off
red_led.toggle()
sleep(1)
print(red_led.is_lit)
```


Example 2: Controlling Buttons

```
from gpiozero importButton

btn = Button(4)

while True:
   btn.wait_for_press()
   print('You pressed me')
   btn.wait_for_release()
   print('You released me')
```

Example 3: Combing LEDs and Buttons





Iteration 1

```
from gpiozero import Button, LED
from time import time, sleep
from random import randint

led = LED(17)
btn = Button(4)

led.on()
btn.wait_for_press()
led.off()

while True:
     led.on()
     btn.wait_for_press()
     led.off()
```

Iteration 2

```
from gpiozero import Button, LED
from time import time, sleep
from random import randint

led = LED(17)
btn = Button(4)

led.on()
btn.wait_for_press()
led.off()

while True:
      sleep(randint(1,10))
      led.on()
      btn.wait_for_press()
      led.off()
```


Iteration 3

```
from gpiozero import Button, LED
from time import time, sleep
from random import randint

led = LED(17)
btn = Button(4)

led.on()
btn.wait_for_press()
led.off()

while True:
      sleep(randint(1,10))
      led.on() start = time()
      btn.wait_for_press()
      led.off()
      end = time()
```

Iteration 4

```python
from gpiozero import Button, LED
from time import time, sleep
from random import randint

led = LED(17)
btn = Button(4)

while True:
    sleep(randint(1,10))
    led.on()
    start = time()
    btn.wait_for_press()
    end = time()
    led.off()
    print(end - start)
```


Check for Understanding

- Which of these python programs would result in a LED (pin 17) which changes state whenever a Button (pin 4) is pressed.

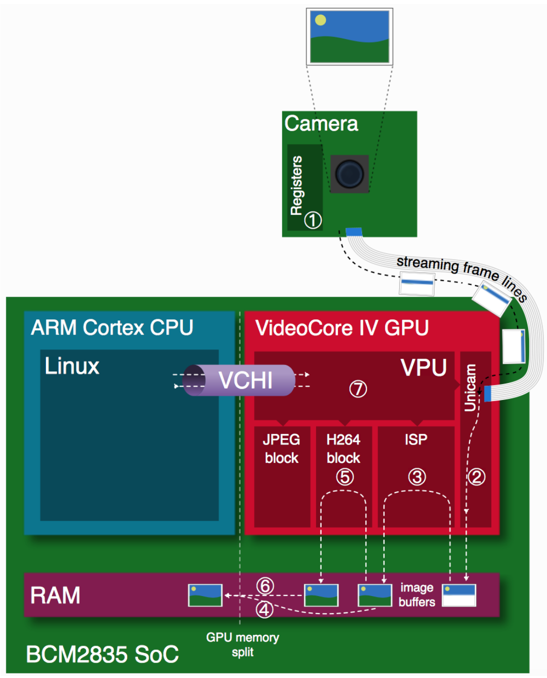
```python
from gpiozero import LED, Button
led = LED(17)
btn = Button(4)
while True:
btn.wait_for_press()
led.toggle()
```

```python
from gpiozero import Button
led = LED(17)
 btn = Button(4)
while True:
    btn.wait_for_press()
    led.toggle()
```

```python
from gpiozero import LED, Button
led = LED(17)
btn = Button(4)
while True:
    btn.wait_for_press()
    led.toggle()
```

```python
from gpiozero import LED, Button
led = LED(17)
btn = Button(4)
while True:
    btn.wait_for_press()
    led.on()
```

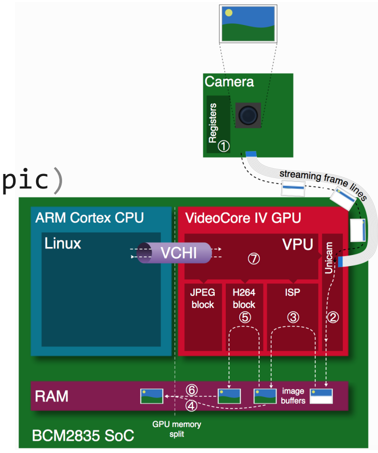
Example 4: Using the Camera



http://picamera.readthedocs.io/en/release-1.13/fov.html?highlight=camera

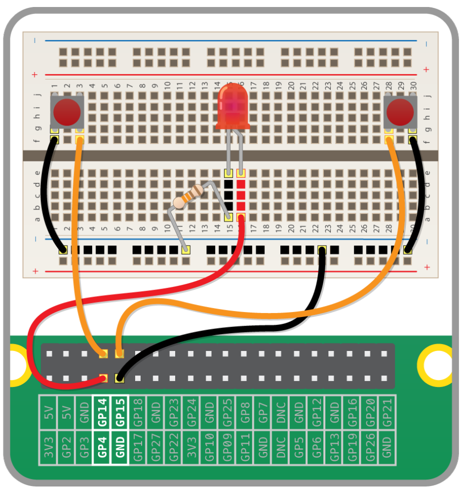
Using the Camera

```
import picamera
from time import sleep

pic = 1
with picamera.PiCamera() as camera:
    camera.resolution = (1024, 768)
    camera.capture('/home/pi/hamster/image%03d.jpg' % pic)
    pic += 1
    sleep(0.2)
```



To Probe Further …

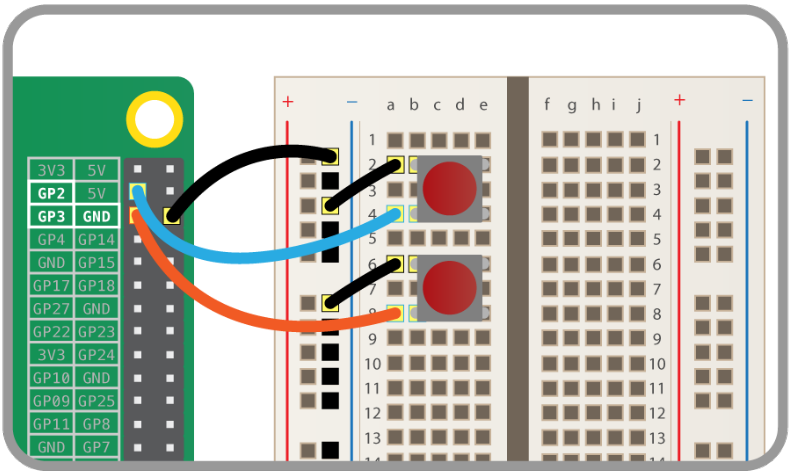
Quick Reaction Game



https://www.raspberrypi.org/learning/python-quick-reaction-game/

To Probe Further …

GPIO Music Box



https://www.raspberrypi.org/learning/gpio-music-box/worksheet/

GPIOzero API Devices

Input Devices
- Button
- Line Sensor
- Motion Sensor
- Light Sensor
- Distance Sensor
- DigitalInputDevice
- SmoothedInputDevice
- GPIODevice

Output Devices
- LED
- PWMLED
- RGBLED
- Buzzer
- Motor
- Servo
- AngularServo
- DigitalOutputDevice
- PWMOutputDevice

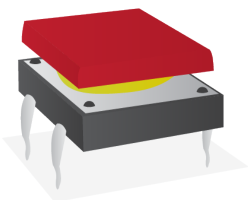
Some Useful GPIOzero API Devices

- Boards and Accessories
  - LEDBoard
  - LEDBarGraph
  - ButtonBoard
  - TrafficLights

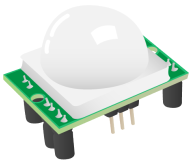
Lab 1

- Using GPIOzero, experiment with the frequency of the LED flashes, what's the fastest you can make it flash? Can you make it flash randomly?
- Can you create short (dot) flashes and long (dash) flashes, giving you the basics of Morse code, with this can you broadcast a message?
- Can you add extra LEDs to your breadboard and control them with other GPIO pins?
- With multiple LEDs could you create a simple light sequence in code, this could be something functional like a traffic light sequence or something fun like some blinky disco lights

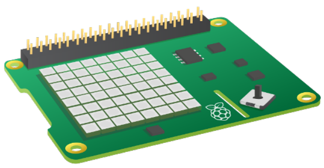
Raspberry Pi Hardware



push button or switch



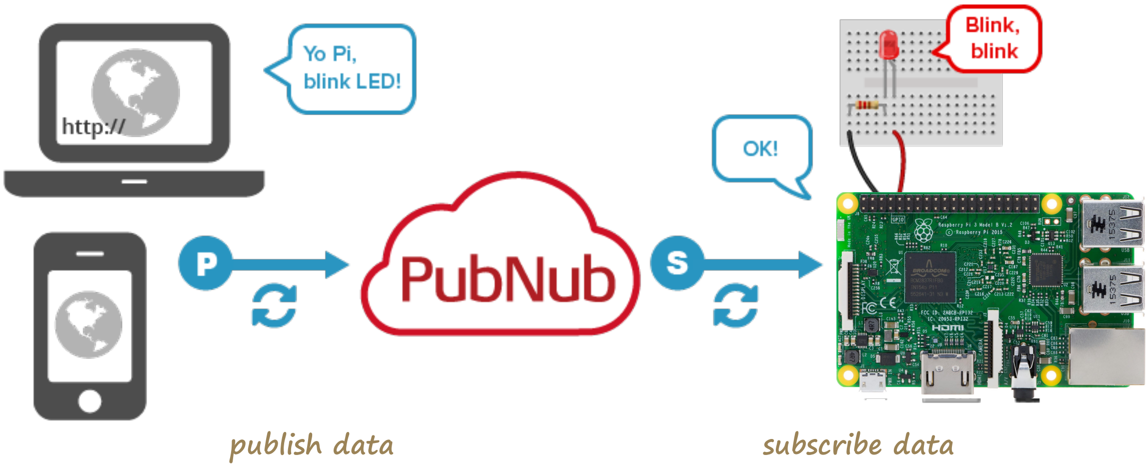
Individual Sensors



Sense HAT

Controlling the Pi from the Internet

Making it IoT: Remote-Controlled LED



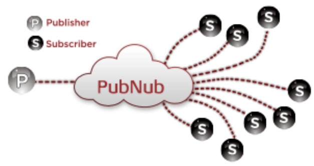
Making it IoT: Remote-Controlled LED

Subscribing data from a web client

```
pubnub = Pubnub(publish_key='demo', subscribe_key='demo')
channel = 'disco'

def _callback(m, channel):
  if m['led'] == 1:
    for i in range(6):
      GPIO.output(LED_PIN,True)
      time.sleep(0.5)
      GPIO.output(LED_PIN,False)
      time.sleep(0.5)
pubnub.subscribe(channels=channel, callback=_callback, error=_error)
```

`button.addEventListener('click', publish);`

When the button is clicked on browser, it publishes data {'led': 1}



To Probe further…

https://github.com/pubnub/workshop-raspberrypi/blob/master/web/disco.html

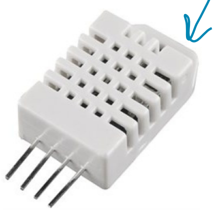
https://github.com/pubnub/workshop-raspberrypi/blob/master/projects-python/remote-led/remote-led.py

https://www.pubnub.com/docs/web-javascript/data-streams-publish-and-subscribe

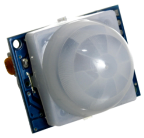
Sample IoT Projects

Hardware

- Pyroelectric IR Motion sensor
- Combinations of sensors and LED
- DHT22 Temperature & Humidity sensor

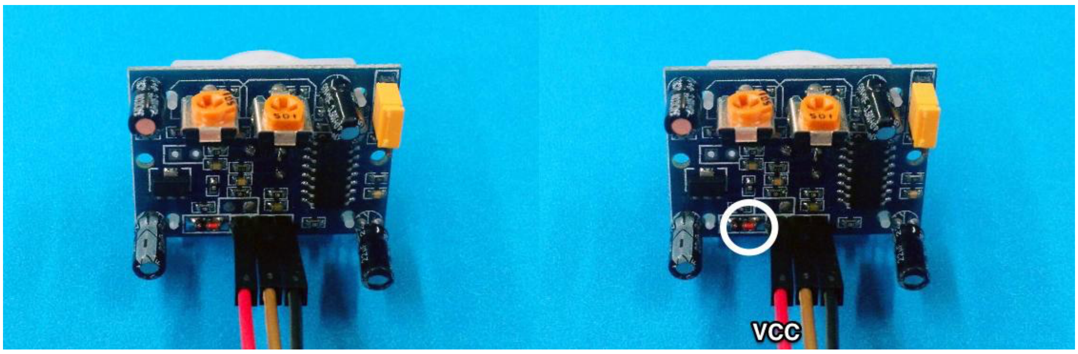


PIR Motion Sensor

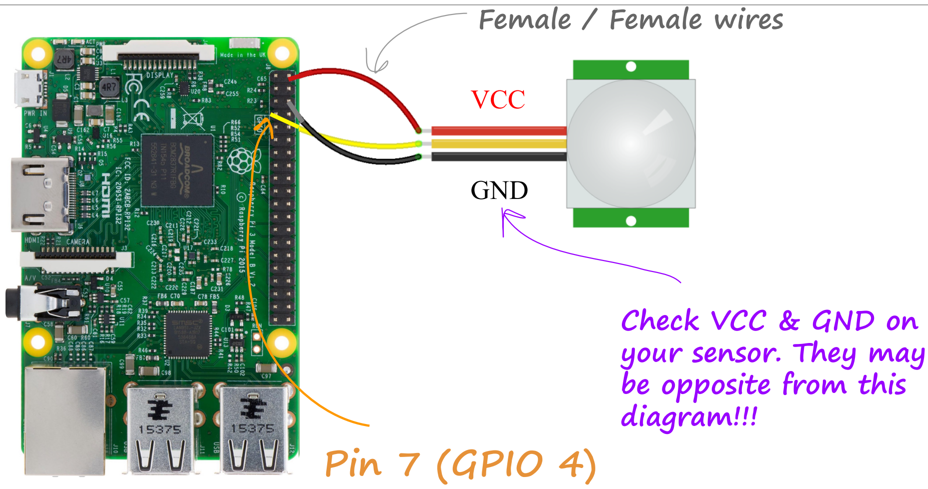


- It detects motions by measuring changes in IR radiation when an object moves around it.
- https://github.com/pubnub/workshop-raspberrypi/tree/master/projects-python/motion-sensor
- http://pubnub.github.io/workshop-raspberrypi/web/motion.html

PIR Motion Sensor



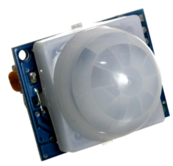
PIR Motion Sensor



PIR Motion Sensor w/ LED

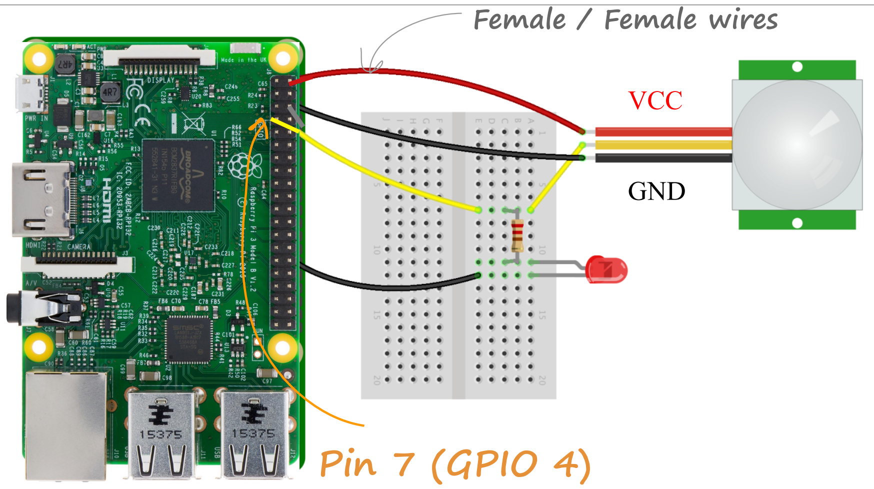
- Combination of the PIR motion sensor with a LED as a visual indicator

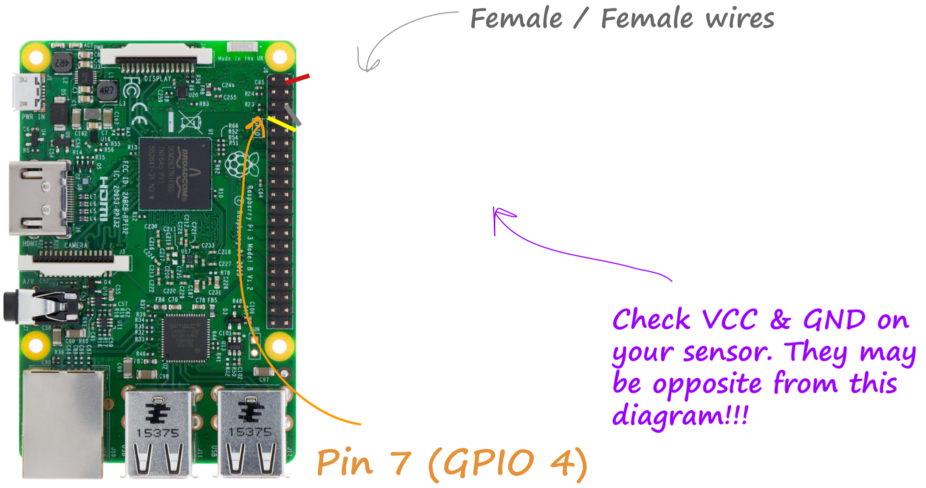
https://github.com/pubnub/workshop-raspberrypi/tree/master/projects-python/motion-led



http://pubnub.github.io/workshop-raspberrypi/web/motion.html

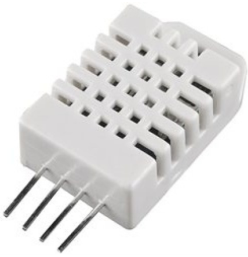
PIR Motion Sensor w/ LED





Data Visualization with Temperature Sensor

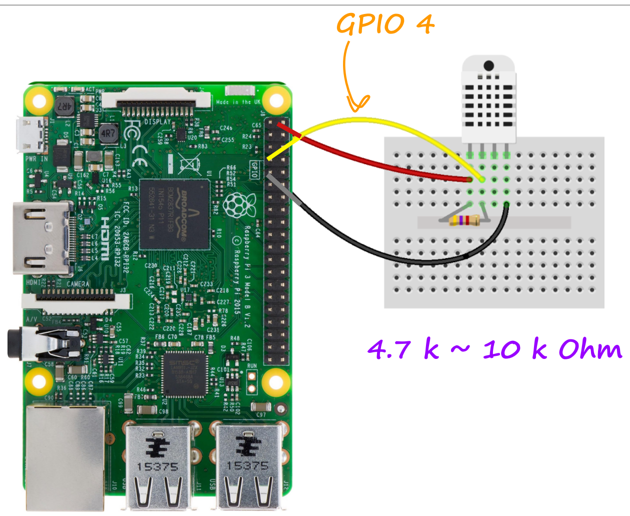
- It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin.



https://github.com/pubnub/workshop-raspberrypi/tree/master/projects-python/dht22

http://pubnub.github.io/workshop-raspberrypi/web/temperature.html

DHT22 Sensor



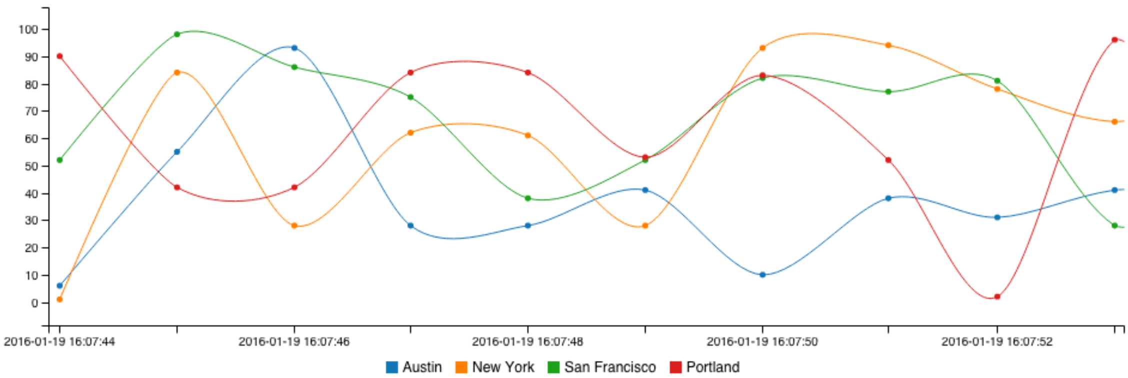
LAU
Lebanese American University

DHT22 Sensor

- Download & Install Adafruit DHT library:

```
~$ git clone
https://github.com/adafruit/Adafruit_Python_DHT.git

~$ cd Adafruit_Python_DHT
~$ sudo python setup.py install
```

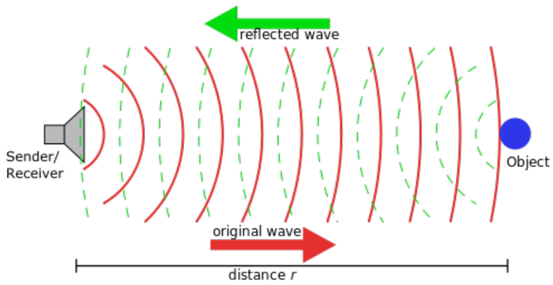
Realtime Data Graphs & Charts

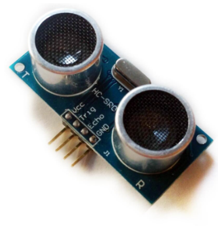


https://github.com/pubnub/eon-chart

Ultrasonic RangeFinder

- The HC-SR04 ultrasonic sensor uses sonar signals to determine distance to an object

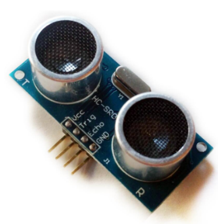




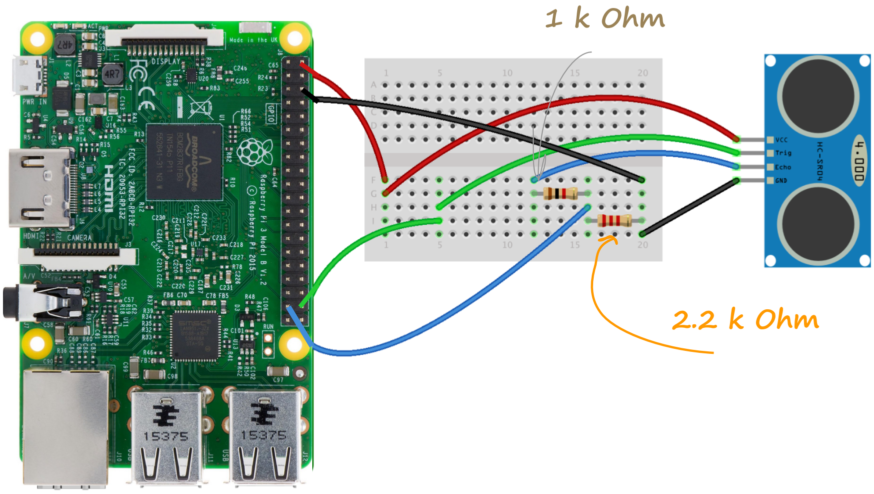
Ultrasonic RangeFinder

https://github.com/pubnub/workshop-raspberrypi/blob/master/projects-python/range-finder/

http://pubnub.github.io/workshop-raspberrypi/web/range.html



Ultrasonic RangeFinder



Recall: Course Project

- Any novel idea
  - Discover what other projects exist, get some inspiration, find out the pitfalls.
  - Generate Ideas of what you would like to achieve, then shortlist which are the best.
  - Design & Build starting small, get the basics right first before tackling the most challenging aspects.
  - Test & Improve your project, be prepared to fail, learn and iterate.
  - Share what you're doing, ask for help where needed, inspire others to tackle their own projects
- Extra credit for new and interesting ideas

Project Themes

- Music
  - https://www.raspberrypi.org/learning/gpio-music-box/worksheet/
- Nature
  - https://www.raspberrypi.org/?s=Nature
- Robotics
  - https://www.raspberrypi.org/?s=robotics
- Space
  - https://www.raspberrypi.org/education/programmes/astro-pi/
- Seasonal Holidays
  - https://www.raspberrypi.org/blog/holidays-with-pi/
- Game
  - https://www.raspberrypi.org/learning/hamster-party-cam/worksheet/

https://www.raspberrypi.org/resources/learn/

Credits

- Creative Commons Attributions
- LED circuit: Wikimedia
- PIR Sensor: Wikimedia / Oomlout
- Ultrasonic: Wikimedia / Georg Wiora (Dr. Schorsch)
- GPIO Pins: RaspberryPi-Spy.co.uk
- Raspberry Pi Foundation resources
- Also, great public domain images from Pixabay!