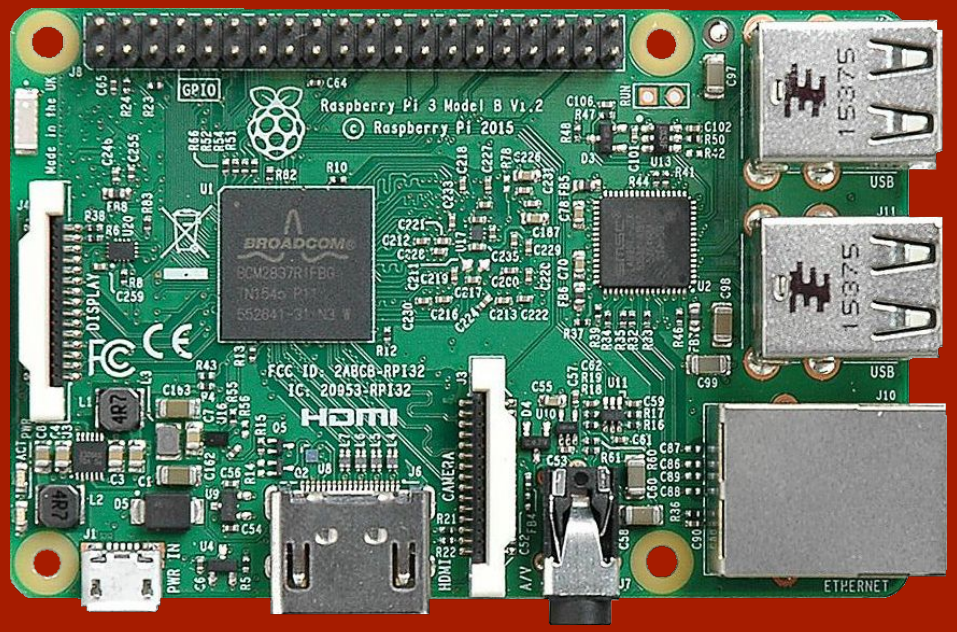
IoT and Practical Applications in Data Science

Lesson 1 - Internet of Things & Raspberry Pi

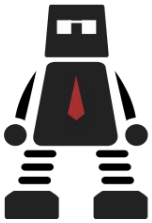


TINKERCADEMY

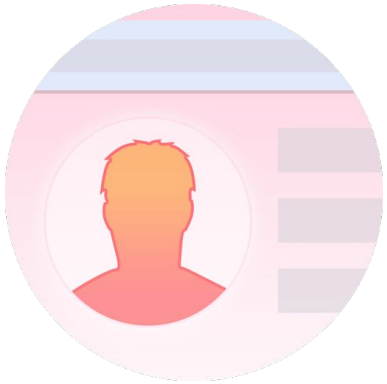
An Infocomm Club Appointed Vendor

About Us

Who We Are



TINKER
TINKER



About You

1. **Go to tk.sg/ricep2017yr2**

2. **Follow the instructions to sign up for Slack. Be sure to fill in your actual first and last names.**

Expectations

1. Feel free to help one another, but your work must be your own

2. Ask questions if you're confused. Feel free to PM me on Slack with any questions you don't want to ask in #discussion

3. Classroom clean-up: after working with Raspberry Pi make sure all wires are plugged back in, and no components are left on the desks

What Are We Doing?

- Learn about the Internet of Things
- Learn how to handle collected data
  - DataFrames in Python
  - Data Wrangling
  - Visualisations
- Learn about APIs
  - Client-side APIs
  - Server-side APIs
  - Creating our own API
- Learn how to use the Raspberry Pi
  - How to connect sensors
  - How to interact with these sensors using Python
- Learn about Design Thinking

Assessment

(Tentatively)

Problem Set (10%)

Quiz (20%)

Midterm Project (25%)

Final Project (40%)

Classroom Participation (5%)

What do you think is the **Internet of Things**?

Think and share

What is the Internet of Things?

Internet of Things (IoT)

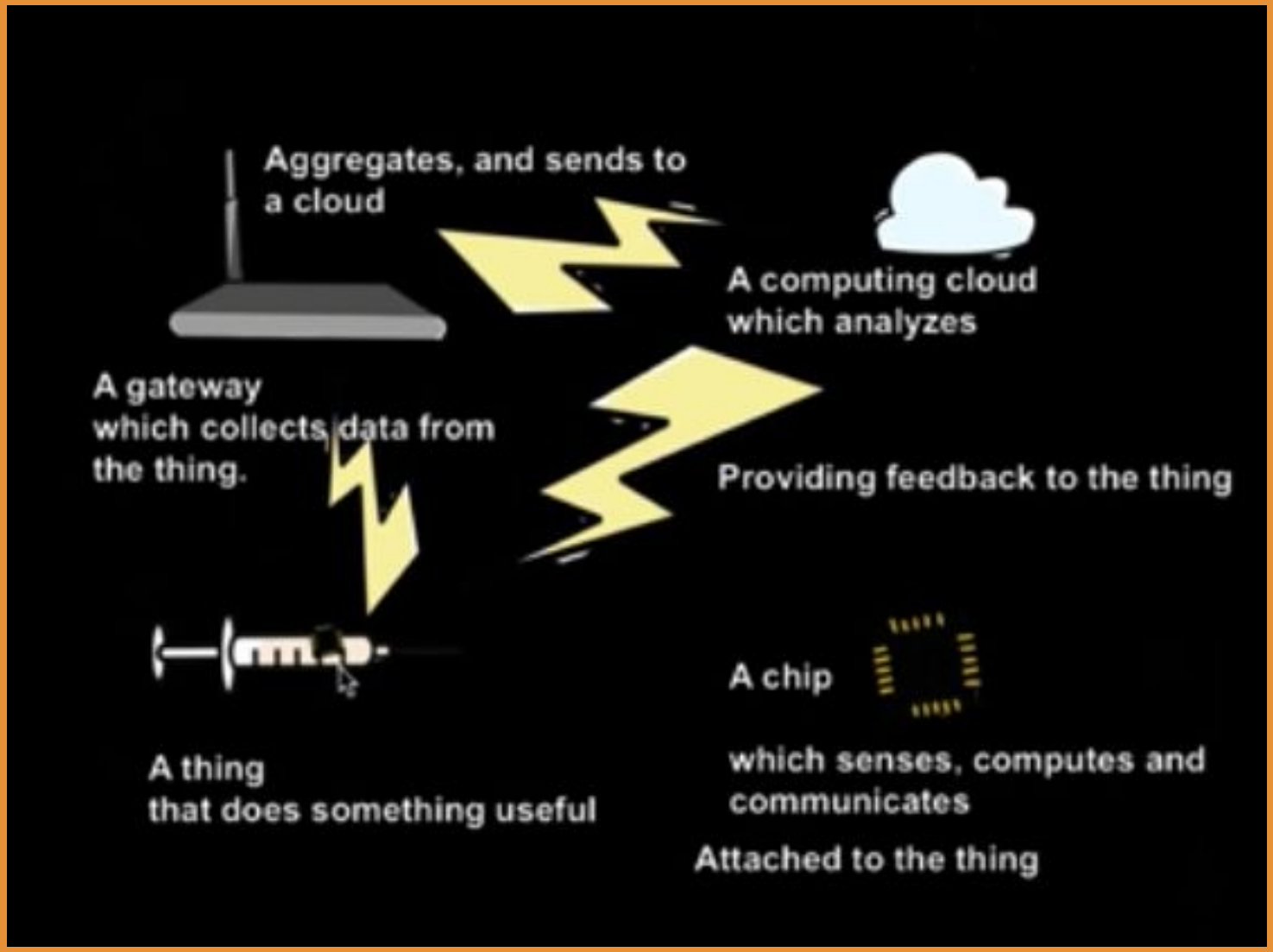
What is IoT? - A collection of **sensors and chips** embedded within physical devices that **gather and upload valuable data** to be analysed in a platform that can **inform decisions**.

In the cloud, data can be cleaned, processed, and analysed. Data creates value, because it can be used to drive business, engineering, development, and consumption decisions.

For example: Google Maps live traffic information.

Data is integral to IoT

- Data creates value from the connected things
- We have to process massive amounts of data -- clean, analyse, and deduce insights
- In order to help make sense of the trends coming from each small data source
- And to be able to provide feedback or take actions
- Techniques learnt in Data Science means we can learn more from the data and provide better insights and take better actions



Simplified Representation of IoT

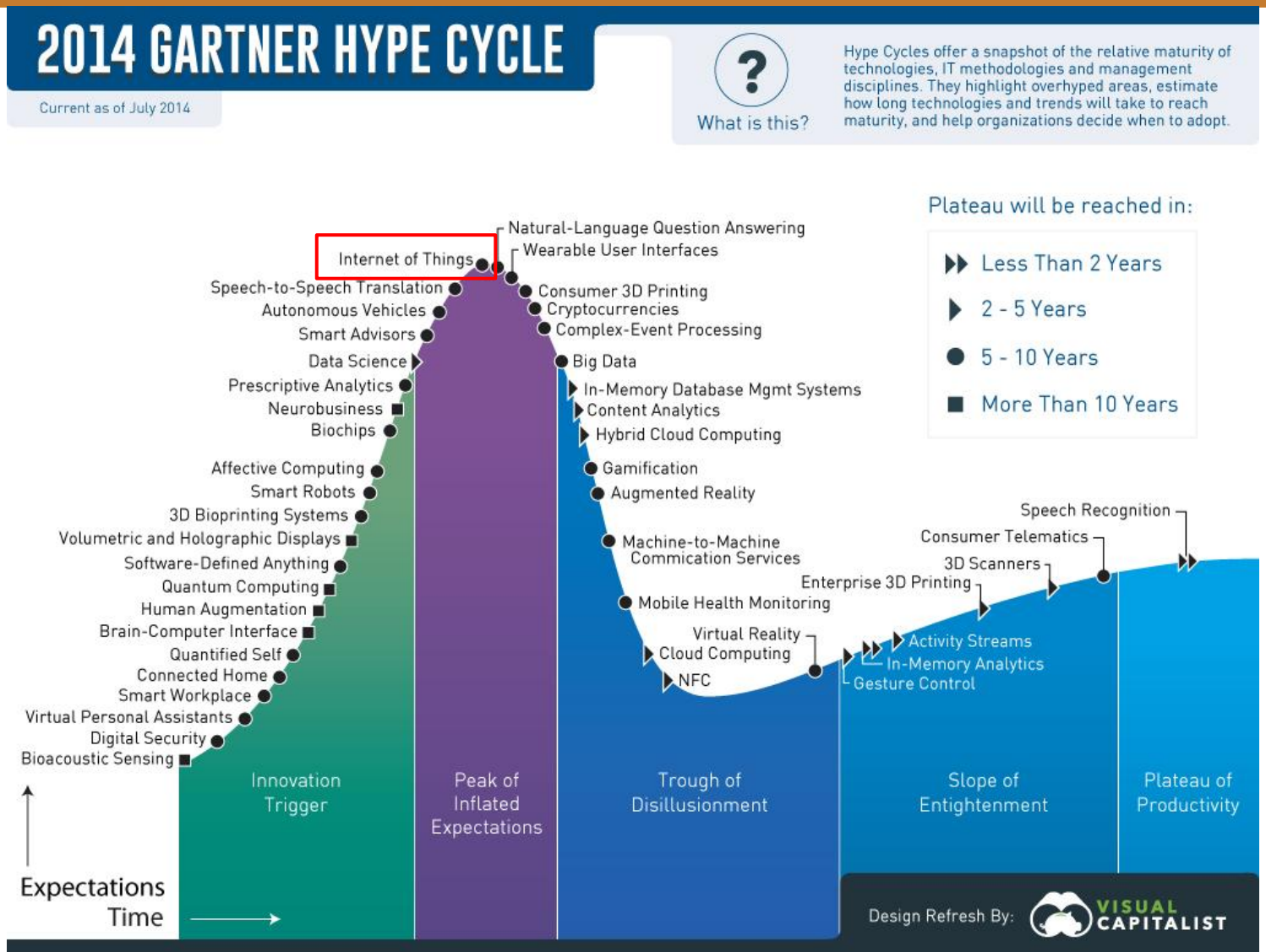
IoT Formal Definition

In 2015, the IEEE published an 86 page paper trying to establish a formal definition. They list a set of features that a system must have to be considered an IoT system:

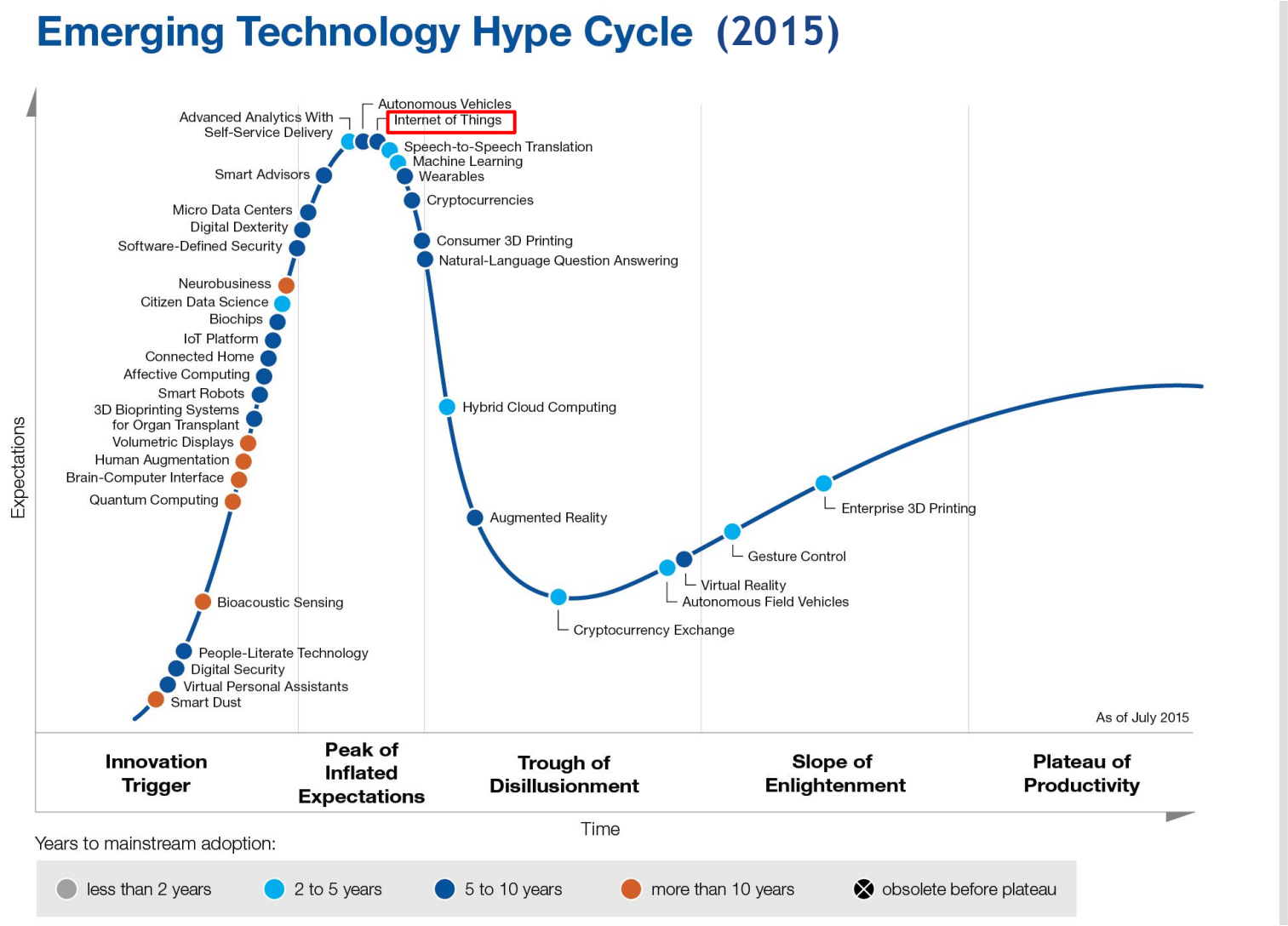
- Interconnection of Things
- Connection to the Internet
- Uniquely Identifiable Things
- Ubiquity
- Sensing/Actuation Capability
- Self-configurability
- Programmability

http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

Internet of Things (IoT)



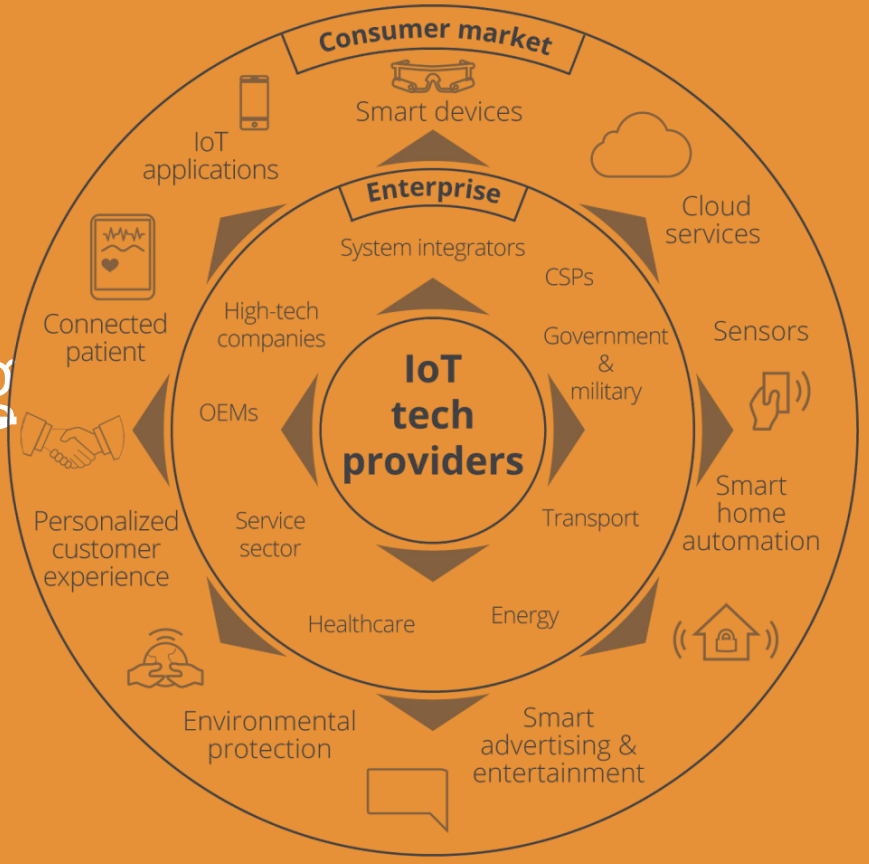
Internet of Things (IoT)



Getting There

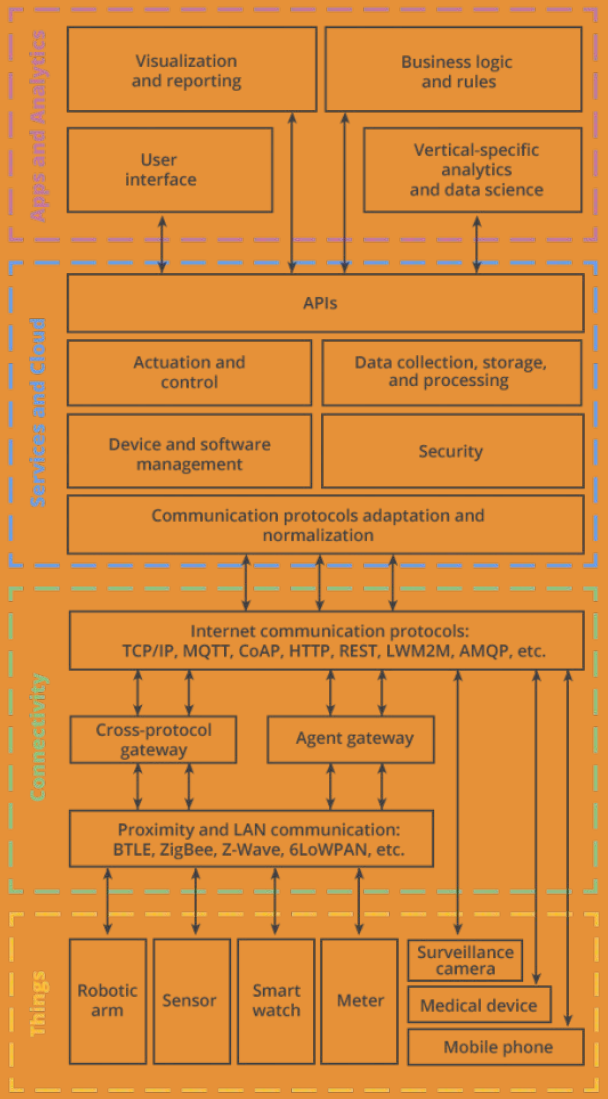
- Companies are building IoT platforms
  - IBM Watson IoT
  - Amazon Web Service IoT
  - Microsoft Azure IoT
  - ThingWorx IoT Platform
  - Cisco IoT Cloud Connect
  - Salesforce IoT Cloud
  - Android Things
  - Apple HomeKit
  - Samsung ARTIK IoT Platform
- Not interoperable; very fragmented
- **What does an IoT platform provide?**

https://www.kaaproject.org/what-is-iot/



IoT Platform

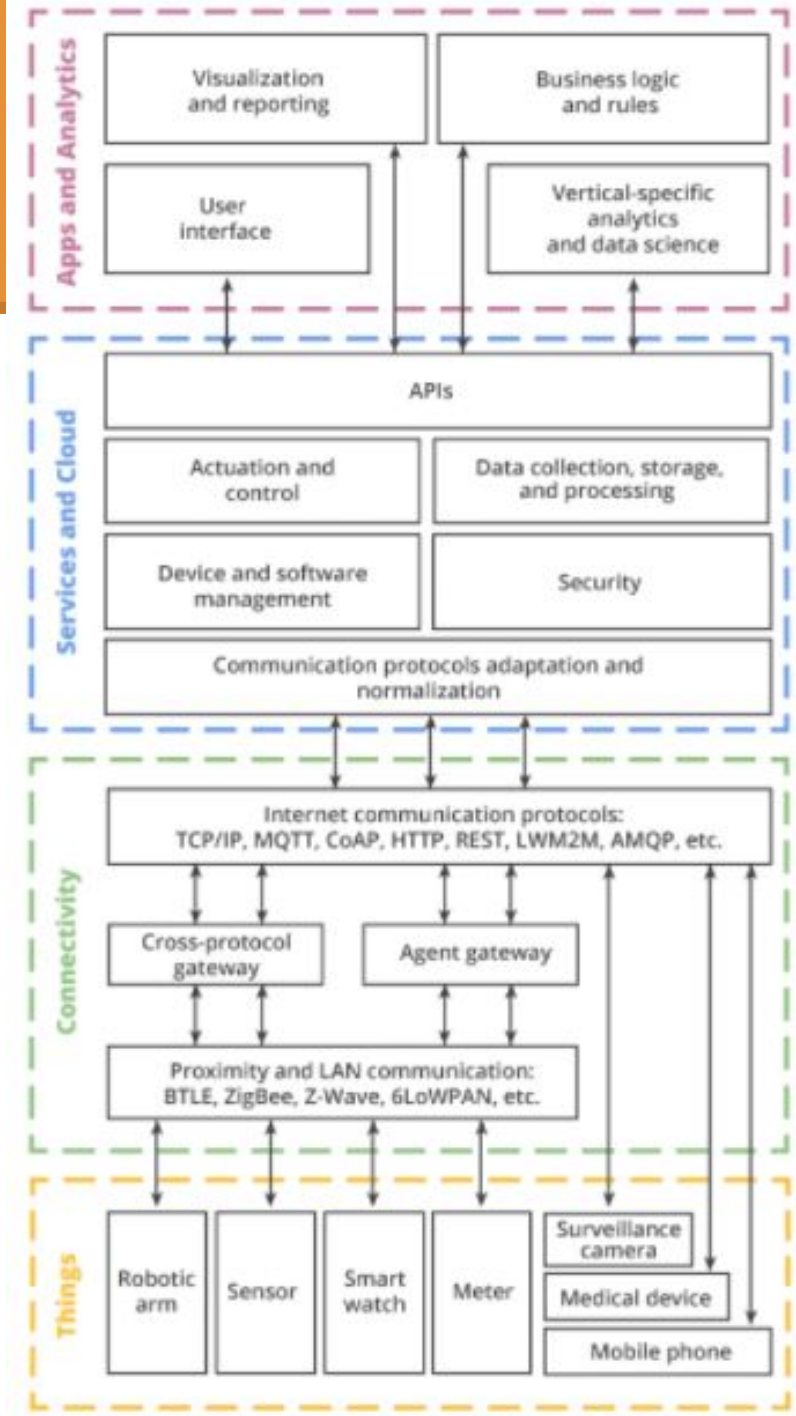
https://www.kaaproject.org/what-is-iot/



IoT Platform

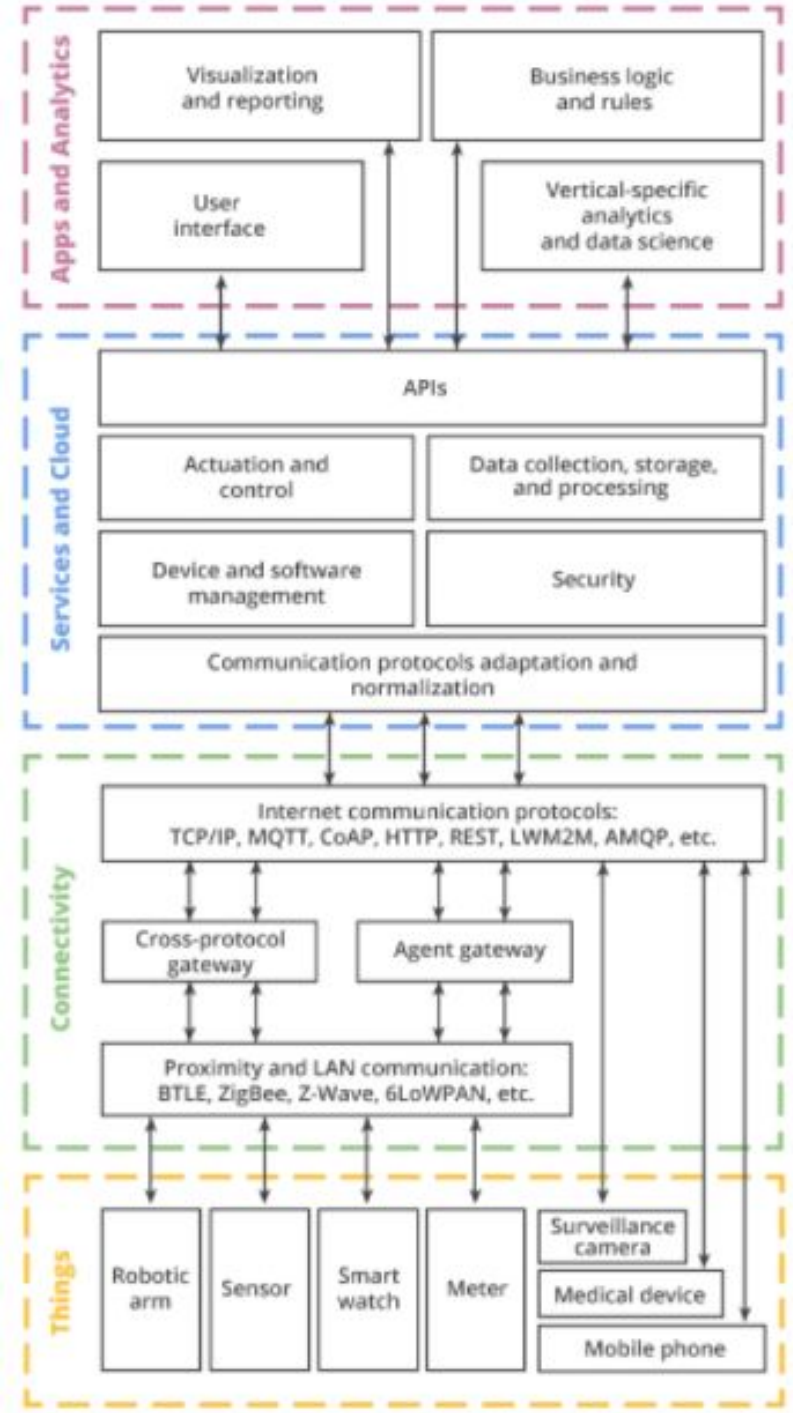
IoT stack

- The Analytics layer is where a lot of Data Science happens.



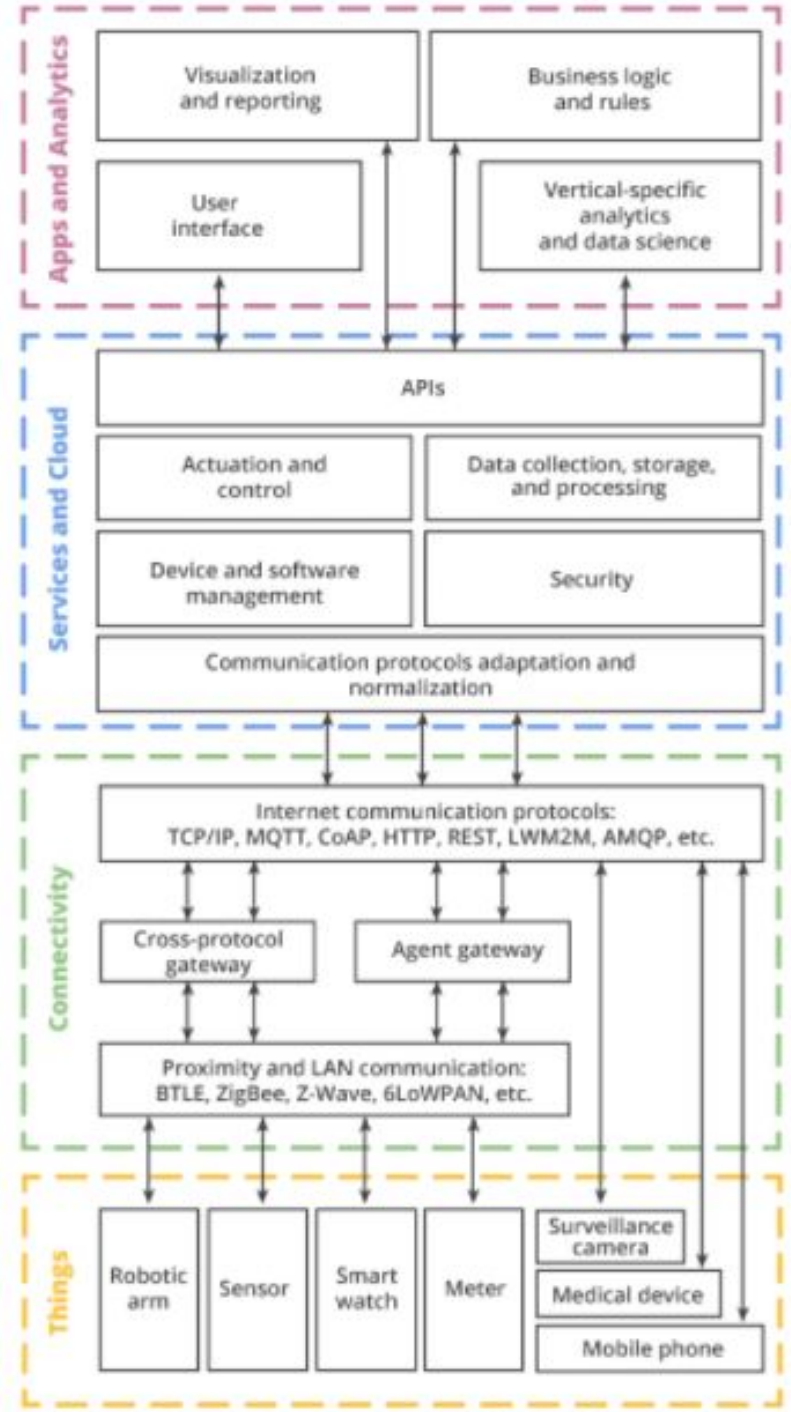
IoT stack

- At the Cloud layer, a big area of concern is Security and Privacy in data collection and storage.



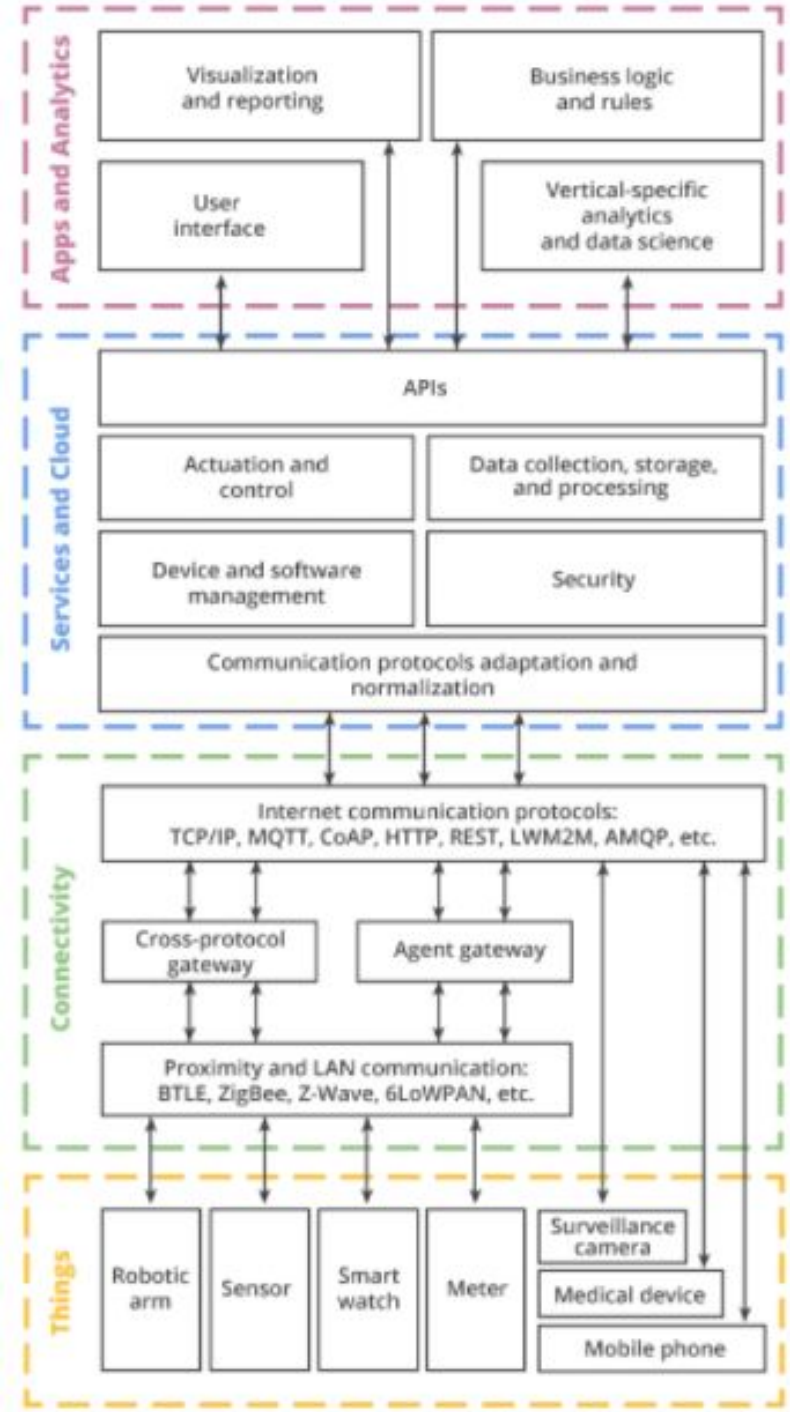
IoT stack

- To achieve connectivity within the local area, possibilities include Bluetooth Low Energy, WiFi, ZigBee.
- Across the Internet, new protocols are used such as MQTT and CoAP
- Why not just HTTP? -- small devices have limited processing power. Energy constraints means efficiency matters!



IoT stack

- At the Things layer, microcontrollers such as Arduino and Raspberry Pi enable us to create low-energy devices with a whole bunch of attached sensors.



Why/How did IoT take off?

1. Circuits became more advanced
    - Smaller
    - Cheaper to build
    - High computing capability
    - Require less power to run

2. Wireless technologies became more advanced
    - More powerful (greater range)
    - Require less power to run (don't require constant power connection)

3. Networks became more advanced
    - Access available everywhere
    - Higher bandwidth

We're using all 3 of these this term!

IoT in the News

IoT is now growing faster than smartphones

U.S. mobile carriers added more cars than phones to their networks in the second quarter

By Stephen Lawson FOLLOW

IDG News Service | Aug 10, 2016 6:36 PM PT

RELATED TOPICS

Internet of Things

Car Tech

Mobile & Wireless

1 COMMENT

If there were any doubt that IoT is for real, one fact ought to dispel it: For the first time, U.S. mobile operators are adding IoT connections to their networks faster than they're adding phones.

In fact, cars alone are getting connected to cellular networks faster than anything else, according to statistics compiled by Chetan Sharma Consulting for the second quarter of this year. Counting all U.S. carriers, about 1.4 million cars got connected to cellular networks in the quarter, compared with 1.2 million phones and less than 900,000 tablets.

http://www.computerworld.com/article/3106446/internet-of-things/iot-is-now-growing-faster-than-smartphones.html

More Possible Examples

In 2014, BusinessInsider brainstormed possible IoT applications:

> It's 2025, and you hail a Google cab to go across town. Your Google cab is self-driving and free! When you sit down, the screens inside the cab begin playing advertisements for products that you were just browsing over the weekend. When you were shopping, the products you were browsing sent cookies to your smartphone. The cab reads these cookies and plays relevant advertisements in return for a "free" cab ride.

Source: http://www.businessinsider.com.au/9-real-life-scenarios-that-show-how-the-internet-of-things-could-transform-our-lives-2014-8

More Possible Examples

In 2014, BusinessInsider brainstormed possible IoT applications:

> In 2025, all cars are now self-driving. What's more is that all the cars on the road can talk to each other, and better yet, talk to the traffic lights. Traffic lights now become completely optimized and efficient. Better yet, the greater traffic system talks to all cars on the road everywhere, spreading out traffic via different routes based on congestion and wait times.

Source: http://www.businessinsider.com.au/9-real-life-scenarios-that-show-how-the-internet-of-things-could-transform-our-lives-2014-8

More Possible Examples

In 2014, BusinessInsider brainstormed possible IoT applications:

> You're out biking 30km away from home. Your bike slips and you fall. The accelerometer in your helmet has detected that you've hit your head. Your helmet 'calls out' to see if your bike is nearby (it is) and if it's moving (it's not). Your bike sounds an emergency alarm for five seconds. If it's not deactivated, your bike sends your exact location to an ambulance, and any other emergency contacts you've pre-listed. Your bike alarm continues to sound, attracting the attention of passers-by. It also sends out a signal to the road sign 1km down the road warning motorists to slow down.

Source: http://www.businessinsider.com.au/9-real-life-scenarios-that-show-how-the-internet-of-things-could-transform-our-lives-2014-8

IoT Example: Connected Cows

IoT Challenges

Many challenges as IoT continues to grow

- Standards/Platforms - many different standards and platforms; not all can interact with one another
- Security - difficult to secure connections, especially without concrete standards
- Adaptability - easy for companies to integrate
- Scalability - easy to grow
- Maintenance - easy to repair/modify/update (imagine updating millions of systems worldwide)

IoT in the News

The Internet Of Things Is a Security And Privacy Dumpster Fire And The Check Is About To Come Due

"Systems are filled with externalities that affect other systems in unforeseen and potentially harmful ways. What might seem benign to the designers of a particular system becomes harmful when it's combined with some other system. Vulnerabilities on one system cascade into other systems, and the result is a vulnerability that no one saw coming and no one bears responsibility for fixing. The Internet of Things will make exploitable vulnerabilities much more common. It's simple mathematics. If 100 systems are all interacting with each other, that's about 5,000 interactions and 5,000 potential vulnerabilities resulting from those interactions. If 300 systems are all interacting with each other, that's 45,000 interactions. 1,000 systems: 12.5 million interactions. Most of them will be benign or uninteresting, but some of them will be very damaging."

https://www.techdirt.com/articles/20160725/09460835061/internet-things-is-security-privacy-dumpster-fire-check-is-about-to-come-due.shtml

RISK ASSESSMENT —

“Internet of Things” security is hilariously broken and getting worse

Shodan search engine is only the latest reminder of why we need to fix IoT security.

J.M. PORUP (UK) - 1/23/2016, 11:30 PM

138

Shodan, a search engine for the Internet of Things (IoT), recently launched a new section that lets users easily browse vulnerable webcams.

The feed includes images of marijuana plantations, back rooms of banks, children, kitchens, living rooms, garages, front gardens, back gardens, ski slopes, swimming pools, colleges and schools, laboratories, and cash register cameras in retail stores, according to Dan Tentler, a security researcher who has spent several years investigating webcam security.

"It's all over the place," he told Ars Technica UK. "Practically everything you can think of."

http://arstechnica.com/security/2016/01/how-to-search-the-internet-of-things-for-photos-of-sleeping-babies/

CRUNCH NETWORK

The IoT threat to privacy

Posted Aug 14, 2016 by ***Christine Bannan***

Christine Bannan

As the Internet of Things becomes more widespread, consumers must demand better

https://techcrunch.com/2016/08/14/the-iot-threat-to-privacy/

Security Risks of IoT

According to documents inside the cache, a CIA program named "Weeping Angel" provided the agency's hackers with access to Samsung Smart TVs, allowing a television's built-in voice control microphone to be remotely enabled while keeping the appearance that the TV itself was switched off, called "Fake-Off mode." Although the display would be switched off, and LED indicator lights would be suppressed, the hardware inside the television would continue to operate, unbeknownst to the owner. The method, co-developed with British intelligence, required implanting a given TV with malware – it's unclear if this attack could be executed remotely, but the documentation includes reference to in-person infection via a tainted USB drive. Once the malware was inside the TV, it could relay recorded audio data to a third party (presumably a server controlled by the CIA) through the included network connection.

Security Risks of IoT

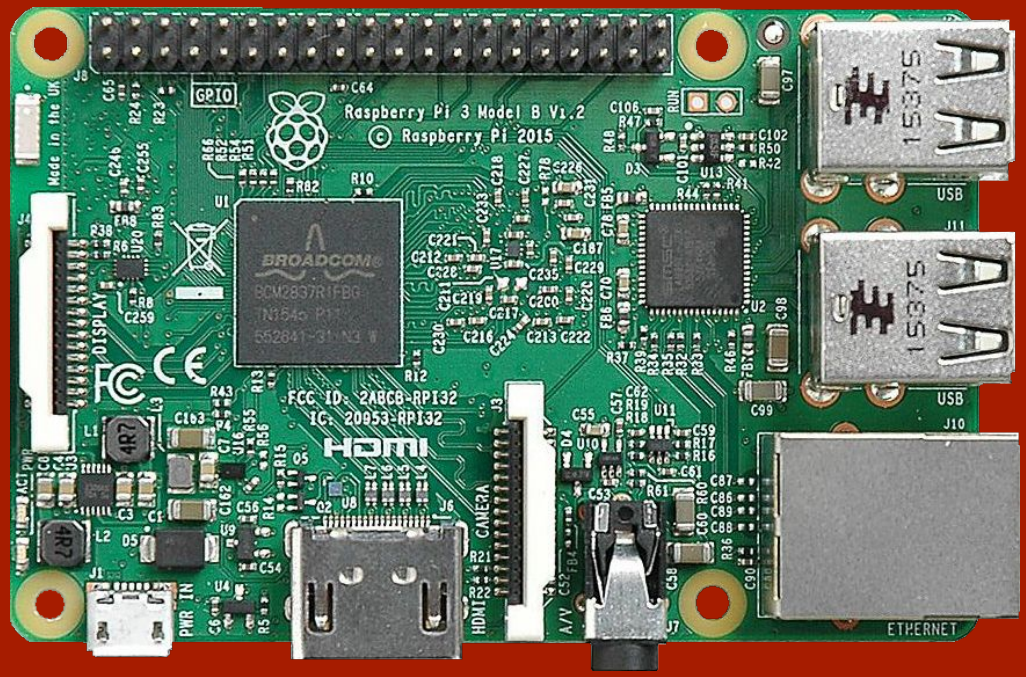
Analysis of a university firewall identified over 5,000 devices making hundreds of Domain Name Service (DNS) lookups every 15 minutes, slowing the institution's entire network and restricting access to the majority of internet services.

"We identified that this was coming from their IoT network, their vending machines and their light sensors were actually making the requests"

How to store data?

Before we get into learning how to connect sensors and create electronic projects with the RPi, we're going to first learn some python.

We're going to start with learning how to store data so that we can access it efficiently.



Raspberry Pi

What is Raspberry Pi?

What is a Raspberry Pi?

from **Raspberry Pi Foundation** PRO 2 years ago | more

More from Raspberry Pi Foundation

Autoplay on

What is Raspberry Pi?

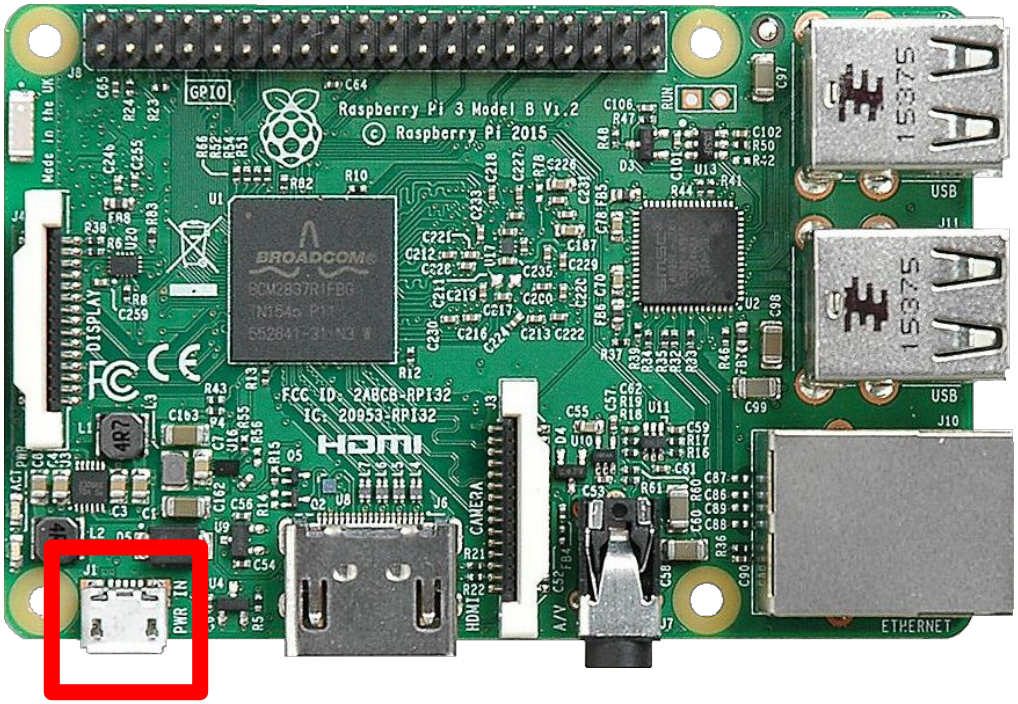
- Single-board computer
- Developed in the UK by the Raspberry Pi Foundation
- Intended to stimulate the teaching of basic computer science in schools
- Also great for home electronics projects!
- We’re not using it to teach coding specifically, we’re taking it one step further and using it to create electronics projects that can gather data for us

What is Raspberry Pi?

- The latest model (Raspberry Pi 3) has 4× ARM Cortex-A53 1.2GHz processor and 1GB RAM
- Besides being a full-fledged computer running a Linux Operating System, it also has GPIO pins which eases the interfacing with electronics components.
- It has USB ports, so it’s easier to attach mouse/keyboard/WiFi.
- The design uses an SD card (instead of a hard drive) for booting and storage.

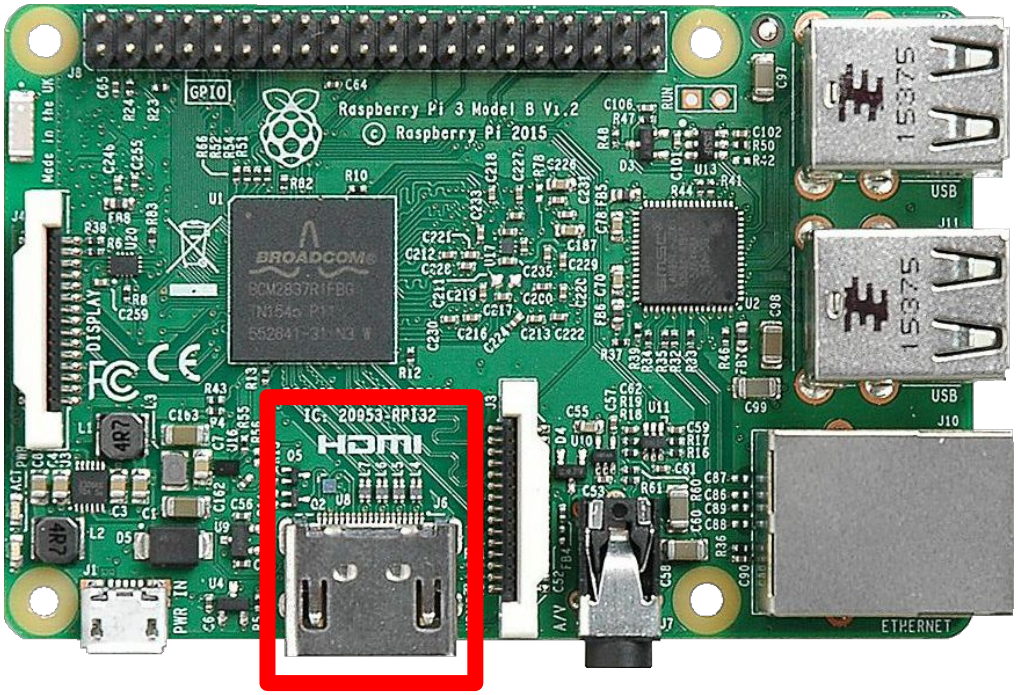
What is Raspberry Pi?

- MicroUSB Power connector



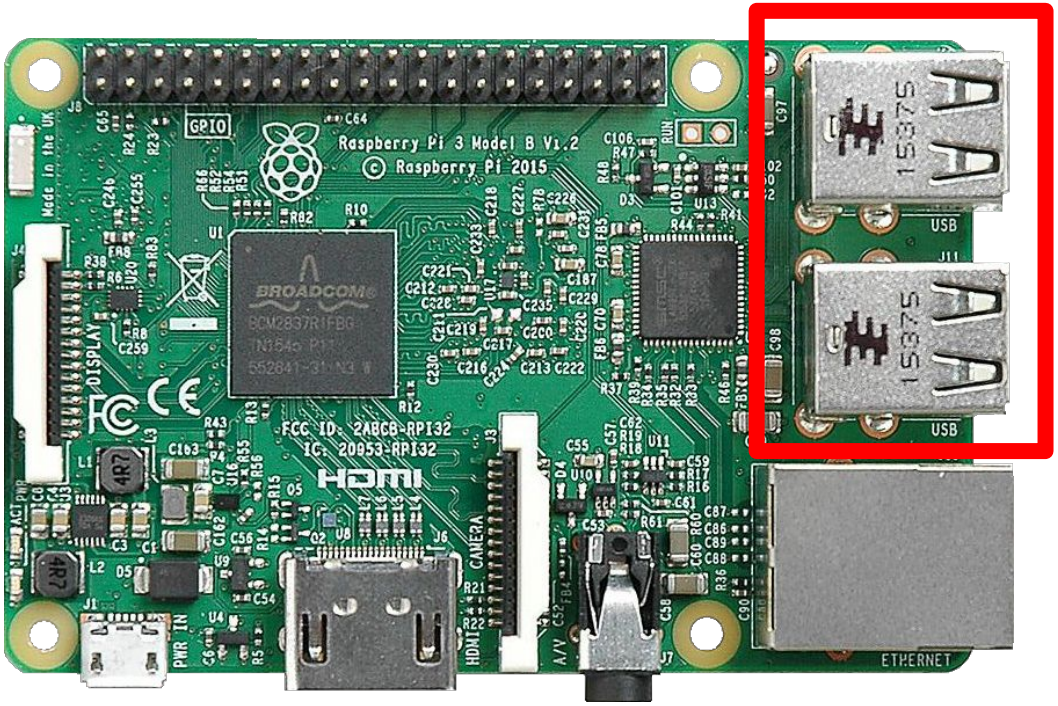
What is Raspberry Pi?

- HDMI connection



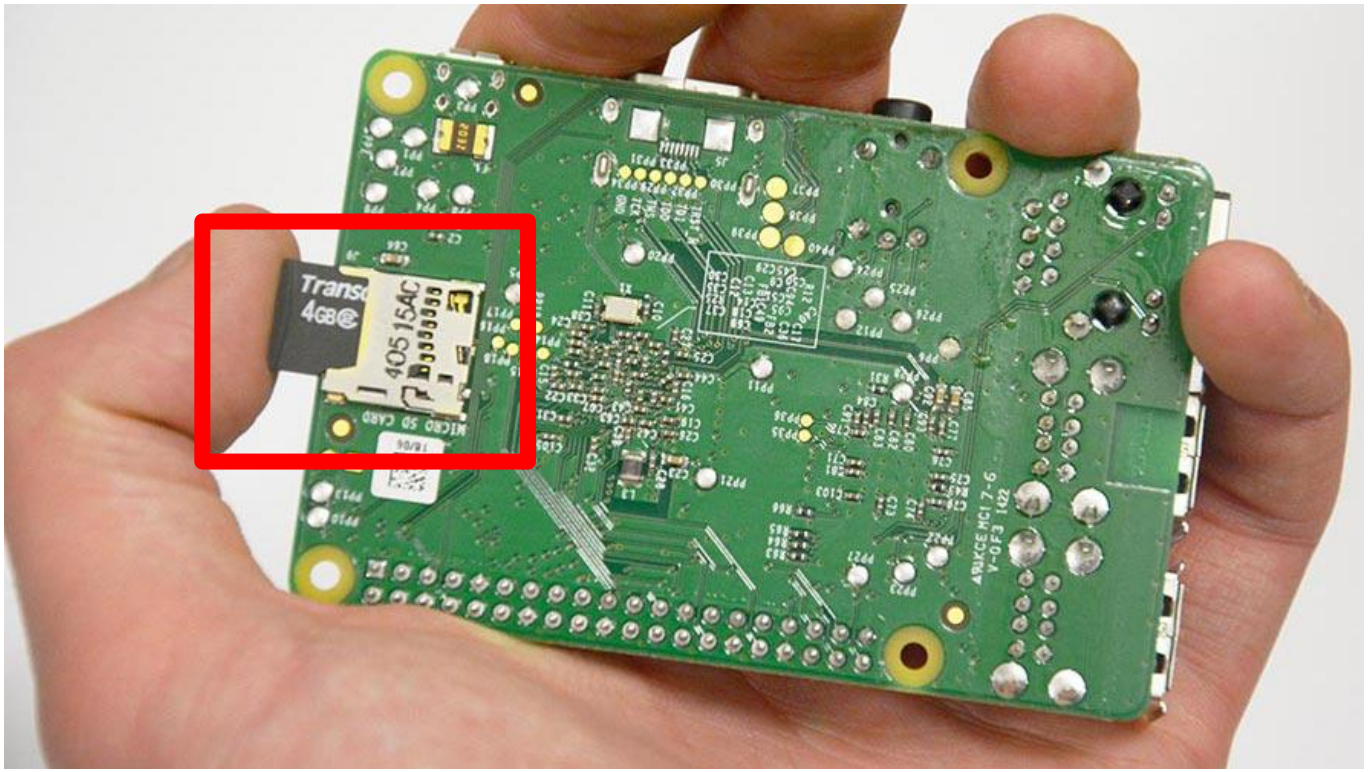
What is Raspberry Pi?

- USB ports



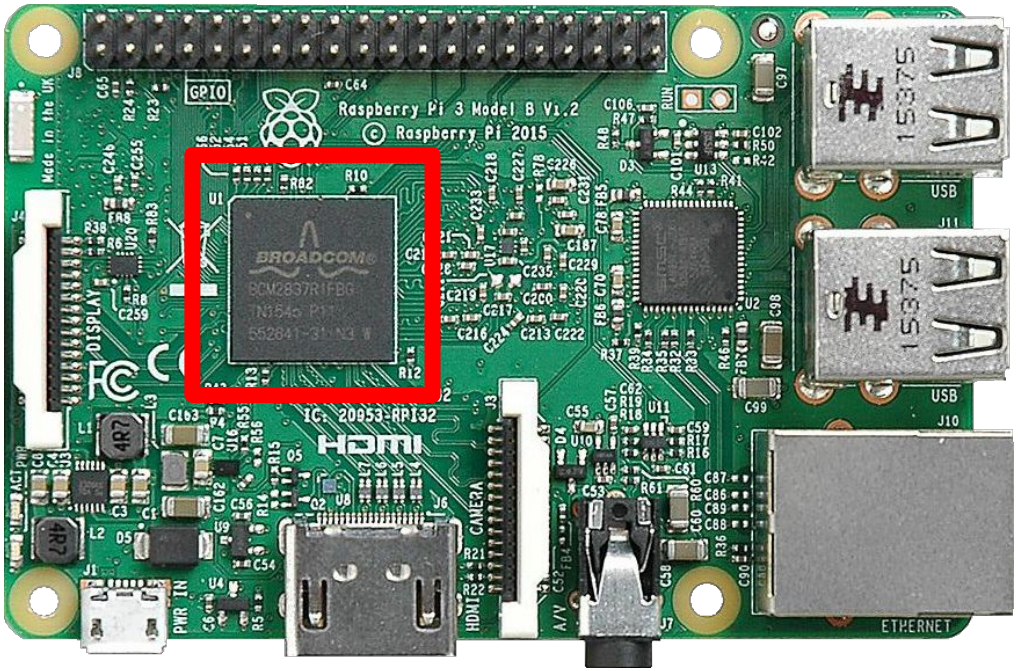
What is Raspberry Pi?

- microSD slot



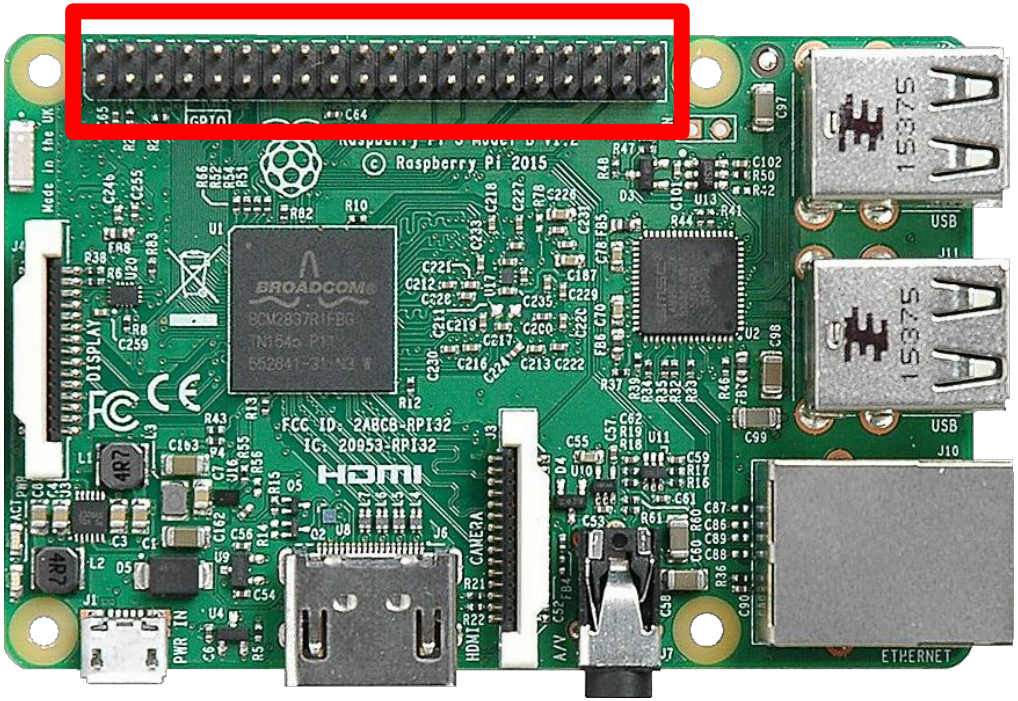
What is Raspberry Pi?

- Processor



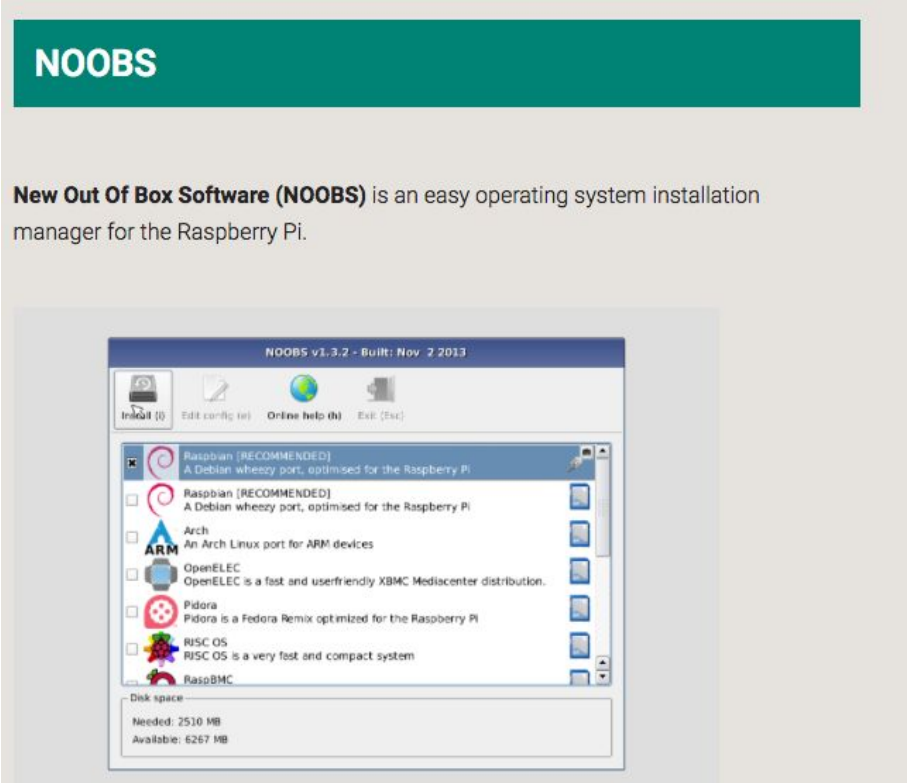
What is Raspberry Pi?

- 40 GPIO (General Purpose Input/Output) pins



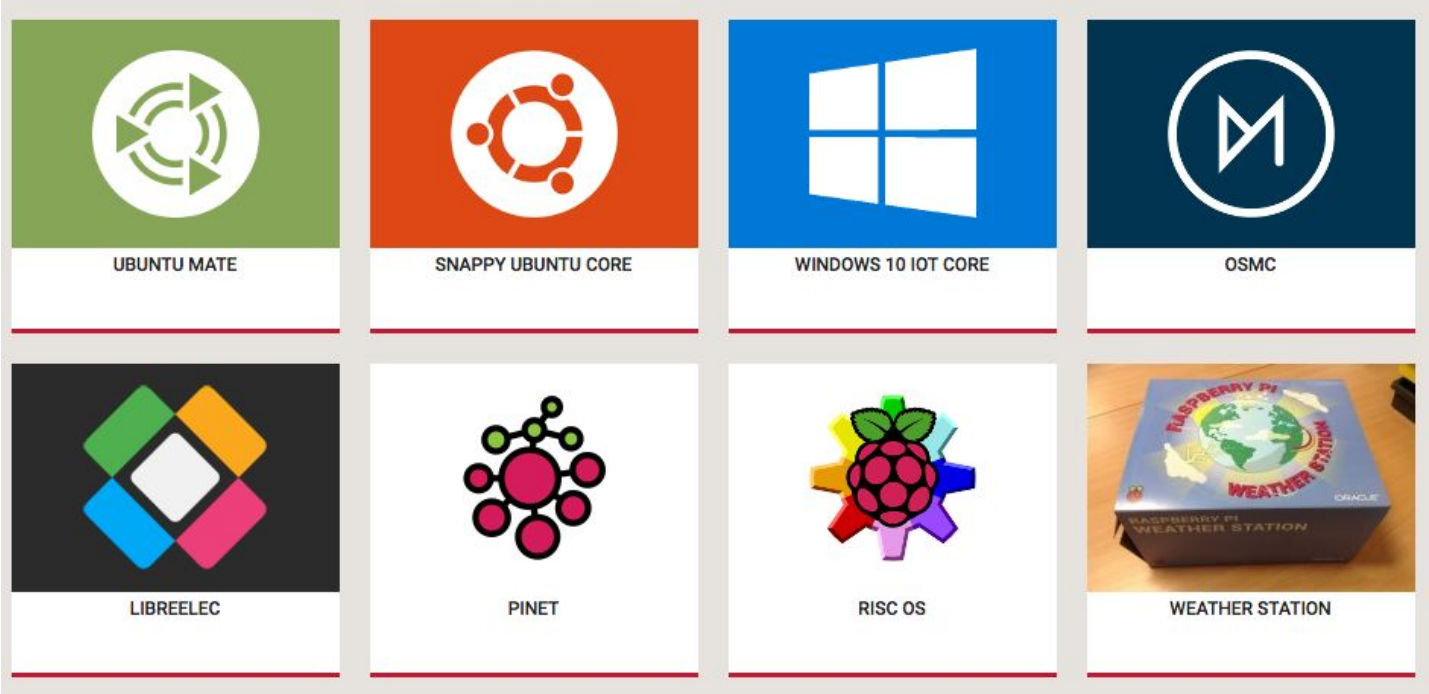
What's on the Raspberry Pi?

We'll be working on the Raspbian operating system. It was installed with NOOBS, an OS installation manager.

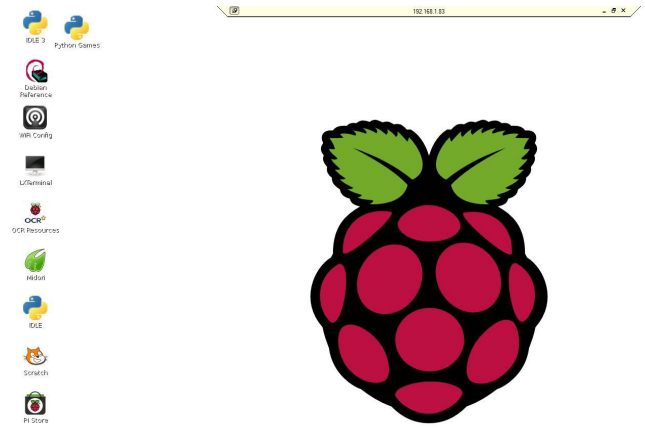


What's on the Raspberry Pi?

We're using Raspbian (Debian-based Linux). You can also install other OSes on RPi including Windows 10 IoT Core and RISC OS. The others are Linux-based.

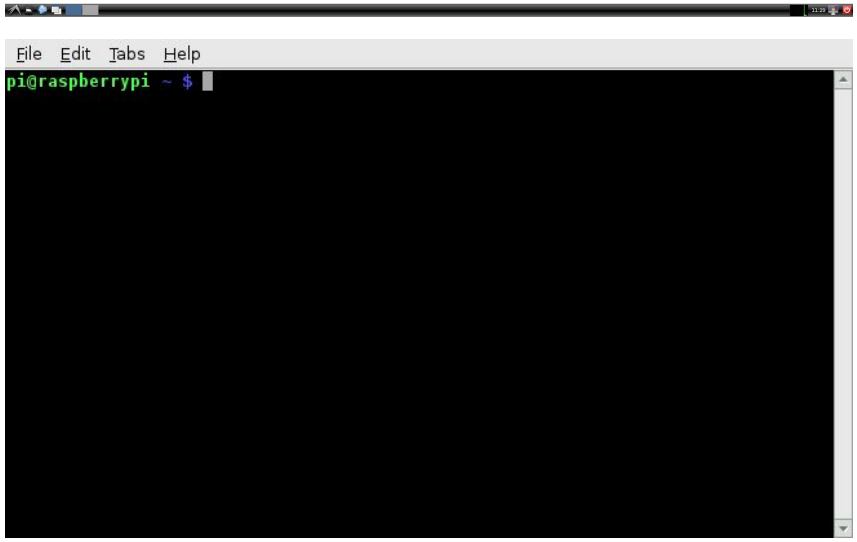


GUI vs Command Line



Benefits of GUI:

- Lower learning curve
- Doesn't require memorisation of many commands

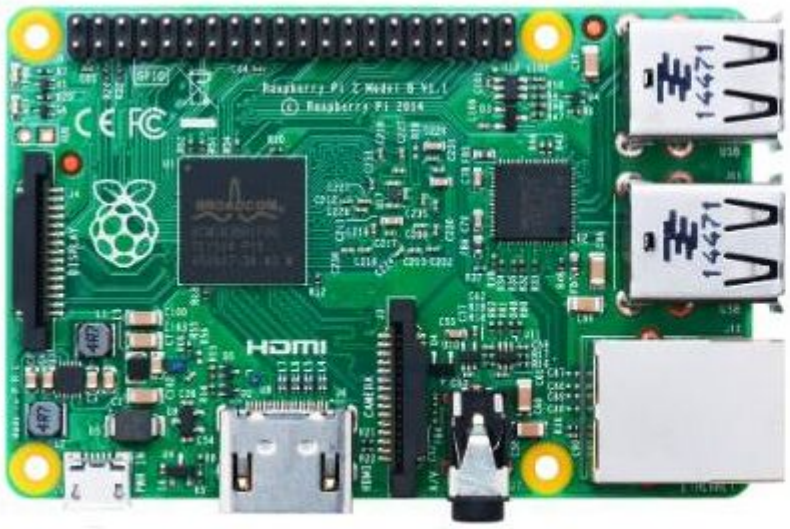


Benefits of Text-Based Interface:

- All commands available at your fingertips (there's only so much you can show on a GUI)
- Need to use it for some functions

Pi 3 vs Pi 2

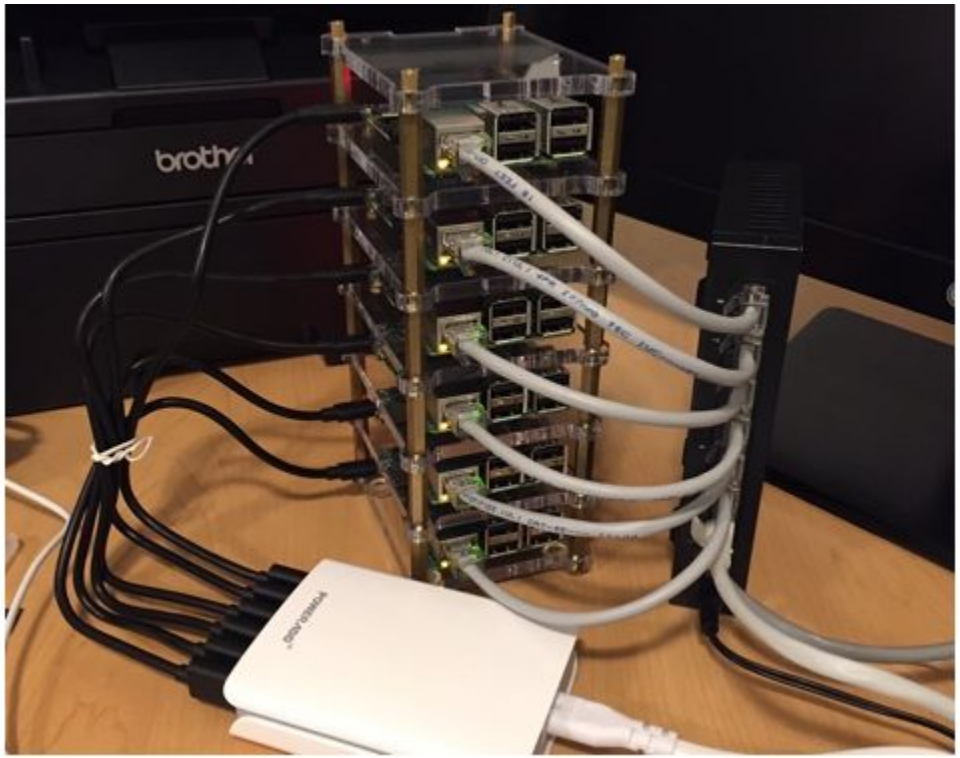
Visually Pi 3 and Pi 2 looks very similar. Some of you have the Pi 2 and some have the Pi 3. Pi 3 has built-in WiFi and Bluetooth unlike Pi 2.



RASPBERRY PI 2 MODEL B

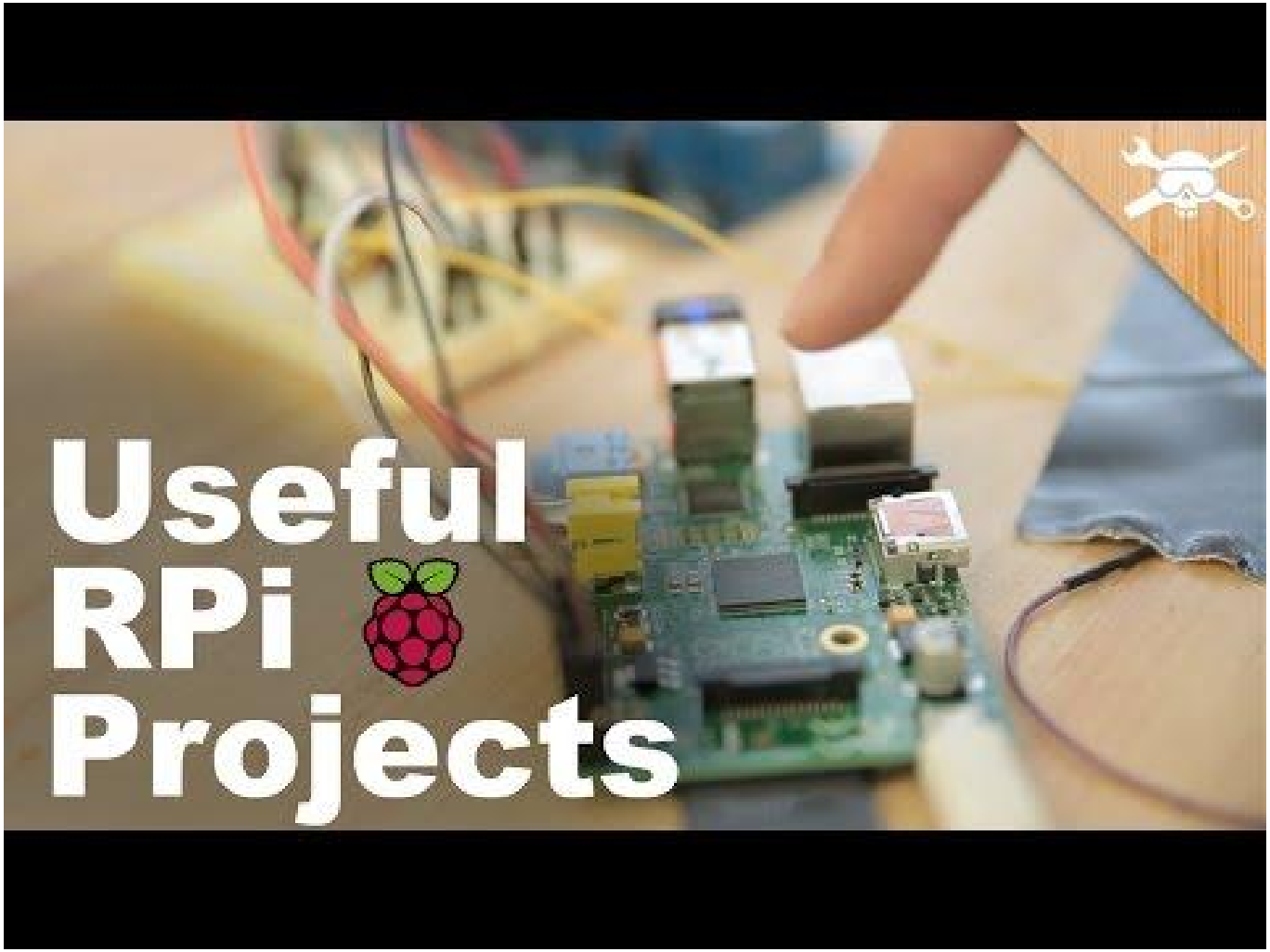
RASPBERRY PI 3 MODEL B

Example RPi Projects

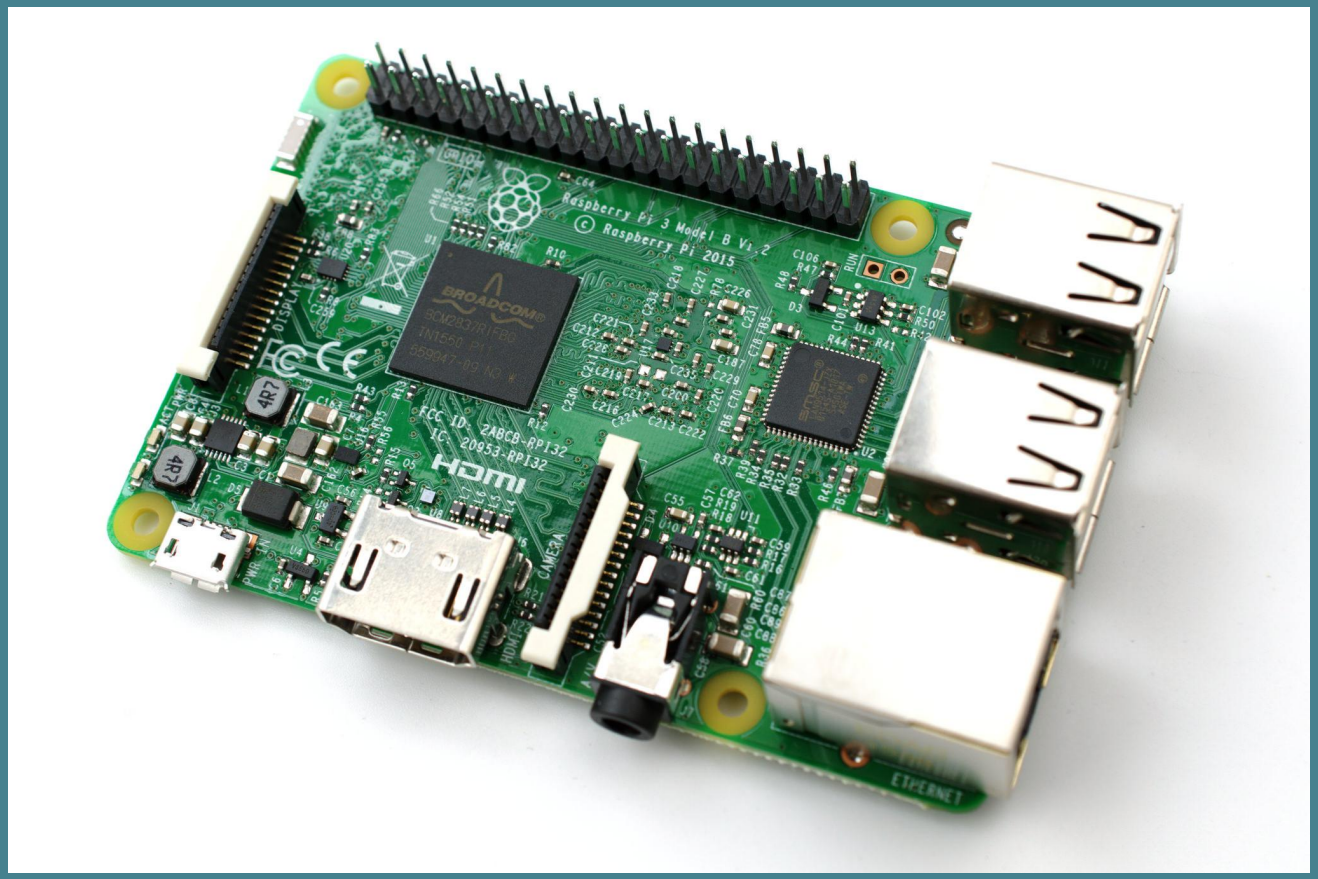


http://www.pcworld.com/article/2895874/10-insanely-innovative-incredibly-cool-raspberry-pi-projects.html

Example RPi Projects

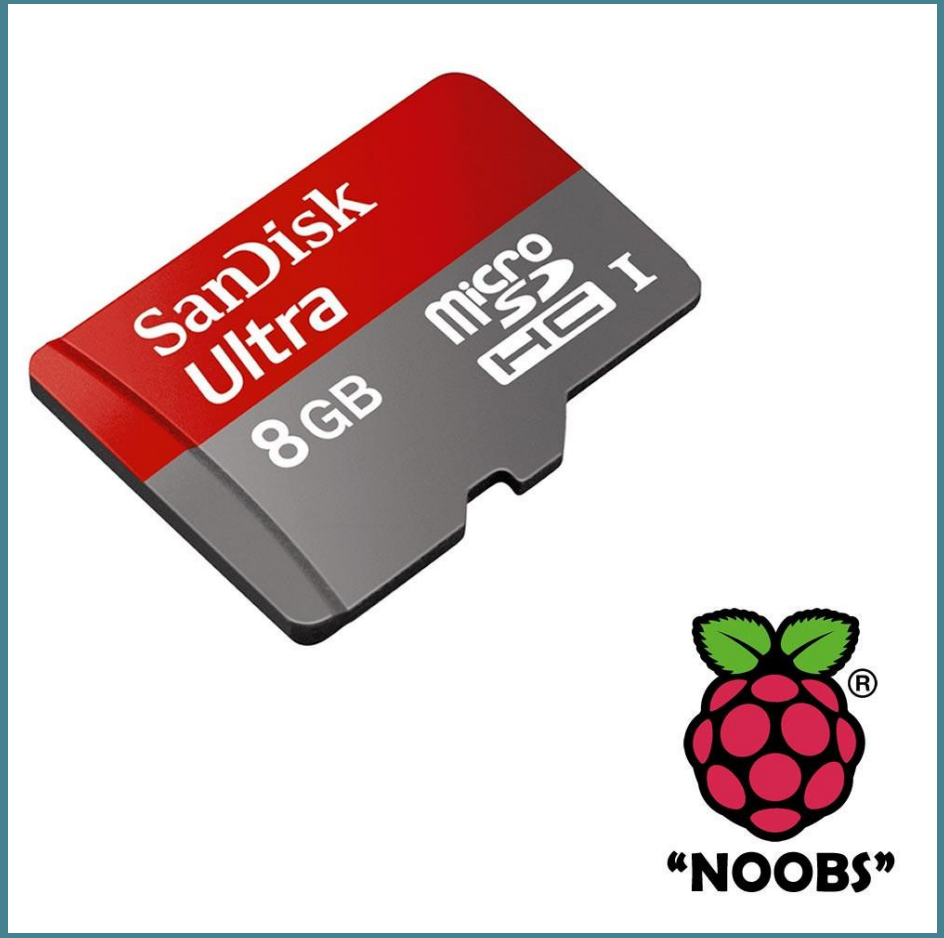


What Will We Be Using?



Raspberry Pi

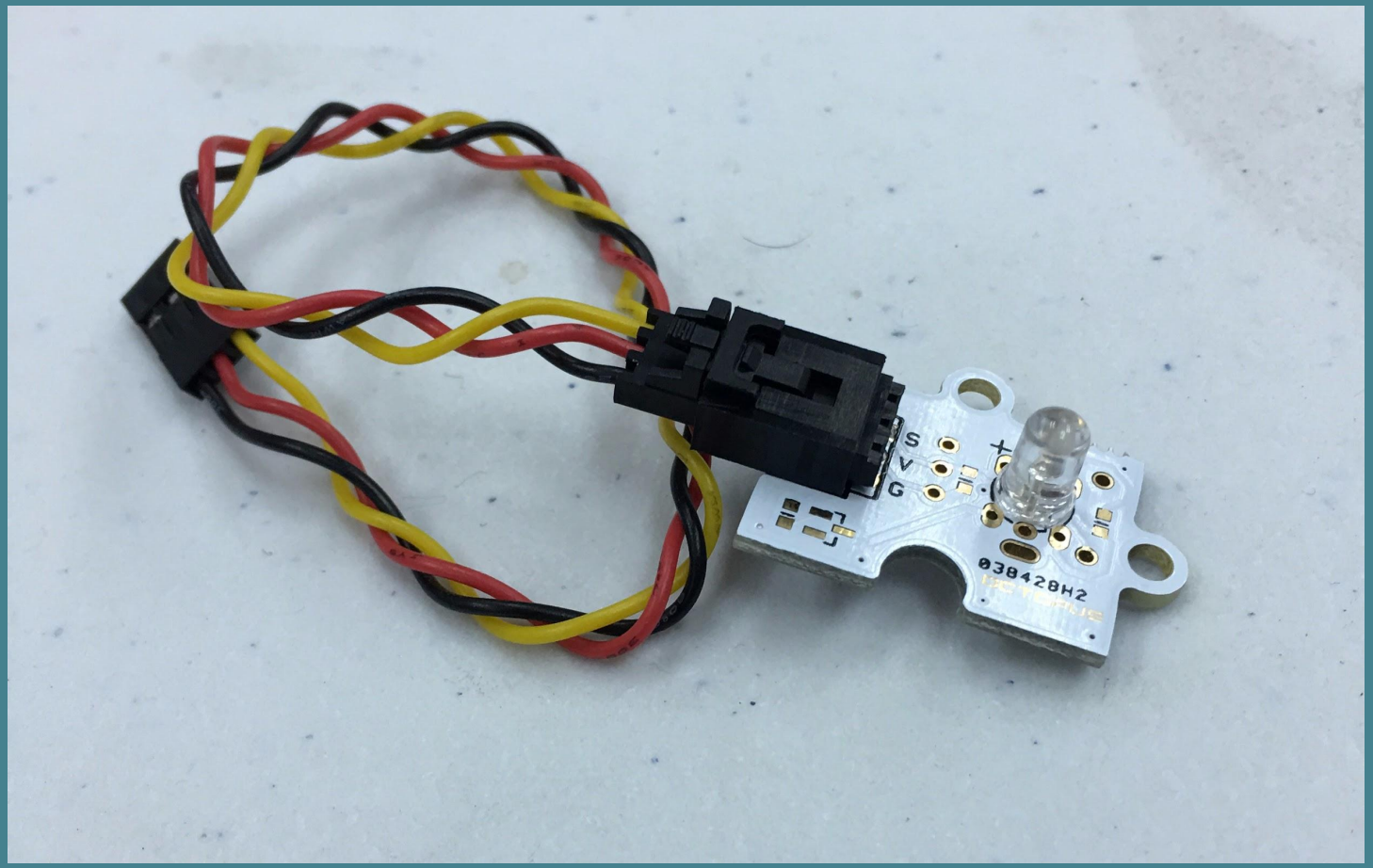
Power Supply



MicroSD card with NOOBS

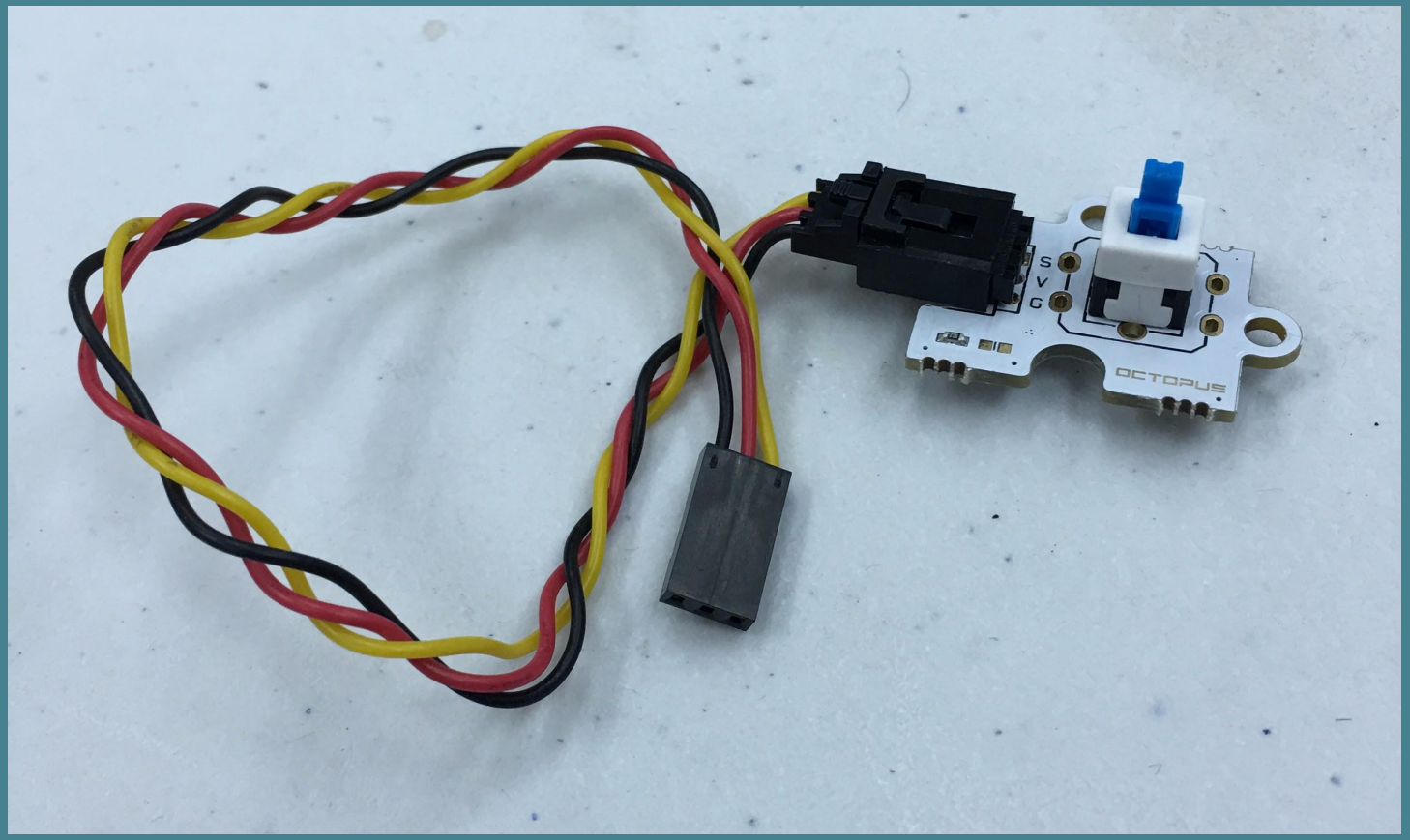
GPIO Shield

(14 Digital Ports, 3 Analog Ports)

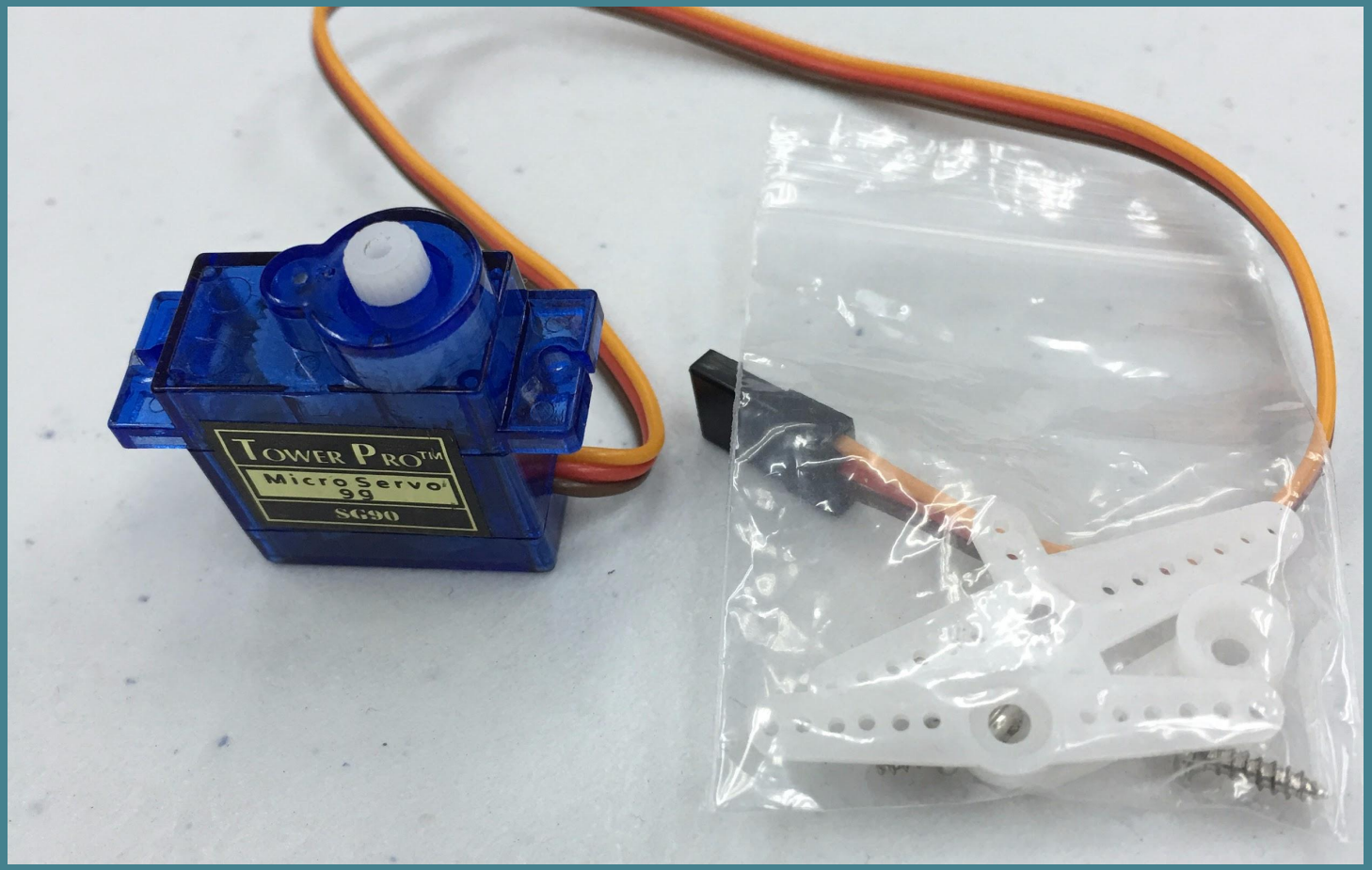


(Hopefully) **3 LEDs**

Active
Buzzer
+
Passive
Buzzer
+
wires

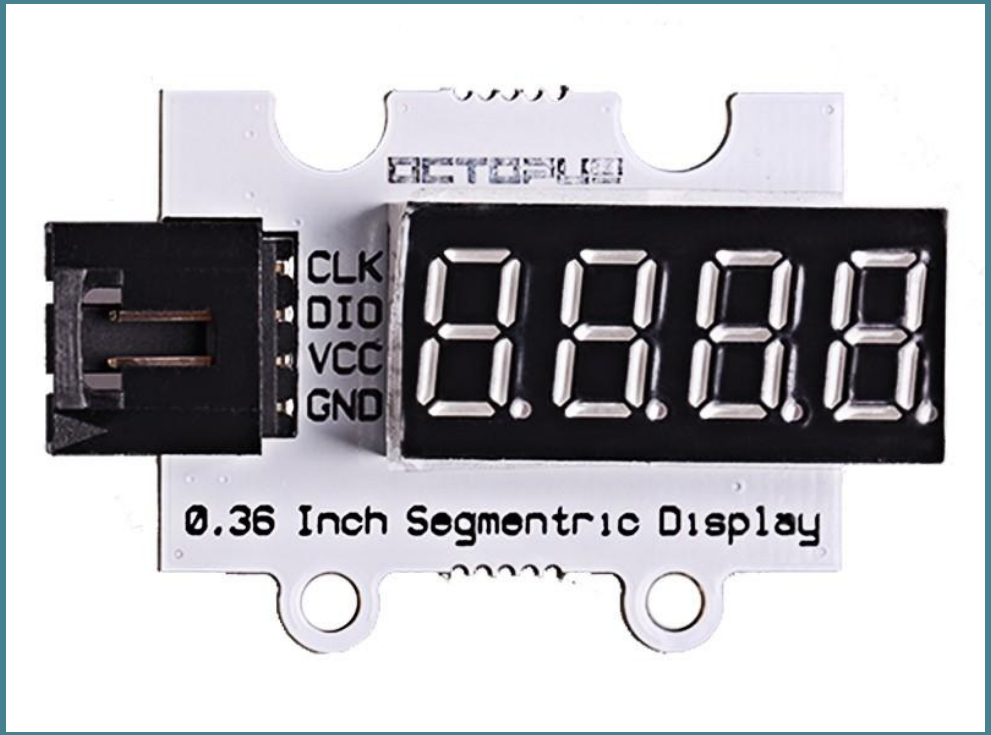


E-Lock Button



Servo Motor

Segment LED



Reed Switch

Booting your Raspberry Pi

- Working in pairs, I'd recommend:
  - One computer for Internet access to do research
  - One computer to connect to the Raspberry Pi

- Insert SD card into RPi
- Connect Monitor (HDMI end to RPi; DVI end to monitor)
- Connect Mouse and Keyboard (USB)
- Connect Ethernet (if possible)
- Connect power (micro-USB port) and it should power up.

Booting your Raspberry Pi

Old versions of the RPi might require you to log in:

Default username: pi

Default password: raspberry

(your password won't show as you type, don't worry, it's still taking your keystrokes)

Raspberry Pi as an IoT Device?

It can be!

If we connect sensors to it, it can function as an IoT device that collects data.

We also have the option of using it as a computer, but connecting a monitor and peripherals (mouse, keyboard, etc.)

Before we get there, we need to learn a bit more python first...

Screen Resolution Problems?

Any trouble getting the correct screen resolution?

1. Open Terminal
2. Type ‘sudo raspi-config’
3. Navigate to Advanced Options > Resolution
4. Click Finish
5. Say ‘Yes’ to reboot your RPi for it to take effect.

(You can also go into IDLE config and edit the text size in IDLE)

Intro to DataFrames

How to Learn?

Google!

Tons of useful resources online. If faced with something you don’t know how to do, chances are high that someone else has already faced this problem and posted about it on the internet!

(StackOverflow is your friend!)

DataFrames

numpy - An extension to the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays

pandas - An open source library providing high-performance, easy-to-use data structures and data analysis tools for Python

DataFrames

Why use DataFrames?

- Arrays require homogeneous data, while a pandas dataframe can store different data types
- Pandas has built in functionality to communicate with a lot of common data-processing apps
- Easy to pull data from a database directly into a dataframe

(Pandas has a **lot** of functionality, we're only going to scratch the surface here!)

But most importantly...

DataFrames

DataFrames are essential when dealing with huge datasets

(There are other non-python options too, but arrays aren't one of them ^.^)

If we try to use arrays with a huge dataset, it will take hours for our computers to crunch the operations!

DataFrames

Your RPis should have numpy and pandas preinstalled (you can tell by trying to import them).

If you do end up needing to install them, you can just use pip.

For example:

```
pip3.4 install pandas
# 3.4 indicates the version of python for
  which you want to install the library
```

DataFrames

Once you have them installed, create a new python file and import pandas and numpy:

```
import pandas as pd
import numpy as np
```

(you can import them as whatever abbreviation you like, but these slides will use ‘pd’ and ‘np’)

How to Create DataFrames

DataFrames are 2-D labelled arrays that can hold any data type (`dtype`). To create them, use pd.DataFrame()

```
df = pd.DataFrame([[1,2],[3,4]])
```

How to Create DataFrames

What if I had no clue how to use pd.DataFrame()?

pandas.DataFrame

class pandas.**DataFrame**(*data=None*, *index=None*, *columns=None*, *dtype=None*, *copy=False*) [source]

Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure

Parameters:

data *: numpy ndarray (structured or homogeneous), dict, or DataFrame*
Dict can contain Series, arrays, constants, or list-like objects

index *: Index or array-like*
Index to use for resulting frame. Will default to np.arange(n) if no indexing information part of input data and no index provided

columns *: Index or array-like*
Column labels to use for resulting frame. Will default to np.arange(n) if no column labels are provided

dtype *: dtype, default None*
Data type to force, otherwise infer

copy *: boolean, default False*
Copy data from inputs. Only affects DataFrame / 2d ndarray input

How to Create DataFrames

If I already have a numpy ndarray, I can pass that to pd.DataFrame too

```
a = np.array([[1,2],[3,4]])
df = pd.DataFrame(a)
```

How to Create DataFrames

We can also optionally label the columns and rows (rows are called index in dataframes):

```
rows = [1,2,3]
cols = list('ABCD')
df = pd.DataFrame([[1,2,3,4],[5,6,7,8],[9,10,11,12]],
                  index=rows, columns=cols)
```

How to Create DataFrames

We can also randomly create an array of data using numpy functions:

```
rows = [1,2,3,4,5,6]
cols = list('ABCD')
df = pd.DataFrame(np.random.randn(6,4), index=rows, columns=cols)
```

In [2]:
```
r = [1,2,3,4,5,6]
df = pd.DataFrame(np.random.randn(6,4), index=r, columns=list('ABCD'))
df
```

Out[2]:

	A	B	C	D
1	-0.090426	1.665410	-0.011287	-1.543680
2	-0.006704	1.213379	-0.402250	-0.462344
3	-0.695395	1.418628	-0.450344	0.464636
4	-0.885897	0.475627	1.072201	1.416538
5	0.352560	-0.645058	1.306389	-0.200803
6	0.630580	0.101905	3.472211	-0.753915

How to Create DataFrames

To check the data type of each column, use df.dtypes!

In [3]:
```
df2 = pd.DataFrame({ 'A' : 1.,
        'B' : pd.Timestamp('20130102'),
        'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
        'D' : np.array([3] * 4,dtype='int32'),
        'E' : pd.Categorical(["test","train","test","train"]),
        'F' : 'foo' })
df2
```

Out[3]:

	A	B	C	D	E	F
0	1	2013-01-02	1	3	test	foo
1	1	2013-01-02	1	3	train	foo
2	1	2013-01-02	1	3	test	foo
3	1	2013-01-02	1	3	train	foo

In [4]:
```
df2.dtypes
```

Out[4]:
```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

Data Types

Series & DataFrames can hold any data type (dtype)

dtypes in pandas include float64, int64, float32, int32, object, category, datetime64[ns] and so on.

Sidenote: Series

A Series is a 1-D labelled array.

s = pd.Series([1,2,3,4,5,6])

If we select a single column from a dataframe it will be a series, so we won't be able to use dataframe methods on it.

DataFrames - Displaying

`.head()` and `.tail()` display the first and last rows of a Series or DataFrame. If no argument is presented, it displays 5 rows.

```
In [3]: df.head()
```

Out[3]:

	A	B	C	D	E	F
1	1	35	4	31	one	NaN
2	53	34	2	9	one	NaN
3	2	21	36	33	two	7
4	7	10	43	5	three	NaN
5	42	4	1	33	two	8

```
In [4]: df.tail(3)
```

Out[4]:

	A	B	C	D	E	F
5	42	4	1	33	two	8
6	23	28	7	3	three	5
7	24	7	18	21	one	NaN

DataFrames – Displaying

.index, .columns and .values display row indexes, column indexes and values.

```
In [5]: df.index
Out[5]: Int64Index([1, 2, 3, 4, 5, 6, 7], dtype='int64')

In [6]: df.columns
Out[6]: Index(['A', 'B', 'C', 'D', 'E', 'F'], dtype='object')

In [7]: df.values
Out[7]: array([[1, 35, 4, 31, 'one', nan],
               [53, 34, 2, 9, 'one', nan],
               [2, 21, 36, 33, 'two', 7.0],
               [7, 10, 43, 5, 'three', nan],
               [42, 4, 1, 33, 'two', 8.0],
               [23, 28, 7, 3, 'three', 5.0],
               [24, 7, 18, 21, 'one', nan]], dtype=object)
```

DataFrames – Sorting

We can also sort data before displaying it.

```
In [9]: df.sort_index(axis=1,ascending=False)
```

Out[9]:

	F	E	D	C	B	A
1	NaN	one	31	4	35	1
2	NaN	one	9	2	34	53
3	7	two	33	36	21	2
4	NaN	three	5	43	10	7
5	8	two	33	1	4	42
6	5	three	3	7	28	23
7	NaN	one	21	18	7	24

```
In [10]: df.sort_values(by='B')
```

Out[10]:

	A	B	C	D	E	F
5	42	4	1	33	two	8
7	24	7	18	21	one	NaN
4	7	10	43	5	three	NaN
3	2	21	36	33	two	7
6	23	28	7	3	three	5
2	53	34	2	9	one	NaN
1	1	35	4	31	one	NaN

DataFrames – Referencing

`.ix[]` is the most common and most useful method: it's smart enough to recognise if you're using integers or indices.

Ex:

```
df.ix[3,'B']
```

will return the value in row 3 and column 'B'

DataFrames – Referencing

`.loc[]` and `.iloc[]` can also do the job:

- **`.loc[]` uses index names**
- **`.iloc[]` uses integer positions**

```
In [7]: df.loc[1]
```

```
Out[7]: A      1
        B     35
        C      4
        D     31
        E    one
        F    NaN
        Name: 1, dtype: object
```

```
In [8]: df.iloc[:,1:5]
```

Out[8]:

	B	C	D	E
1	35	4	31	one
2	34	2	9	one
3	21	36	33	two
4	10	43	5	three
5	4	1	33	two
6	28	7	3	three
7	7	18	21	one

DataFrames – Referencing

By default, `df[ ]` uses integer positions for rows and index labels for columns.

```
In [9]: df[1:4]
```

Out[9]:

	A	B	C	D	E	F
2	53	34	2	9	one	NaN
3	2	21	36	33	two	7
4	7	10	43	5	three	NaN

```
In [10]: df.loc[1:4]
```

Out[10]:

	A	B	C	D	E	F
1	1	35	4	31	one	NaN
2	53	34	2	9	one	NaN
3	2	21	36	33	two	7
4	7	10	43	5	three	NaN

DataFrames – Referencing

`.loc[]` and `.iloc[]` referencing the same section:

```
In [11]: df.loc[1:3,['A','E']]
```

Out[11]:

	A	E
1	1	one
2	53	one
3	2	two

```
In [12]: df.iloc[0:3,[0,4]]
```

Out[12]:

	A	E
1	1	one
2	53	one
3	2	two

DataFrames - Referencing

`.at[]` and `.iat[]` reference a single scalar value. You can also use `.loc[]` and `.iloc[]`, but it is slower.

```
In [13]: df.at[2,'A']
Out[13]: 53

In [14]: df.iat[1,0]
Out[14]: 53
```

DataFrames – Referencing

We can also use boolean expressions to find data.

```
In [15]: df[df.D >= 30]
```

Out[15]:

	A	B	C	D	E	F
1	1	35	4	31	one	NaN
3	2	21	36	33	two	7
5	42	4	1	33	two	8

```
In [16]: df[df < 30]
```

Out[16]:

	A	B	C	D	E	F
1	1	NaN	4	NaN	one	NaN
2	NaN	NaN	2	9	one	NaN
3	2	21	NaN	NaN	two	7
4	7	10	NaN	5	three	NaN
5	NaN	4	1	NaN	two	8
6	23	28	7	3	three	5
7	24	7	18	21	one	NaN

This is a really, really useful way to slice dataframes!

DataFrames – Editing

Editing data is as simple as adding '=' to a reference.

Original:

	A	B	C	D	E	F
1	1	35	4	31	one	NaN
2	53	34	2	9	one	NaN
3	2	21	36	33	two	7
4	7	10	43	5	three	NaN
5	42	4	1	33	two	8
6	23	28	7	3	three	5
7	24	7	18	21	one	NaN

In [3]:

```
df.index = ['a','b','c','d','e','f','g']

df.iat[4,4] = 'OVER 9000'

df.loc[:,'B'] = np.array(['hurhurhur']*len(df))

s = pd.Series([np.nan,np.nan,np.nan,np.nan,np.nan,np.nan])
s.index = ['A','B','C','D','E','F']
df.iloc[1,:] = s

df[df['C'].isin([43])] = df * 2

df
```

Out[3]:

	A	B	C	D	E	F
a	1	hurhurhur	4	31	one	NaN
b	NaN	NaN	NaN	NaN	NaN	NaN
c	2	hurhurhur	36	33	two	7
d	14	hurhurhurhurhurhur	86	10	threethree	NaN
e	42	hurhurhur	1	33	OVER 9000	8
f	23	hurhurhur	7	3	three	5
g	24	hurhurhur	18	21	one	NaN

DataFrames – Editing

Adding rows to DataFrames is very different from adding columns. (columns are easy, rows are hard)

In [4]:
```
df['G'] = [4,6,4,6,4,6,4]
df2 = pd.Series([99,99,99,99,99,99,99],index=['A','B','C','D','E','F','G'],name=['h'])
df.append(df2)
```

Out[4]:

	A	B	C	D	E	F	G
a	1	hurhurhur	4	31	one	NaN	4
b	NaN	NaN	NaN	NaN	NaN	NaN	6
c	2	hurhurhur	36	33	two	7	4
d	14	hurhurhurhurhurhur	86	10	threethree	NaN	6
e	42	hurhurhur	1	33	OVER 9000	8	4
f	23	hurhurhur	7	3	three	5	6
g	24	hurhurhur	18	21	one	NaN	4
h	99	99	99	99	99	99	99

DataFrames - Editing

`.concat()` can also be used to merge larger DataFrames.

In [4]:
```
df3 = pd.DataFrame([4,6,4,6,4,6,4],index=['a','b','c','d','e','f','g'],columns=['G'])
df3
dfs = [df,df3]
pd.concat(dfs,axis=1)
```

Out[4]:

	A	B	C	D	E	F	G
a	1	hurhurhur	4	31	one	NaN	4
b	NaN	NaN	NaN	NaN	NaN	NaN	6
c	2	hurhurhur	36	33	two	7	4
d	14	hurhurhurhurhurhur	86	10	threethree	NaN	6
e	42	hurhurhur	1	33	OVER 9000	8	4
f	23	hurhurhur	7	3	three	5	6
g	24	hurhurhur	18	21	one	NaN	4

In [5]:
```
df2 = pd.DataFrame([[99,99,99,99,99,99,99]],index=['h'],columns=['A','B','C','D','E','F','G'])
dfs = [df,df2]
pd.concat(dfs)
```

Out[5]:

	A	B	C	D	E	F	G
a	1	hurhurhur	4	31	one	NaN	NaN
b	NaN	NaN	NaN	NaN	NaN	NaN	NaN
c	2	hurhurhur	36	33	two	7	NaN
d	14	hurhurhurhurhurhur	86	10	threethree	NaN	NaN
e	42	hurhurhur	1	33	OVER 9000	8	NaN
f	23	hurhurhur	7	3	three	5	NaN
g	24	hurhurhur	18	21	one	NaN	NaN
h	99	99	99	99	99	99	99

DataFrames – Editing

`.dropna()` removes rows with NaN values.

```
In [4]: df.dropna(how='any')
```

Out[4]:

	A	B	C	D	E	F
c	2	hurhurhur	36	33	two	7
e	42	hurhurhur	1	33	OVER 9000	8
f	23	hurhurhur	7	3	three	5

```
In [5]: df.dropna(how='all')
```

Out[5]:

	A	B	C	D	E	F
a	1	hurhurhur	4	31	one	NaN
c	2	hurhurhur	36	33	two	7
d	14	hurhurhurhurhurhur	86	10	threethree	NaN
e	42	hurhurhur	1	33	OVER 9000	8
f	23	hurhurhur	7	3	three	5
g	24	hurhurhur	18	21	one	NaN

There are tons of different methods to manipulate DataFrames.

We won't cover them all in class, but you can explore more on your own.

DataFrame & Plotting Exercises

As we go through, submit your answers in #discussion

(gogo class participation!)

(be sure to try them yourself before peeking at submitted answers)

Exercise 1

Create the following DataFrame (hint: use np.nan for NaNs!)

	A	B	C	D	E	F
1	1	35	4	31	one	NaN
2	53	34	2	9	one	NaN
3	2	21	36	33	two	7
4	7	10	43	5	three	NaN
5	42	4	1	33	two	8
6	23	28	7	3	three	5
7	24	7	18	21	one	NaN

Note: NaN stands for 'not a number' and is used to represent an undefined or unrepresentable value

Exercise 2

Try it out:

Using the previous DataFrame, how can we access the cell with the value ‘42’ ?

Exercise 2

Try it out:

Using the previous DataFrame, how can we access the cell with the value ‘42’ ?

Answer:

```
Many different answers, here’s two using df.ix:
      print(df.ix[4,'A'])
      print(df.ix[4,0])
```

Exercise 3

Try it out:

Using the previous DataFrame, how can we access the cell with the value ‘28’ ?

Exercise 3

Try it out:

Using the previous DataFrame, how can we access the cell with the value ‘42’ ?

Answer:

```
Many different answers, here’s two using df.ix:
      print(df.ix[5,'B'])
      print(df.ix[5,1])
```

Exercise 4

Try it out:

Using the previous DataFrame, how can we access column ‘C’ ?

Exercise 4

Try it out:

Using the previous DataFrame, how can we access column ‘C’ ?

Answer:

```
df.C
```

Exercise 5

Try it out:

Using the previous DataFrame, how can we access row 5?

Exercise 5

Try it out:

Using the previous DataFrame, how can we access row 5?

Answer:

```
df[df.index=5]
```

Other DataFrame functions

Mathematical operations:

`df.add()`

`df.sub()`

`df.multiply()`

`df.divide()`

`df.apply()` - this allows us to use numpy functions

Note that these can be applied to slices as well.

Other DataFrame functions

Importing CSV (Comma Separated Values) files:

Use `pandas.read_csv()`

```
e.g. data = pd.read_csv('filename.csv')
```

CSV files are frequently used to store data!

Sidenote: CSV files

Moar DataFrame Exercises

Exercise 1:

Read the weather_year.csv file (sent via Slack) into a dataframe and print it in python to get an idea of what it looks like. What is the length of the DataFrame? How many data observations are there?

Moar DataFrame Exercises

Exercise 2:

Print out the second to last date in the data set.

```
*Bonus points for doing it multiple ways!
```

Moar DataFrame Exercises

Exercise 3:

Rename the columns to the following names:

data
maxtemp
meantemp
mintemp
maxdew
mindew
maxhumidity
meanhumidity
minhumidity
maxpressure
meanpressure
minpressure
maxvisibility
meanvisibility
minvisibility
maxwind
meanwind
minwind
precipitation
cloudcover
events
winddir

Moar DataFrame Exercises

Exercise 4:

Currently, the row indices have been auto-generated by python (integers 0 to 365). Change them to be dates in datetime format.

Hint: The dates in the csv file are currently in string format. To convert them to datetime, use the following:

```
from datetime import datetime
def string_to_date(date_string):
    return datetime.strptime(date_string, "%Y-%m-%d")
```

Hint2: Use the .apply method to convert an entire col at once

Moar DataFrame Exercises

Exercise 5:

Now that we no longer need the date column, drop it from the dataframe.

Hint: use .drop()

Moar DataFrame Exercises

Exercise 6:

Using our new row indices, print out the cloud cover data for the month of May.

Hint: use the datetime(year, month, day) function

Moar DataFrame Exercises

Exercise 7:

Drop all rows with NaN values.

Do you think this is a good idea for this data set?

Moar DataFrame Exercises

Exercise 8:

Currently the temperatures are all in Fahrenheit. Convert them all to Celsius and replace the Fahrenheit values.

Hint: to convert F to C, subtract 32, then multiply by 5/9

Moar DataFrame Exercises

Exercise 9:

A lot of time with data sets is often spent selecting specific rows of interest. Print out all the rows where the temperature never exceeded 0 degrees Celsius.

Moar DataFrame Exercises

Exercise 10:

Let's work with the precipitation column. Currently, it's filled with strings. Most values are numbers in text format, but some values are 'T' which means 'trace of precipitation'.

Convert the column to type float, and replace the 'T' values with small numbers (i.e. .005).

Hint: Use the .apply() method as we did before!

Moar DataFrame Exercises

Exercise 11:

Now that we can work with the Precipitation column, what was the coldest temperature observed when there was no cloud cover and trace amounts of precipitation?