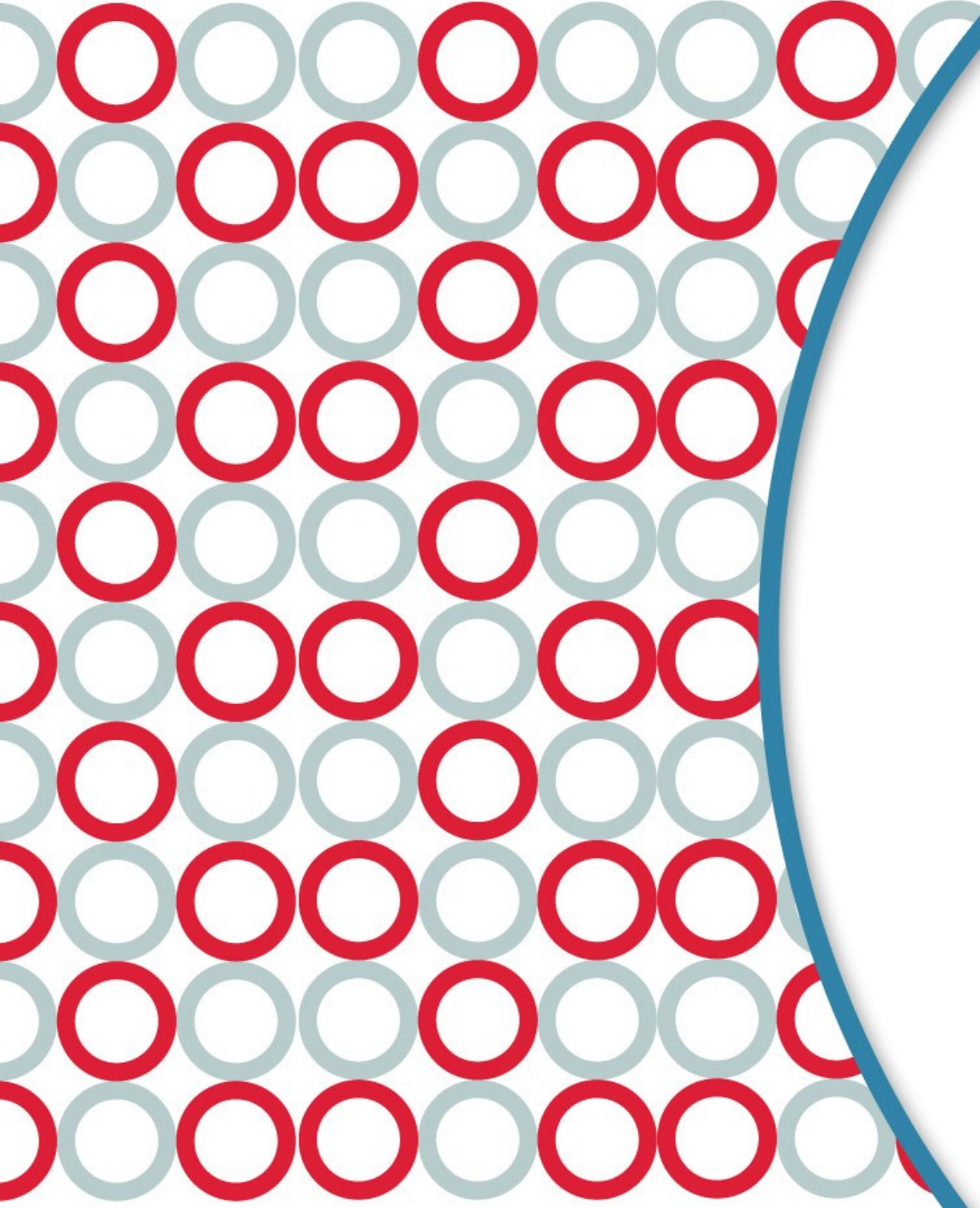


Mar 18, 2020

IoT & Fog Computing

Trayan Iliev

tiliev@iproduct.org
http://iproduct.org

Disclaimer

All information presented in this document and all supplementary materials and programming code represent only my personal opinion and current understanding and has not received any endorsement or approval by IPT - Intellectual Products and Technologies or any third party. It should not be taken as any kind of advice, and should not be used for making any kind of decisions with potential commercial impact.

The information and code presented may be incorrect or incomplete. It is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall the author or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the information, materials or code presented or the use or other dealings with this information or programming code.

Trademarks

OpenFog™ is trademark or registered trademark of OPEN FOG CONSORTIUM, INC.

Oracle®, Java™ and JavaScript™ are trademarks or registered trademarks of Oracle and/or its affiliates.

LEGO® is a registered trademark of LEGO® Group. Programs are not affiliated, sponsored or endorsed by LEGO® Education or LEGO® Group.

Raspberry Pi™ is a trademark of Raspberry Pi Foundation.

Other names may be trademarks of their respective owners.

About me

Trayan Iliev

– CEO of IPT – Intellectual Products & Technologies

– Oracle® certified programmer 15+ Y

– End-to-end reactive fullstack apps with Java, ES6/7, TypeScript, Angular, React and Vue.js

– 12+ years IT trainer

– Voxxed Days, jPrime, jProfessionals, BGOUG, BGJUG, DEV.BG speaker

– Organizer RoboLearn hackathons and IoT enthusiast (http://robolearn.org)

– Lecturer @ Sofia University – courses: Internet of Things (with SAP), Multiagent Systems and Social Robotics

Since 2003: IT Education Evolved

- Spring 5, Webflux, Java SE/Web/EE 7/8/9
- Reactive Robotics & IoT with Reactor / RxJava / Akka
- Node.js + Express + React + Redux + GraphQL
- Angular + TypeScript + Redux (ngrx)
- SOA & REST HATEOAS
- DDD & Reactive Microservices

Fog Computing – between IoT Devices and The Cloud

- Edge, Fog, Mist & Cloud Computing
- Fog domains and fog federation, wireless sensor networks, multi-layer IoT architecture
- Fog computing standards and specifications
- Practical use-case scenarios & advantages of fog
- Fog analytics and intelligence on the edge
- Technologies for distributed asynchronous event processing and analytics in real time
- Lambda architecture – Spark, Storm, Kafka, Apex, Beam, Spring Reactor & WebFlux
- Eclipse IoT platform

We Live in Exponential Time!

"The future is widely misunderstood. Our forebears expected it to be pretty much like their present, which had been pretty much like their past."

–Ray Kurzweil, The Singularity Is Near

We Live in Exponential Time

- "intuitive linear" view of technological progress vs. the "historical exponential" view.
- The Law of Accelerating Returns: Evolution applies positive feedback in that the more capable methods resulting from one stage of evolutionary progress are used to create the next stage. As a result, the rate of progress of an evolutionary process increases exponentially over time.
- A correlate of the above observation is that the "returns" of an evolutionary process (e.g., the speed, cost-effectiveness, or overall "power" of a process) increase exponentially over time.

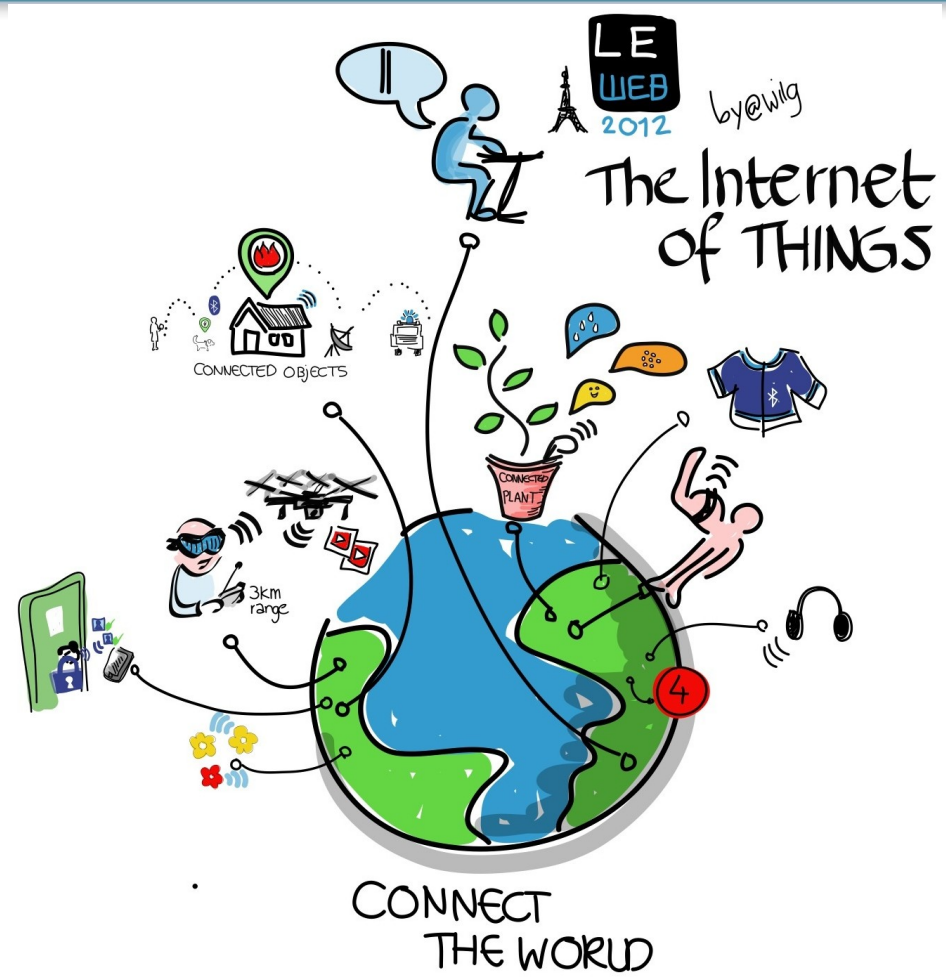
http://www.kurzweilai.net/the-law-of-accelerating-returns, By – Ray Kurzweil, The Singularity Is Near

IoT, IoE, WoT – What It Means?

The **Internet-of-Things (IoT)** is a self-configuring and adaptive network which connects real-world things to the Internet enabling them to communicate with other connected objects leading to the realization of a new range of ubiquitous services.

R. Minerva et al. - Towards a definition of IoT
Technical report, IEEE, 2015

Example: Internet of Things (IoT)



CC BY 2.0, Source: https://www.flickr.com/photos/wilgengebroed/8249565455/

Radar, GPS, lidar for navigation and obstacle avoidance (2007 DARPA Urban Challenge)

Интернет на нещата (IoT, IoE)

Today computers—and, therefore, the Internet—are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes (a petabyte is 1,024 terabytes) of data available on the Internet were first captured and created by human beings—by typing, pressing a record button, taking a digital picture, or scanning a bar code. ... The problem is, people have limited time, attention and accuracy—all of which means they are not very good at capturing data about things in the real world. ... We're physical, and so is our environment ... If we had computers that knew everything there was to know about things ... we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so.

— Kevin Ashton, 'That 'Internet of Things' Thing', RFID Journal, July 22, 2009

Интернет на нещата (IoT, IoE)

Интернет на нещата има потенциала да промени света, така както го направи Интернет. Дори повече.

— Kevin Ashton, 'That 'Internet of Things' Thing', RFID Journal, 2009

- 50 петабайта данни се създават и въвеждат от хора
- Хората имат ограничено време, внимание и точност
- Данни за нещата от реалния свят в реално време
- Проследяване и управление на всичко, намаляване на проблемите, загубите и цената
- Ще знаем кога нещата се нуждаят от замяна, ремонт или извеждане от експлоатация

Интернет на нещата в цифри

- According to Gartner, there will be nearly 26 billion devices on the Internet of Things by 2020.

  [Gartner, 2013-12-12, http://www.gartner.com/newsroom/id/2636073]

- According to ABI Research, more than 30 billion devices will be wirelessly connected to the Internet of Things by 2020 (Internet of Everything)

  [ABI Research, https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne]

- It's expected to be a 19 Trillion USD market

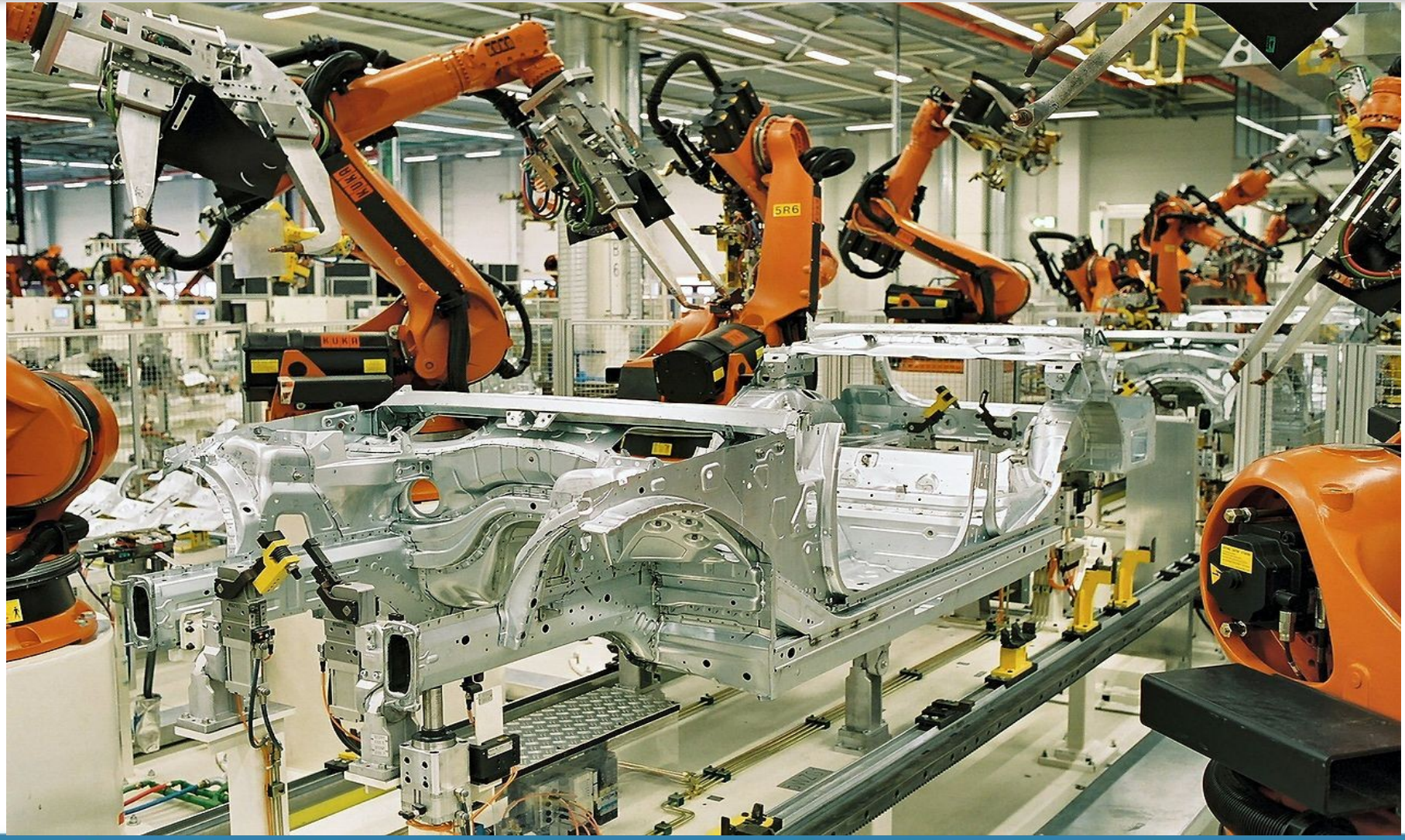
  [John Chambers, Cisco CEO, http://www.bloomberg.com/news/2014-01-08/cisco-ceo-pegs-internet-of-things-as-19-trillion-market.html]

Internet of Things & AI

"Artificial intelligence (AI) is viewed as vital to realising the value of IoT data. Just over a quarter of survey respondents (26%) say that IoT data are pivotal to their current or planned use of AI, with 56% identifying IoT as "one of many important sources" for AI initiatives. Furthermore, 64% agree that "the value of IoT data to my organisation has increased as we have developed our AI capabilities". Many interviewees view IoT and AI as two components of an advanced analytics capability. Reportedly, algorithms trained on data sources including IoT provide the greatest value and competitive differentiation."

[The Economist Intelligence Unit Limited 2020: https://learn.arm.com/economist-iot-report-typ.html]

В промишлеността ...



Industrial IoT – перспективи

“The IIoT market is expected to grow from USD 77.3 billion in 2020 to USD 110.6 billion by 2025, at a CAGR of 7.4% during the forecast period. The growth of the IIoT industry is driven by factors such as technological advancements in semiconductor and electronic devices, increased use of cloud computing platforms, standardization of IPv6, and support from governments of different countries for R&D activities related to IIoT.”

Maarkets and Markets,2019 - Industrial IoT (IIoT) Market: https://www.marketsandmarkets.com/Market-Reports/industrial-internet-of-things-market-129733727.html

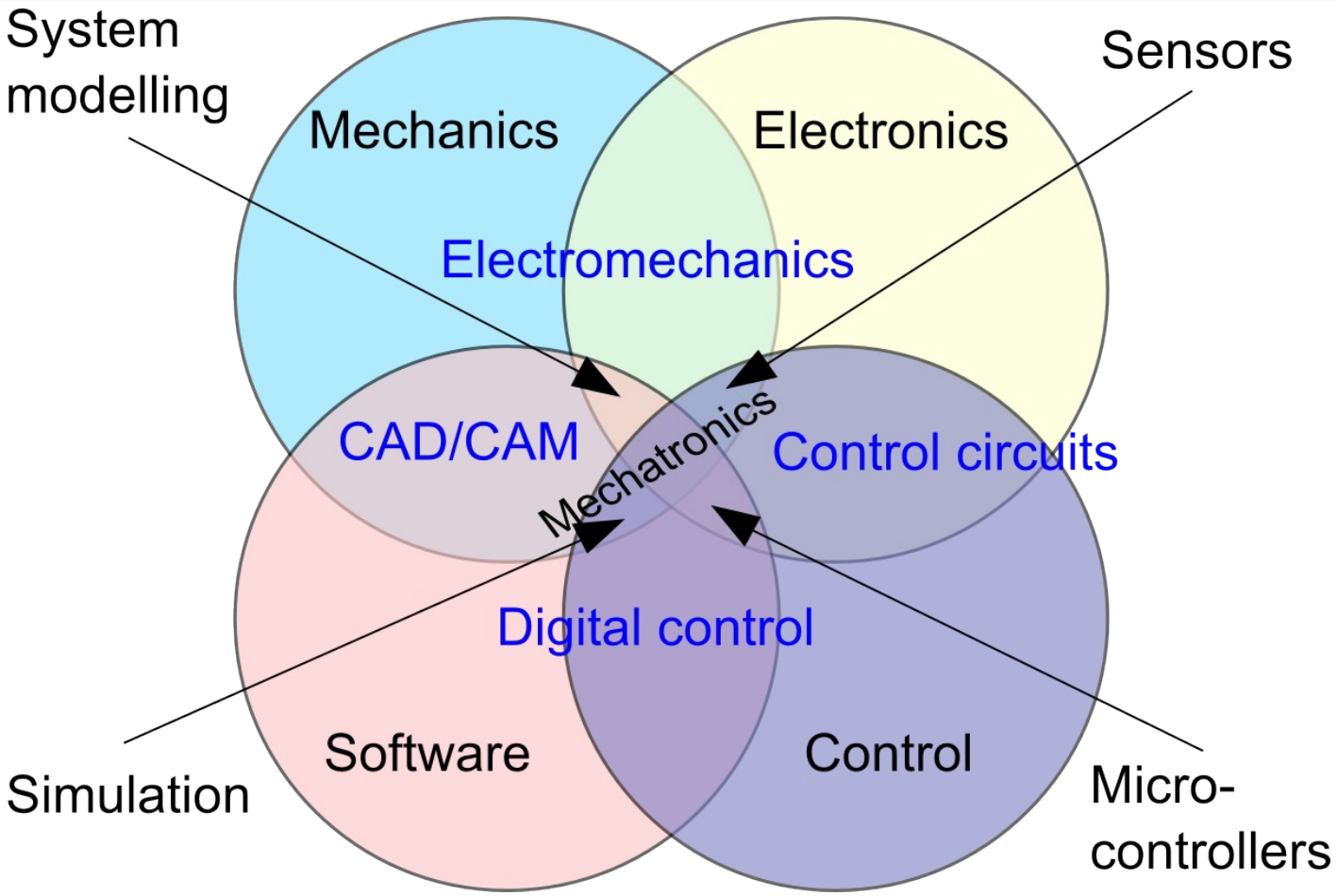
Internet of Things (IoT) – архитектурни изисквания

"Basket of remotes" problem – we'll have hundreds of applications to interface with hundreds of devices that don't share protocols for speaking with one another

[Jean-Louis Gassée, Apple initial alumni team, and BeOS co-founder, http://www.mondaynote.com/2014/01/12/internet-of-things-the-basket-of-remotes-problem/]

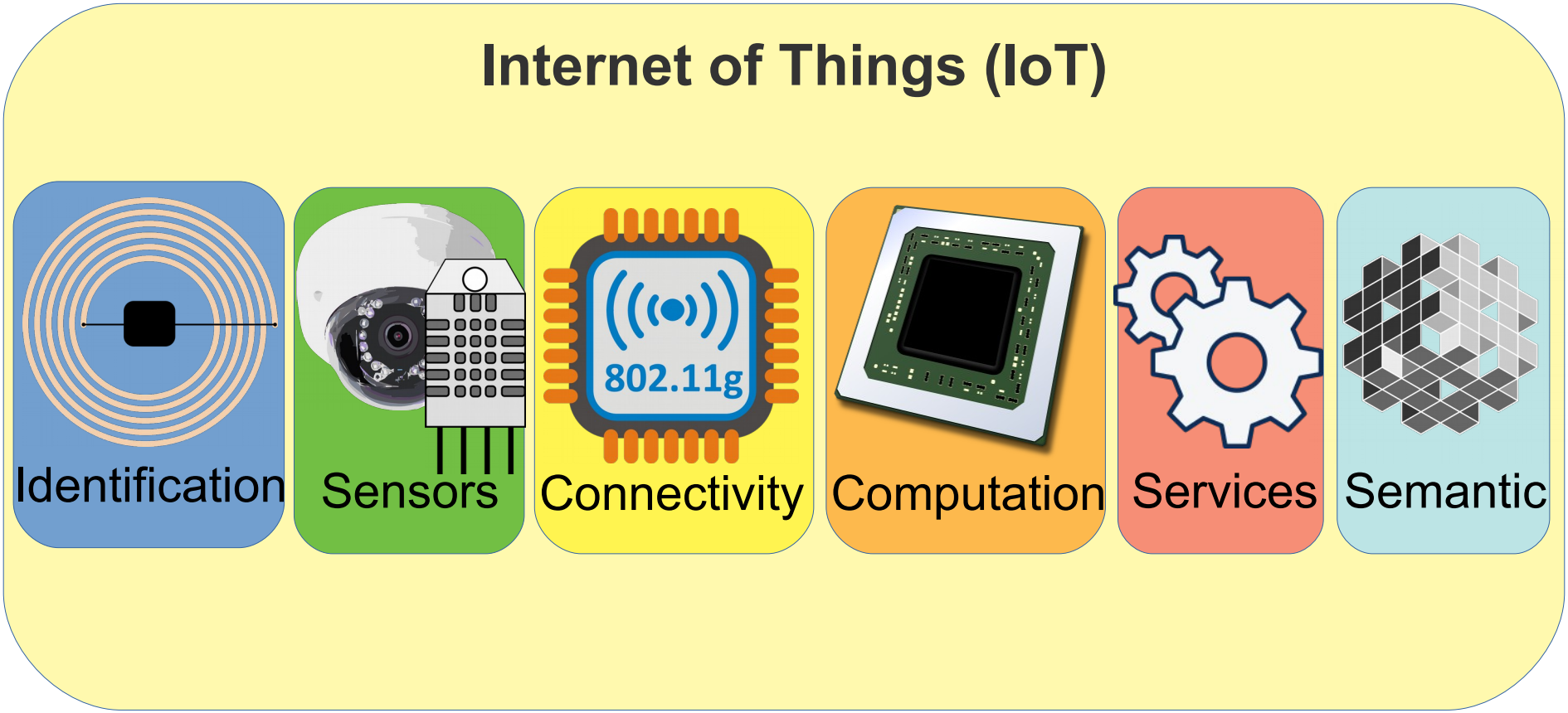
- IoT устройствата трябва да бъдат лесно достъпни за своите потребители и техните агенти
- IoT архитектурите трябва да бъдат конкурентни разпределени, устойчиви на грешки, високо-производителни, еластични и децентрализирани, **динамично** разширяеми и еволюиращи

Engineering, Science & Art

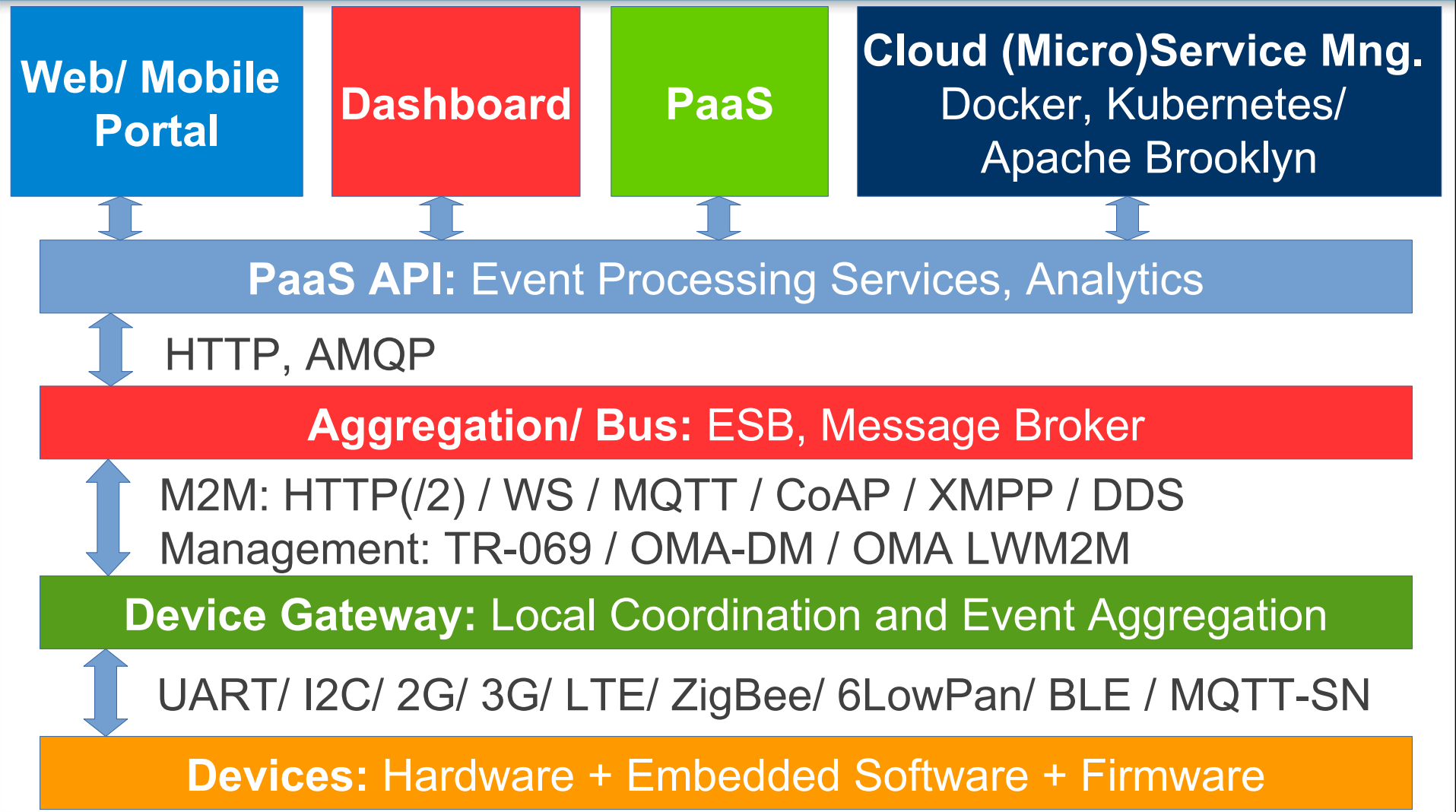


Source: https://commons.wikimedia.org/w/index.php?curid=551256, CC BY-SA 3.0

Key Elements of IoT



IoT Services Architecture



Cloud Computing - Definition

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

National Institute of Standards and Technology

Essential Characterisics of Cloud

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

Cloud Service Models

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

Cloud Deployment Models

- Private cloud
- Community cloud
- Public cloud
- Hybrid cloud

Edge Computing (Mesh Computing)

"… places applications, data and processing at the logical extremes of a network rather than centralizing them. Placing data and data-intensive applications at the Edge reduces the volume and distance that data must be moved."

IoT Guide

http://internetofthingsguide.com/d/edge_computing.htm

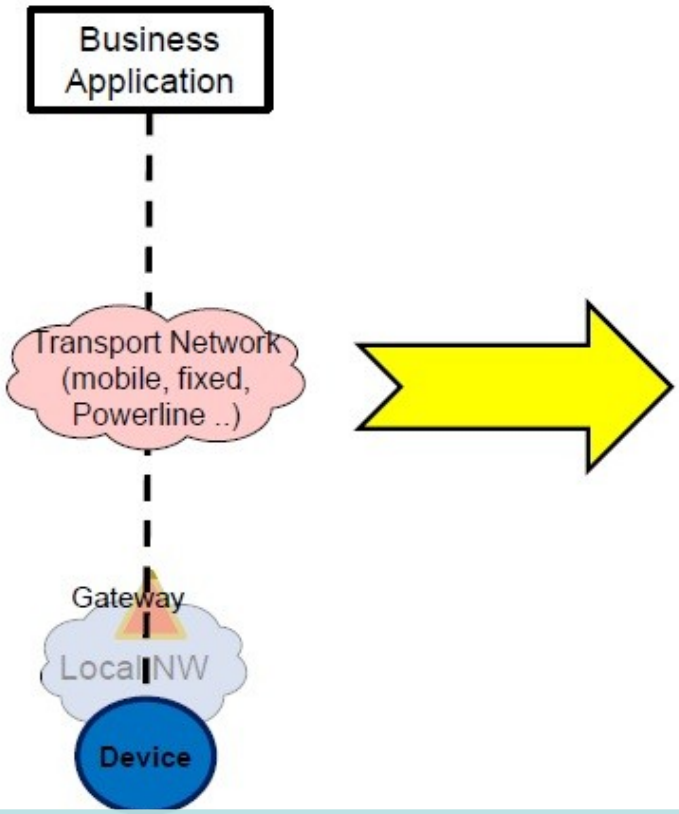
Fog Computing: Definition

A horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to he users along a cloud-to-thing continuum.

OpenFog™ Consortium

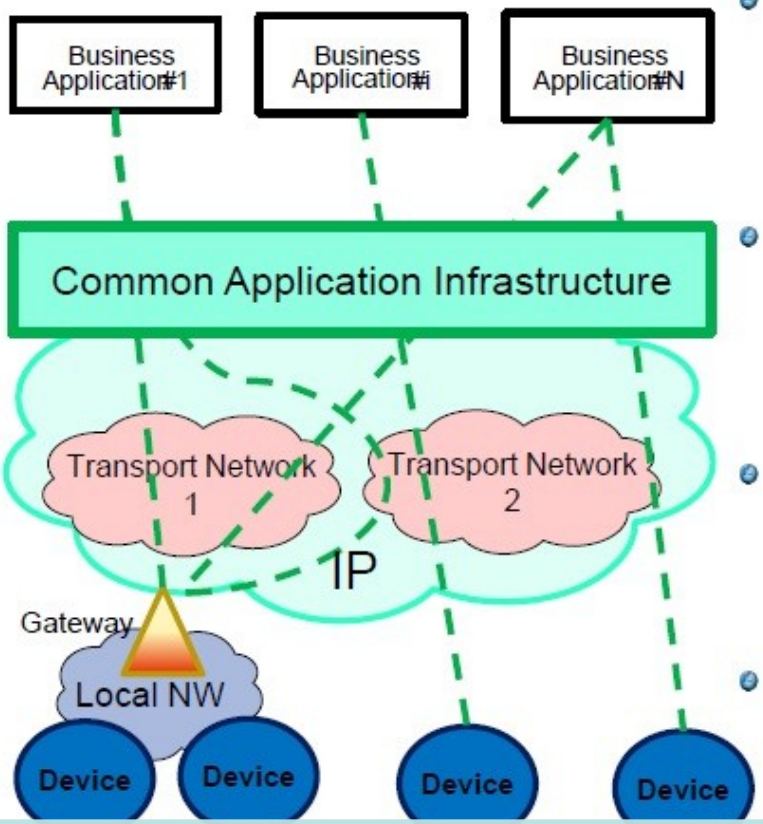
Vertical vs. Horizontal IoT

Pipe (vertical):
1 Application, 1 NW,
1 (or few) type of Device



Horizontal (based on common Layer)

Applications share common infrastructure, environments and network elements



- *M2M Applications providers* run individual M2M services. Customer is Device owner
- *M2M Service provider* hosts several M2M Applications on his Platform.
- *Wide Area Transport Network operator(s)* Customer is the M2M service provider
- *End user* owns / operates the Device or Gateway

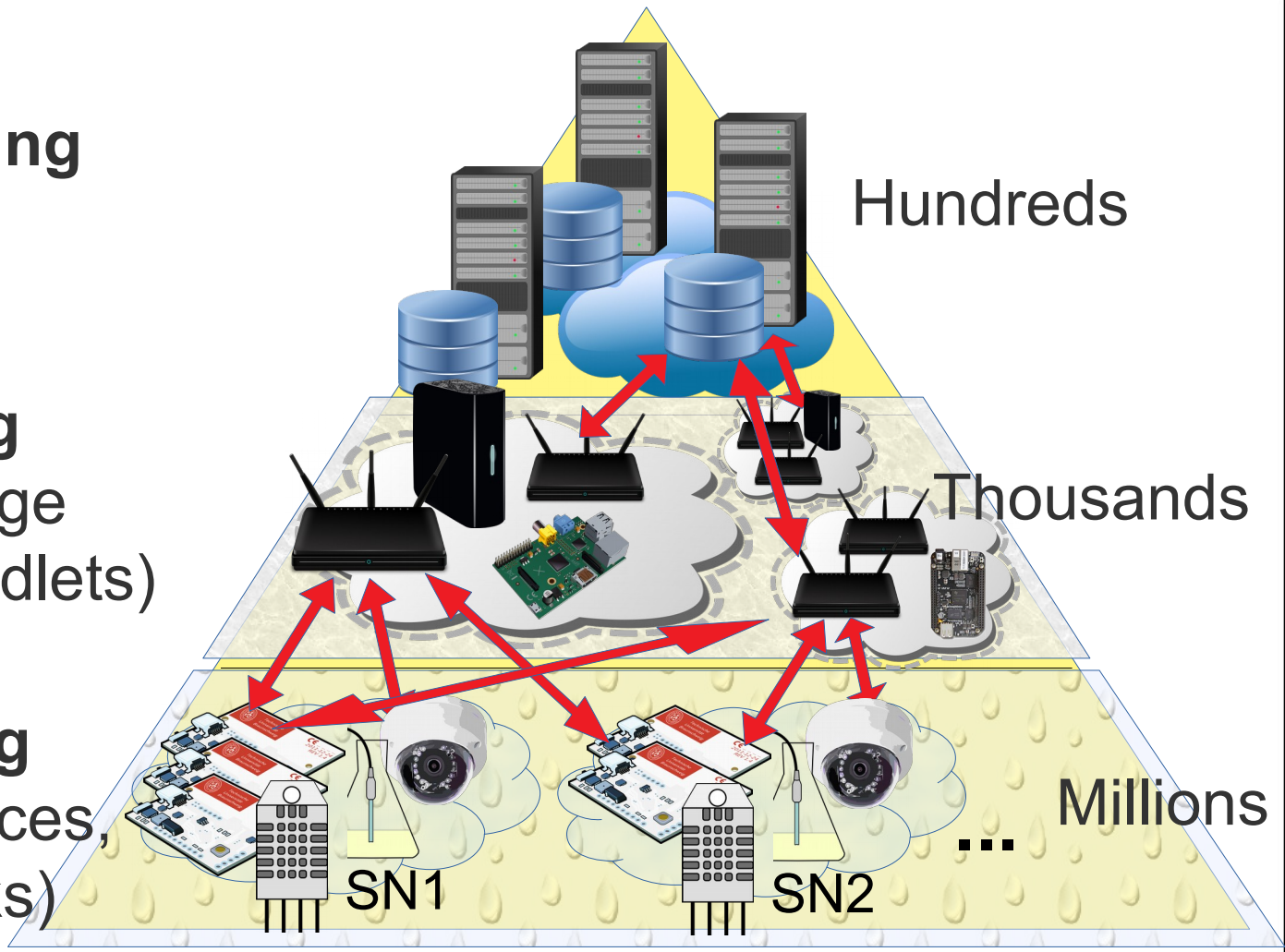
http://www.iot-week.eu/overview_2013/iot-week-2012/programme-1/tuesday-1/presentations/semantic-interoperability/K1_ETSI-M2M-oneM2M-and-the-need-for-semantics-IoTWeek2012.pdf, By Joerg Swetina (NEC)- ETSI M2M / oneM2M and the need for semantics

Cloud, Fog and Mist Computing

Cloud Computing
(Data-centers)

Fog Computing
(Fog Nodes, Edge Gateways, Cloudlets)

Mist Computing
(Smart IoT Devices, Sensor Networks)



Mist Computing - Definition

Mist computing is a lightweight and rudimentary form of computing power that resides directly within the network fabric at the edge of the network, the fog layer closest to the smart end-devices, using microcomputers and microcontrollers to feed into fog computing nodes and potentially onward towards the cloud computing services.

National Institute of Standards and Technology

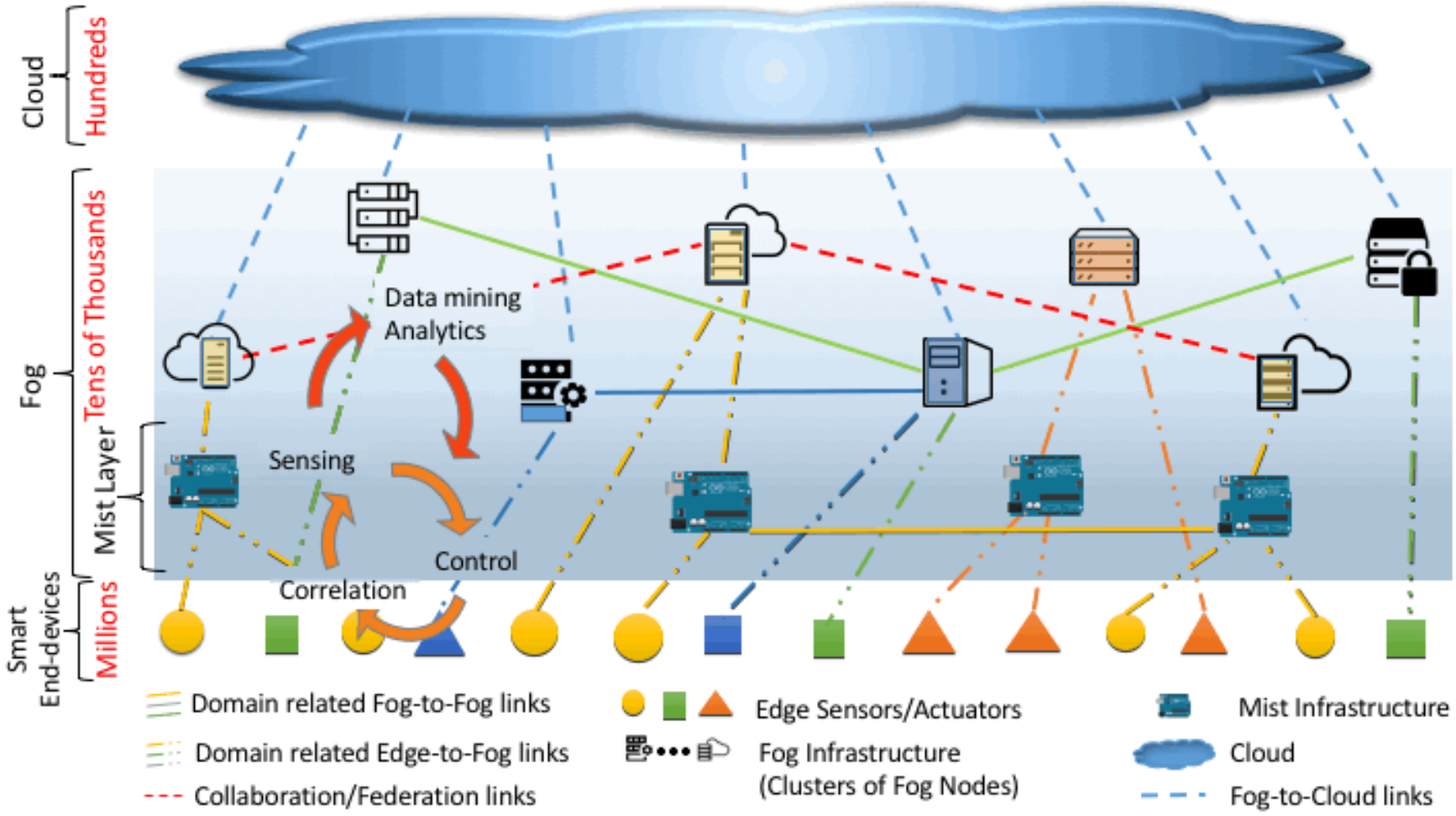
Why Fogging?

- Exponential Data in Realtime – data generated by IoT devices is growing exponentially – especially high bandwidth devices and apps like: LIDARs, 3D cameras, US arrays, gaming, streaming, augmented reality, etc.
- If all data should go to the cloud for analysis and speech/ image/ 3D scene recognition → a recipe for network congestion, high-latency, and self-made DoS.
- Many metrics like performance, efficiency, scalability, latency, security, bandwidth, reliability, privacy could be greatly improved if the processing, analysis, and partially decision making are done locally – close to the source of data.

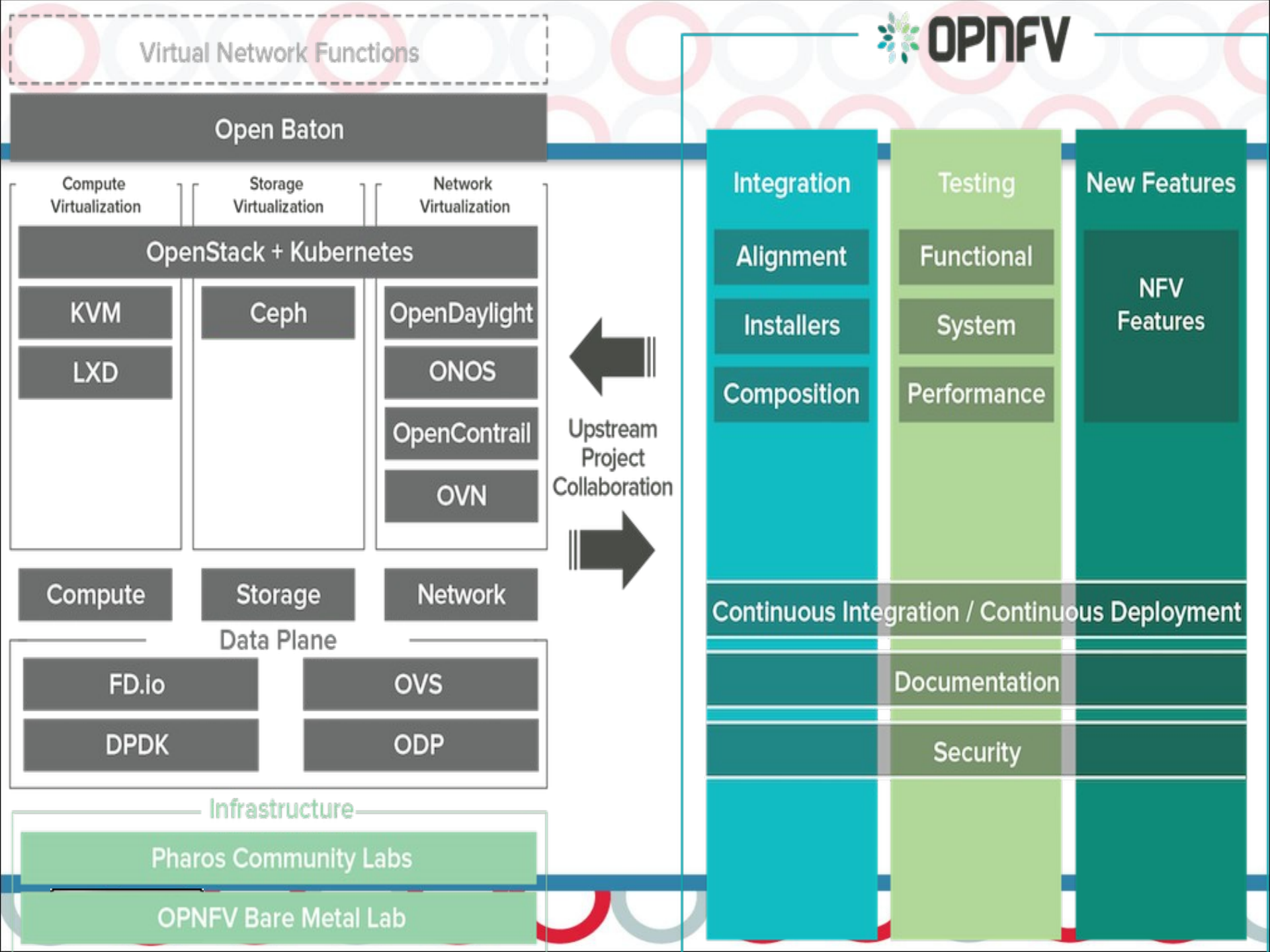
Fog Nodes, Domains & Federation

- Fog Node – The physical and logical network element that implements fog computing services. It is somewhat analogous to a server in cloud computing.
- Fog Domain – seamlessly extends cloud computing to the edge for secure control and management of domain specific hardware, software, and standard compute, storage and network functions.
- Fog Federation – secure control and management of ***multiple fog domain instances***, including edge devices, computes, networking, storage & services in a distributed and consistent manner, providing horizontal expansion of functionality over disperse geolocations.

Fog Comuting – a Cloud-Based Ecosystem for Smart End-Devices



https://csrc.nist.gov/csrc/media/publications/sp/800-191/draft/documents/sp800-191-draft.pdf, National Institute of Standards and Technology (NIST) Definition of Fog Computing – Special Publication Draft 800-191


Virtual Network Functions
Open Baton
Compute Virtualization
Storage Virtualization
Network Virtualization
OpenStack + Kubernetes
KVM
LXD
Ceph
OpenDaylight
ONOS
OpenContrail
OVN
Upstream Project Collaboration
Compute
Storage
Network
Data Plane
FD.io
DPDK
OVS
ODP
Infrastructure
Pharos Community Labs
OPNFV Bare Metal Lab
OPNFV
Integration
Alignment
Installers
Composition
Testing
Functional
System
Performance
New Features
NFV Features
Continuous Integration / Continuous Deployment
Documentation
Security

IoT Networking & Identification

- Time-Sensitive Networking (TSN) – IEEE 802.1
- ISO/IEC 20248 Automatic Identification and Data Capture Techniques – QR Code, RFID, NFC – secure identification of Things.
- GS1's EPC Tag Data Standard (TDS) and EPCglobal Architecture Framework (EPC Network) – allows storing and querying of data related to objects identified with Electronic Product Code numbers.
- IPv6 – allows identification of network interaces and IP package routing, not permanent

Fog Standards and Specifications

- National Institute of Standards and Technology (NIST) *Definition of Fog Computing* – Special Publication Draft 800-191
- OpenFog Consortium – *OpenFog Reference Architecture for Fog Computing* – medium to high level description of system architectures for fog nodes and networks, including of multiple viewpoints (Functional, Deployment), views (Software, System, Node), and Perspectives (cross-cutting concerns). Based on:
- ISO/IEC/IEEE 42010:2011 – *Systems and software engineering — Architecture description*

NIST Definition of Fog Computing

Fog computing is a horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional cloud or data centers. This paradigm supports vertically-isolated, latency-sensitive applications by providing ubiquitous, scalable, layered, federated, and distributed computing, storage, and network connectivity.

National Institute of Standards and Technology

Fog Computing Characteristics - I

- Contextual location awareness, and low latency
- Geographical distribution & distributed deployment
- Large-scale sensor networks – environment monitoring, Smart Grid → distributed computing and storage
- Very large number of nodes – geo-distributed
- Support for mobility – mobile devices: smartphones, etc.
- Real-time interactions – streaming, lambda architecture
- Predominance of wireless IoT access: analytics, compute
- Heterogeneity – deployed in wide variety of environments

Fog Computing Characteristics - II

- Interoperability and federation - seamless support of certain services (e.g. real-time streaming) requires the cooperation of different providers.
- Support for real-time analytics and interplay with the Cloud -while Fog nodes provide localization, therefore enabling low latency and context awareness, the Cloud provides global centralization. Many applications require both Fog localization and Cloud globalization, particularly for analytics and Big Data.
- Fog is particularly well suited to real-time streaming analytics as opposed to historical, Big Data batch analytics that is normally carried out in a data center.

Fog as a Service (FaaS)

- Pay-as-you-go model
- Includes:
  - Software as a Service (SaaS)
  - Platform as a Service (PaaS)
  - Infrastructure as a Service (IaaS)

OpenFog Consortium

- Founded in November 2015 by ARM, Cisco, Dell, Intel, Microsoft and Princeton University
- Open participation from across industry, academia and non-profit organizations that have an interest in the emerging IoT landscape
- Defines open architectural framework enabling IoT industry convergence and game-changing innovation through fog computing
- Efforts complementary to other initiatives like: Industrial Internet Consortium (IIC), ETSI-MEC (Mobile Edge Computing), OPC-UA, Open Connectivity Foundation (OCF), OpenNFV, etc.

Example Use Cases

- Traffic Control and Smart Cars – Vehicle-to-X – including: Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Manufacturer (V2M), multiple public and private fog and cloud networks.
- Security Cameras and Surveillance – in smart cities / homes, retail stores, factories, public transportation, airports – terabytes per day by single camera → local processing, anomaly detection, and decision making.
- Smart cities- parking, shopping, healthcare, infrastructure
- Smart Buildings – HVAC, lighting, doors, parking, security, elevtors, support for smartphones, tablets, etc.

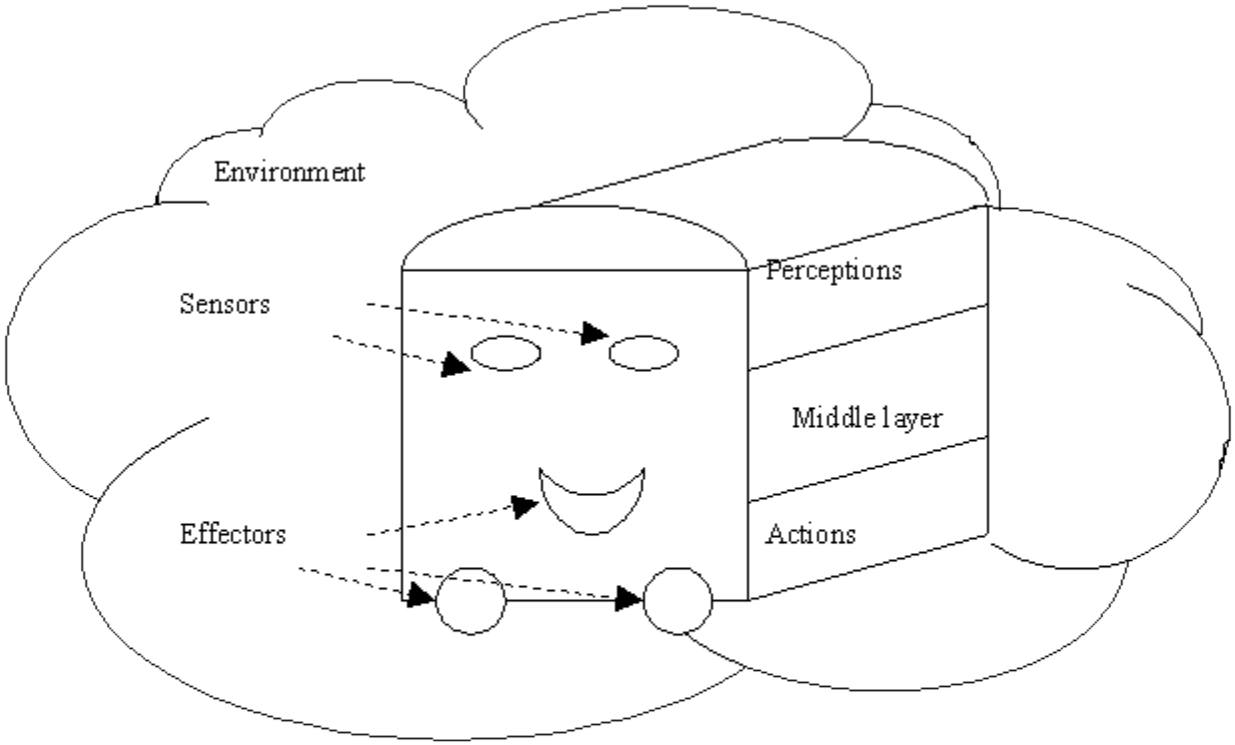
Core Principles

- Scalable
- Agile and open
- Secure
- Autonomous
- Flexible & programmable
- Highly available
- Reliable
- Remotely serviceable
- Hierarchically structured based on business needs

Fog Analytics

- Descriptive Analytics
- Reactive (Diagnostic) Analytics
- Predictive Analytics
- Prescriptive Analytics

Intelligent Agents - Looks Similar?



File Edit View History Bookmarks Tools Help

EV3JLib | WPA sup... | wpa_sup... | WPA sup... | dhcpcd - ... | Wireless ... | Raspberry Pi •... | Raspberry Pi •... | Akka

akka.io

akka

akka

Documentation FAQ Download Mailing List Code Commercial Support

Build powerful concurrent & distributed applications **more easily.**

Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM.

Simple Concurrency & Distribution

Asynchronous and Distributed by design. High-level abstractions like Actors, Futures and STM.

Resilient by Design

Write systems that self-heal. Remote and/or local supervisor hierarchies.

High Performance

50 million msg/sec on a single machine. Small memory footprint; ~2.5 million actors per GB of heap.

Elastic & Decentralized

Adaptive load balancing, routing, partitioning and configuration-driven remoting.

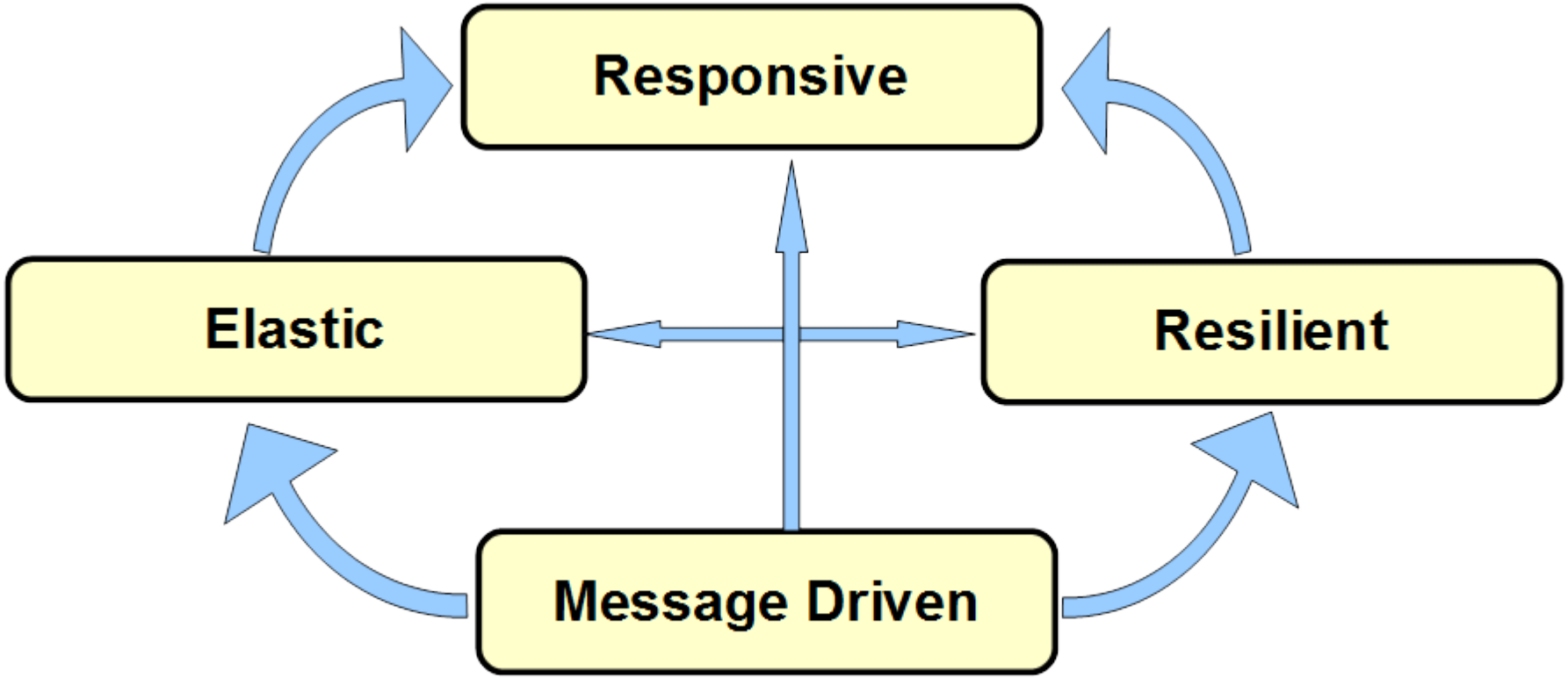
Extensible

Use Akka Extensions to adapt Akka to fit your needs.

EN 6:42 AM 3/4/2015

Reactive Manifesto

[http://www.reactivemanifesto.org]



Scalable, Massively Concurrent

- **Message Driven** – asynchronous message-passing allows to establish a boundary between components that ensures loose coupling, isolation, location transparency, and provides the means to delegate errors as messages [Reactive Manifesto].
- The main idea is to separate concurrent producer and consumer workers by using **message queues.**
- **Message queues** can be **unbounded** or **bounded** (limited max number of messages)
- **Unbounded** message queues can present memory allocation problem in case the producers outrun the consumers for a long period → **OutOfMemoryError**

Data / Event / Message Streams

"Conceptually, a stream is a (potentially never-ending) flow of data records, and a transformation is an operation that takes one or more streams as input, and produces one or more output streams as a result."

Apache Flink: Dataflow Programming Model

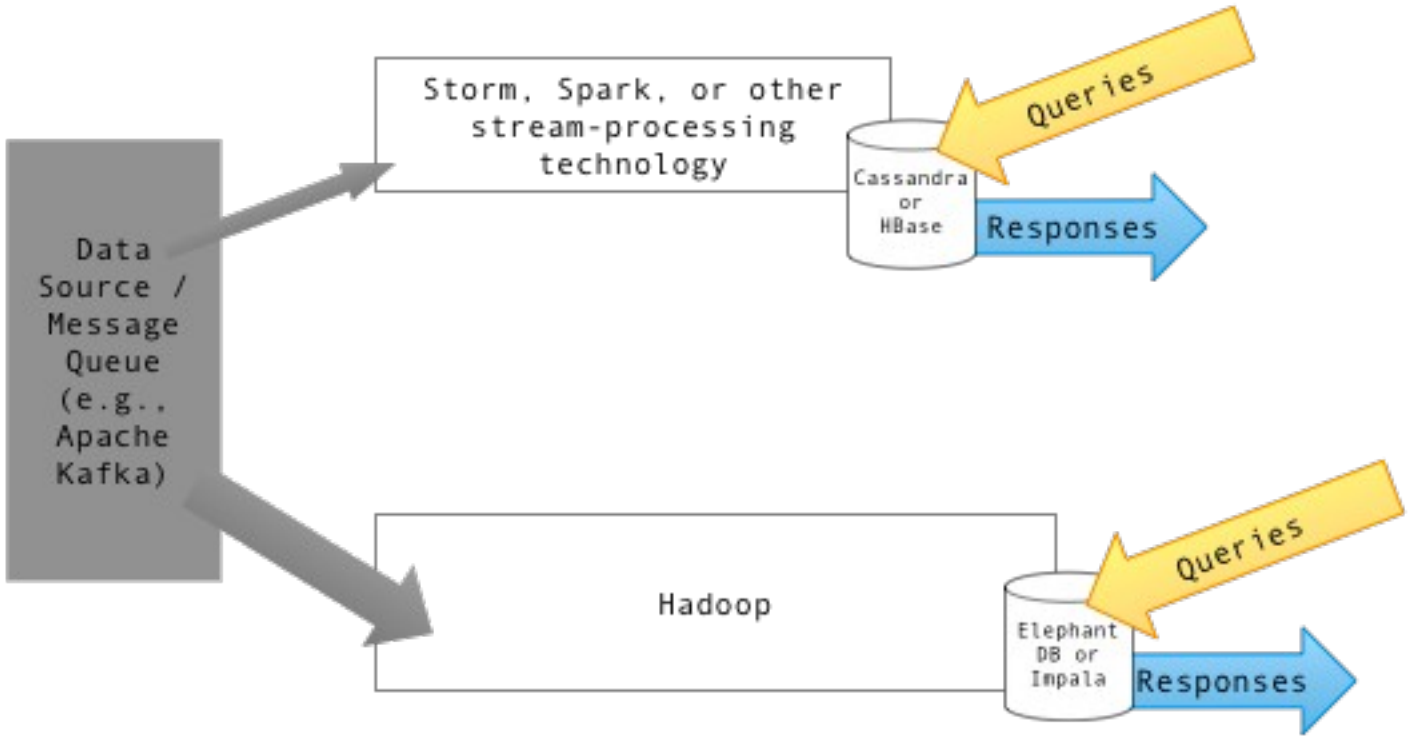
Data Stream Programming

The idea of abstracting logic from execution is hardly new -- it was the dream of SOA. And the recent emergence of microservices and containers shows that the dream still lives on.

For developers, the question is whether they want to learn yet one more layer of abstraction to their coding. On one hand, there's the elusive promise of a common API to streaming engines that in theory should let you mix and match, or swap in and swap out.

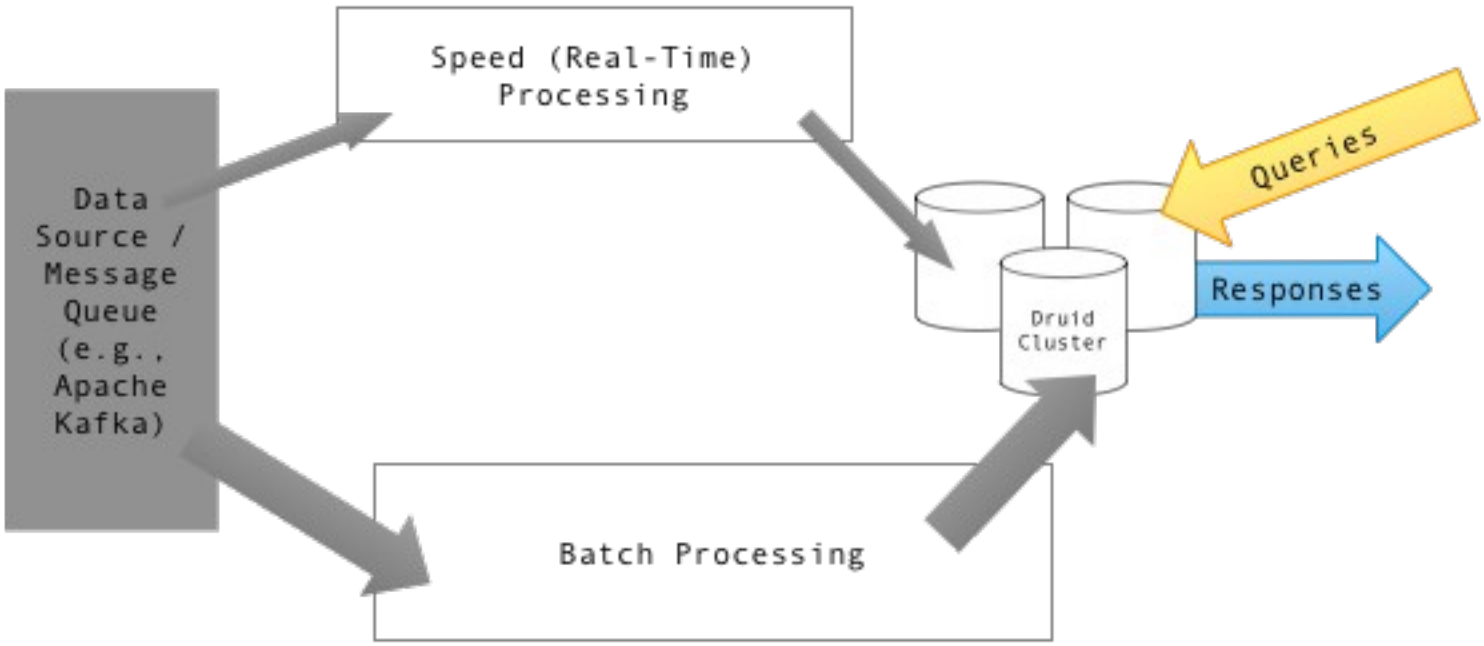
Tony Baer (Ovum) @ ZDNet - Apache Beam and Spark: New coopetition for squashing the Lambda Architecture?

Lambda Architecture - I



https://commons.wikimedia.org/w/index.php?curid=34963986, By Textractor - Own work, CC BY-SA 4

Lambda Architecture - II

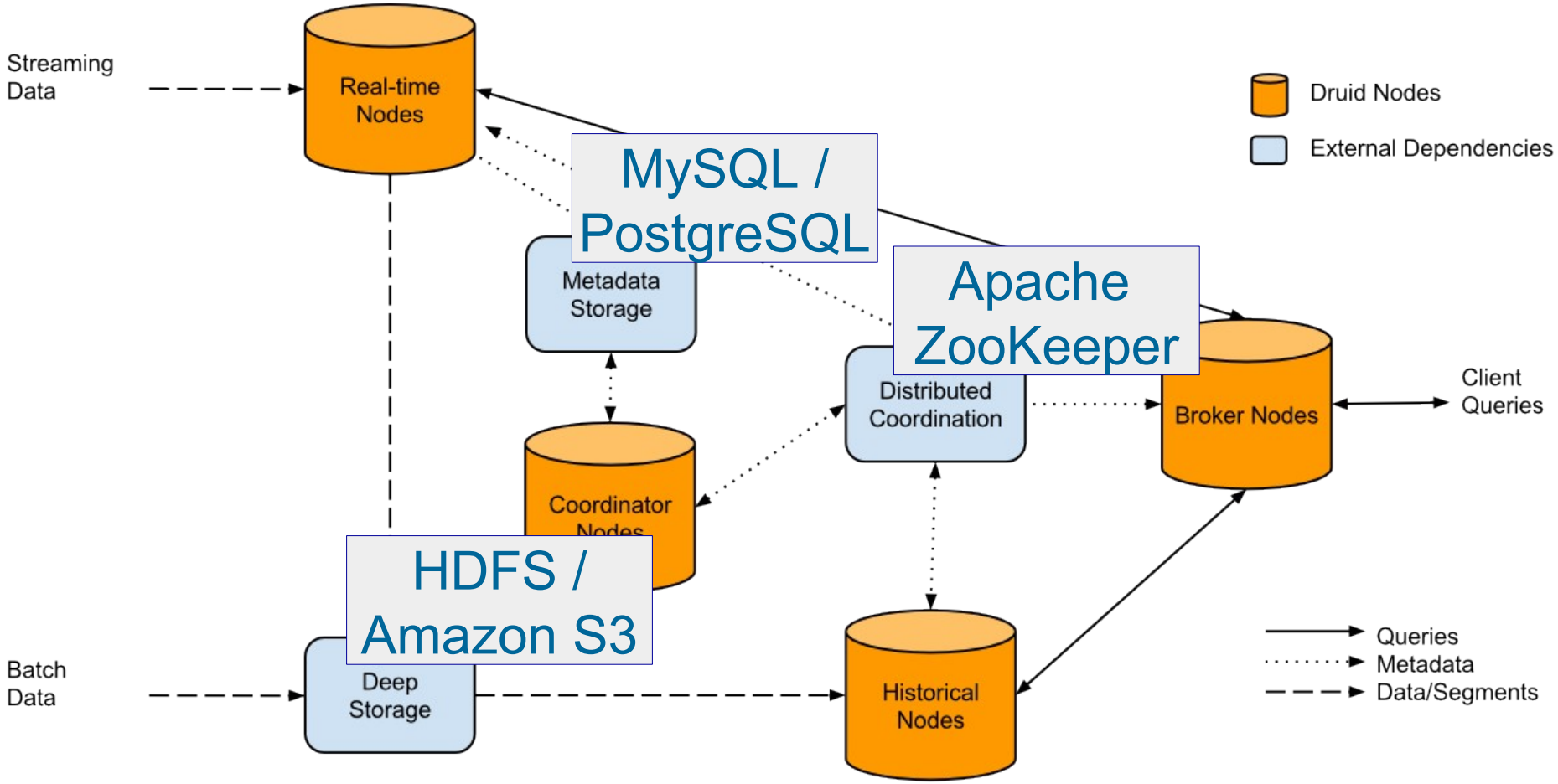


https://commons.wikimedia.org/w/index.php?curid=34963987, By Textractor - Own work, CC BY-SA 4

Lambda Architecture - III

- Data-processing architecture designed to handle massive quantities of data by using both *batch-* and *stream-processing* methods
- Balances *latency, throughput, fault-tolerance, big data, real-time analytics*, mitigates the latencies of map-reduce
- Data model with an append-only, *immutable data* source that serves as a system of record
- Ingesting and processing *timestamped events* that are appended to existing events. State is determined from the *natural time-based ordering* of the data.

Druid Distributed Data Store (Java)



https://commons.wikimedia.org/w/index.php?curid=33899448 By Fangjin Yang - sent to me personally, GFDL

Lambda Architecture: Projects - I

- Apache Spark is an open-source cluster-computing framework. Spark Streaming, Spark Mllib

- Apache Storm is a distributed stream processing – streams DAG

- Apache Apex™ unified stream and batch processing engine.

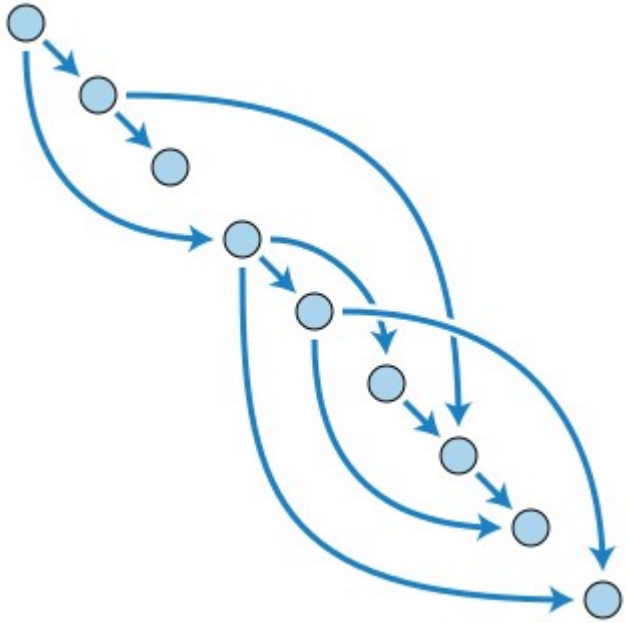
Lambda Architecture: Projects - II

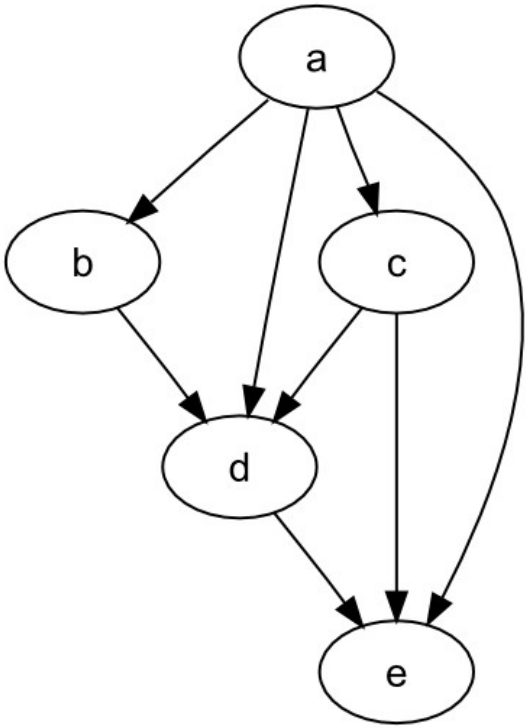
- Apache Flink - open source stream processing framework – Java, Scala

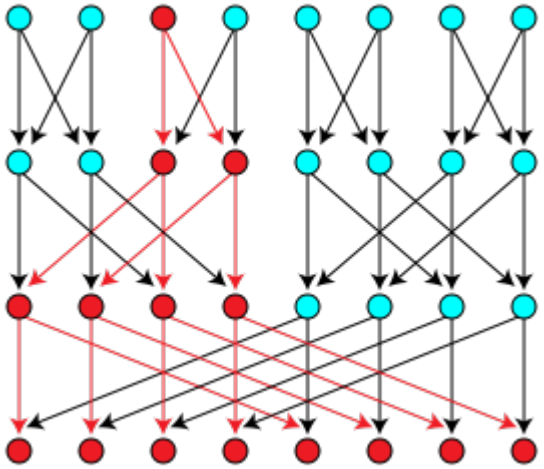
- Apache Kafka - open-source stream processing (Kafka Streams), real-time, low-latency, high-throughput, massively scalable pub/sub

- Apache Beam – unified batch and streaming, portable, extensible

Direct Acyclic Graphs - DAG

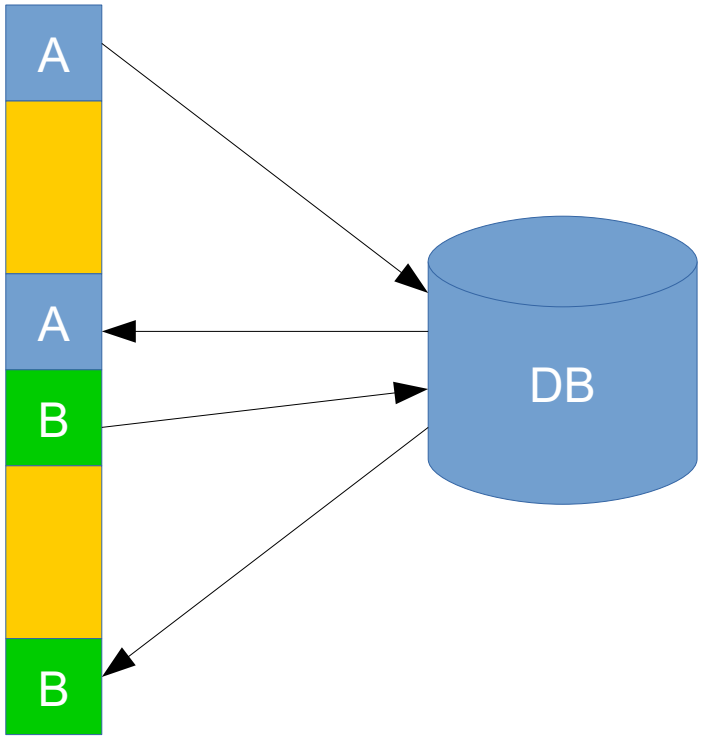




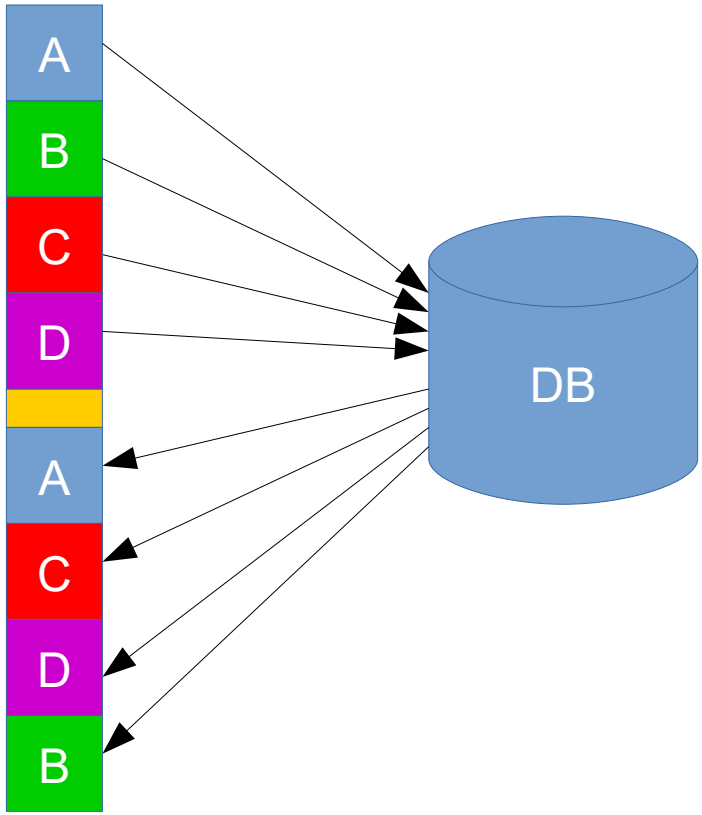


Synchronous vs. Asynchronous IO

Synchronous



Asynchronous



Reactive Programming

- Microsoft® opens source polyglot project ReactiveX (Reactive Extensions) [http://reactivex.io]:

Rx = Observables + LINQ + Schedulers :)

Java: RxJava, JavaScript: RxJS, C#: Rx.NET, Scala: RxScala, Clojure: RxClojure, C++: RxCpp, Ruby: Rx.rb, Python: RxPY, Groovy: RxGroovy, JRuby: RxJRuby, Kotlin: RxKotlin ...

- Reactive Streams Specification [http://www.reactive-streams.org/] used by:
- (Spring) Project Reactor [http://projectreactor.io/]
- Actor Model – Akka (Java, Scala) [http://akka.io/]

Reactive Streams Spec.

- Reactive Streams – provides standard for asynchronous stream processing with non-blocking back pressure.
- Minimal set of interfaces, methods and protocols for asynchronous data streams
- April 30, 2015: has been released version 1.0.0 of Reactive Streams for the JVM (Java API, Specification, TCK and implementation examples)
- Java 9: java.util.concurrent.Flow

Reactive Streams Spec.

- **Publisher** – provider of potentially unbounded number of sequenced elements, according to Subscriber(s) demand.

Publisher.subscribe(Subscriber) => **onSubscribe onNext* (onError | onComplete)?**

- **Subscriber** – calls **Subscription.request(long)** to receive notifications
- **Subscription** – one-to-one **Subscriber** ↔ **Publisher**, request data and cancel demand (allow cleanup).
- **Processor** = **Subscriber** + **Publisher**

FRP = Async Data Streams

- **FRP** is asynchronous data-flow programming using the building blocks of functional programming (e.g. map, reduce, filter) and explicitly modeling time
- Used for GUIs, robotics, and music. Example (RxJava):

```
Observable.from(
     new String[]{"Reactive", "Extensions", "Java"})
    .take(2).map(s -> s + " : on " + new Date())
    .subscribe(s -> System.out.println(s));
```

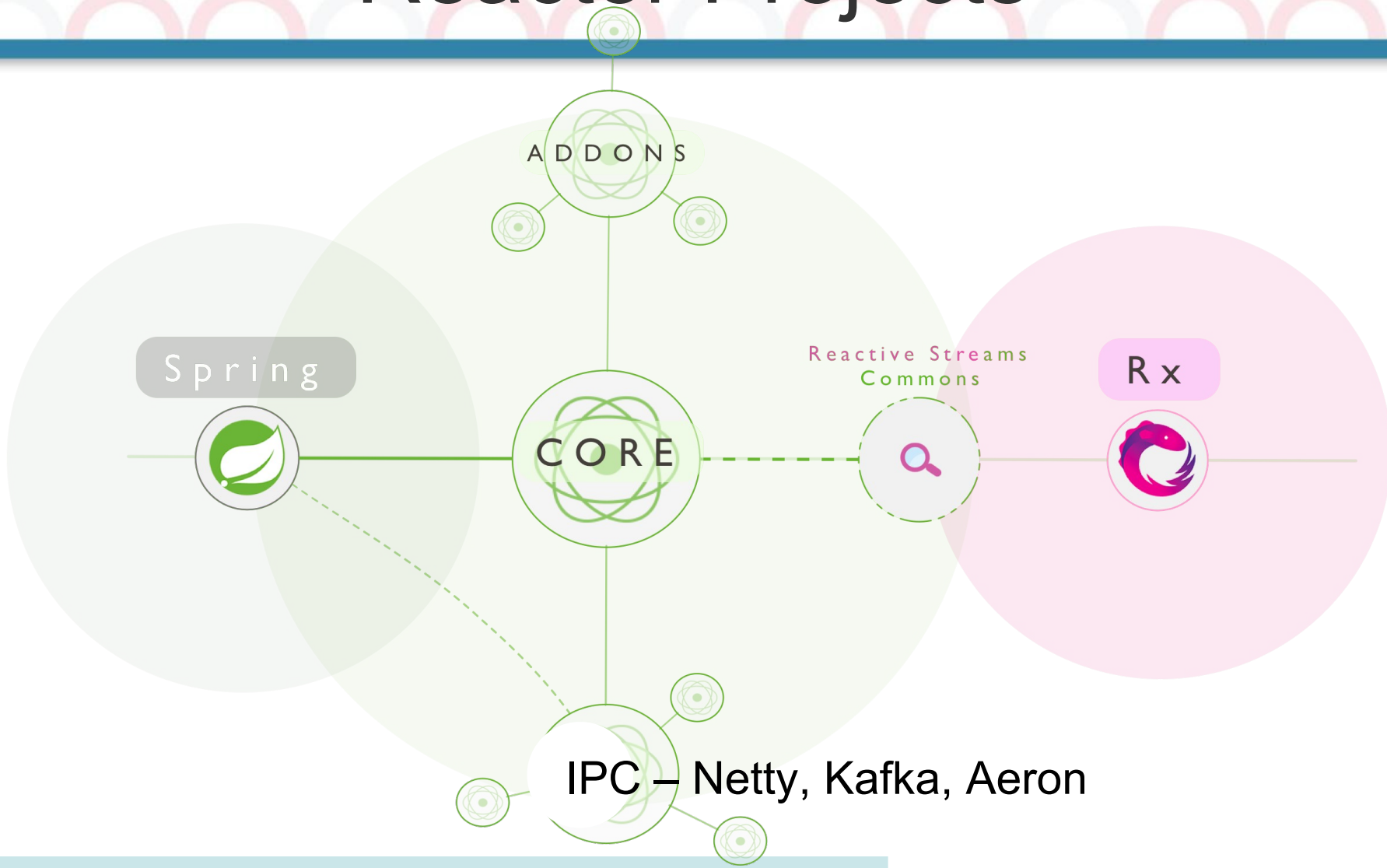
Result:

```
Reactive : on Wed Jun 17 21:54:02 GMT+02:00 2015
Extensions : on Wed Jun 17 21:54:02 GMT+02:00 2015
```

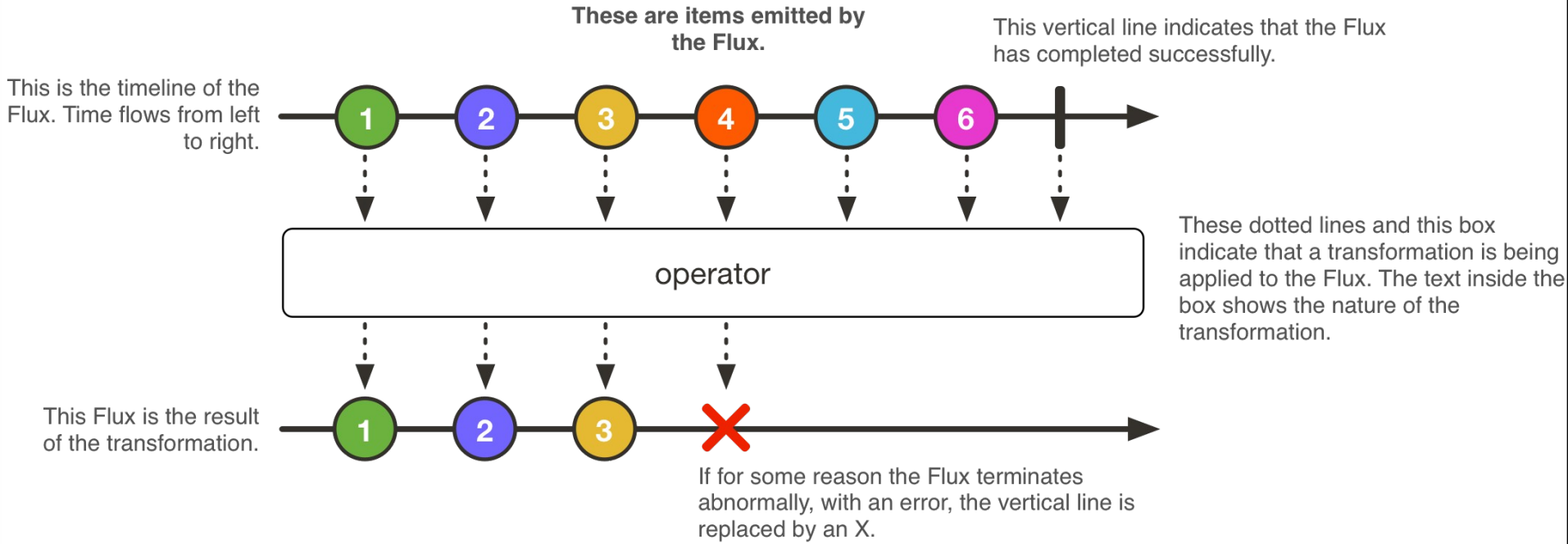
Project Reactor

- Reactor project allows building high-performance (low latency high throughput) non-blocking asynchronous applications on JVM.
- Reactor is designed to be extraordinarily fast and can sustain throughput rates on order of 10's of millions of operations per second.
- Reactor has powerful API for declaring data transformations and functional composition.
- Makes use of the concept of Mechanical Sympathy built on top of Disruptor / RingBuffer.

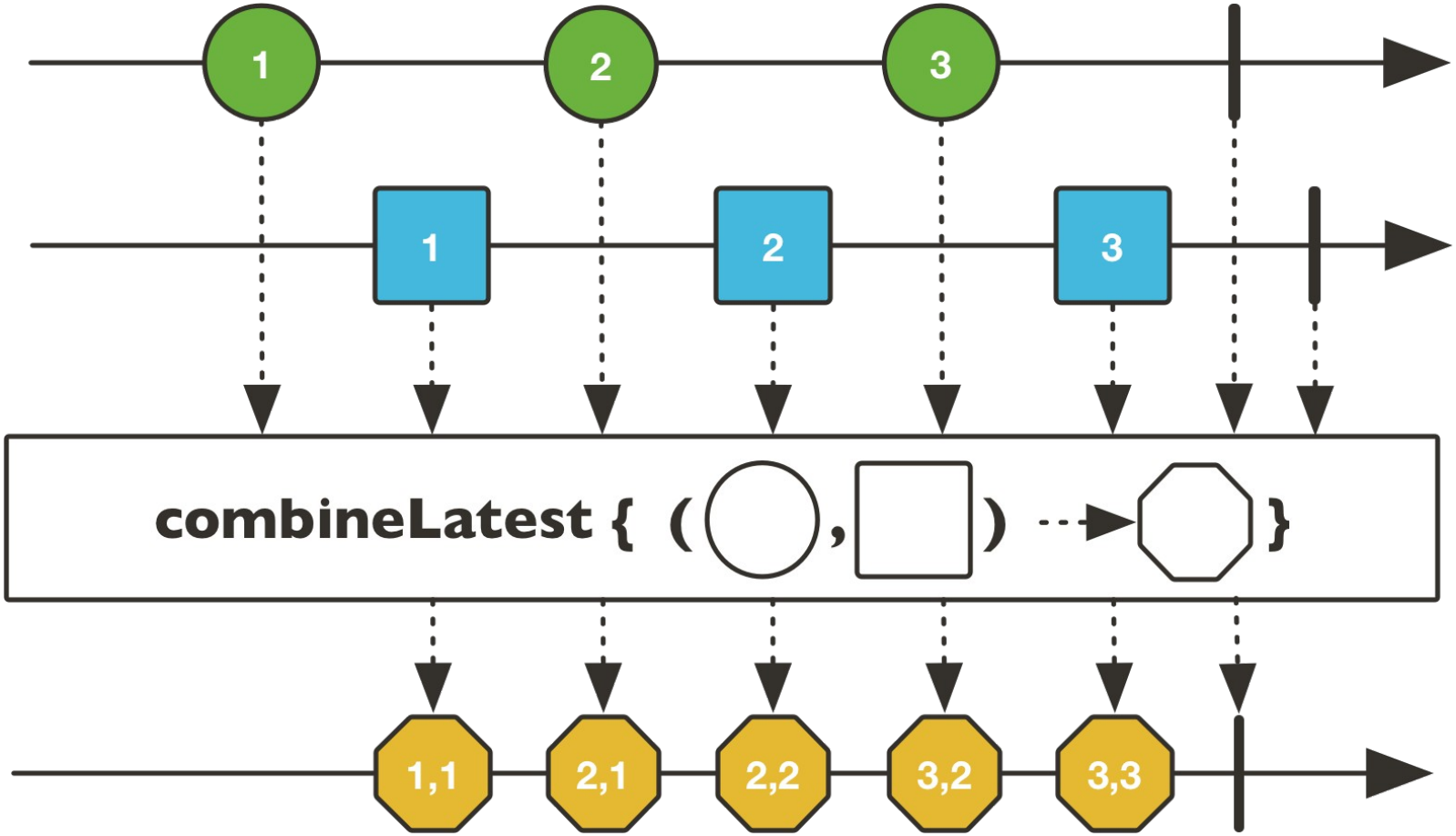
Reactor Projects



Reactor Flux

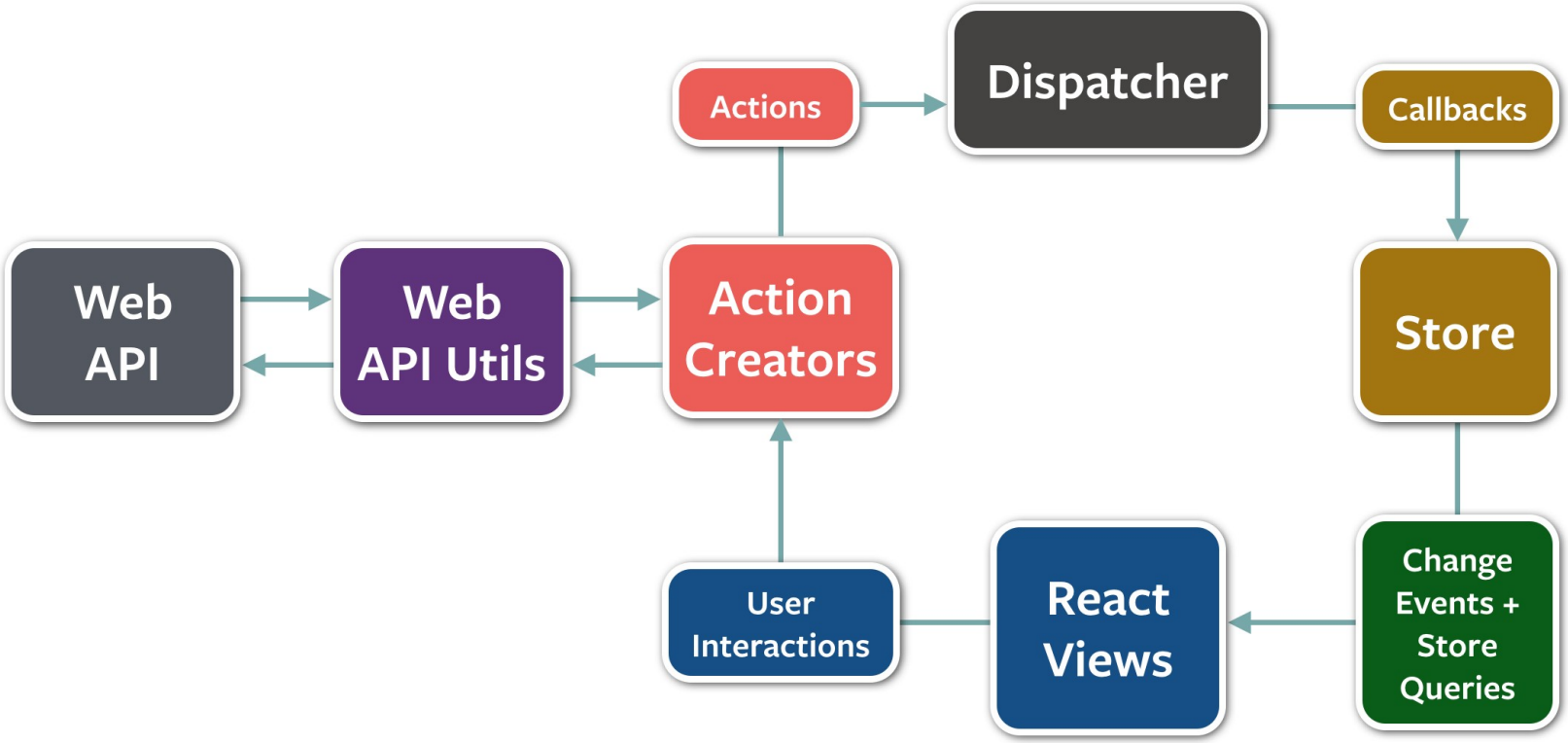


Example: Flux.combineLatest()



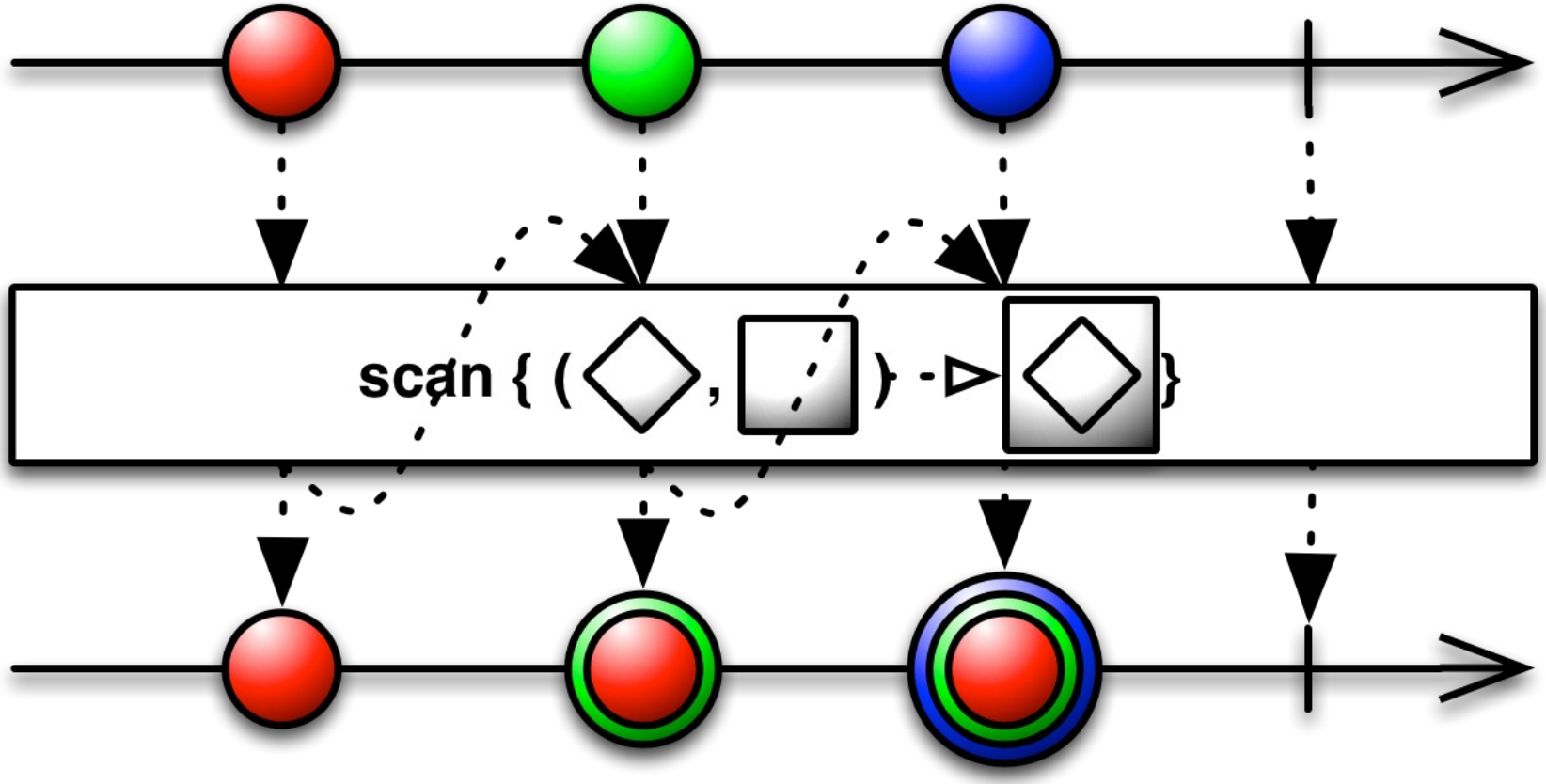
https://projectreactor.io/core/docs/api/, Apache Software License 2.0

Flux & Redux Design Patterns



Source: Flux in GitHub, https://github.com/facebook/flux, License: BSD 3-clause "New" License

Redux == Rx Scan Opearator

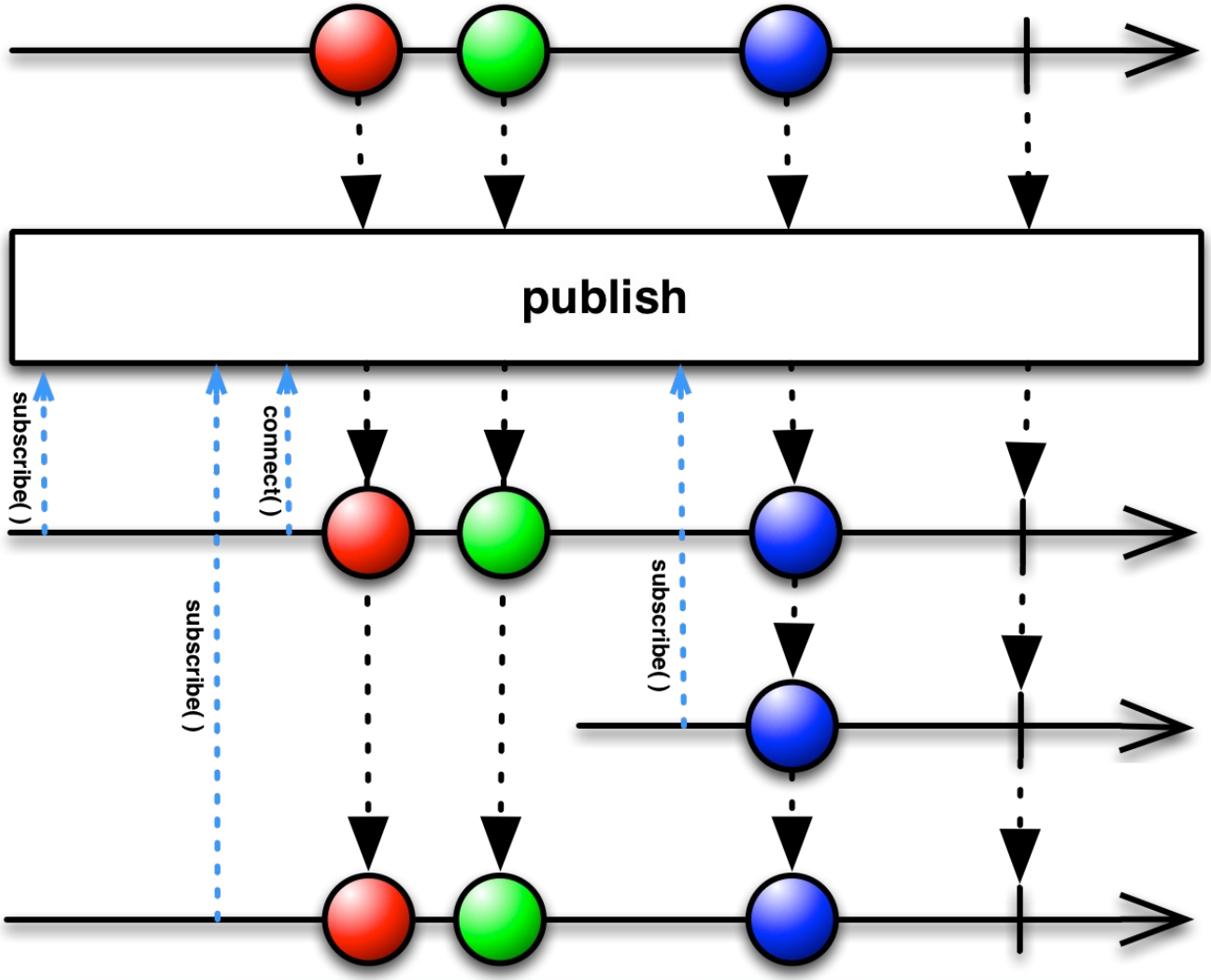


Source: RxJava 2 API documentation, http://reactivex.io/RxJava/2.x/javadoc/

Hot and Cold Event Streams

- PULL-based (Cold Event Streams) – Cold streams (e.g. RxJava Observable / Flowable or Reactor Flow / Mono) are streams that run their sequence when and if they are subscribed to. They present the sequence from the start to each subscriber.
- PUSH-based (Hot Event Streams) – emit values independent of individual subscriptions. They have their own timeline and events occur whether someone is listening or not. When subscription is made observer receives current events as they happen.
  Example: mouse events

Converting Cold to Hot Stream



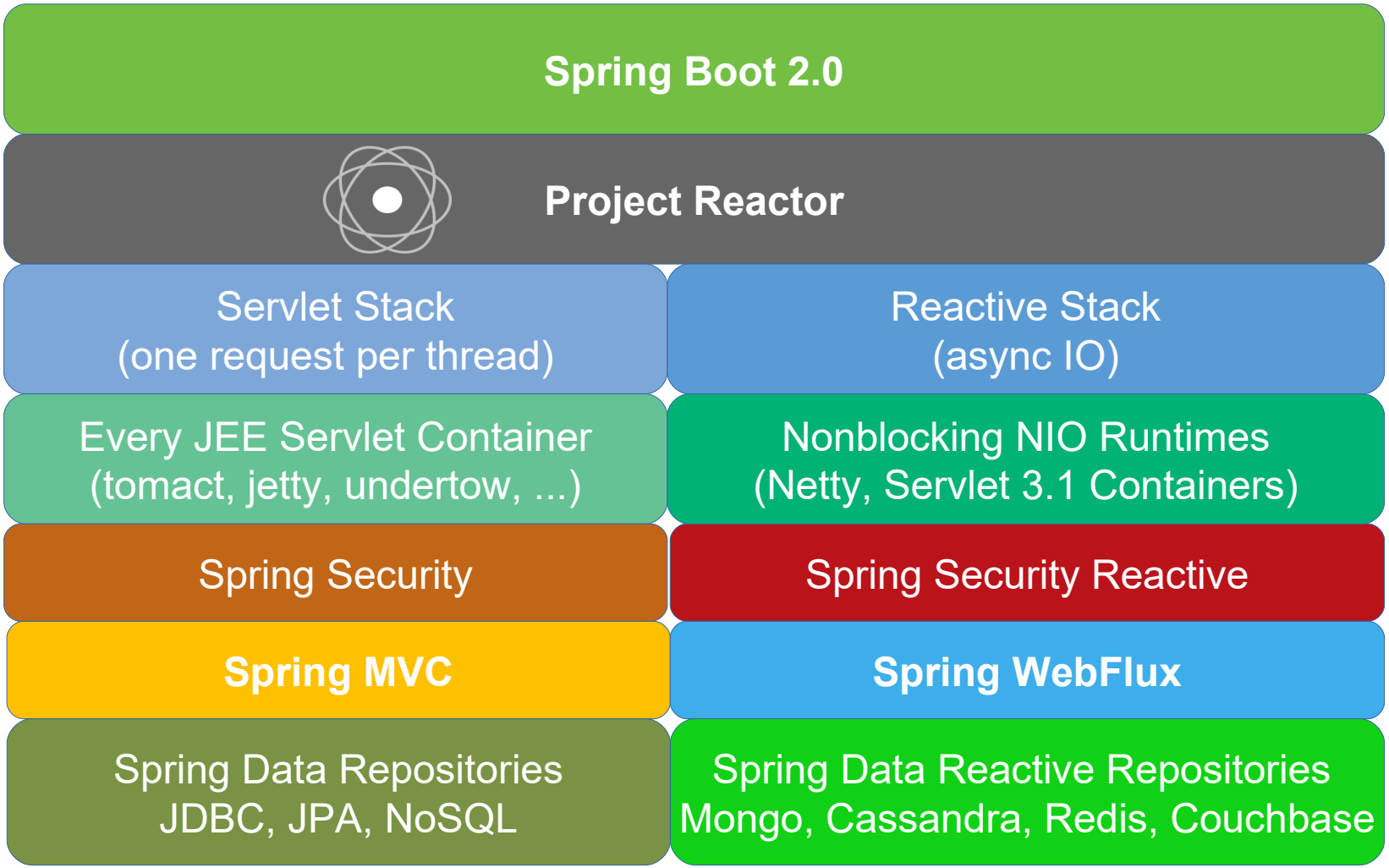
Hot Stream Example - Reactor

```
EmitterProcessor<String> emitter =
        EmitterProcessor.create();
FluxSink<String> sink = emitter.sink();
emitter.publishOn(Schedulers.single())
        .map(String::toUpperCase)
        .filter(s -> s.startsWith("HELLO"))
        .delayElements(Duration.of(1000, MILLIS))
        .subscribe(System.out::println);
sink.next("Hello World!"); // emit - non blocking
sink.next("Goodbye World!");
sink.next("Hello Trayan!");
Thread.sleep(3000);
```

Top New Features in Spring 5

- Reactive Programming Model
- Spring Web Flux
- Reactive DB repositories & integrations + hot event streaming: MongoDB, CouchDB, Redis, Cassandra, Kafka
- JDK 8+ and Java EE 7+ baseline
- Testing improvements – WebTestClient (based on reactive WebFlux WebClient)
- Kotlin functional DSL

Spring 5 Main Building Blocks



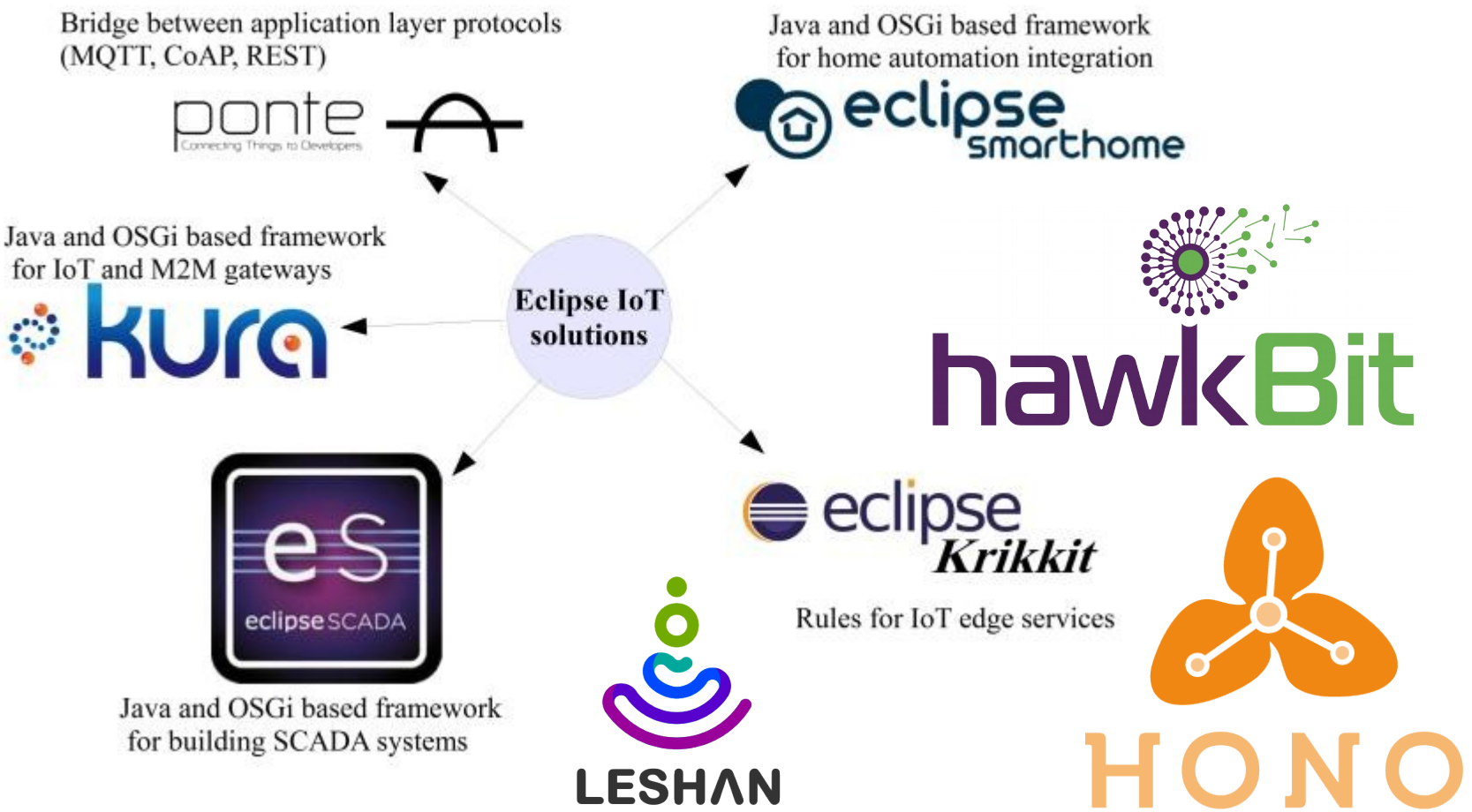
Source: *https://spring.io*

WebFlux: Best Expalined in Code

Lets see REST service Spring 5 WebFlux demos. Available @GitHub:

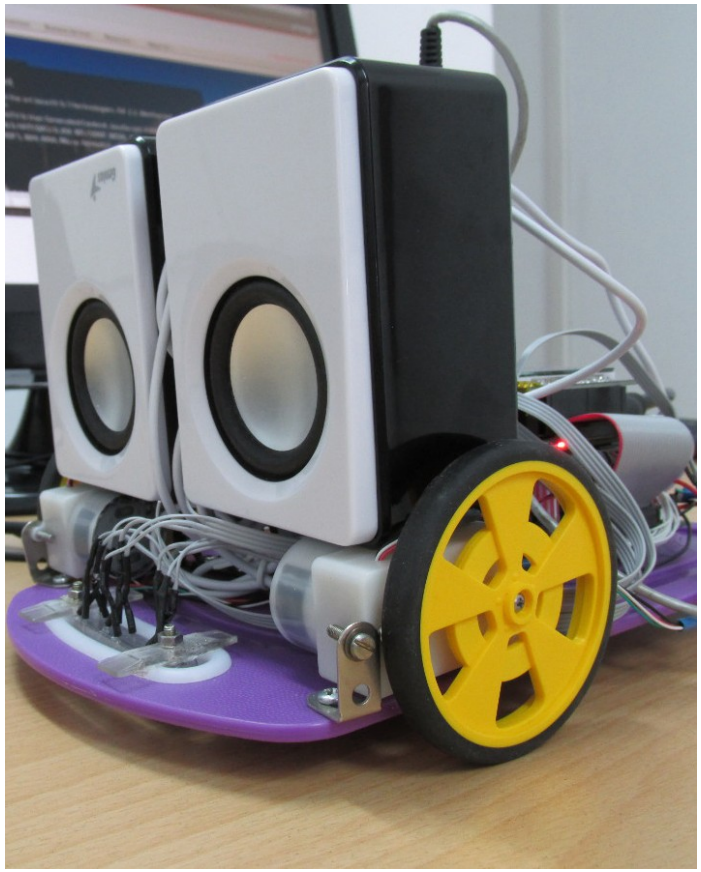
https://github.com/iproduct/spring-5-webflux

Eclipse IoT Platform



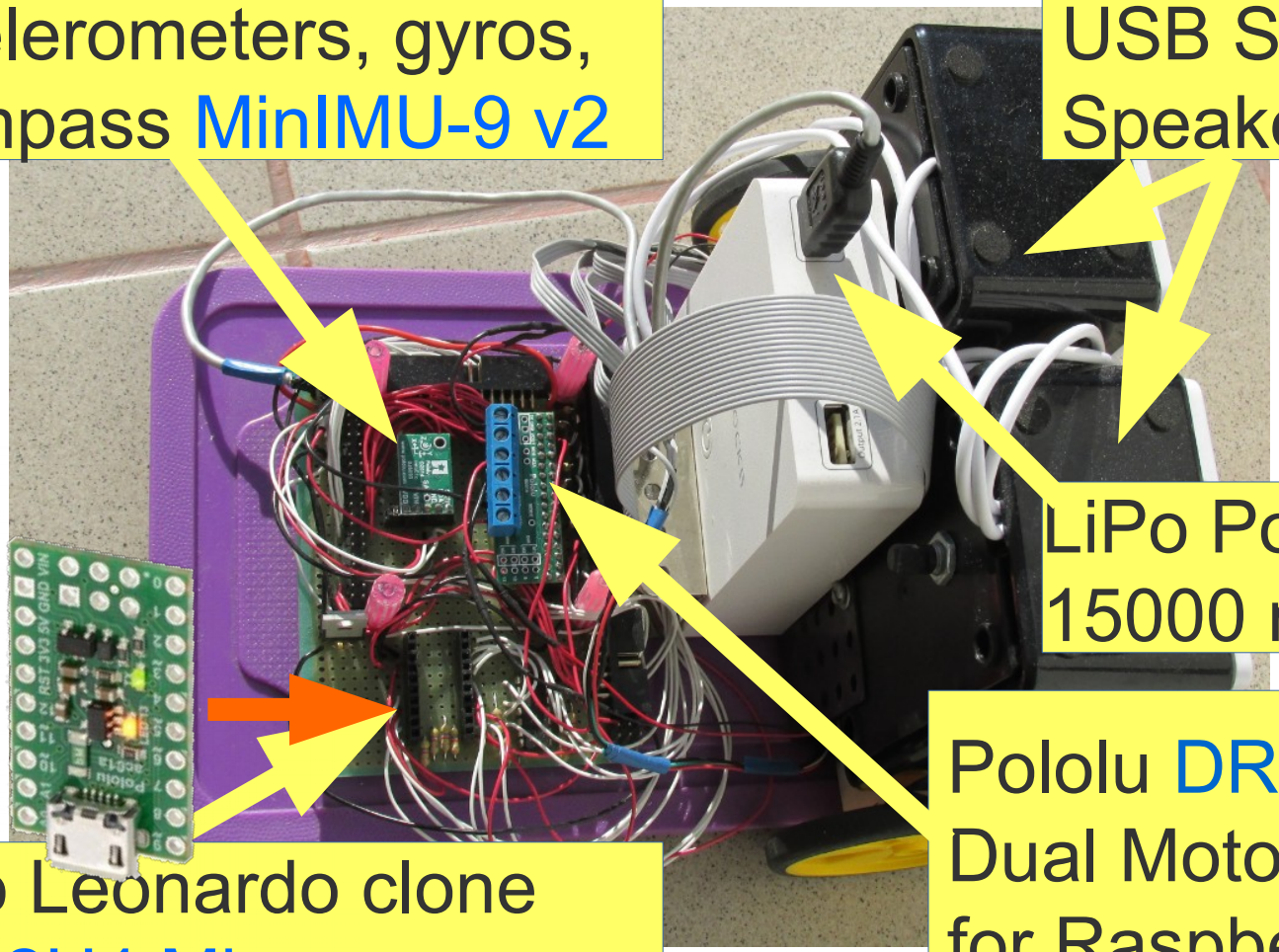
Based on: https://www.researchgate.net/publication/279177017_Internet_of_Things_A_Survey_on_Enabling_Technologies_Protocols_and_Applications, By Ala Al-Fuqaha et al. - Internet of Things: A Survey on Enabling Technologies, Protocols and Applications

IPTPI: RPi2 + Ardunio Robot



- Raspberry Pi 2 (quad-core ARMv7 @ 900MHz) + Arduino Leonardo cloneA-Star 32U4 Micro
- *Optical encoders* (custom), IR optical array, 3D accelerometers, gyros, and compass MinIMU-9 v2
- **IPTPI** is programmed in Java using Pi4J, Reactor, RxJava, Akka
- More information about IPTPI: http://robolearn.org/iptpi-robot/

IPTPI: RPi2 + Ardunio Robot



3D accelerometers, gyros, and compass MinIMU-9 v2

USB Stereo Speakers - 5V

LiPo Powebank 15000 mAh

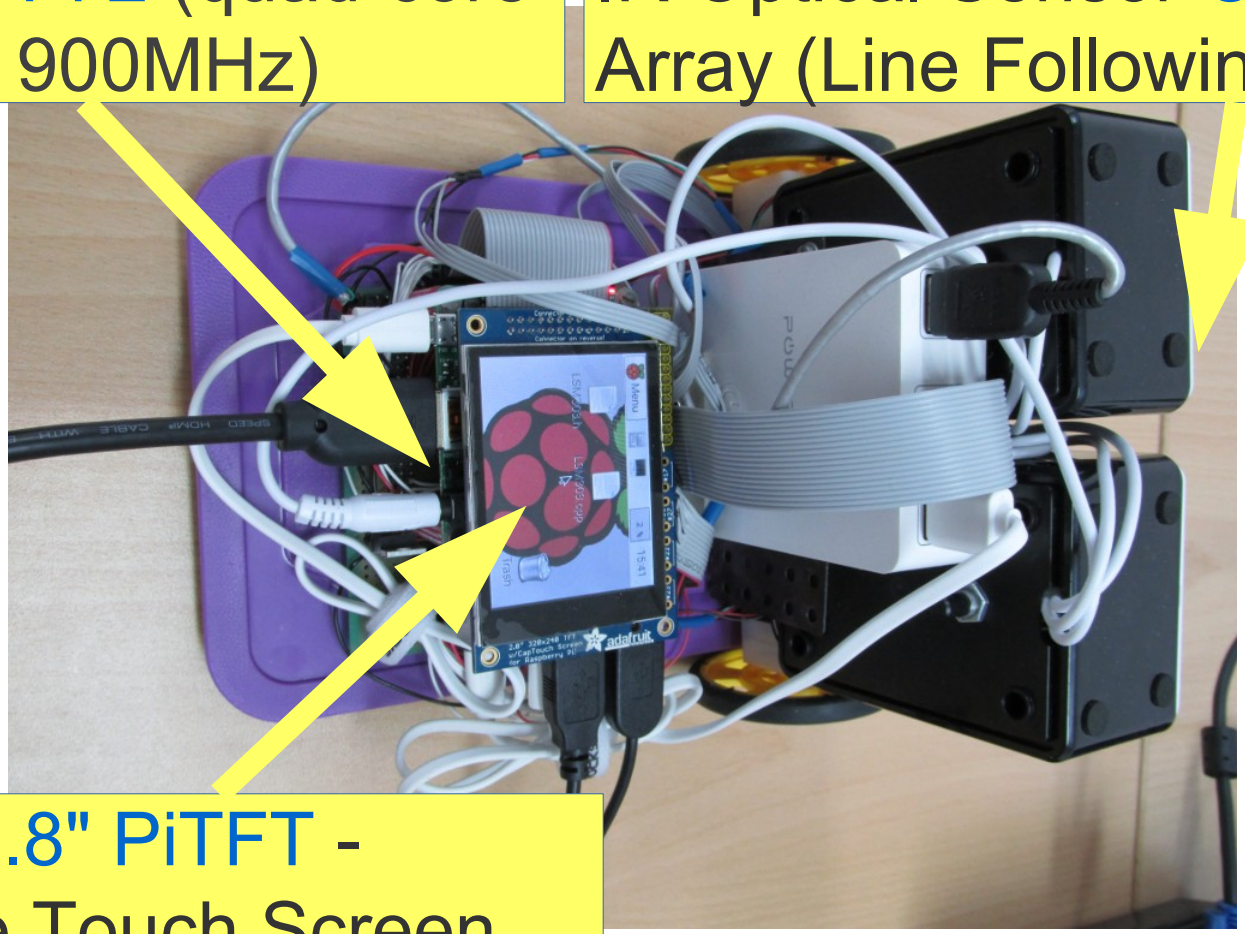
Pololu DRV8835 Dual Motor Driver for Raspberry Pi

Arduino Leonardo clone A-Star 32U4 Micro

IPTPI: RPi2 + Ardunio Robot

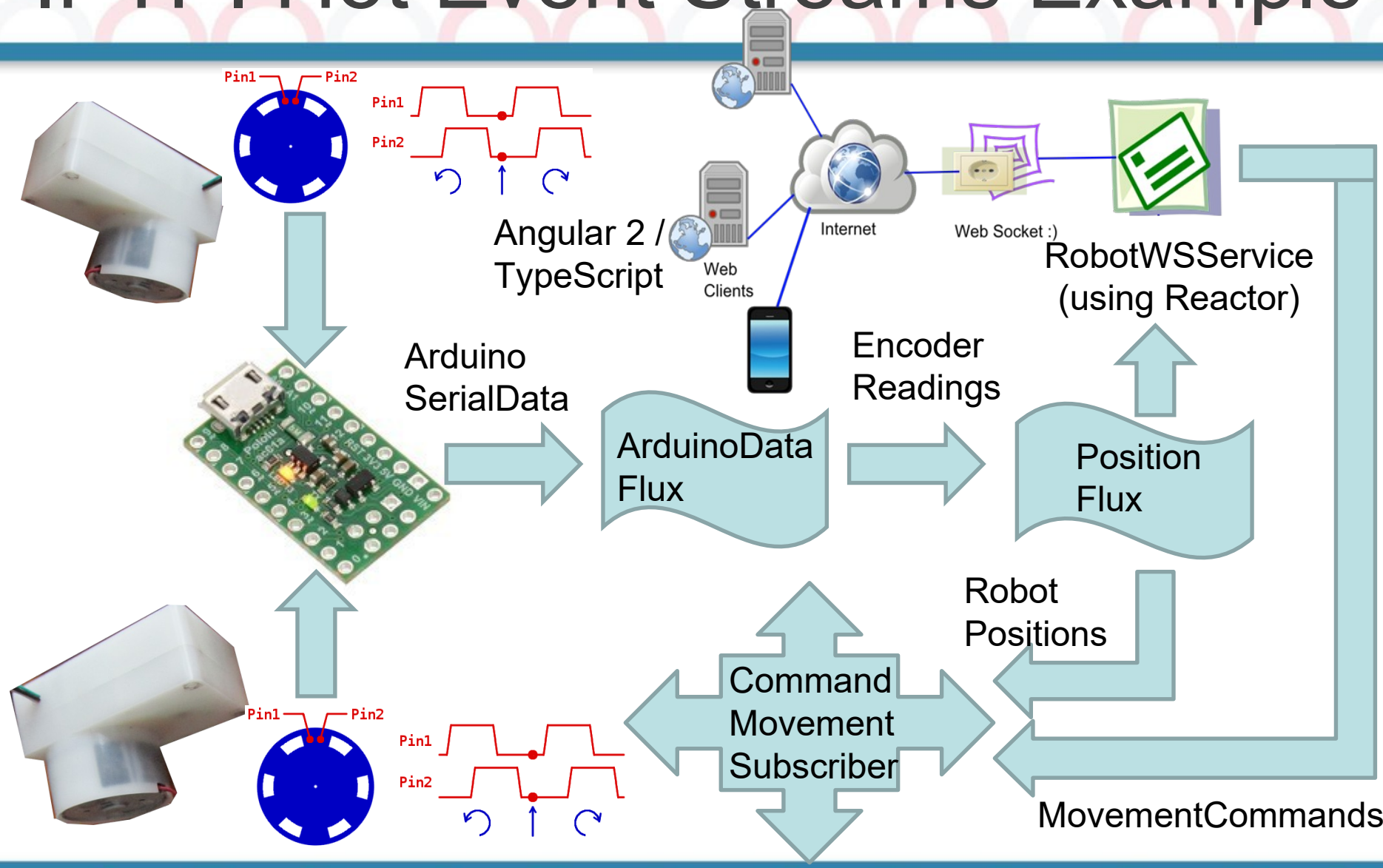
Raspberry Pi 2 (quad-core ARMv7 @ 900MHz)

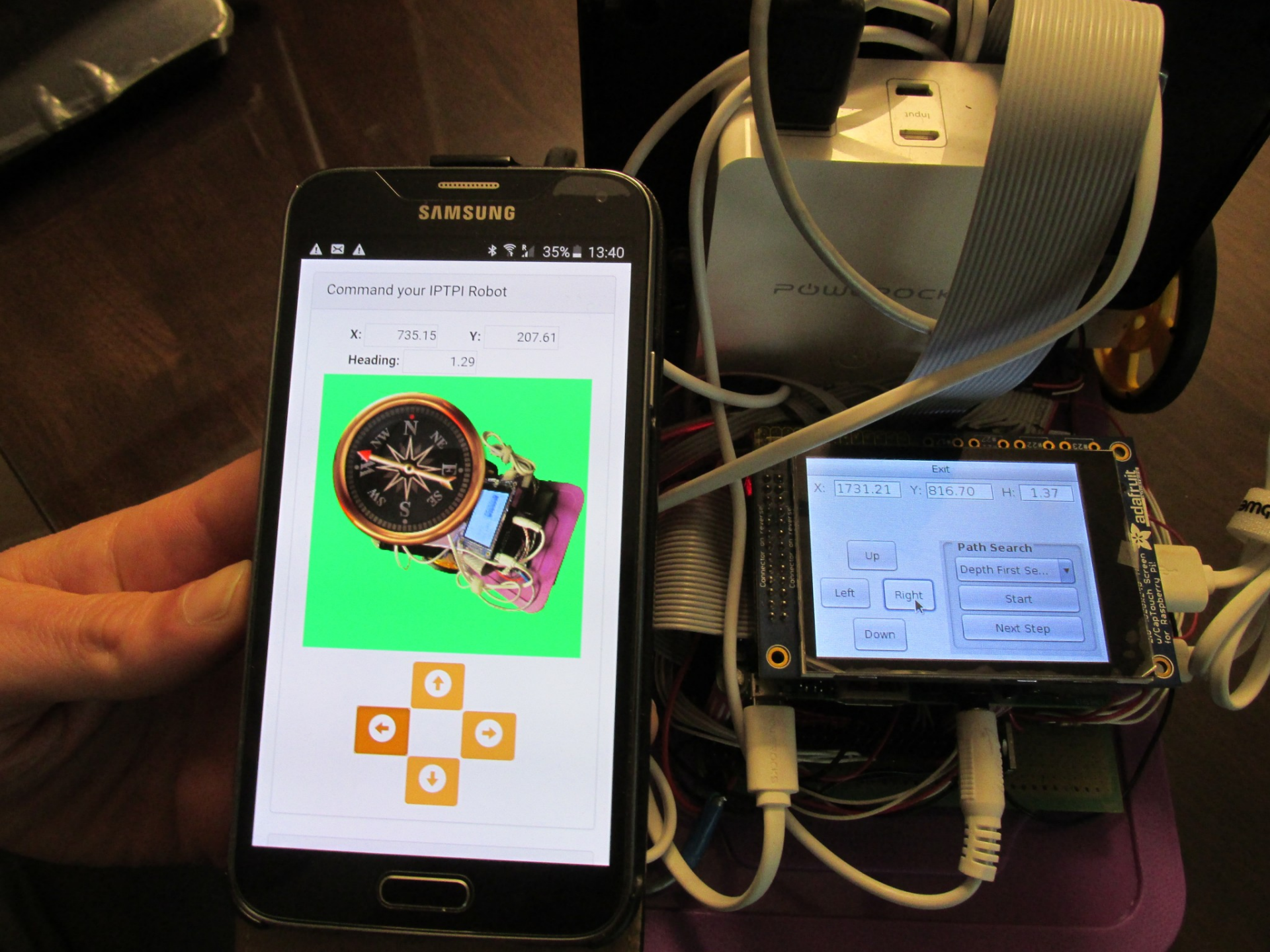
IR Optical Sensor QRD1114 Array (Line Following)

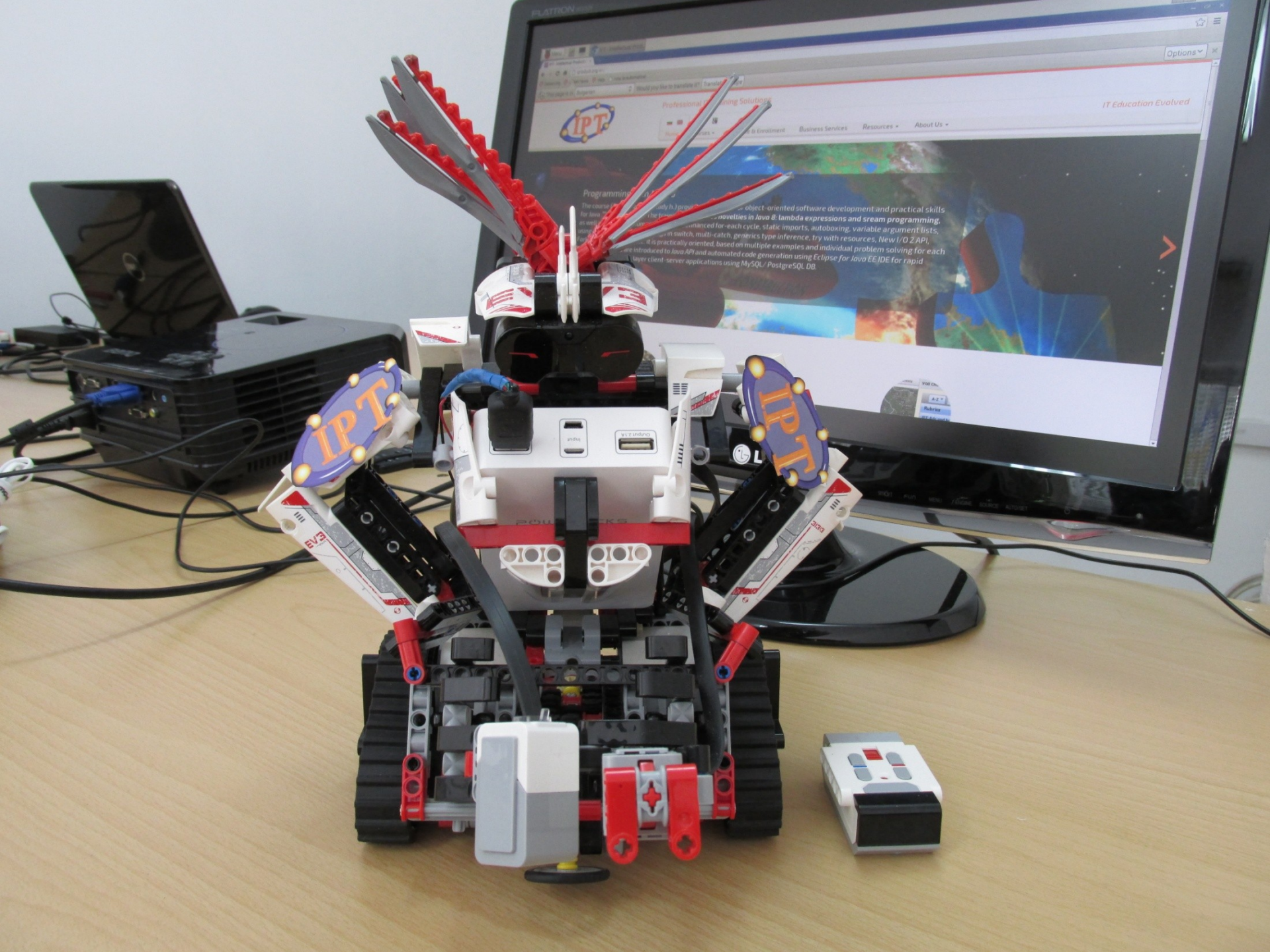


Adafruit 2.8" PiTFT - Capacitive Touch Screen

IPTPI Hot Event Streams Example

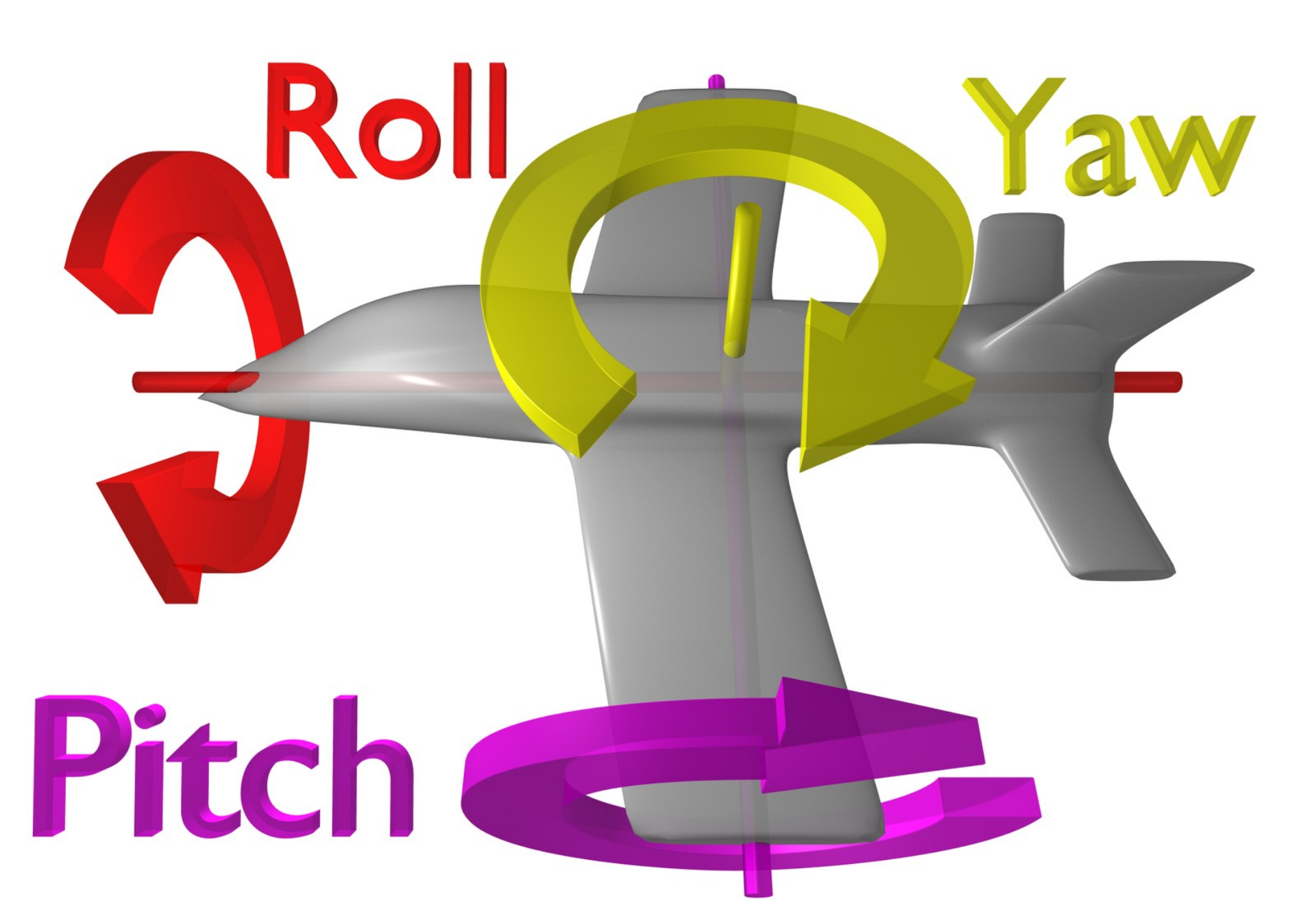



SAMSUNG
35% 13:40
Command your IPTPI Robot
X: 735.15
Y: 207.61
Heading: 1.29
Exit
X: 1731.21
Y: 816.70
H: 1.37
Up
Left
Right
Down
Path Search
Depth First Se...
Start
Next Step

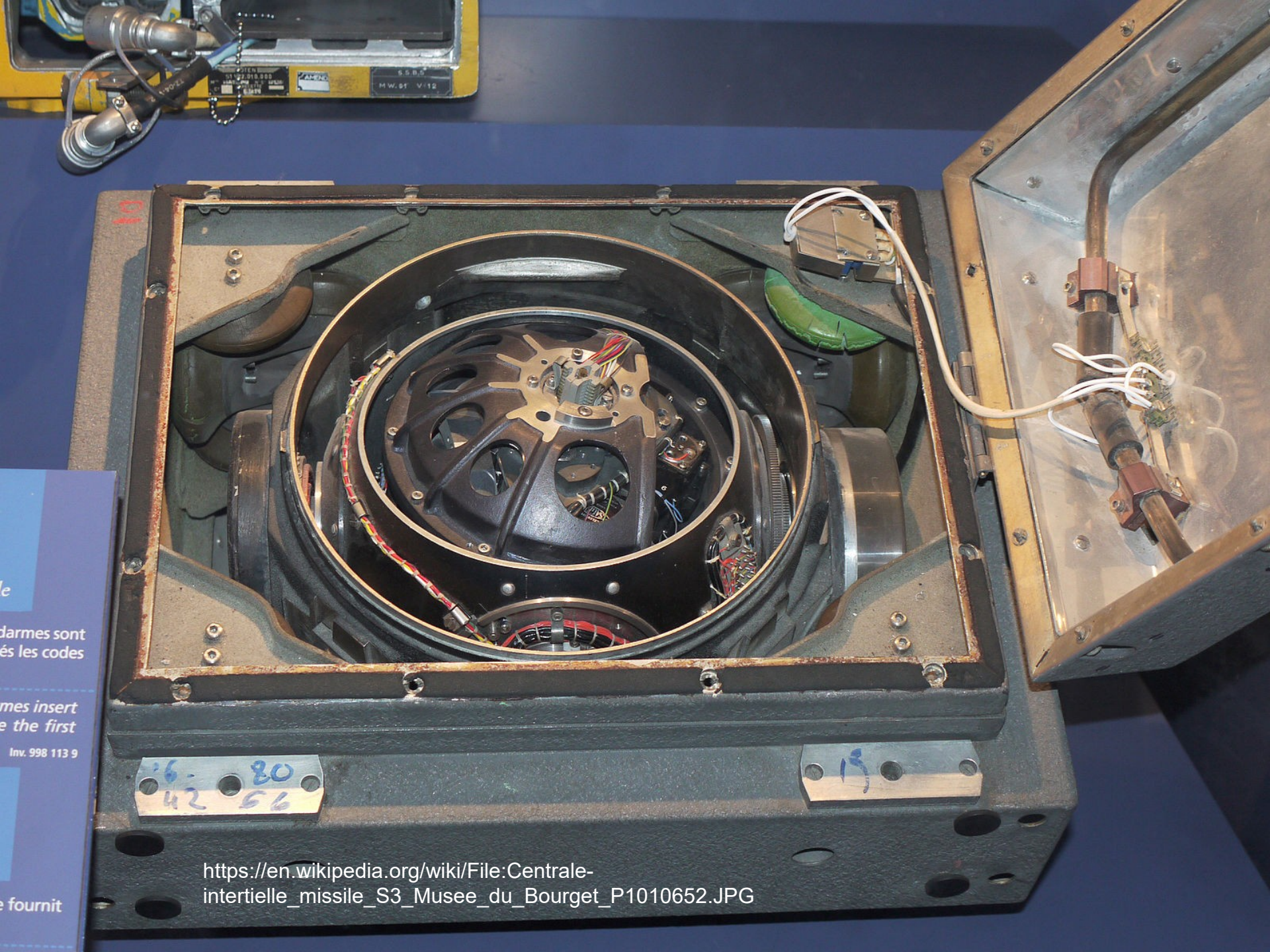

IPT
IT Education Evolved
Business Services
Resources
About Us
Options
IPT
IPT

LeJaRo: Lego® Java Robot

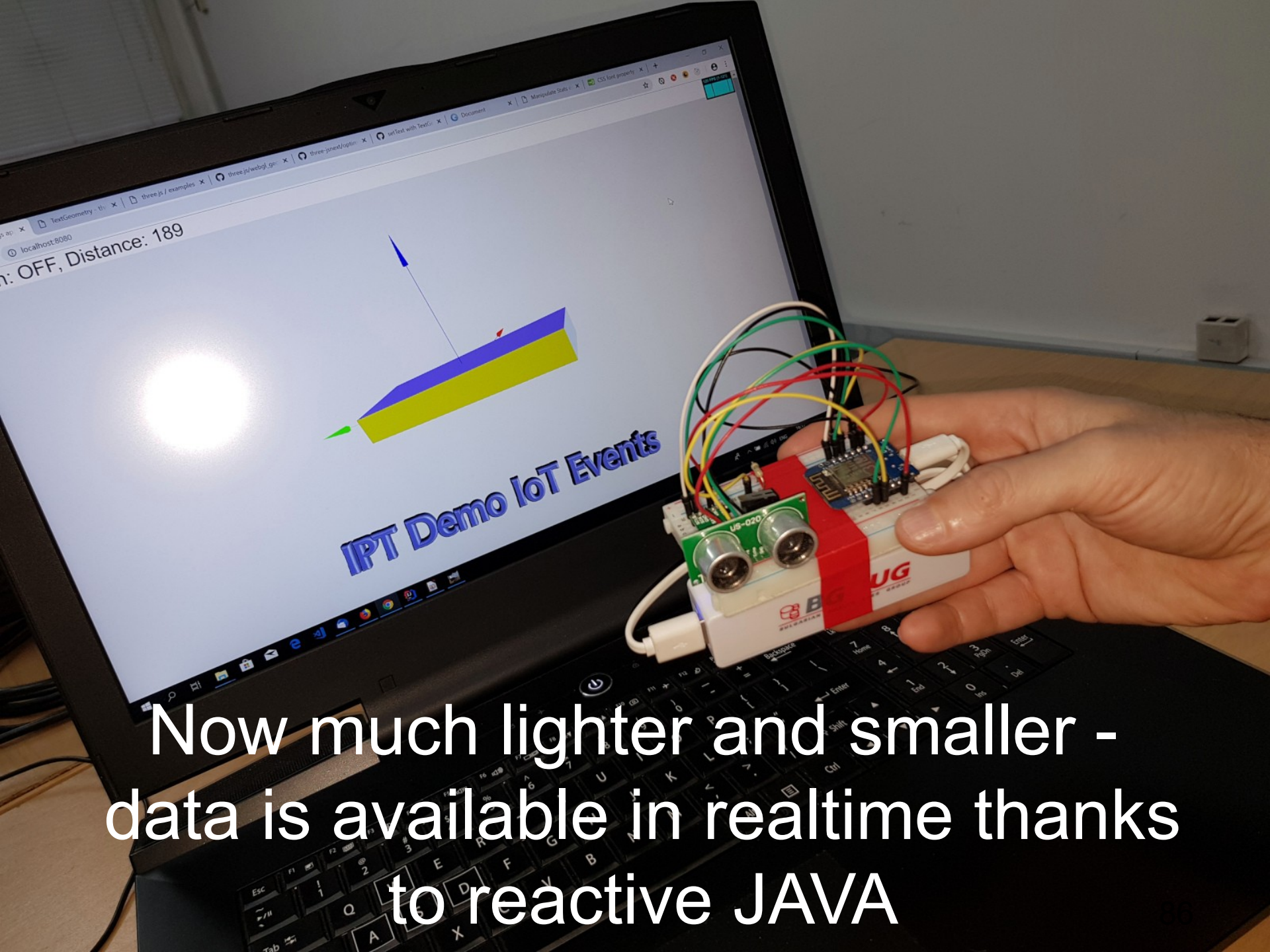
- Modular – *3 motors (with encoders)* – one driving each track, and third for robot clamp.
- Three sensors: *touch sensor* (obstacle avoidance), *light color sensor* (follow line), *IR sensor* (remote).
- LeJaRo is programmed in Java using ***LeJOS*** library.
- More information about LeJaRo: http://robolearn.org/lejaro/
- Programming examples available @GitHub: https://github.com/iproduct/course-social-robotics/tree/master/motors_demo



https://en.wikipedia.org/wiki/File:Flight_dynamics_with_text.png



https://en.wikipedia.org/wiki/File:Centrale-intertielle_missile_S3_Musee_du_Bourget_P1010652.JPG



Now much lighter and smaller - data is available in realtime thanks to reactive JAVA

Demo Code

Reactive Java Robotics and IoT with Reactor, RxJS, Angular 2 Demos are available @ GitHub:

https://github.com/iproduct/course-social-robotics

Thank's for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg