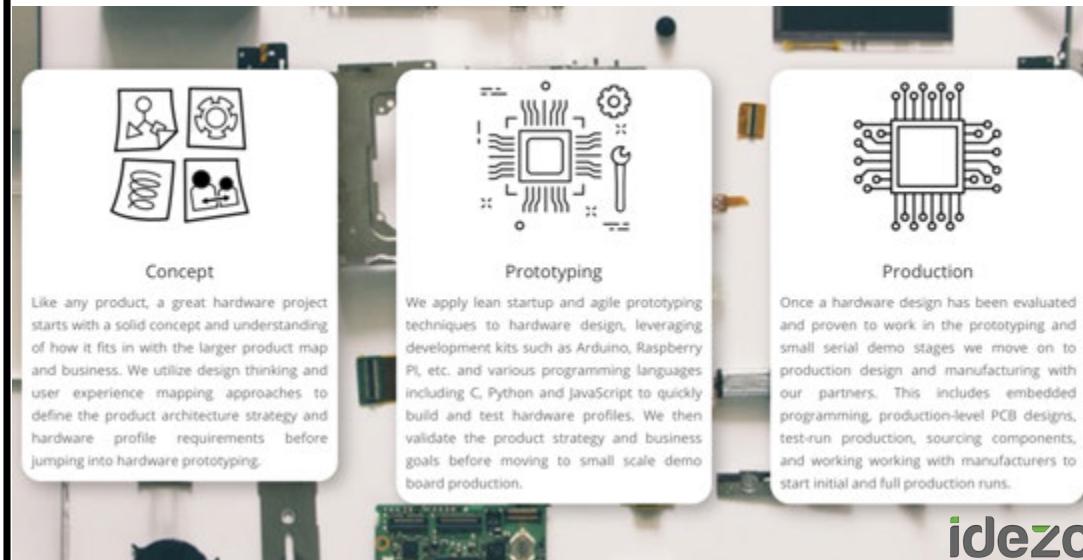


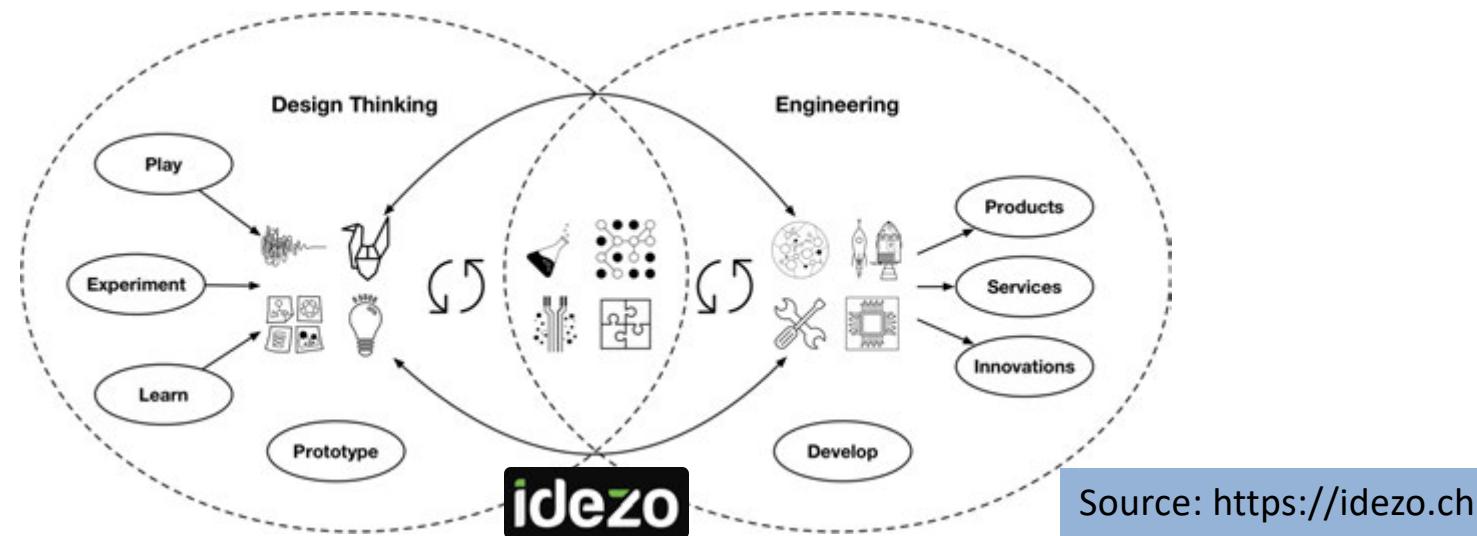
HANDS ON APPROACH TO

Data Science for (the) IoT



Rob van der Willigen

HANDS ON APPROACH TO DATA SCIENCE for (the) IoT



Source: <https://idezo.ch>

This Course material is distributed under the Creative Commons Attribution- NonCommercial-ShareAlike 3.0 license. You are free to copy, distribute, and transmit this work. You are free to add or adapt the work. You must attribute the work to the author(s) listed above.

You may not use this work or derivative works for commercial purposes. If you alter, transform, or build upon this work you may distribute the resulting work only under the same or similar license.

This Data Science Course was developed for keuzevak- program of the [School of Communication, Media and Information Technology \(CMI\)](#) at the Hogeschool Rotterdam (**Rotterdam University of Applied Sciences, RUAS**).

If you find errors or omissions, please contact the author, Rob van der Willigen, at r.f.van.der.willigen@hr.nl. Materials of this course and code examples used will become available at:

<https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT>

Course Setup

Lesson 01: week 02 **Discovering the IoT Data Science Domain**

Lesson 02: week 03 **Defining project requirements**
 + Cost calculation/estimate

Lesson 03: week 04 **Learn to write code**

Lesson 04: week 05 **Data Science: How to start your own IoT Project**

Lesson 05: week 06 **IoT Platforms & MiddleWare**

Lesson 06: week 07 **Core IoT Concepts + setting-up your IoT-device**

Lesson 07: week 08 **Grading + Summary + Q & A**

Week 09 / 10: FEEDBACK + GRADING

DEADLINE OPLEVERING

Om voor beoordeling in aanmerking te komen lever je een ascii text file (.txt) aan met de link (URL) van jouw Git-Hub account met de IoT-repository.

De repository bevat een beschrijving van het IoT-Project waarin bewijzen

---zoals video-materiaal / code / design concept / data organisatie---

zijn opgenomen van op basis van de door jouw bedachte/geteste IoT data-pipeline/Prototype zoals beschreven op <https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT>

Deze text-file moet je uploaden via de LMS **INLEVER_MAP_OP2**
van de CURSUS CMIDAT01k Data Science for IoT (2020-2021).

Deadline is: 9 februari (vanaf 10 februari kun je niet meer inleveren).

Beoordeling vindt plaats conform het beoordelingsmodel:

https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT/blob/master/Docs/BEOORDELINGS_MODEL_DS_for_IoT_V24_npv_2020.pdf

Deze mededeling wordt als E-mail bericht wordt verstuurd (2 februari)
in week 10 OP2 collegejaar 2020-2021

Nota Bene! Stuur een Reply op deze E-Mail
als je voor beoordeling van de cursus: CMIDAT01k Data Science for IoT (2020-2021) in aanmerking wilt komen.

lesson seven



Internet of Things 101: Building IoT
Prototype with Raspberry Pi
Feb 9 and 11, 2016 at Forward 4 Conf

Examples of Git-Hub Repository reporting on --how to create an-- IoT-pipeline scenario + Code

Note evidence of a working prototype is NOT included!

<https://docs.google.com/presentation/d/1kHjN8ClbggBlr44m2JpsRpR6IH-B-Yu7b5bECC-ntK4/edit#slide=id.p4>

<https://github.com/pubnub/workshop-raspberrypi>

<https://github.com/pubnub/workshop-raspberrypi/tree/master/projects-python/range-finder>

<https://github.com/pubnub/workshop-raspberrypi>

Computational Thinking Core-Concepts

Computational thinking is a way of solving problems, designing systems, & understanding human behavior that draws on concepts fundamental to data science.

Viewpoint | Jeannette M. Wing

Computational Thinking

It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.



Computational thinking builds on the power and limits of computing processes, whether they are executed by a human or by a machine. Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the niddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions.

Computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability. Just as the printing press facilitated the spread of the three Rs, what is appropriately incutious about this vision is that computing and computers facilitate the spread of computational thinking.

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

Having to solve a particular problem, we might ask: How difficult is it to solve? and What's the best way to solve it? Computer science rests on solid theoretical underpinnings to answer such questions pre-

cisely. Stating the difficulty of a problem accounts for the underlying power of the machine—the computing device that will run the solution. We must consider the machine's instruction set, its resource constraints, and its operating environment.

In solving a problem efficiently, we might further ask whether an approximate solution is good enough, whether we can use randomization to our advantage, and whether false positives or false negatives are allowed. Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.

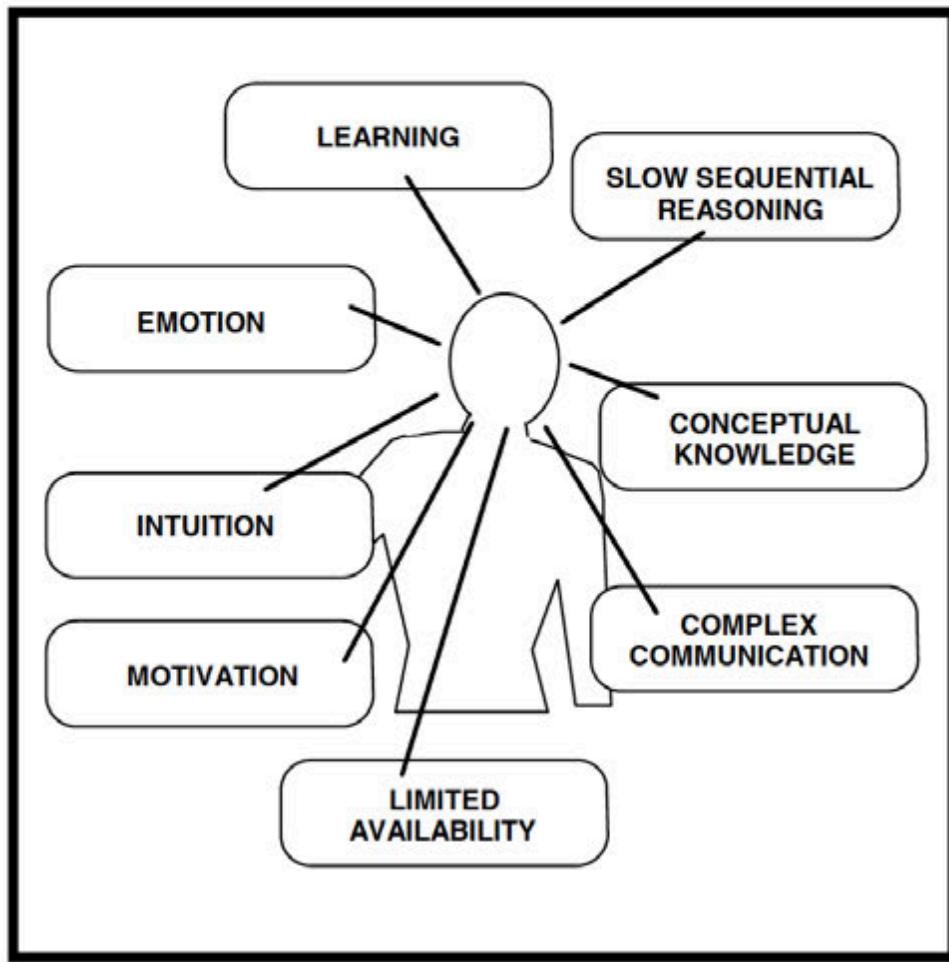
Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code. It is type checking as the generalization of dimensional analysis. It is recognizing both the virtues and the dangers of aliasing, or giving someone or something more than one name. It is recognizing both the cost and power of indirect addressing and procedure call. It is judging a program not just for correctness and efficiency but for aesthetics, and a system's design for simplicity and elegance.

Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system's behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is

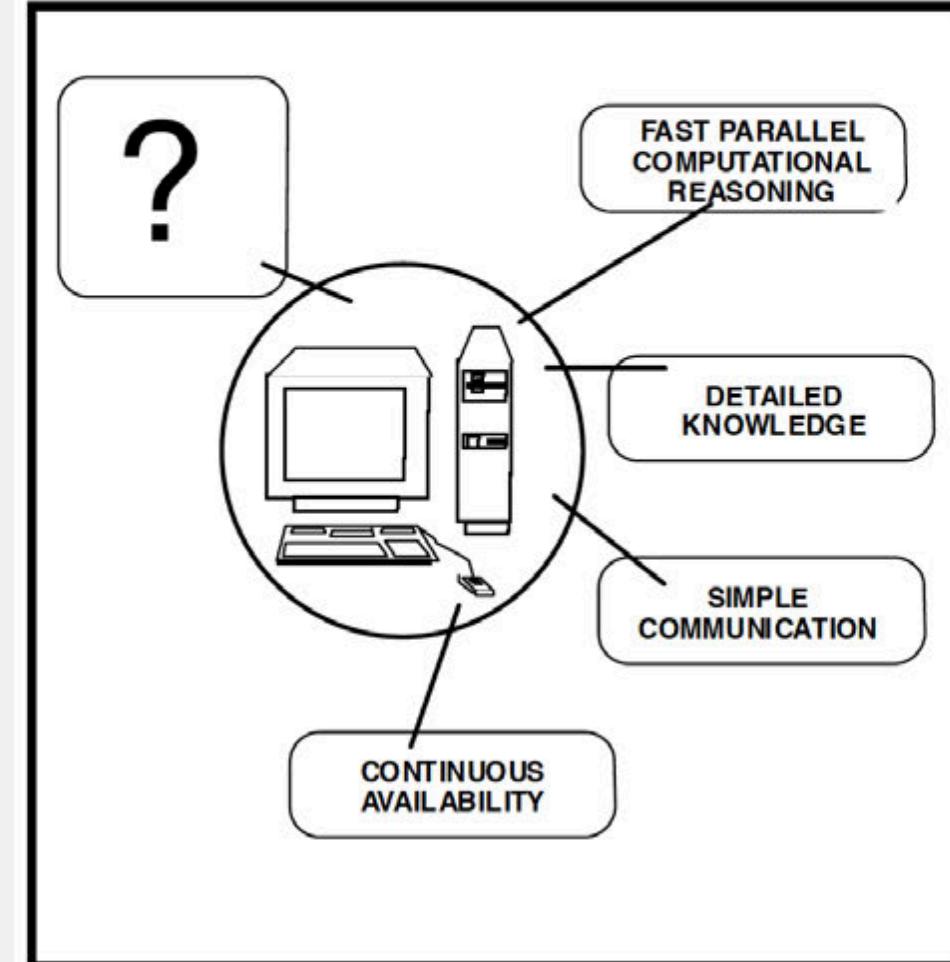
LIBRARY

Human versus Computer

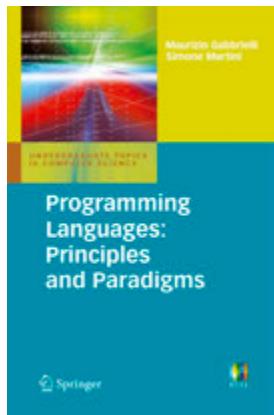
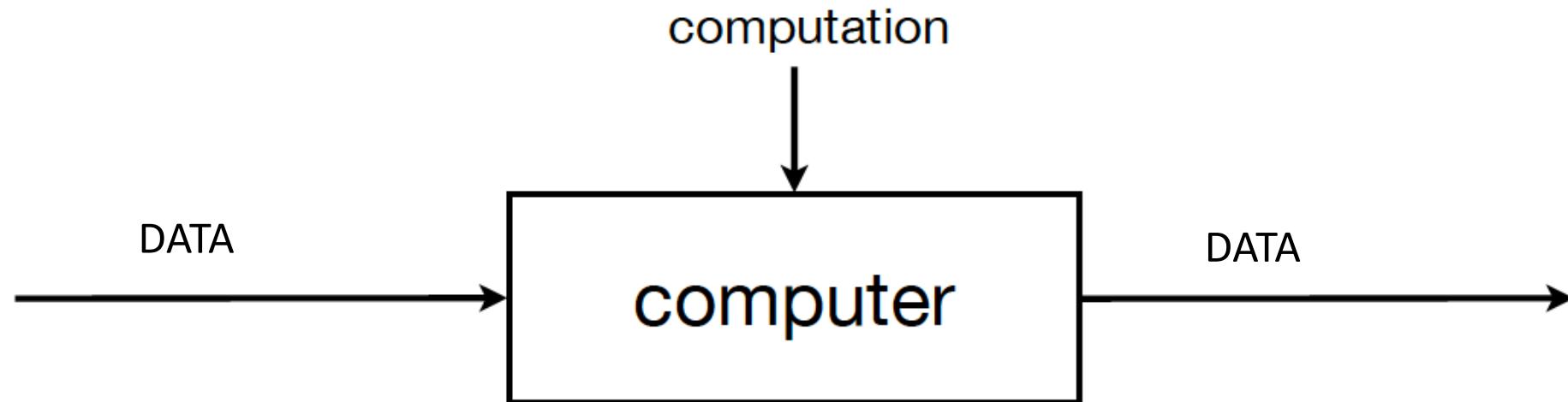
Humans are pattern-driven



Computers are data-driven



What is a Computer?



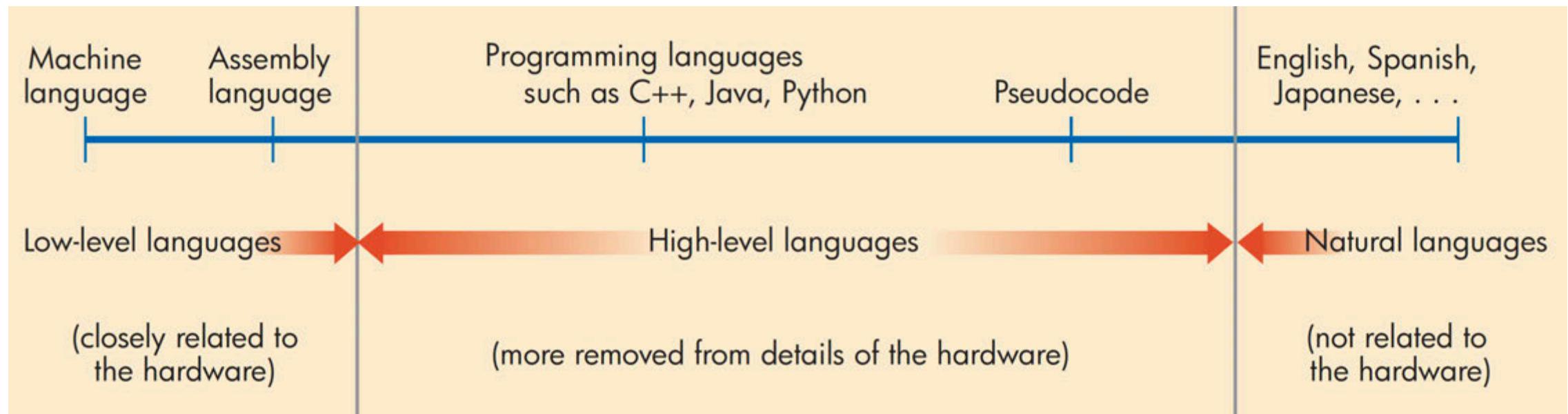
A computer is a **programmable machine** that receives input, stores and manipulates **data**, and provides output in a **useful format**.

Software

Software is a generic term, which is used to describe a group of computer programs & procedures which perform some task on a computer (system).

It is an ordered sequence of instructions given for changing the state of the computer hardware, in a certain predefined manner.

Coding-paradigma's



Command Line Terminal (interface)

What Is the Command Line?

The command-line interface, sometimes referred to as the CLI, is a tool into which you can type text commands to perform specific tasks—in contrast to the mouse's pointing and clicking on menus and buttons.

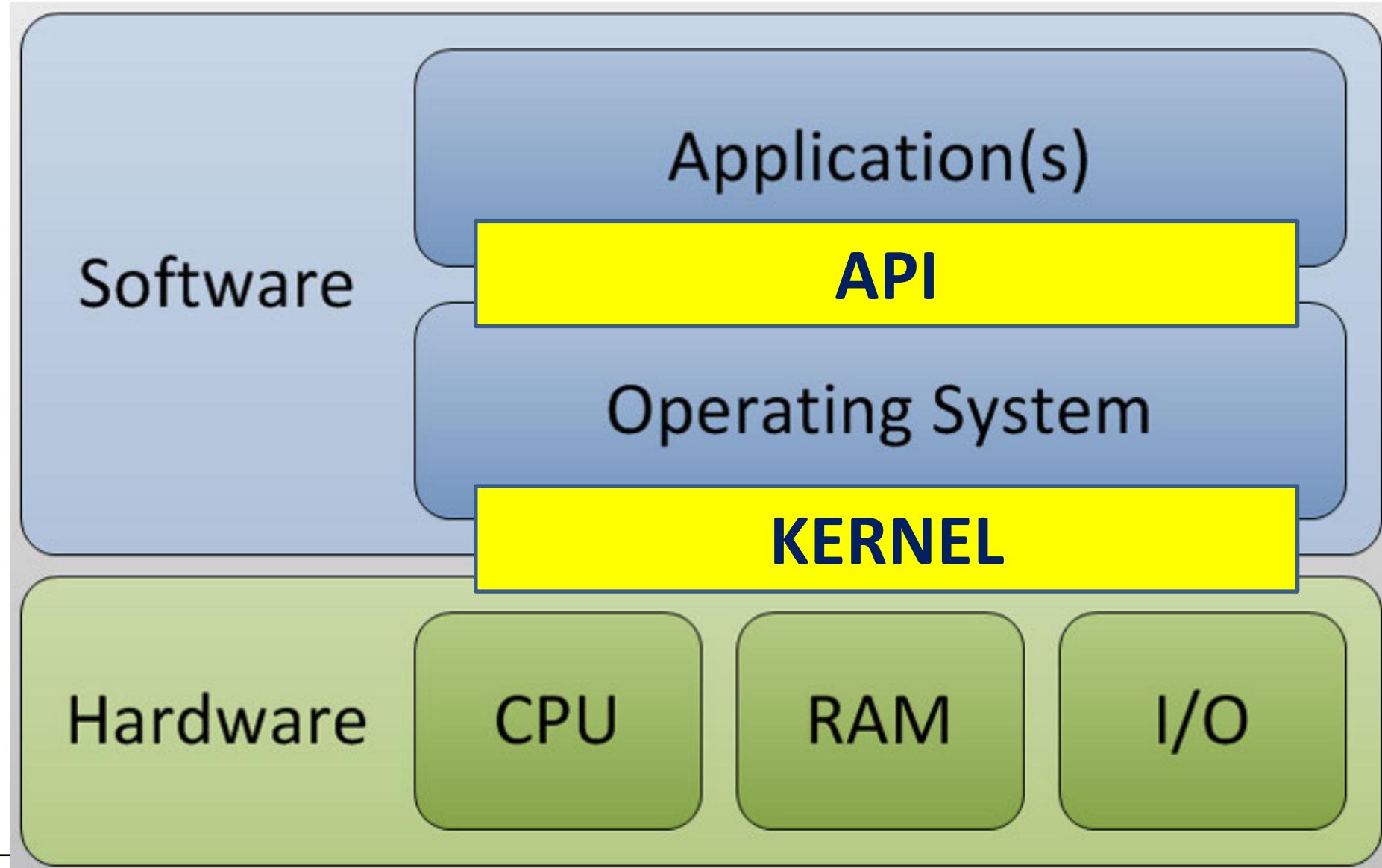
Since you can directly control the computer by typing, many tasks can be performed more quickly, and some tasks can be automated with special commands that loop through and perform the same action on many files—saving you, potentially, loads of time in the process.

Coding

Anyone interested in **developing software**, such as a source-code, game, or IoT pipeline, **must start by learning a programming language & decide which (source-code) Editor/Debugger to use.**

<https://www.computerhope.com/issues/ch000675.htm>

Software vs Hardware



Integrated Development Environment

Text Editor

Syntax Coloring

Autocomplete

Indexer

Doc Viewer

Code Templates



Builder

Compiler

Lexer

Parser

Assembler

Linker



Debugger GUI

Debugger



What is (**source-code**) Program Software: Integrated Development Environment (**IDE**)?

Code editor: This feature is a **text editor** designed for writing and editing source code.

Source code editors are distinguished from text editors because they enhance or simplify the writing and editing of code.

Compiler/Interpreter: This tool **transforms source code** written in a human readable/writable language into a form **executable** by a computer.

Debugger: This tool is used during testing to help **debug application** programs.

Build automation tools: These tools automate common **developer** tasks.

Source-Code Editor

A **source code editor** is a text editor program **designed specifically for editing source code** of computer programs by programmers.

It may be a **Standalone Application** or it may be built into an **Integrated Development Environment (IDE)** or **Web Browser**.

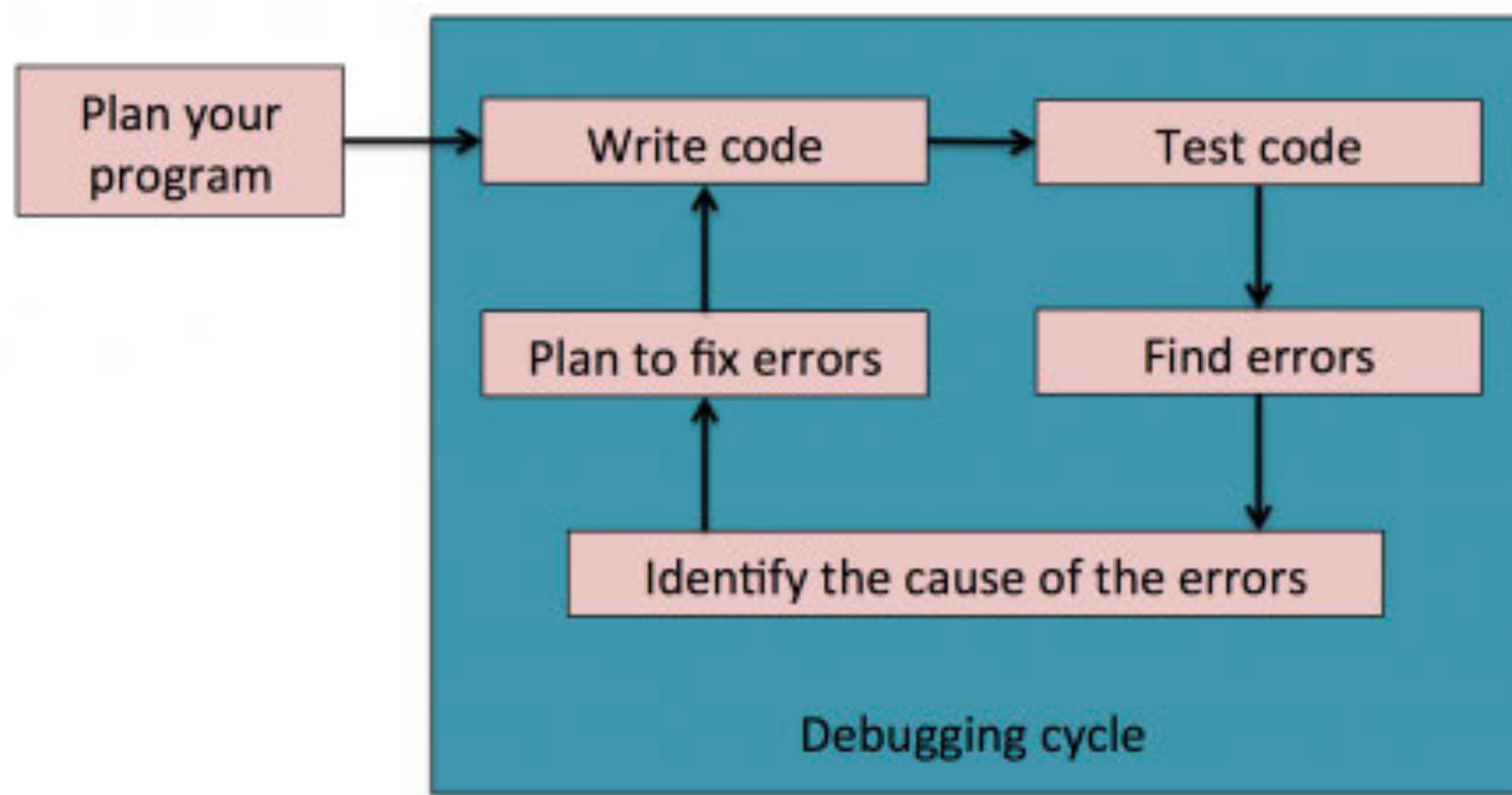
Source code editors are the **most fundamental programming tool**, as the fundamental job of programmers is **to write and edit source code**.

Debugging

Debugging is a process of analyzing a computer program **source-code** and **removing** its **logical** or **syntactical errors**.

Software which assists this process is a **debugger**.

Using a **debugger**, a **software developer** can step through a **program's code** and **analyze** its variable values, searching for **errors**.

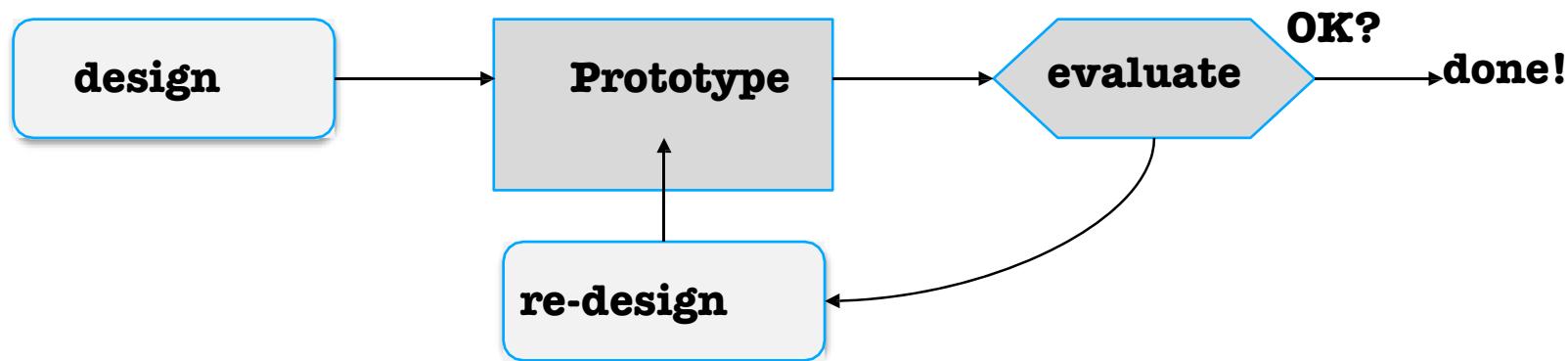
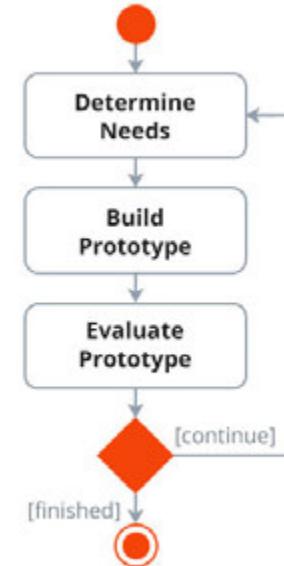


Rapid Prototyping

Core-Concepts

PROTOTYPING

- You never get it right first time
- If at first you don't succeed ...



PITFALLS OF PROTOTYPING

- Moving little by little ... but to where



1. need a good start point
2. need to understand what is wrong

Google
for Entrepreneurs

Rapid Prototyping

Part 1: Paper Prototyping

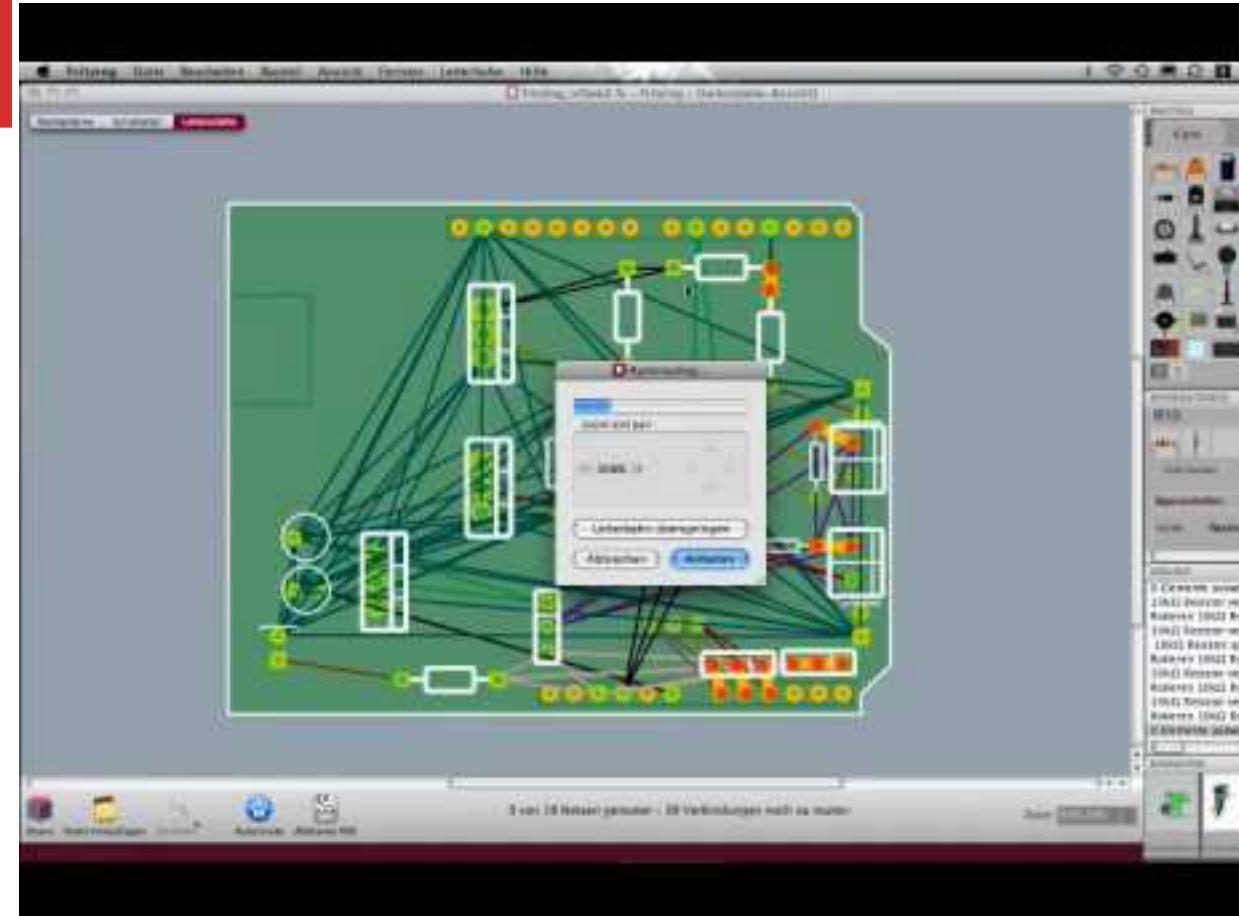
Fritzing

The Fritzing website features a red header with the word "fritzing" in white and the tagline "electronics made easy". Below the header is a navigation bar with links: Projects, Parts, Download, Learning, Services, Contribute, FORUM (highlighted in blue), and FAB. A large image of a Raspberry Pi 3 Model B board is displayed, with a red box highlighting the "Raspberry Pi 3" text. Below the image is a text block: "Releasing a custom fritzing part for the new Raspberry Pi 3. Download "Raspberry_Pi3_3.fzpz"".

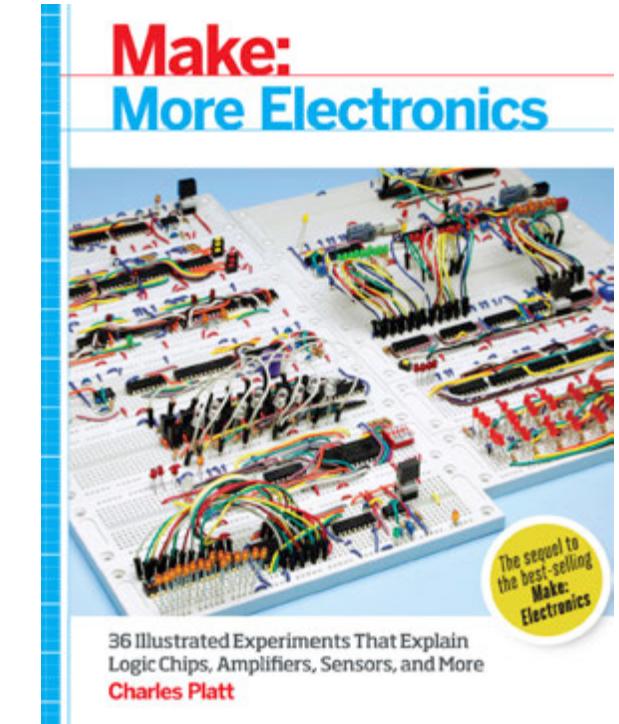
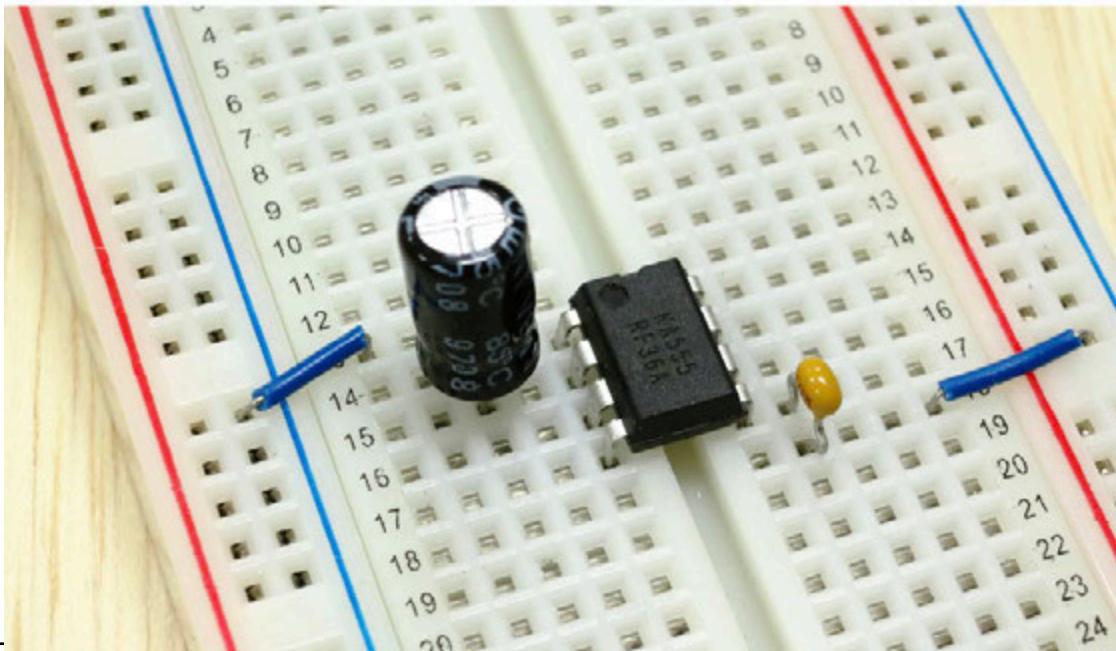
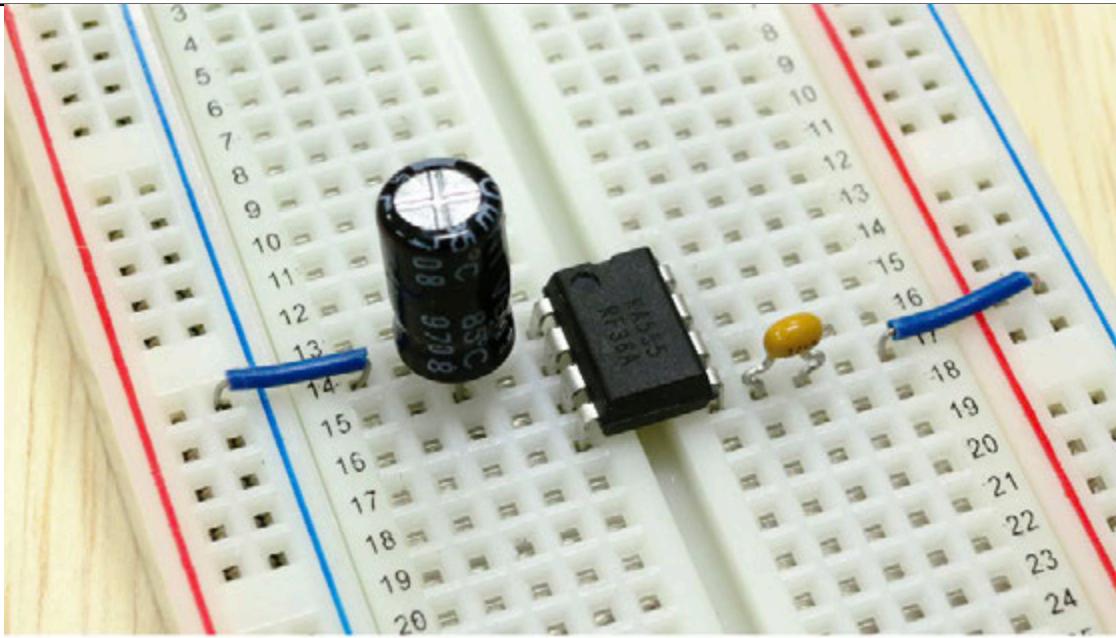
So new Raspberry Pi 3 Model B has just been released. There are two giant upgrades in the Pi 3. The first is a next generation Quad Core Broadcom BCM2837 64-bit ARMv8 processor, making the processor speed increase from 900 MHz on the Pi 2 to up to 1.2GHz on the Pi 3.

The second giant upgrade (and this is the one we're personally most excited about) is the addition of a BCM43143 WiFi chip BUILT-IN to your Raspberry Pi. No more pesky WiFi adapters - this Pi is WiFi ready. There's also Bluetooth Low Energy (BLE) on board making the Pi an excellent IoT solution (BLE support is still in the works, software-wise)

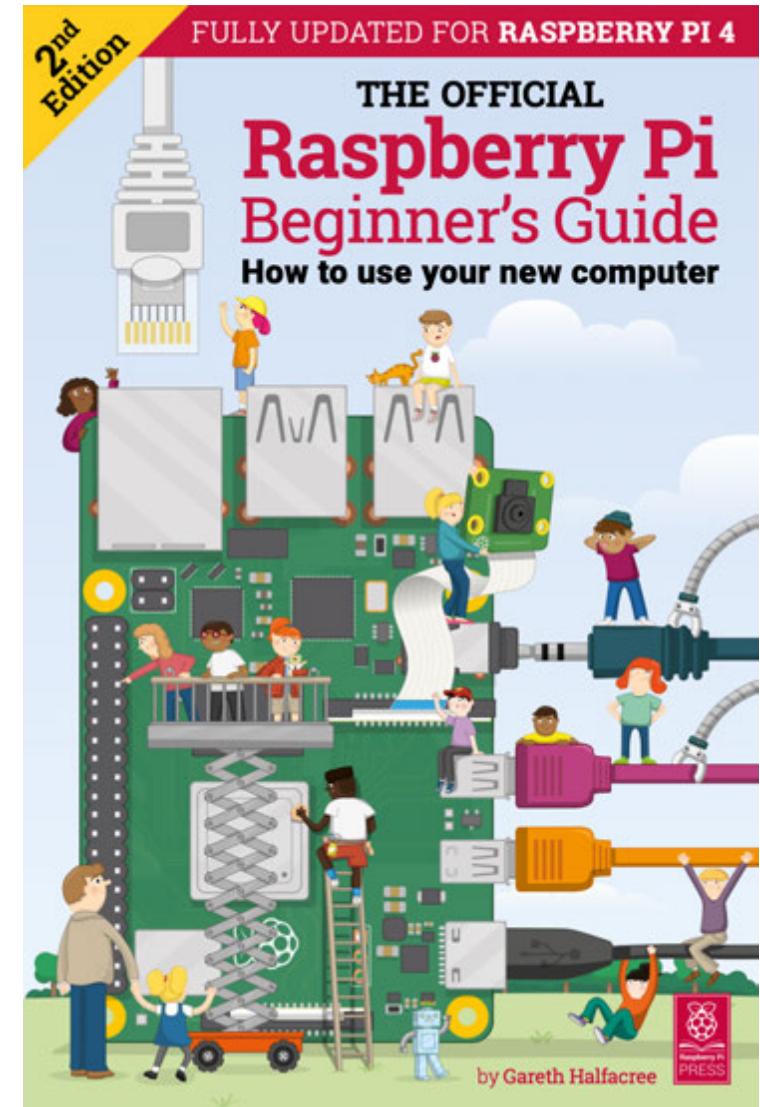
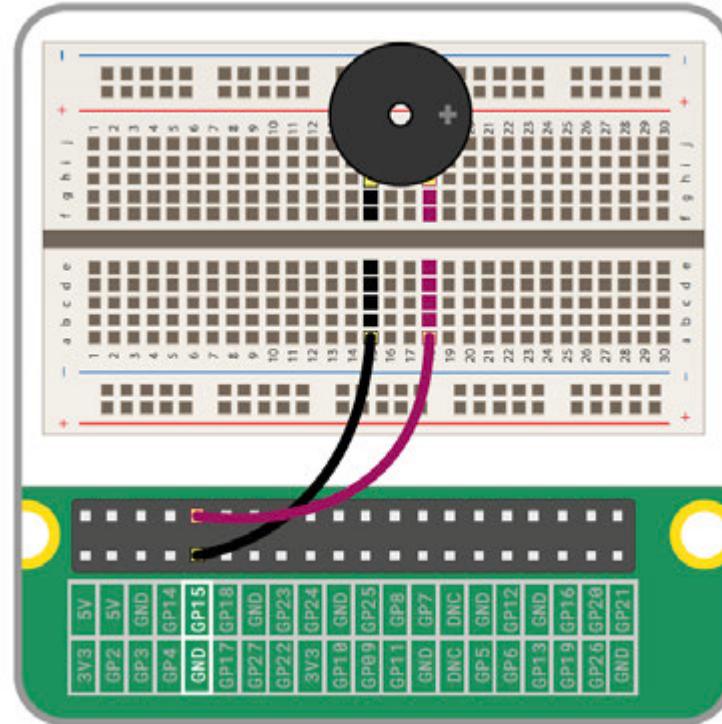
<https://fritzing.org/projects/raspberry-pi-3>



Breadboard

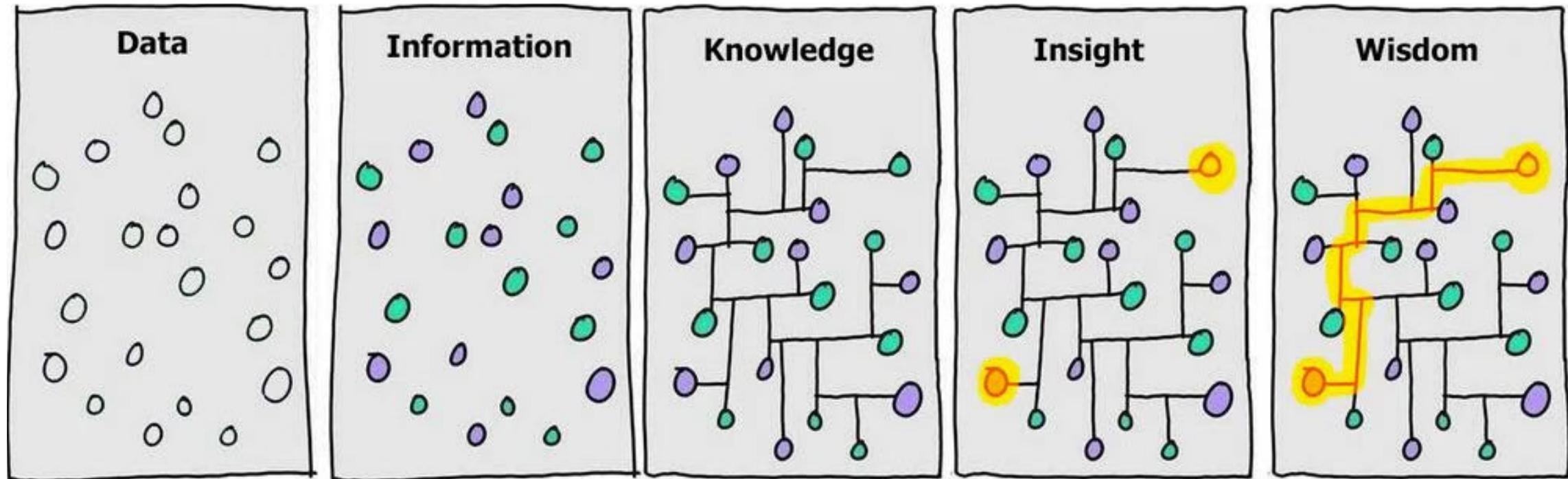


Rapid Prototyping

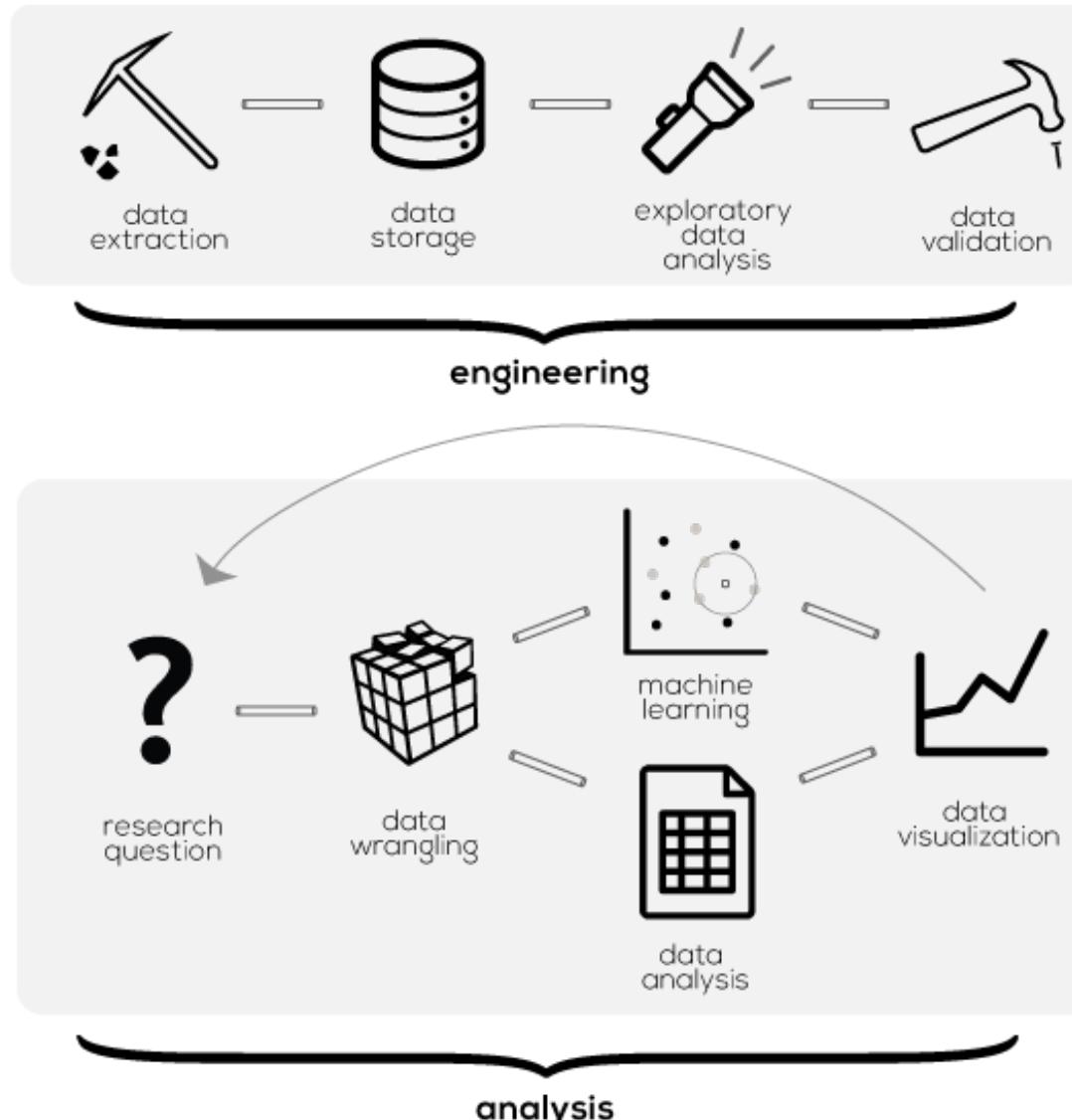


Data Science Core-Concepts

How to make sense of IoT Data (Science)



the data pipeline



DATA-DRIVEN IoT: WHAT IS DATA?

Data [gegevens]

Raw Facts

No Context

Numbers

Symbols

Data comes from the Latin word, "datum," meaning a "thing given."

Although the term "data" has been used since as early as the 1500s, modern usage started in the 1940s and 1950s as practical electronic computers began to input, process, and output data.

98734975471894614398734578

20875980542158009258202908

12349823094823048002343423

98734975471894614398734578

20875980542158009258202908

12349823094823048002343423

DATA STRUCTURING: Observations versus Variables

Data Table [DATA MATRIX]

A generalized version of the data table is shown. This table can represent any number of **observations** described over multiple **variables**.

This table describes a series of observations (from o1 to on) where each observation is described using a series of variables (from x1 to xp). A value is provided for each variable of each observation.

Patient ID	Treated	Age	Outcome	Random
1	Yes	Young	Positive	0.24
2	No	Young	Positive	0.85
3	Yes	Old	Negative	0.64
4	No	Old	Negative	0.70
5	No	Old	Negative	0.87
6	No	Old	Negative	0.72
7	No	Old	Negative	0.86
8	No	Young	Negative	0.16
9	No	Young	Positive	0.17

Observations	Variables				
	x ₁	x ₂	x ₃	...	x _p
o_1	x_{11}	x_{12}	x_{13}	...	x_{1p}
o_2	x_{21}	x_{22}	x_{23}	...	x_{2p}
o_3	x_{31}	x_{32}	x_{33}	...	x_{3p}
...
o_n	x_{n1}	x_{n2}	x_{n3}	...	x_{np}

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

variables

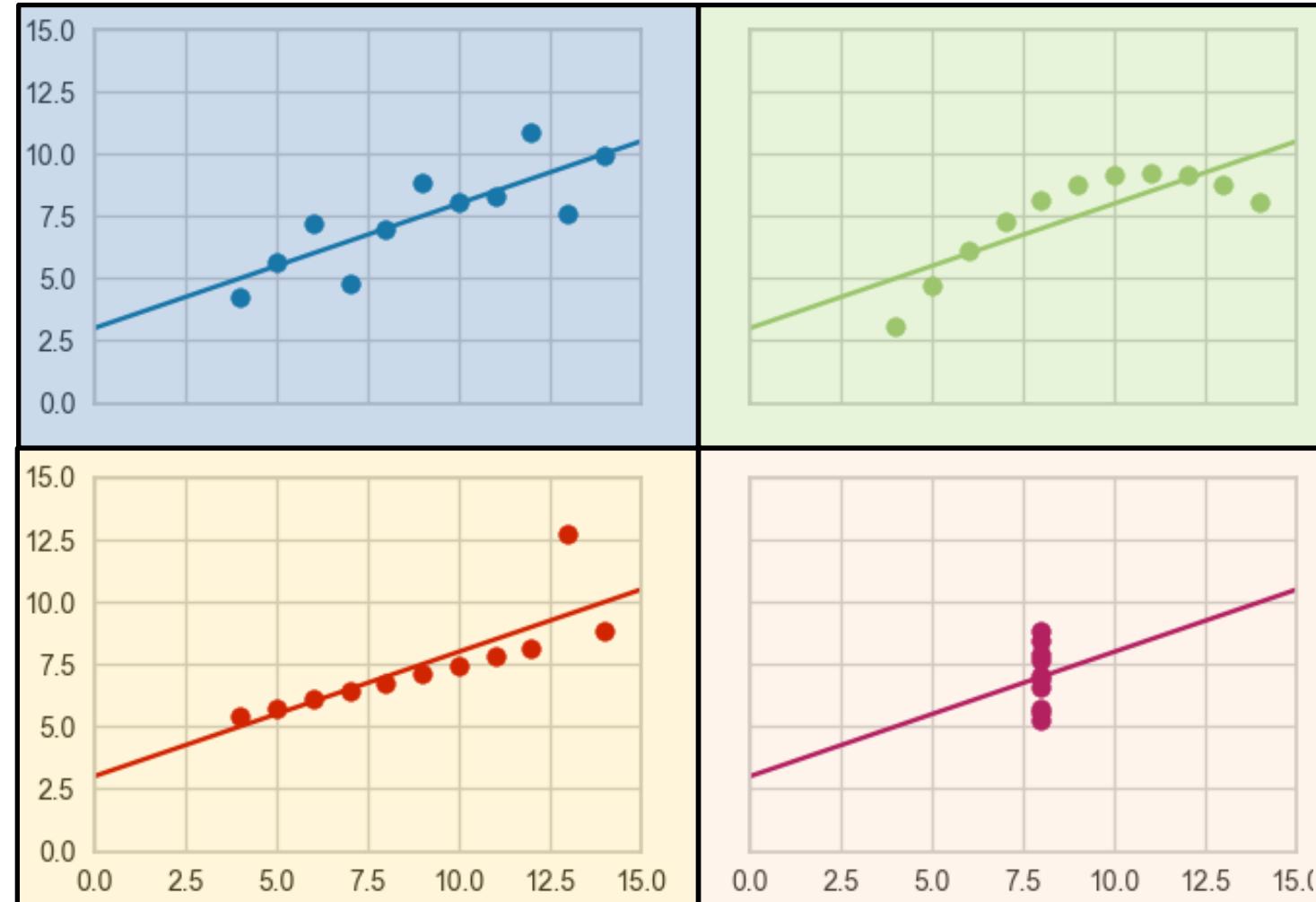
country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

observations

DATA STRUCTURING: Relationships & Patterns [Anscombe Quartet]

Anscombe's quartet

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89



ORGANIZING DATA

- STRUCTURED DATA
 - lists
 - $n \times p$ tables, arrays
 - hierarchies
 - (e.g., organization chart)
 - networks
 - (e.g., travel routes,
 - hypertext = links)
- Generic data-interchange formats:
XML, JSON
- UNSTRUCTURED DATA
 - text
 - images
 - video
 - sound
- Often can be made structured by, e.g., parsing language, segmenting images, etc.

JSON

- Similar to XML but simpler

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "isAlive": true,  
    "age": 25,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": "10021-3100"  
    },  
    "phoneNumbers": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "office",  
            "number": "646 555-4567"  
        },  
        {  
            "type": "mobile",  
            "number": "123 456-7890"  
        }  
    "children": [],  
    "spouse": null  
}
```

PARSING DATA INTO USEFUL INFORMATION

- Given a known grammar, unstructured text data can be parsed
- “It ain’t over till the fat lady sings”
((it, (ain't, over)), (till, ((the, (fat, lady)), sings)))
- Similarly, images can be segmented into parts

PARSING

- Similarly, images can be segmented into parts



(a) Detection



(b) Instance Segmentation



(c) Human Parsing

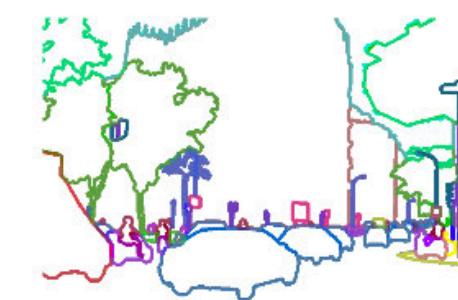
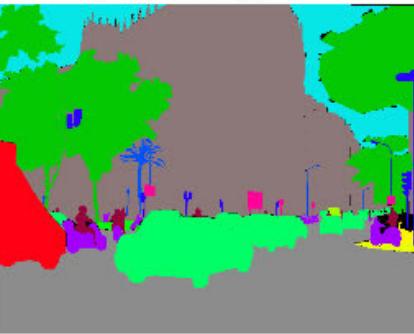
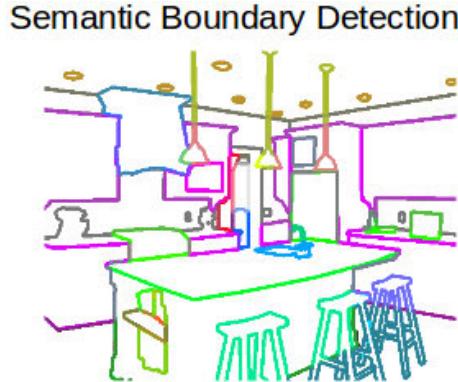
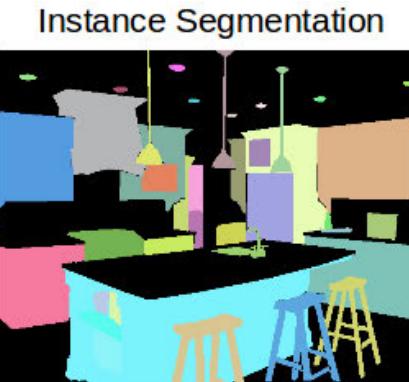


(d) Multi-Human Parsing



PARSING

- Similarly, images can be segmented into parts



TYPES OF DATA: Quantitative versus Qualitative [numerical vs categorical]

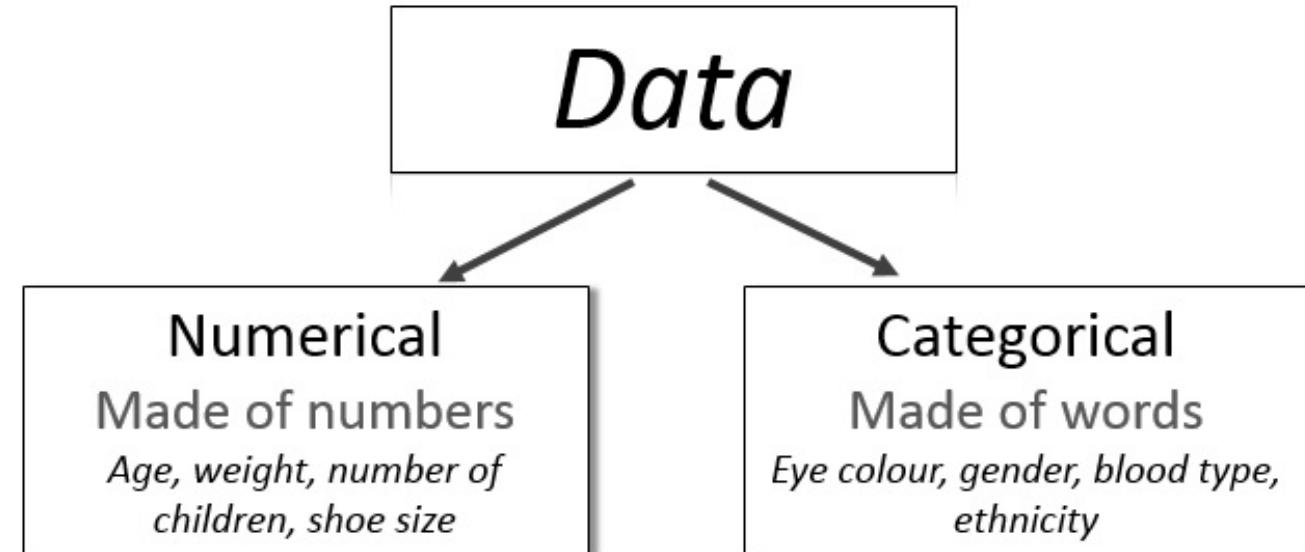
Data Quantification

Quantitative [Numerical] data:

This data can be described using **numbers**, and basic mathematical procedures, including addition, are possible on the set. It can be **discrete** (countable numbers) or **continuous** (infinitely large or small)

Qualitative [Categorical] data:

This data are categories. It cannot be described using numbers and basic mathematics. Is generally thought of as being described using "**natural**" **categories** and **language**.



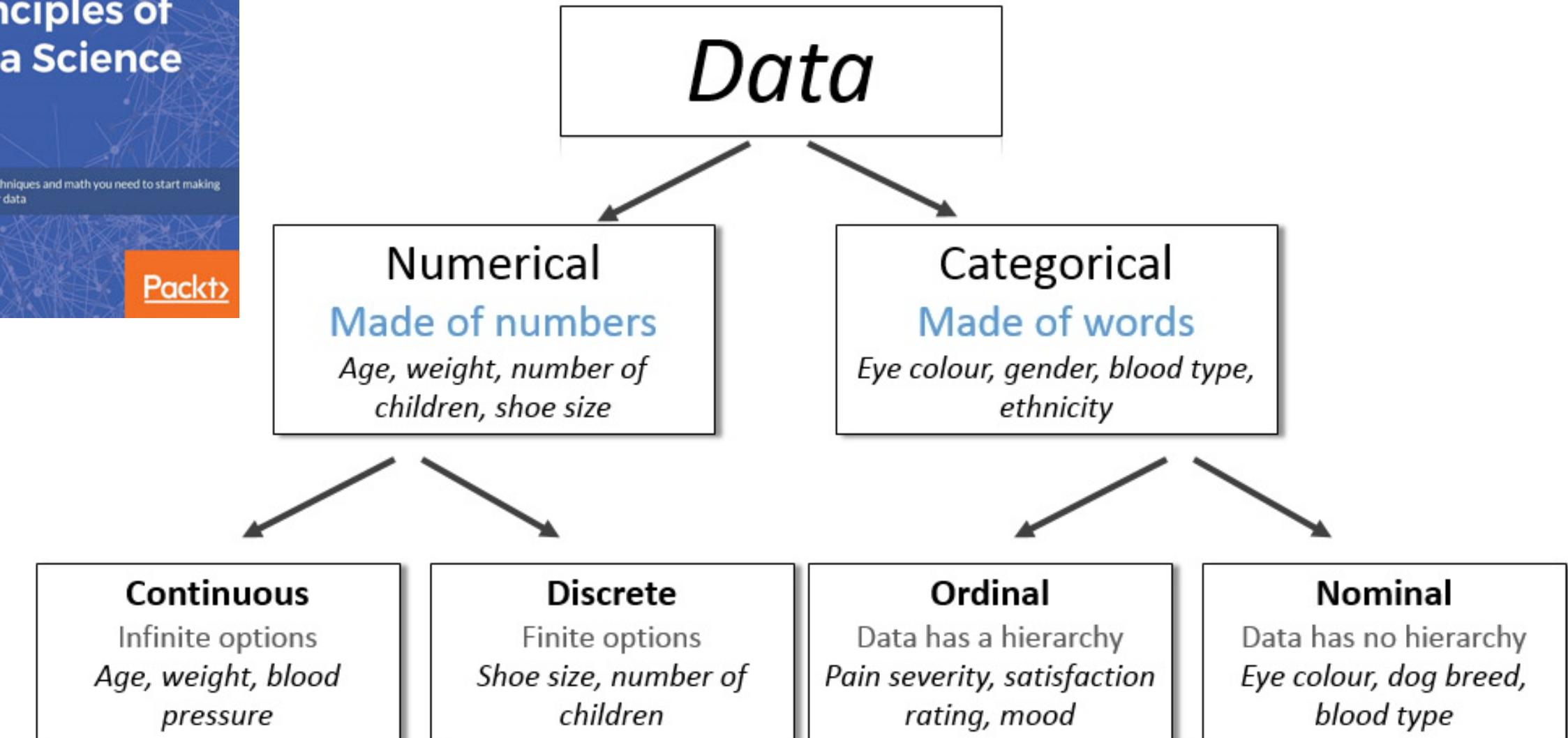
- Quantitative values
 - **Measure** things
 - *Revenue, Units, Marketshare, Duration, Customer Satisfaction, Visits, Price, etc.*
- Categorical values
 - Subdivide things into **groups**
 - *Region, product, category, employee, etc.*

Principles of Data Science

Learn the techniques and math you need to start making sense of your data

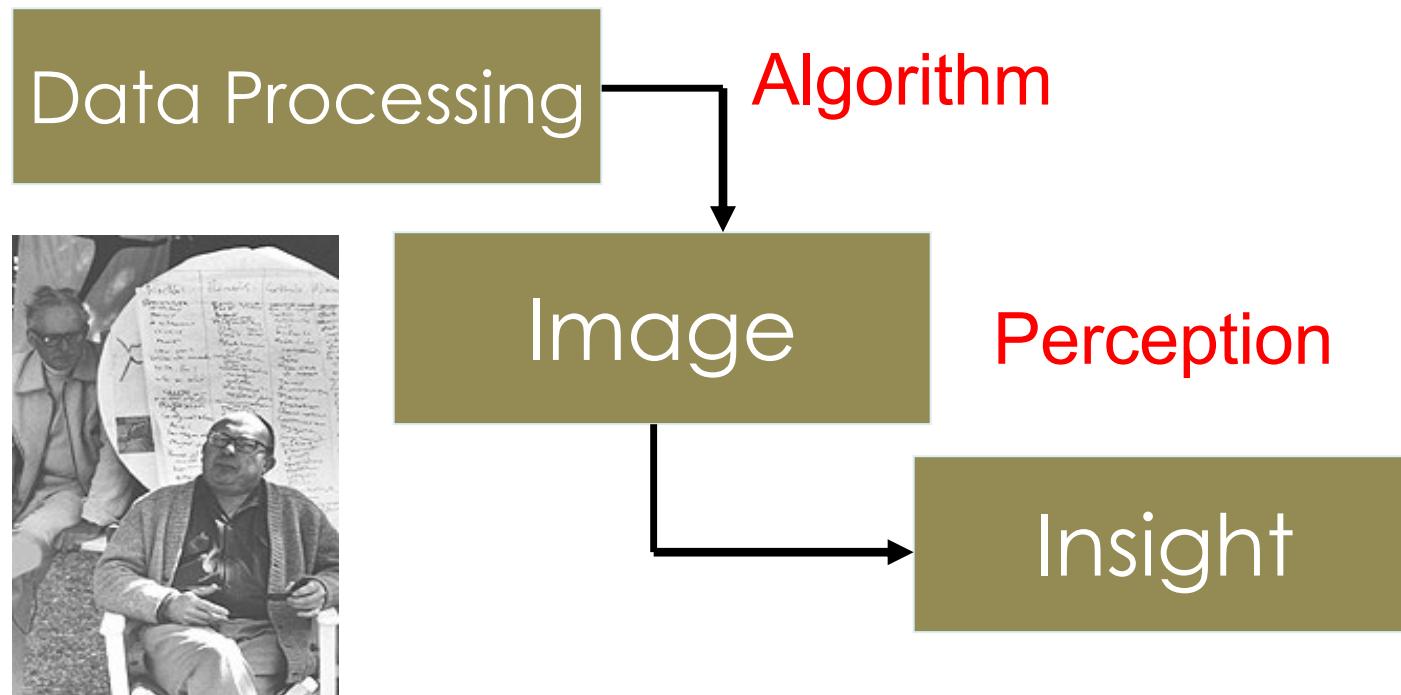


Packt



DATA STRUCTURING: data worden pas inzichtelijk als (beeld)figuur (graphical visualization) of GRAAF (graph)

Jacques Bertin who wrote the classic works of graphical visualization "Semiology of Graphics" states that the "transformation from numbers to insight requires two stages"



SEE ALSO: http://www.cs.wright.edu/~jgallii/hfe306/Data_Visualization_Quenin.ppt

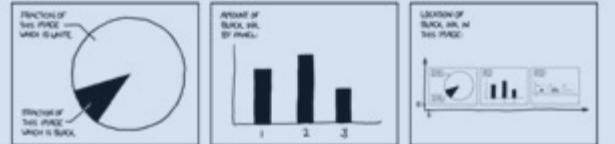
Practical Data Visualization

March 18, 2015
 COMPSCI 216:
 Everything Data

Angela Zoss
 Data Visualization Coordinator
 Data and Visualization Services

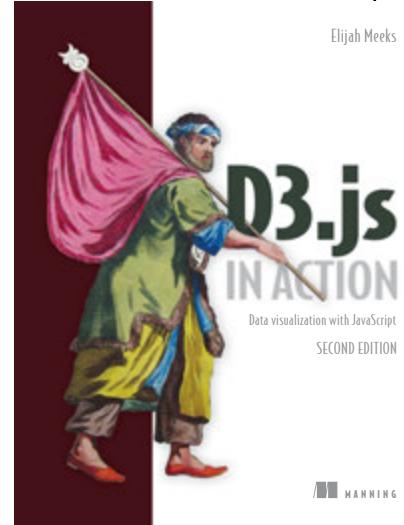
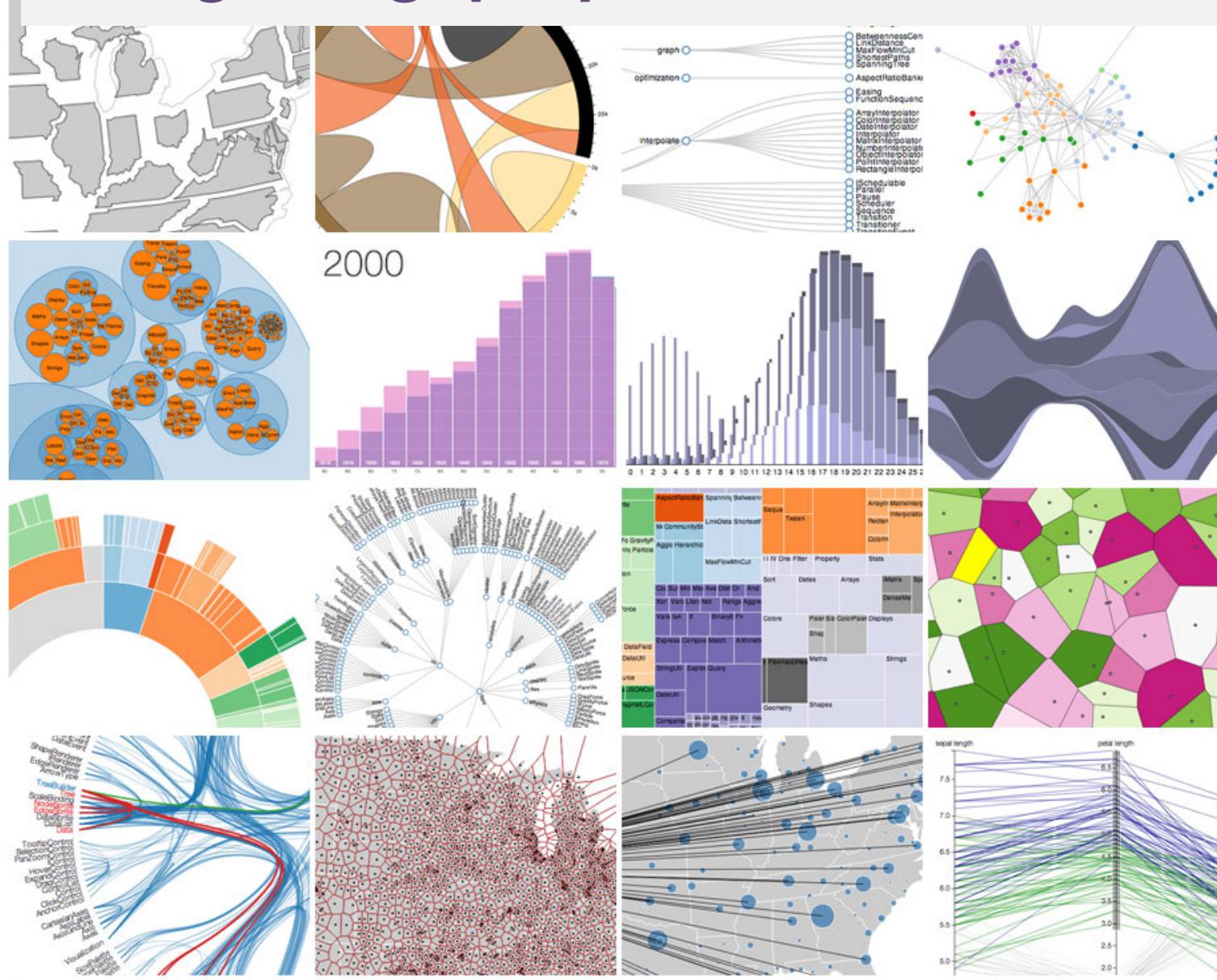
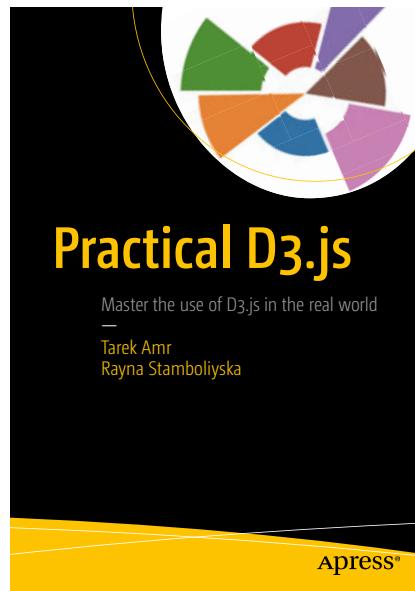


Communicating through infographics: visualizing scientific and engineering information



IEEE
 Christa Kelleher
 Nicholas School of the Environment
 Duke University

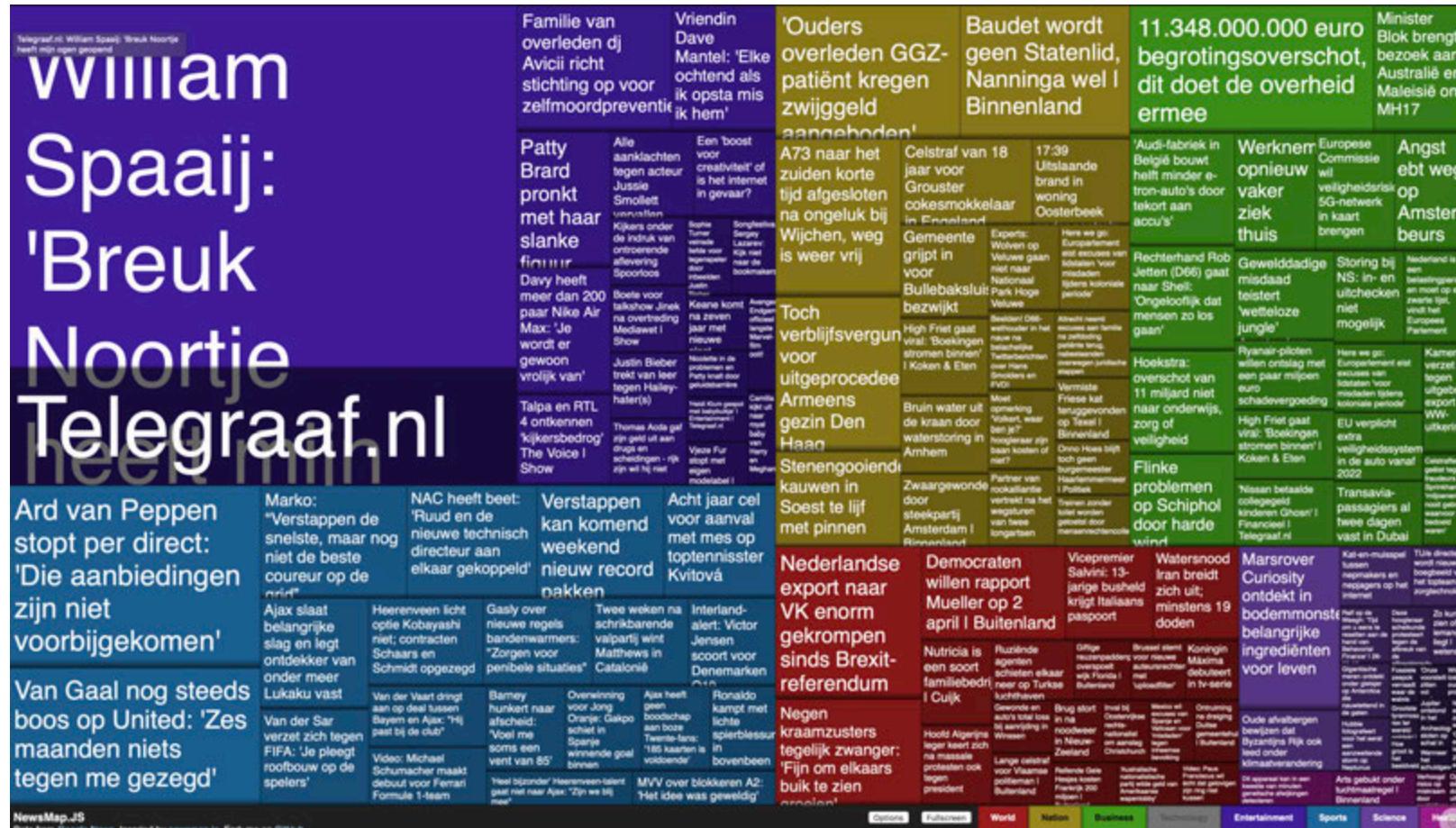
Wat geeft grip op DATA?



DATA STRUCTURING: TREE MAP [example]

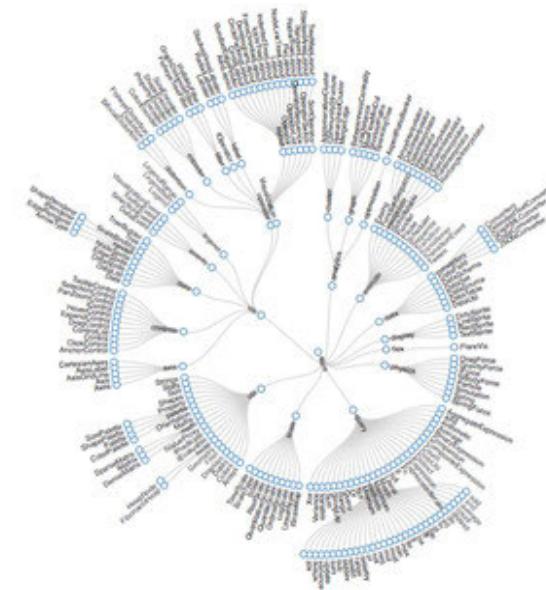
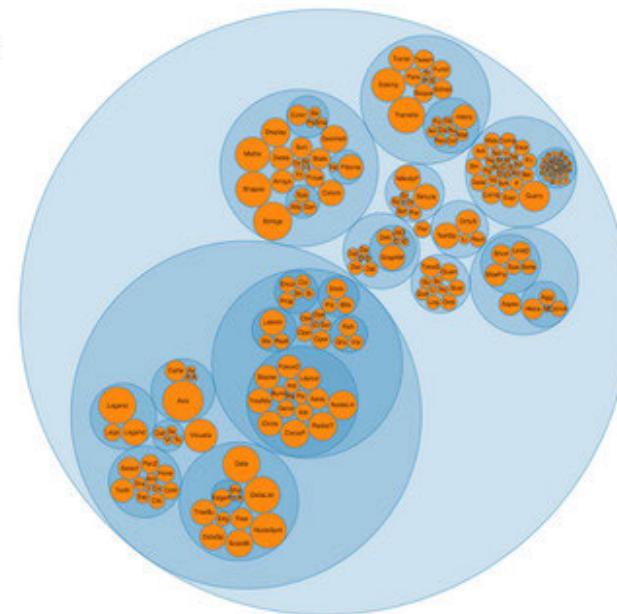
<https://github.com/IJMacD/newsmap-js>

<https://newsmap.ijmacd.com>

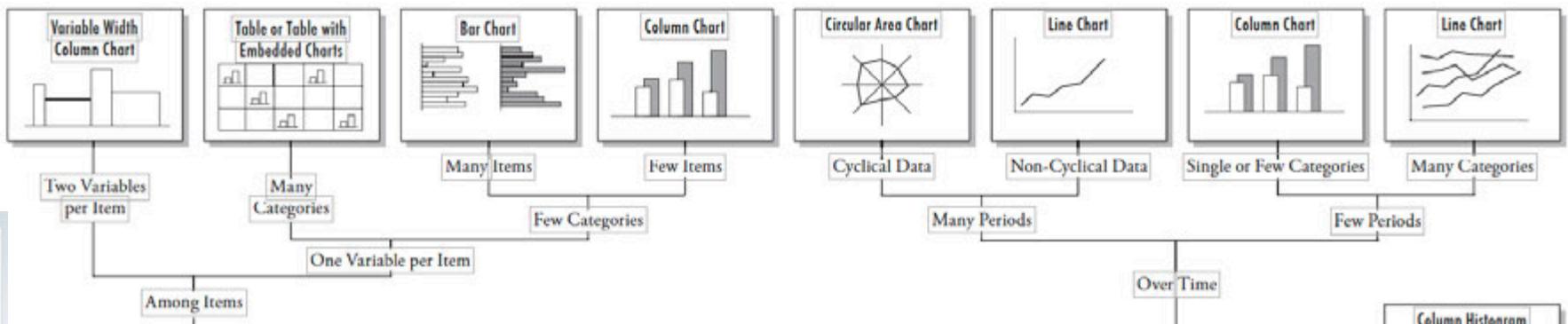
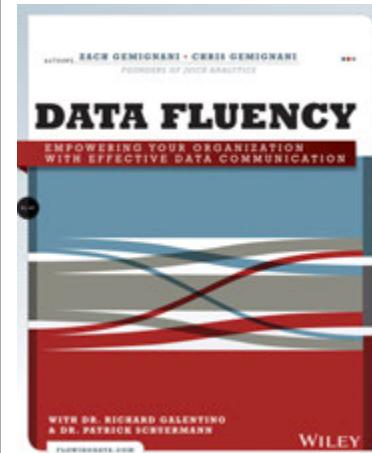


Design for
Information

DATA STRUCTURING: TREE MAP [example]

A**B****C****D**

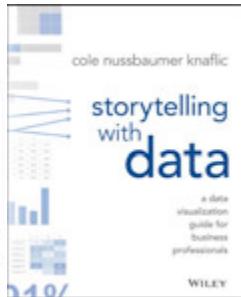
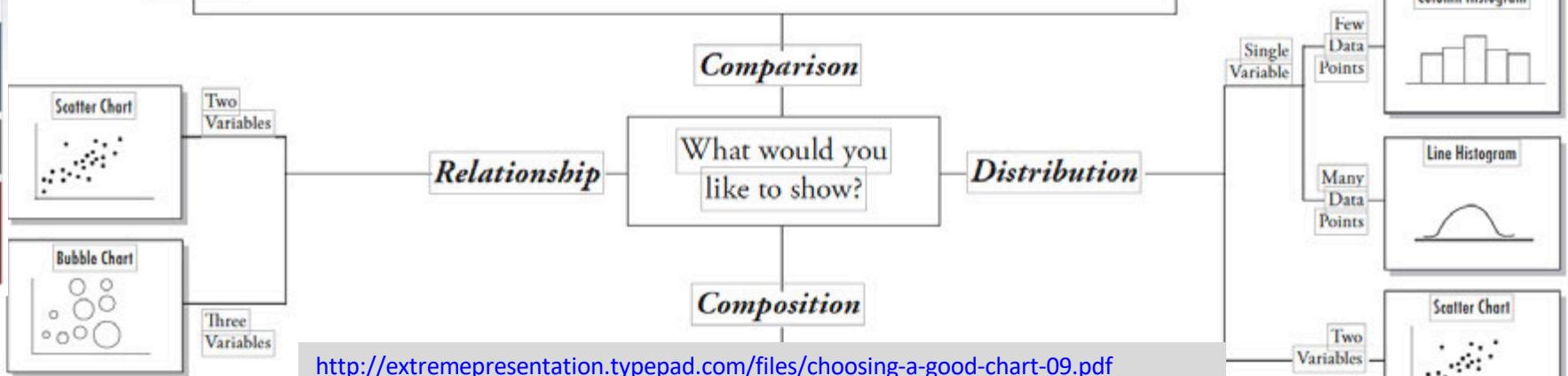
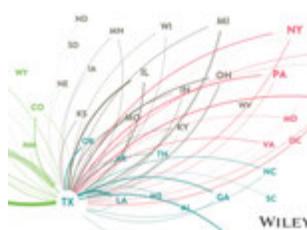
<http://labs.juiceanalytics.com/chartchooser/index.html>



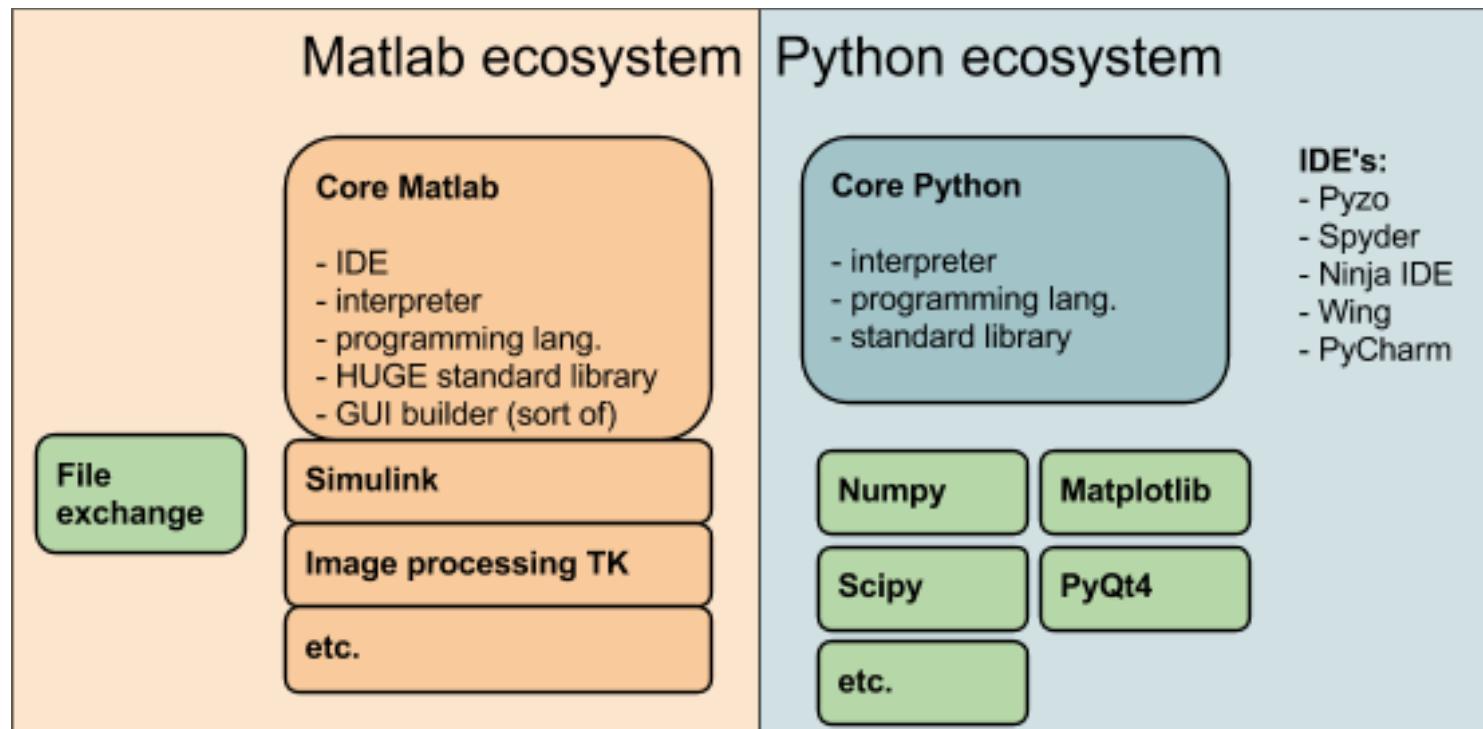
Graph Analysis and Visualization

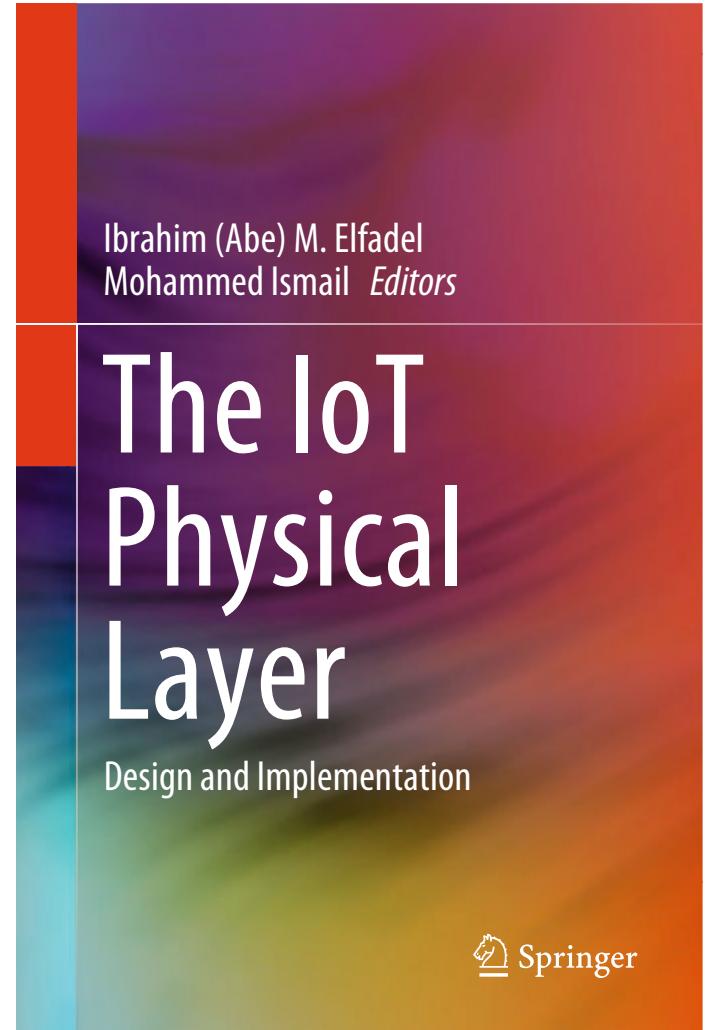
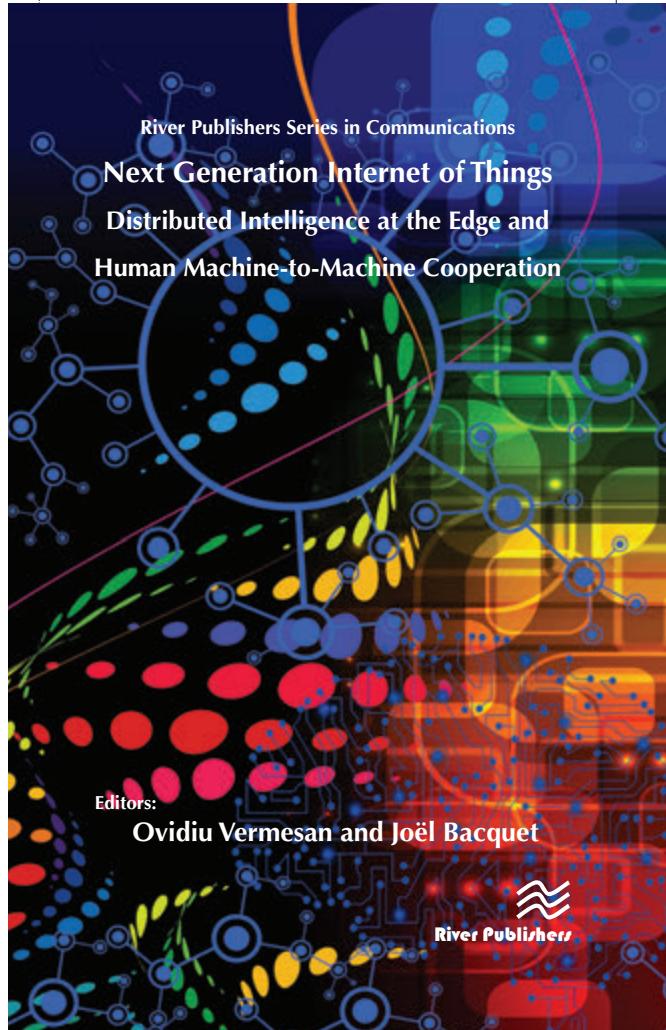
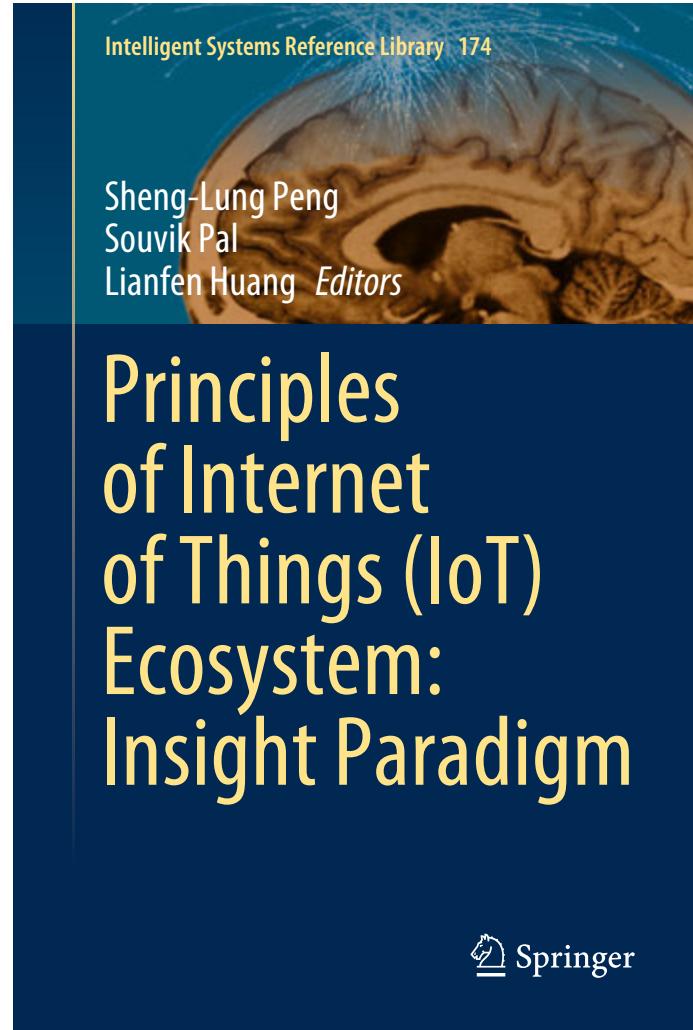
Discovering Business Opportunity in Linked Data

Richard Brath and David Jorner



Tools to Organize/Structure Data to allow for Data-analysis





IoT Core-Concepts

World Wide Web (WWW) vs IoT

WWW:

In the Internet of Computers (WWW), the main data producers and consumers are human beings.

IoT:

The main actors become things, where things are the majority of data producers and consumers.

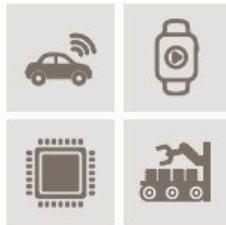
Qin, Y., Sheng, Q. Z., & Curry, E. (2015). Matching over linked data streams in the internet of things. *IEEE Internet Computing*, 19(3), 21-27.



IoT Core Concepts

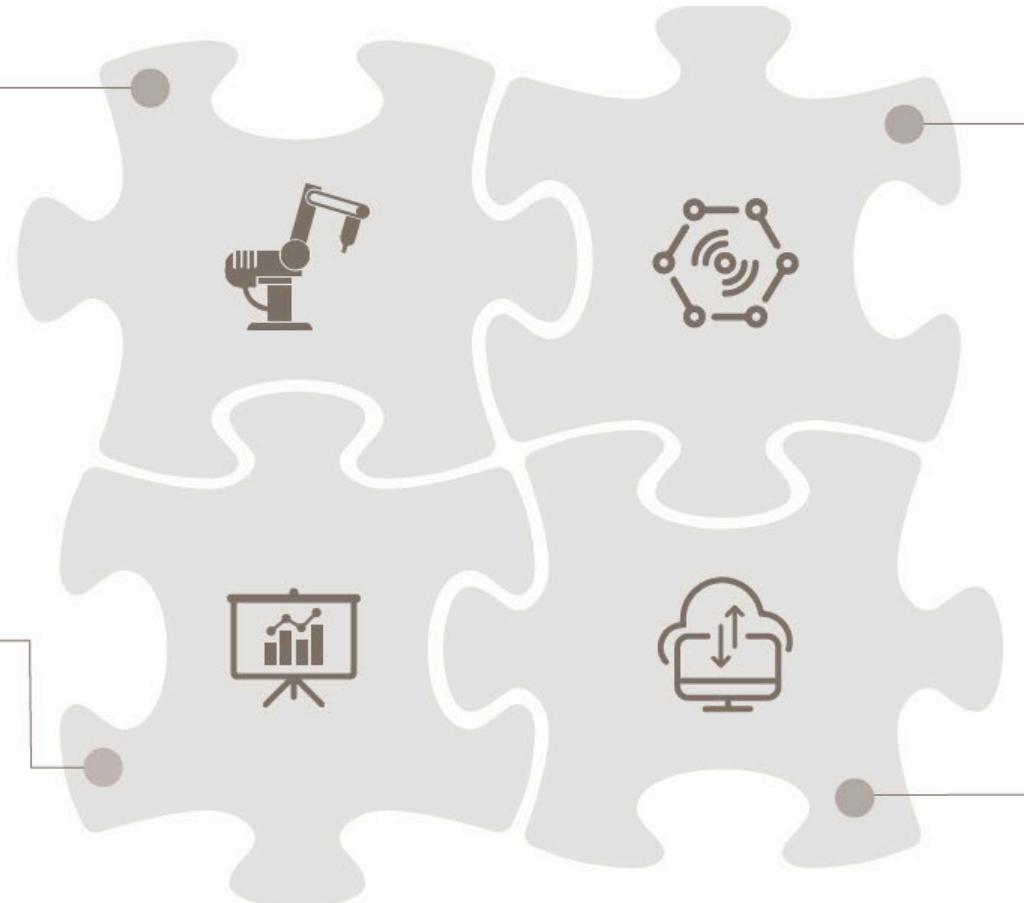
Devices

- Sensors
- Actuators
- Machines



Business Model

- Outcome driven
- Product as a service
- Leverage existing investment



Connectivity

- Network
- Protocol
- Security



Platform

- Storage
- Analytics
- Visualization
- Integration



IoT Platform + User-Types



Hobbyists



Consumer



Industrial



Industry

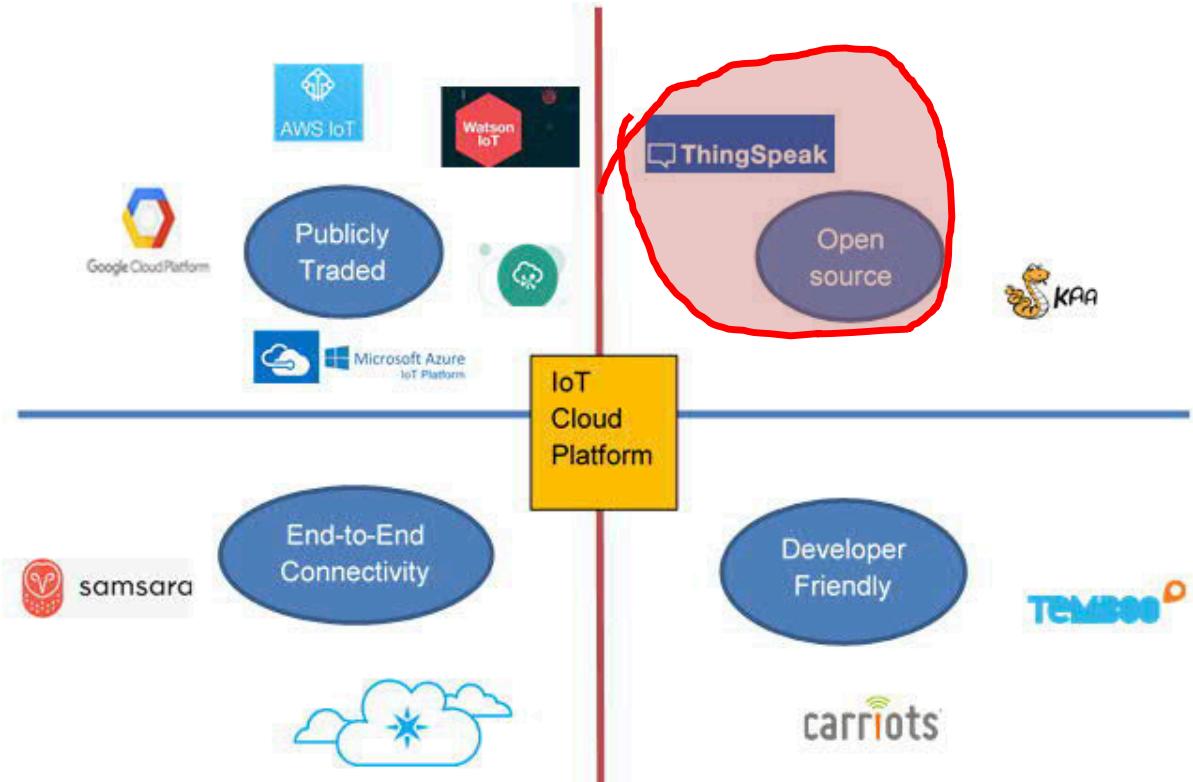
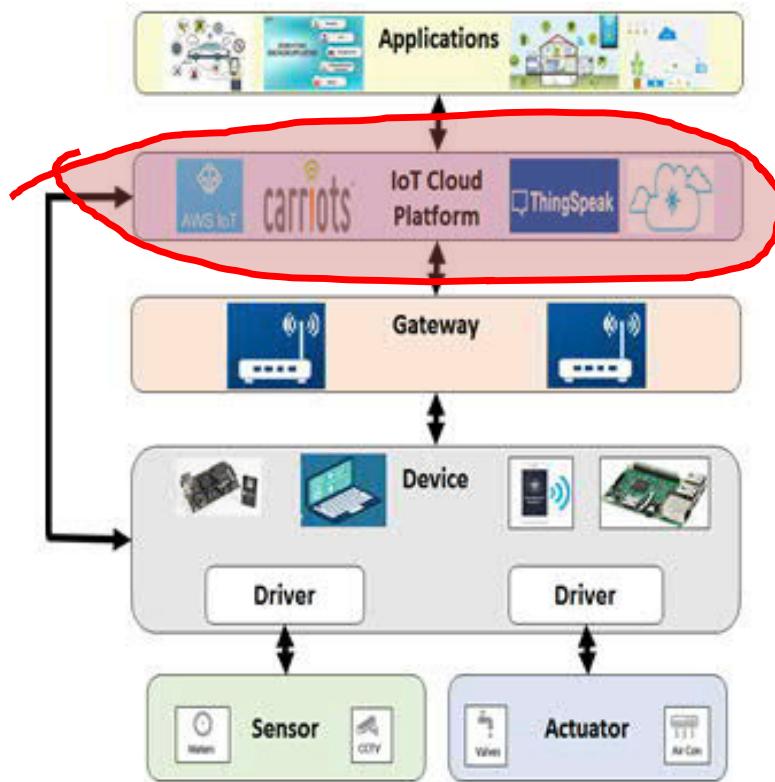
The Internet of Things is a state where the things on the face of the earth connect to the internet and start talking to each other. Things here can be electronic, electrical, mechanical, or electro-mechanical objects.



The number of connected things is projected to grow at an annual compound rate of 23.1% between 2014 to 2020, reaching 50.1 billion things in 2020.¹



IoT Platforms == MiddleWare



<http://www.thetips4you.com/iot-best-open-source-applications-open-source-industrial-iot-platform>

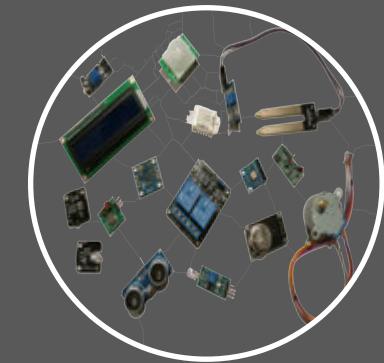
Technologies enabling IoT

WIRELESS TECHNOLOGIES	TCP/IP PROTOCOLS	AUTHORIZATION & AUTHENTICATION TECHNOLOGIES
802.11, 802.15.4, Zigbee, Bluetooth, BLE, CDMA, GSM, LTE	Sockets, IPv4, IPv6, TCP, UDP, ICMP, QoS, SNMP, IMAP, POP3, IPMI, etc.	Oauth2, PAM, LDAP
NETWORK TECHNOLOGIES	MOBILE TECHNOLOGIES	PLATFORMS AND CPU ARCHITECTURES
Ethernet, CAN, rs485/rs422/rs232, 1-wire, i2c, SPI, ModBus/MudBusRTU, IPMI, iSCSI	Android SDK, Qt, iOS SDK, Objective C, Java, Swift	ARM, X86, PowerPC, AVR, PIC
WEB SERVICE TECHNOLOGIES	PROTOCOL TECHNOLOGIES	PROGRAM / SCRIPTING LANGUAGES
SOAP, REST, WSDL, XML, JSON, UDDI, WebSockets	HTTP, JMS, AMQP, D-Bus	Java, C/C++, C#, JavaScript, Ruby, Groovy, Python, Tcl/Tk, ASM, Bash

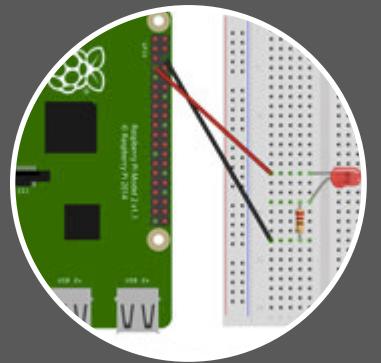
<https://doi.org/10.15779/Z38WW0V>

<https://doi.org/10.1016/j.eswa.2019.05.014>

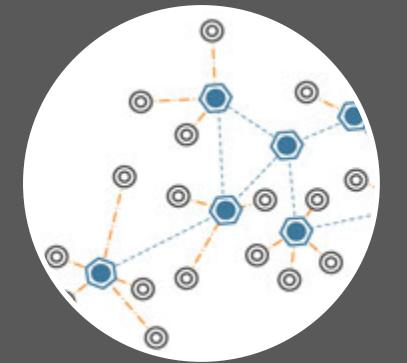
Engineering a IoT Data Pipeline



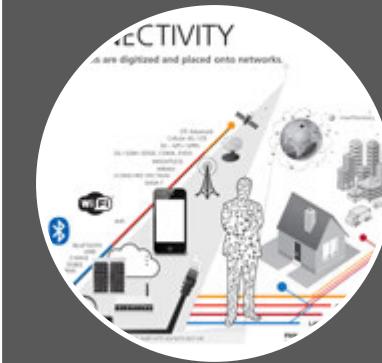
Sensors &
Actuators



Micro-
Processors/
Controllers
(Development
Boards)

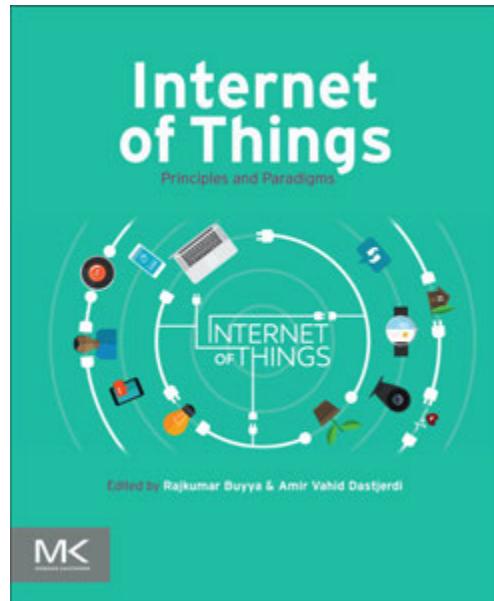


Network /
Protocols



Platforms,
Cloud &
Data Analytics

IoT-device communication protocols

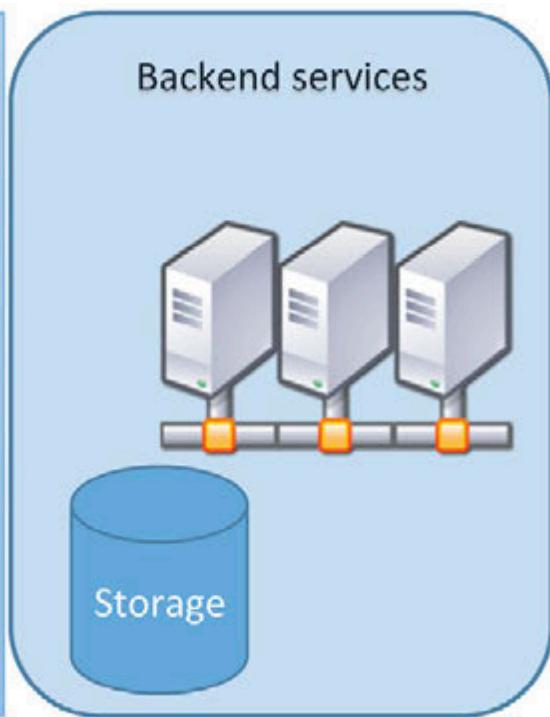
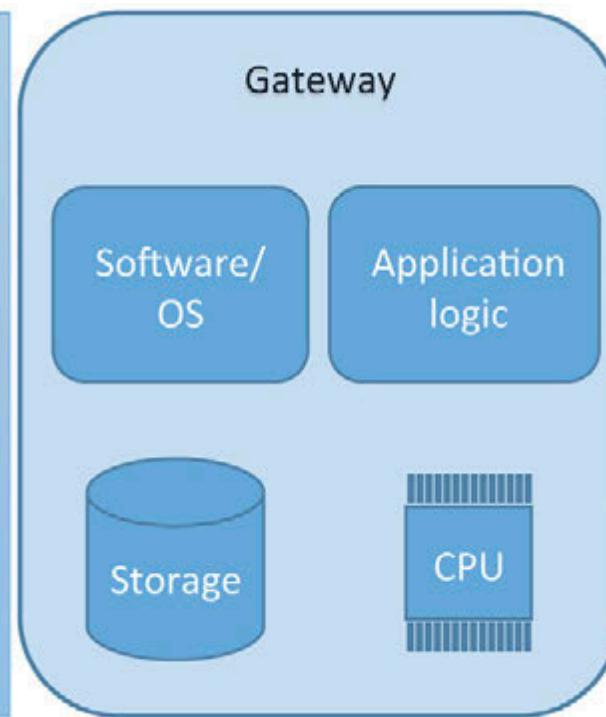
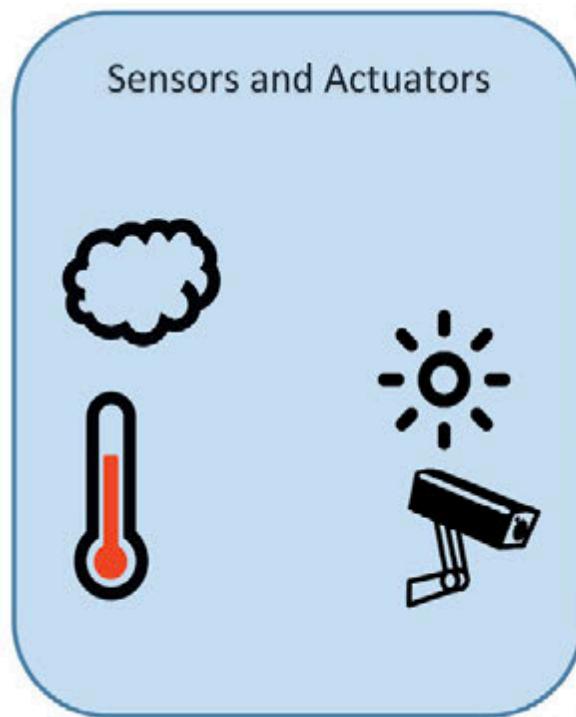


Protocol Name	Transport Protocol	Messaging Model	Security	Best-Use Cases	Architecture
AMPQ	TCP	Publish/Subscribe	High-Optional	Enterprise integration	P2P
CoAP	UDP	Request/Response	Medium-Optional	Utility field	Tree
DDS	UDP	Publish/Subscribe and Request/Response	High-Optional	Military	Bus
MQTT	TCP	Publish/Subscribe and Request/Response	Medium-Optional	IoT messaging	Tree
UPnP	—	Publish/Subscribe and Request/Response	None	Consumer	P2P
XMPP	TCP	Publish/Subscribe and Request/Response	High-Compulsory	Remote management	Client server
ZeroMQ	UDP	Publish/Subscribe and Request/Response	High-Optional	CERN	P2P

IoT communication Protocols

Publisher

Broker



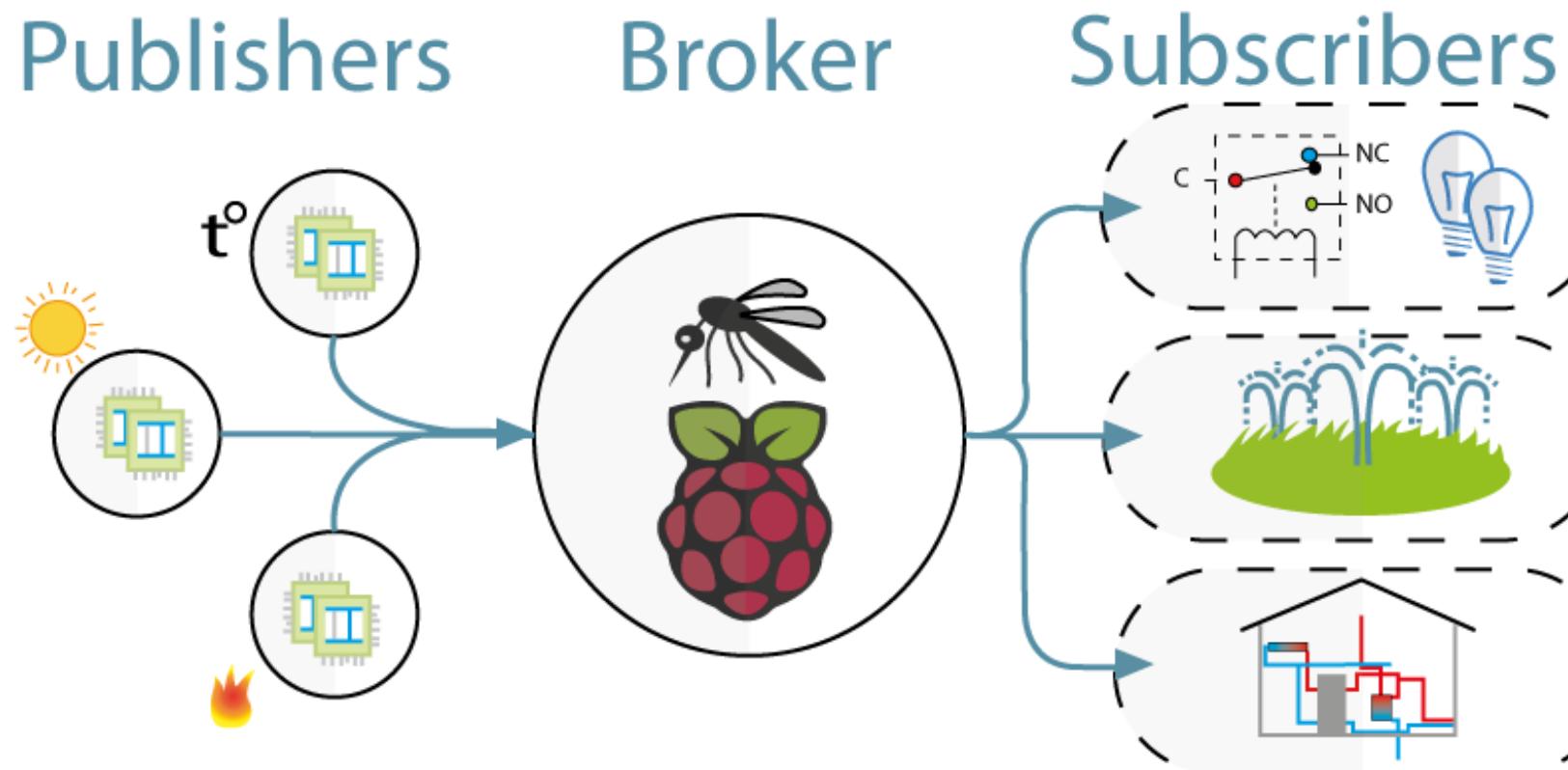
SENSOR TO GATEWAY COMMUNICATION

IoT Communication

The Broker: The broker acts as a gateway; it receives messages from a publisher (a client) and delivers the messages to a subscriber (another client). Brokers are sometimes referred to as *servers*.

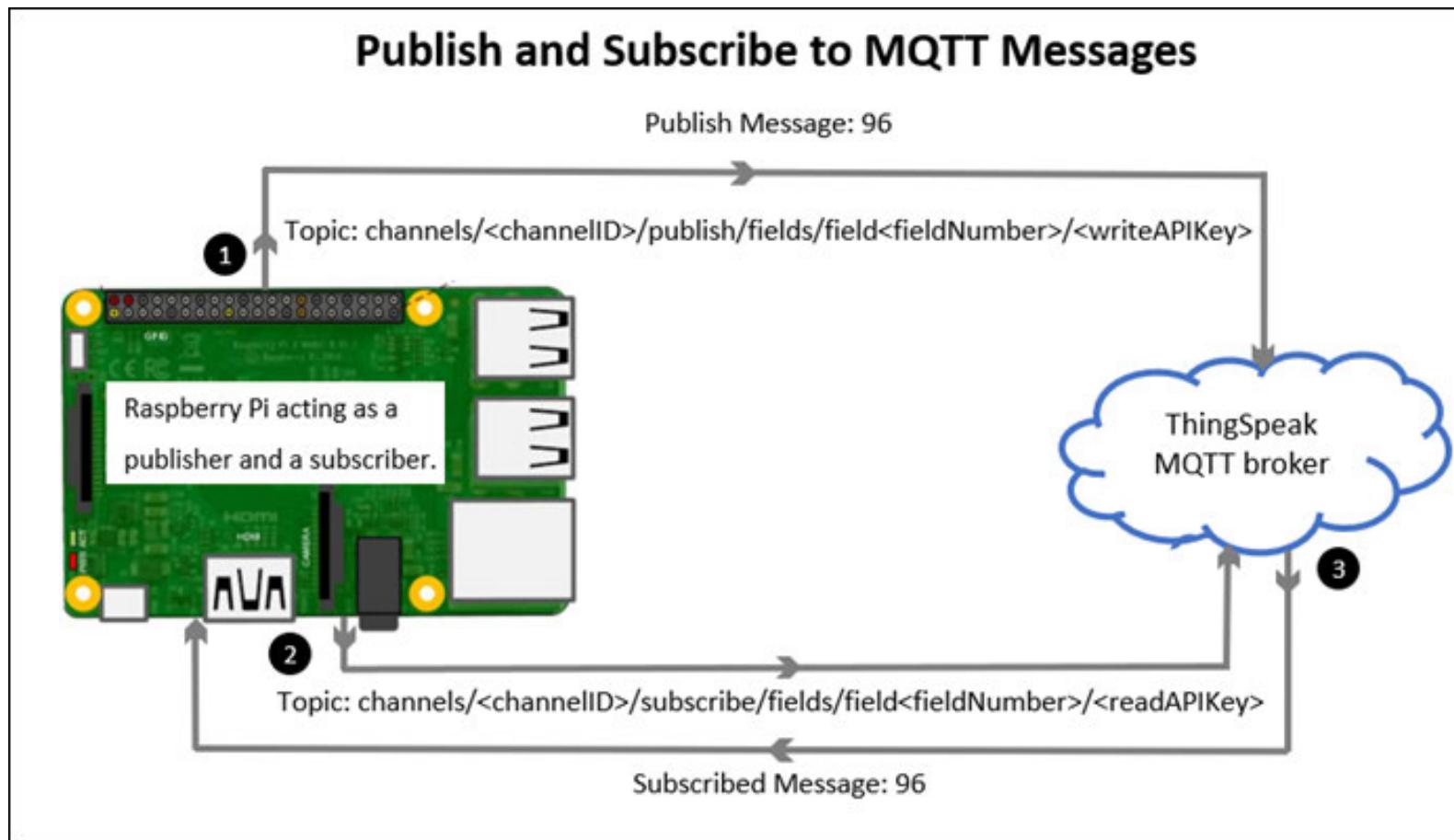
The Subscriber: The subscriber declares its topics of interest to the broker, and the broker sends messages published to those topics.

The Publisher: The publisher sends messages to the broker using a name-space or a topic name, and the broker forwards the messages to the respective subscribers.

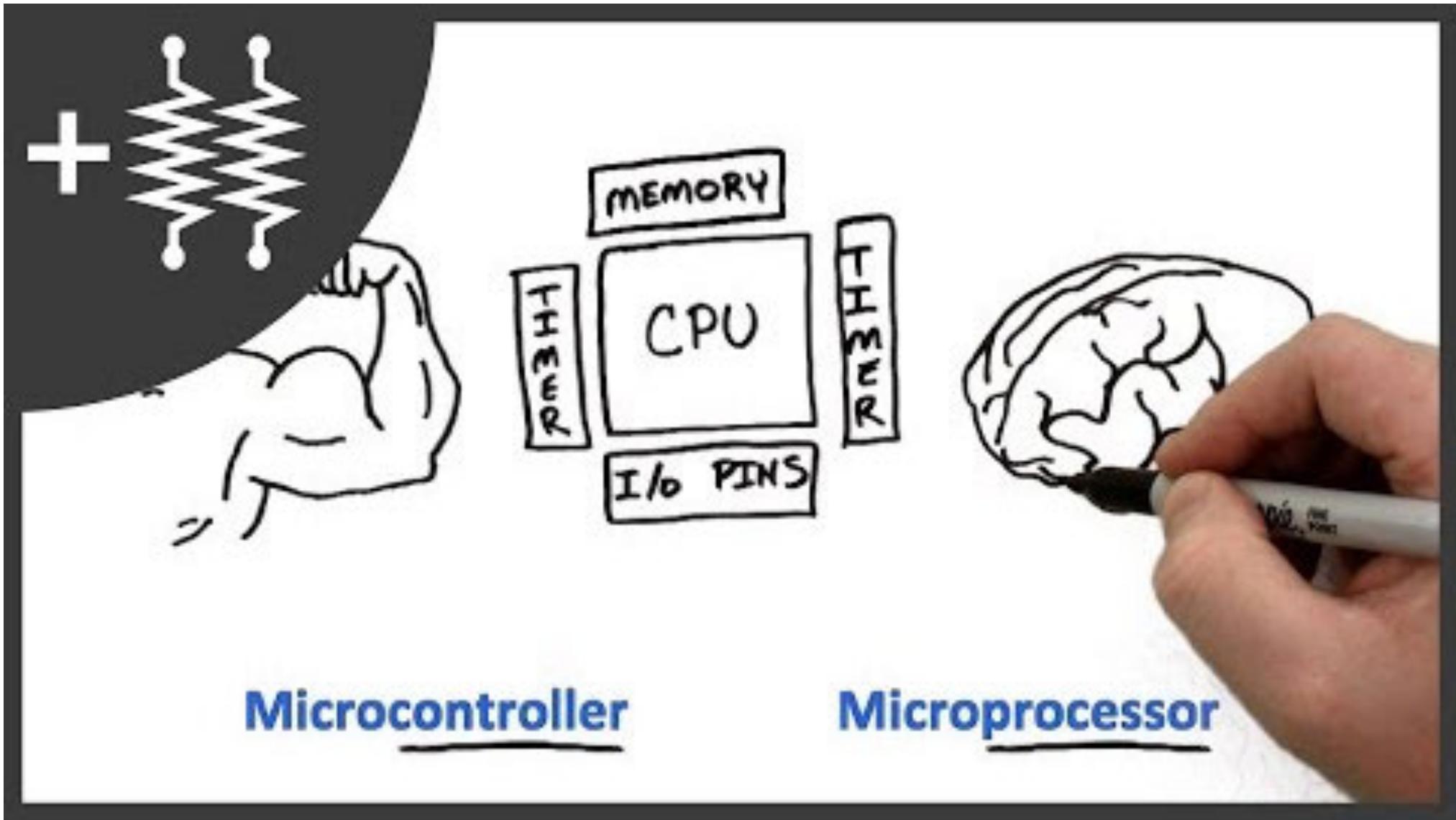


Setting Up Your IoT Device

{IoT device == micro-controller== Raspberry Pi}



Micro- controller/processor Core-Concepts



<https://www.youtube.com/watch?v=7vhvnaWUZjE>



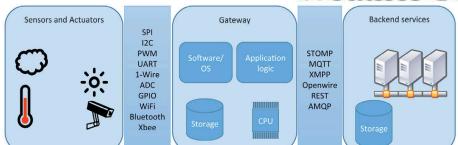
Microprocessor Microcontroller

Only CPU; other peripherals are interfaced via the CPU bus	CPU + (Flash) Memory + RAM + ROM + I/O + Timers + UARTS + ADC + DAC and so on on a single chip
General purpose	Single purpose
High processing power	Low processing power
Power hungry	Can work on a battery

Sensor / Actuator Core-Concepts

(connecting) SENSORS

Sensor	Interface	Processor/Hardware	Power
Wind vane	Analog	Switched resistors	—
Anemometer	GPIO interrupt	Reed switch	—
Rain gauge	GPIO interrupt	Reed switch	—
UV intensity	Analog	ML8511	1,000 µW
Humidity and temperature	2-wire serial	SHT15	80 µW
Barometric pressure	I ² C	BMP180	30 µW
Luminosity	I ² C	TSL2561	720 µW
Lightning sensor	SPI + GPIO interrupt	AS3935	100 µW
Geiger counter	RS-235	LND712	147,000 µW
Weather board	I ² C	Si7020	540 µW
Weather board	I ² C	BMP180	30 µW
Weather board	I ² C	Si1132	1,419 µW

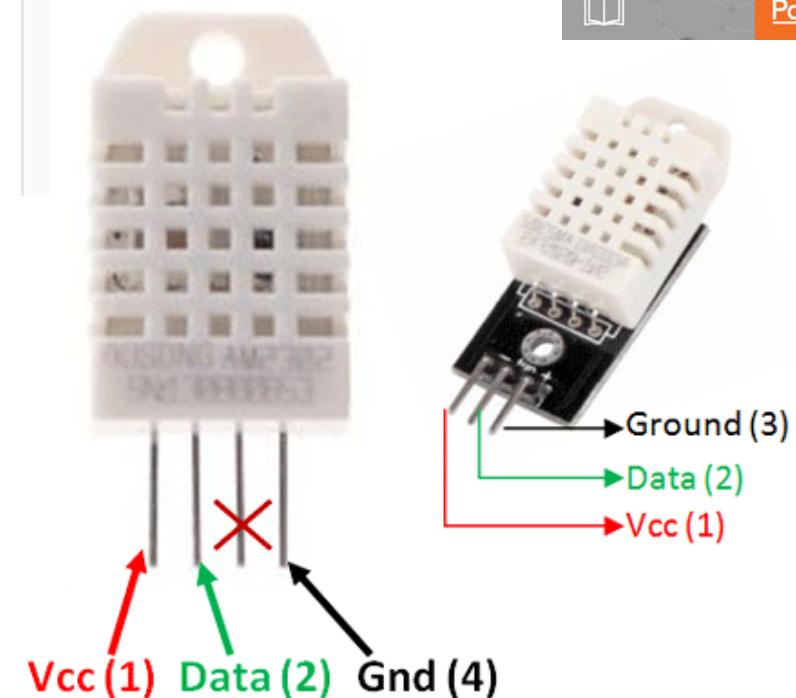
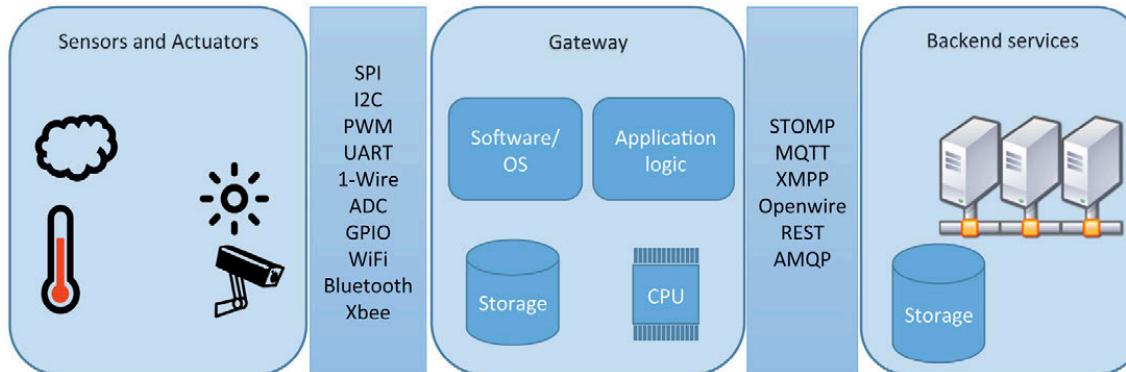


Sensor name

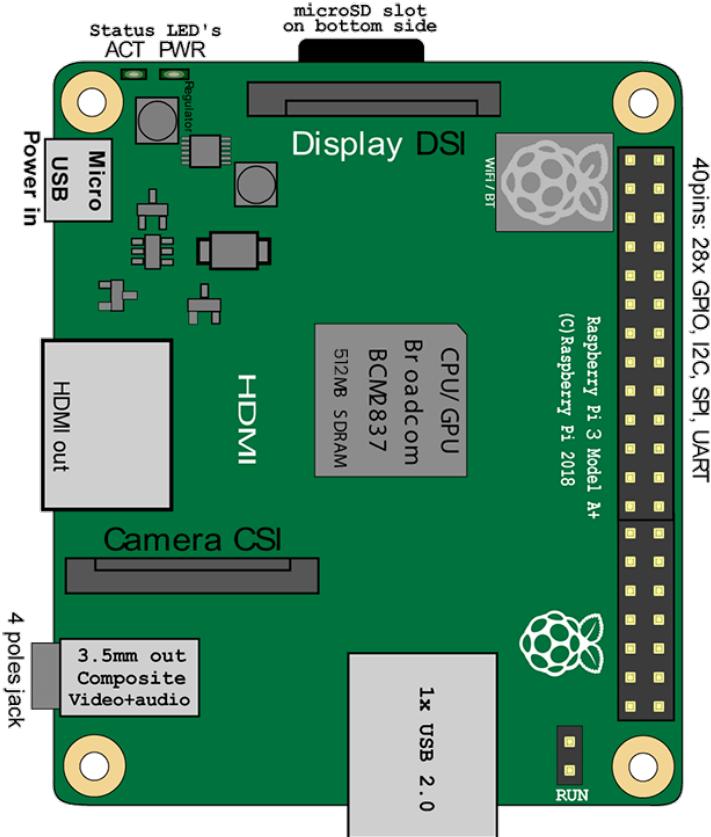
Temperature and humidity sensor (DHT11)

Datasheet link

<https://akizukidenshi.com/download/ds/aosong/DHT11.pdf>



Raspberry Py 3A+



GPIO

schematics of the Raspberry Pi 3 A Plus hardware.

Shown also are general positioning of all the vital circuitry on the board.

The Raspberry Pi 3 A+ is a cut down version of the Pi 3B. As you can tell by its diagram, it features a single USB 2.0 Port. Its only means of network connectivity is the inbuilt Wi-Fi.

CPU: 1.4 GHz quad core ARM Cortex-A53

GPU: 250MHz Broadcom VideoCore IV

RAM: 512mb (Shared with GPU)

Storage: Micro SD

USB 2.0 Ports: 1

USB 3.0 Ports: 0

Networking: 802.11b/g/n/ac dual band 2.4/5 GHz wireless, Bluetooth 4.2 LS BLE

Video Input: 15-pin MIPI camera interface (CSI) connector

Video Outputs: HDMI 1.3, MIPI display interface, DSI

Audio Inputs: Audio over I2S

Audio Outputs: 3.5mm phone jack, Digital Audio via HDMI

Low-Level peripherals: 17 x GPIO, +3.3v, +5v, ground, Plus the following that can be used as GPIO: UART, I2C Bus, SPI bus with two chip select, I2S audio

Power Source: 5v via MicroUSB or GPIO header

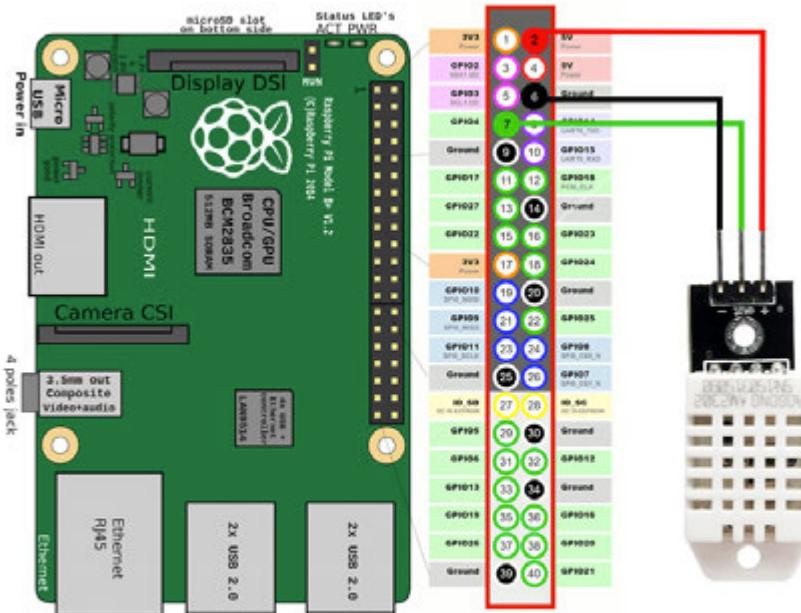
Size: 65.00mm x 56.50mm x 17mm

Weight: 23 g (0.81 oz)

Raspberry Pi Sensor to ThingsBoard using Constrained Application Protocol

Installation

Connecting the sensor



To connect the DHT22 sensor to the RPi:

1. Connect the - output on the sensor to pin 6 on the RPi.
2. Connect the + input on the sensor to pin 2 on the RPi.
3. Connect the output pin on the sensor to pin 7 on the RPi.

<https://github.com/Silver292/rpi-coap>

Introduction

This script is designed to be run on Raspberry Pi 3 hardware running the Raspbian 9.6 operating system.

It is also assumed that a DHT22 temperature and humidity sensor is connected to the Raspberry Pi (RPi) using GPIO 4 pin.

The process of connecting the sensor is described here.

The script creates a Constrained Application Protocol (CoAP) client on the RPi and repeatedly polls the attached DHT22 sensor for temperature and humidity data.

This data is then formatted to JavaScript Object Notation (JSON) and is sent to a CoAP endpoint provided by the ThingsBoard Cloud Platform.

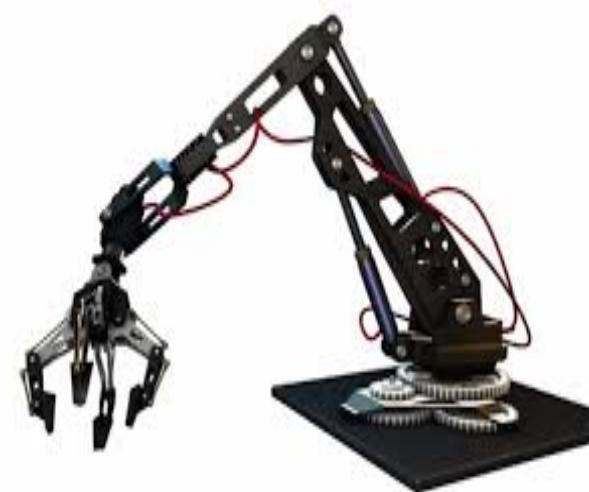
This data is then displayed on the ThingsBoard dashboard that shows the current temperature and humidity as well as historical readings in a graph form.

ACTUATORS

An actuator is a component of a machine that is responsible for moving or controlling a mechanism or system. An actuator requires a control signal and a source of energy.

The control signal is relatively low energy and may be electric voltage or current, pneumatic or hydraulic pressure, or even human power.

ACTUATORS



ACTUATORS

Actuators

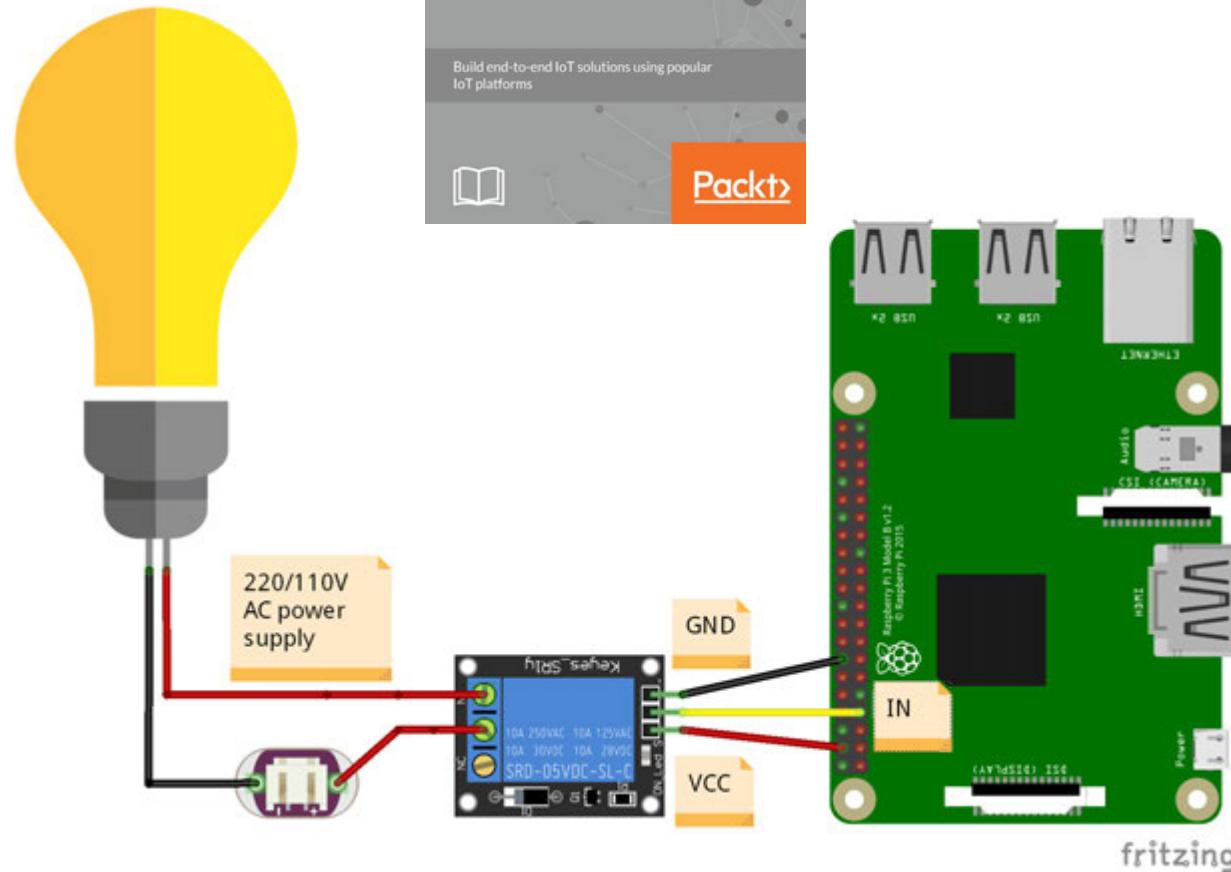
While sensors sense the environment around them, an actuator is responsible for controlling a system by actuating (open/close or on/off). For example, a switch that can be used to turn a light on or off can be controlled via an actuator.

An actuator generally takes a signal which indicates the nature of an operation. For example, if we need to turn a switch on or off, we use a relay.

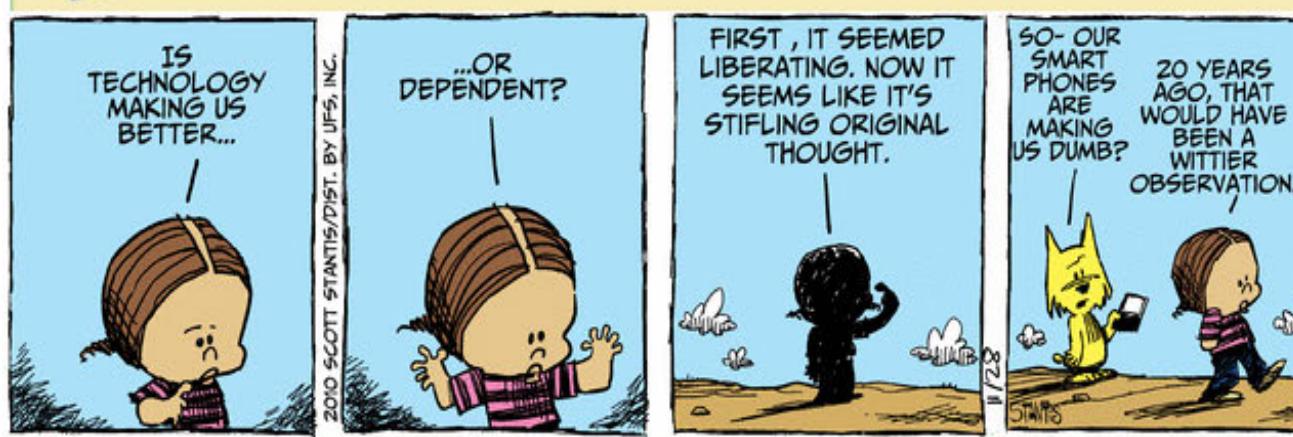
A relay is an electromagnetic switch that takes in the AC mains power supply at one end and a digital signal at other. The general state of the relay is open and that means there is no power supply flowing through from the mains to our load. Once there is a digital high or a digital low signal (depending on the type of relay), the relay moves to a closed state where the load and mains are short and the power flows through.

To get a better understanding of how a relay works and how to use it, take a look at *How to use a relay, the easy way*: <https://www.youtube.com/watch?v=T1fNQjel0js>.

A relay is one of several core actuator components, which are then reused in other actuators such as a solenoid valve.







Creative Commons License Types

	Can someone use it commercially?	Can someone create new versions of it?
Attribution		
Share Alike		 Yup, AND they must license the new work under a Share Alike license.
No Derivatives		
Non-Commercial		 Yup, AND the new work must be non-commercial, but it can be under any non-commercial license.
Non-Commercial Share Alike		 Yup, AND they must license the new work under a Non-Commercial Share Alike license.
Non-Commercial No Derivatives		

SOURCE

<http://www.masternewmedia.org/how-to-publish-a-book-under-a-creative-commons-license/>

This lesson was developed by:

Rob van der Willigen
CMD, Hogeschool Rotterdam
NOV 2021



<http://creativecommons.org/licenses/by-nc-sa/3.0/>

This lesson is licensed under a Creative Commons Attribution-Share-Alike license. You can change it, transmit it, show it to other people. Just always give credit to RFvdW.

<http://creativecommons.org/licenses/>

<http://empoweringthenatives.edublogs.org/2012/03/15/creative-commons-licenses/>

Grading

Data Science for IoT

A hands-on introductory course exploring the Internet of Things from a Data Science point of view.



© 2020

OPDRACHT OMSCHRIJVING

Learning Objective

1. Prepare a (Rapid) prototype for an Internet of Things (IoT) data-pipeline scenario
2. Become familiar with Data Science concepts surrounding IoT applications by building them
3. Learn to design and develop simple lightweight IoT prototypes
4. Learn to report on IoT prototypes & data-pipelines

Choosing IoT data-pipeline scenario

The objective is to choose a topic you care deeply about. A subject or theme that's genuinely interesting to you is of essence to succeed and learn the most. Keep in mind that it should be reasonably well scoped

→→something small, discrete and easy to implement ←←.

Identify a small solvable problem that illustrates your key idea.

Keep it constrained but conceptually interesting.

The lectures outline the scope of Data Science for IoT and relevant literature.

Deliverables:

The course requires the following to be delivered:

- A working prototype that includes software (code), hardware and electronics elements. Code (commented) for the application
- Documentation of the project (contributed to a personal GitHub Repository).
- An online demo (video or IoT Platform implementation) of the completed IoT data-pipeline/project.

Data Science for IoT

A hands-on introductory course exploring the Internet of Things from a Data Science point of view.



© 2020

OPDRACHT OPLEVERING

Submitting your work:

You'll submit your work to the HR LMS as follows:

Documentation should be accessible a personal GitHub Repository submitted in the format of a text file (.txt) containing the URL to your Repository to the lms.hr.nl

INLEVER_MAP_OP2 (vakcode: CMIDAT01K | 2020-2021)

You should document your IoT data-pipeline comprehensively, including:

- ❑ Add/upload code and any supporting documentation and files to your GitHub Repository. Describe the concept and goals (include video or diagrams if needed.)
- ❑ Describe relevant prior projects, approaches or methods you researched that inspired the project. Be thorough and show what informed the project.
- ❑ Process Outline you underwent to reach the outcome (experiments, hacks, tests, refinements, iterations, failures). Describe the realized outcome and how it works.
- ❑ Include supporting images, a video of the working prototype, circuit diagrams, etc. Outline next steps and future directions.

Data Science for IoT

A hands-on introductory course exploring the Internet of Things from a Data Science point of view.



© 2020

Grading

BEOORDELING

Take note:

Main goal is to come up with a compelling IoT data-pipeline scenario.

Provide evidence for a working prototype as proof-of-concept.

A strong grade will result by creating interesting, well-crafted and well documented IoT prototype + data-pipeline. As such, each data driven IoT prototype will be graded as follows:

20% Approach and Topic

Merit, creativity, and context for the outcome/proposal

40% Proof-of-Concept Documentation

Well illustrated with appropriate use of code, video, diagrams, repeatability, etc.

30% Technical Implementation

Quality of data + Executability of Electronics & Code

10% Process

Description and Narration of the Process (ideation, iteration, etc.) and personal/critical reflection on the realized learning objectives.

Anote on documentation:

Each Repository has to be accompanied with a written description.

This is a starting point for your exploration.

Everyone is encouraged to use Jupiter Notebooks.

Data Science for IoT

A hands-on introductory course exploring the Internet of Things from a Data Science point of view.



© 2020

CIJFER

Grading Rubric (Part1)

Merit - Approach and Topic:

How interesting is the IoT concept? Does it represent a data pipeline approach, or an original perspective, based on Data Science principles? Does it deviate from known or standard approaches? Does it use materials or code in an innovative way?

0: incomplete: does not satisfy brief and no originality or creativity demonstrated (e.g. direct replication of prior approach without extension)

1: passing: satisfies minimum requirements of the brief and/or a minor increment over existing work (extension/adaptation of a precedent)

2: good: shows engagement, exploration and insight; uses precedents and/or materials in relatively original ways.

3: excellent: shows deep insight, and significant understanding of the problem; goes beyond the brief and demonstrates significant originality in the ideas and their application; uses precedents and/or materials in unexpected ways; surprising and delightful outcome.

Proof-of-Concept - Documentation:

Is the problem space or scenario clearly explained? How clearly are the key principles and goals of the work articulated? How informed is the work? Does it show connections to Data Science, IoT ideas & research, sensor electronics, IoT frameworks or other elements of the domain?

0: incomplete: No mention of problem space, scenario and/or does not include precedents

1: passing: satisfies requirements, provides core-functionality, is basic in operation; requires improvement

2: good: provides a thoughtful and considered introduction to the work with limited references and limited analysis of past work

3: excellent: provides a thoughtful and considered introduction to the work and supports the context with relevant references and critical analysis of past work.

Data Science for IoT

A hands-on introductory course exploring the Internet of Things from a Data Science point of view.



© 2020

CIJFER

Grading Rubric (Part2)

Technical implementation - Data & Code + Sensory/Actuator Electronics:

How well implemented is the electronics, data & code? Is it well commented, well formatted, well-structured and functioning? Does it show sophisticated approaches? How well composed is it? Does it show technical skill and mastery of programming?

0: incomplete: does not work – electronics, data or code are not included or not error-free

1: passing: satisfies requirements - provides core functionality only, is basic in operation and requires improvement.

2: good: functional; provides reasonably well-structured approach; well commented; and shows technical competence.

3: excellent: provides a considered well organized, well commented and structured data implementation; and/or has implemented complex functionality beyond the brief and/or demonstrated technical skill

Process – Description & Narration:

How well authored, curated, illustrated is the documentation? Is it sufficiently detailed to repeat the outcome? Does it include a critical reflection? Does it communicate the project and its goals succinctly and effectively?

0: incomplete: documentation is missing or doesn't provide any illustration reasonably poor quality, verbose, unclear or shows other communication issues.

1: passing: satisfies minimum requirements - provides the minimum core functionality, is basic in operation and could be improved.

2: good: functional; provides reasonably well-structured approach; well commented; and shows technical competence.

3: excellent: provides a considered well organized, well commented and structured data implementation; and/or has implemented complex functionality beyond the brief and/or demonstrated technical skill