A technical introduction to IoT

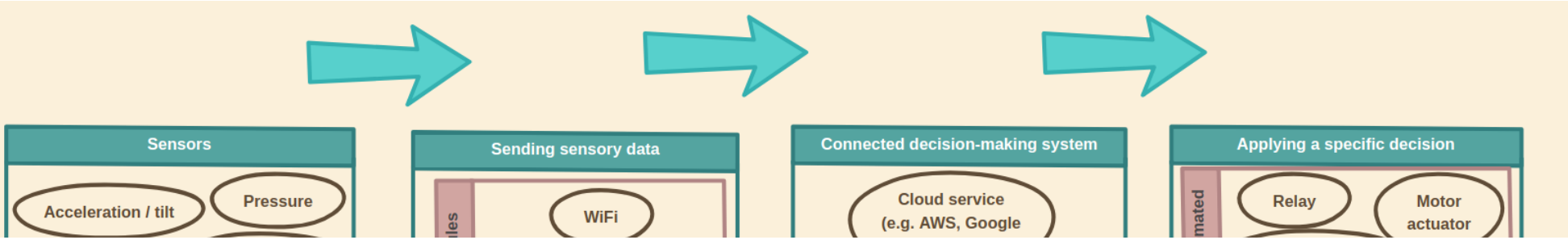
Andriy Lazorenko Jun 13, 2018 · 8 min read

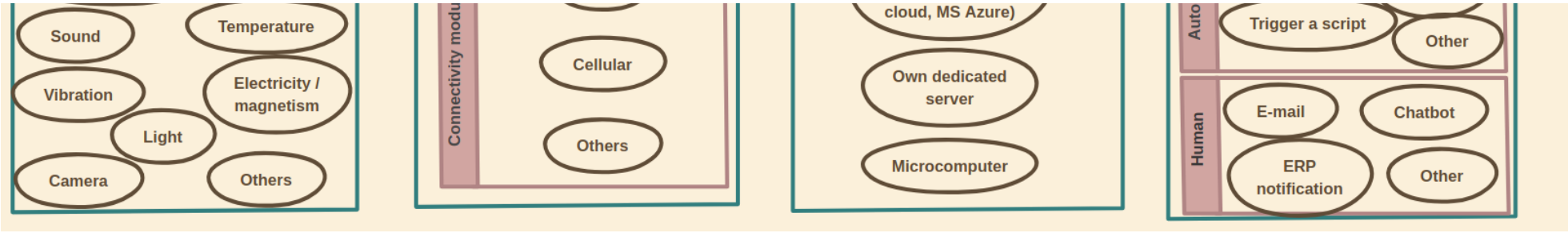
Let us remember a sketch from my article on short introduction to IoT:

IoT for everybody

A light introduction to IoT

medium.com





4 pillars of IoT system

Let us focus on toolbox used to build IoT systems. Toolbox consists of hardware and software instruments.

Disclaimer: all specific devices and clouds services are used in the article for illustration purposes only.

Hardware of IoT

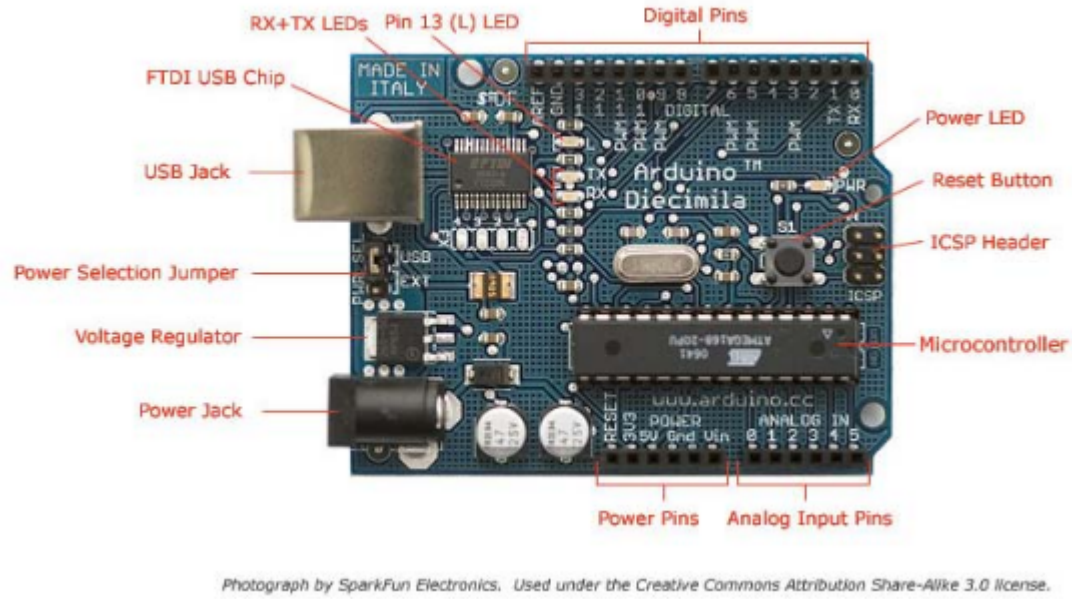
1. Sensors

Hardware always depends on a specific optimization task at hand. Going back to components of IoT system, different **sensors** could be used for IoT systems: temperature, humidity, vibration, electricity, light, sound, pressure, proximity sensors, accelerometers, magnetometers, gyroscopes, cameras, etc. They usually have specific drivers to work with written for most common platforms.

Example of diversity of sensors. A sensor kit for Arduino controller.

2. Devices and connectivity

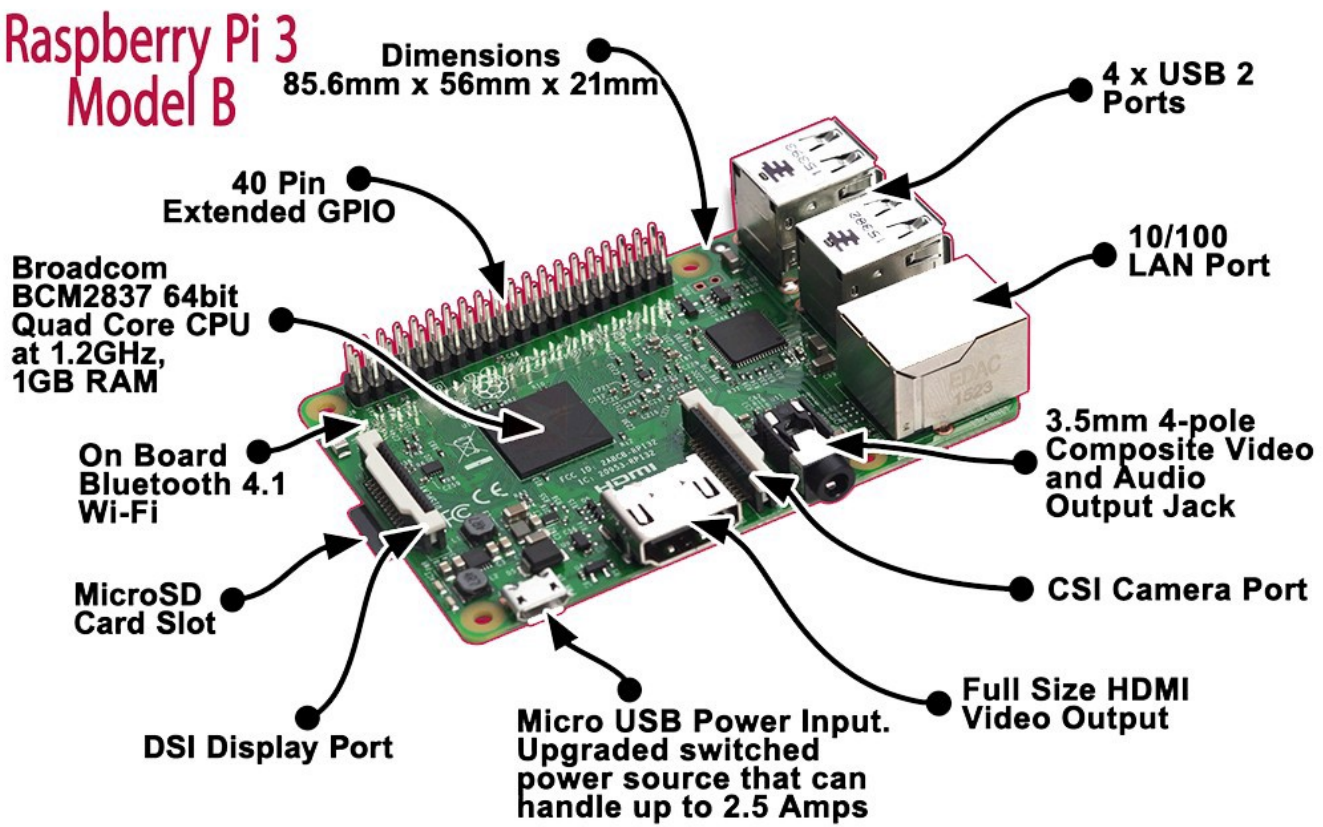
Microcontrollers are programmable devices used for tasks such as connecting different sensors, fusing their readings, sending them via communication (WiFi or cellular) modules. Popular ones are Arduino, Adafruit, STM32, and others.



Arduino microcontroller

Microcomputers are sometimes used for similar tasks as microcontrollers. Unlike many microcontrollers, which have a custom-built Operation System (OS) integrated within controller, many microcomputers are able to use more general OS (mostly based on Linux core) and function as general-purpose PCs (with low power consumption and mediocre performance).

Examples include Raspberry, Orange, Banana Pi series, Nvidia Jetson, and others.



Raspberry Pi 3 microcomputer

Also, a mobile phone is an example of device integrated with multiple sensors (camera, accelerometer, GPS, etc).

Besides the common wireless connectivity technologies such as **WiFi/Bluetooth** (low distance coverage, high energy consumption) and cellular (**GSM, 3G, LTE**: high costs: a data plan is needed for communications, high energy consumption, operator-dependent coverage), there is another option gaining popularity: Low-Power Wide Area Networks (**LPWAN**). It has the following nice niche features:

- Oriented to devices with low data volume and sporadic communications
- Low power consumption, so battery operated devices could work for many years
- Wide area coverage to send and receive data without problems inside buildings, basements, industrial boxes, etc.

There are now a lot of technologies in the same path like LoRa, LTE-MTC, RPMA, UNB, and others more. I find **LoRa** and **LoRaWAN** technologies

worth looking at, because they could be set up independently of network operator and provide means of connectivity to IoT devices even in remote areas.

Articles on IoT connectivity (with examples):

IoT Connectivity 101

Here's everything you need to know about IoT connectivity.

medium.com

Arduino Internet of Things (IoT) Using NiceRf LoRa1276

LPWAN and LORA Internet of things (IoT) has evolved in such way that every day is cheaper, smaller and less power...

www.instructables.com

LoRaWAN Technology for Arduino, Waspmote and Raspberry Pi

Electronics for DIY and the Internet of Things (IoT) Community. Arduino, Raspberry Pi, 3D Printers and Robotic...

www.cooking-hacks.com

3. Connected decision-making system

Decision-making components in IoT, in contrast to embedded solutions, are often placed in a cloud or a remote server for ease of modification and access. That means that hardware-wise either a cloud solution, or a pre-configured server are necessary as a part of usual IoT pipeline, which requires from a technical person some familiarity with cloud services (such as AWS, Google Cloud, MS Azure and many others) or Linux OS for deployment of IoT system.

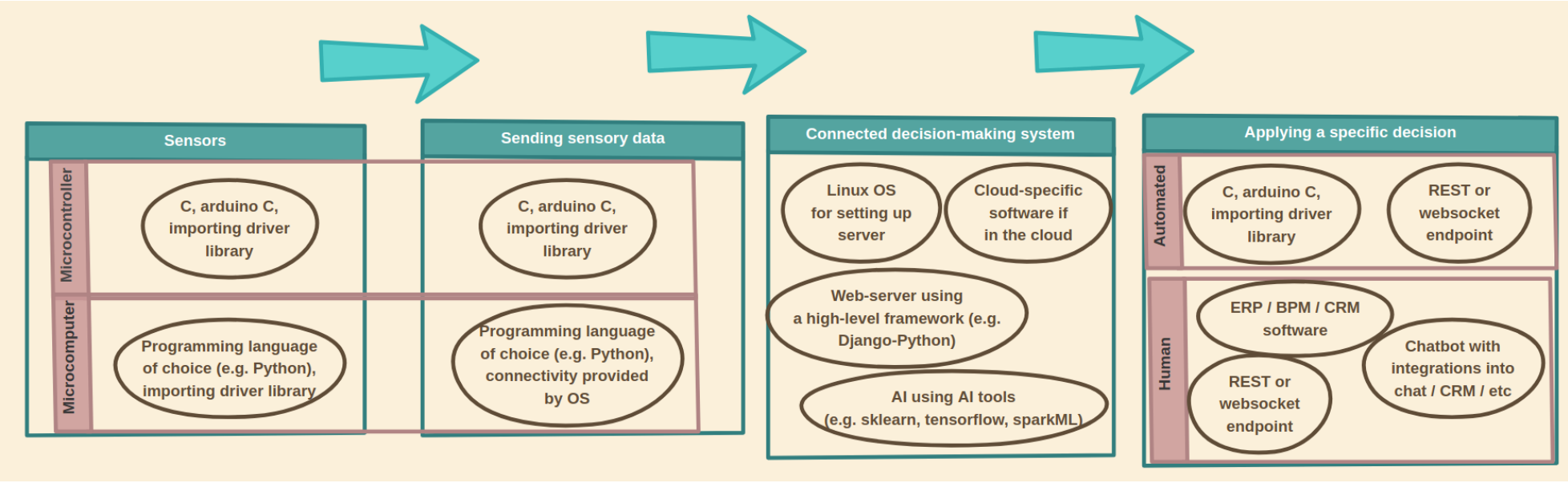
Sometimes, when decision-making logic does not require significant computational capacities, microcomputers connected to internet are used as decision-making hardware.

4. Applying a specific decision

Hardware components that **apply the decision** are, perhaps, the most diverse. In case of **fully-automated action**, component might consist of microcontrollers and motor actuators, switches, relays, etc all the way up to autonomous vehicles, factory robots, drones and other complex machinery. In case the action is dependent on other parties (**indirect decision-making**), actions typically consist of interactions via software deployed on a connected server: sending an e-mail, having a chatbot write a pre-defined

message via instant messengers (e.g. Slack), making a robotic phone call (e.g. if you forgot a credit card in ATM), displaying a notification inside a custom software (e.g. CRM or ERP system), or otherwise attracting attention of human decision-maker, who is needed for execution of a specific task at hand.

Software of IoT



4 steps software-wise

1. Sensors and devices

It is customary to use programmable **controllers** that operate in a C-like language, e.g. Arduino C. Such controllers excel at execution of commands related to communication of sensory results at near-real time. Usually a manufacturer provides a library to access functionality of a specific sensor. The library is then uploaded along with custom code representing the logic of gathering and transmitting the data. Unless communication with a sensor is a part of built-in controller logic, the process is repeated for every sensor.

Microcomputers can support many programming languages, a choice of programming language is subject to project requirements and personal preferences of an engineer. My favorite is Python, as it allows quick prototyping, has strong machine and deep learning capabilities along with decent web capabilities (and has native support in Linux OS). The process of communication with a specific sensor is similar to one described above for controllers.

2. Devices and connectivity

The process of connecting controller to internet is similar to connecting controller to sensor, and same goes for microcomputers, with a slight difference: it might be a lot easier to connect microcomputer to WiFi if the WiFi module's driver is standard for a microcomputer's OS, then

connectivity of microcomputer becomes similar to task of sending data via internet from a computer via programming language of choice.

3. Connected decision-making system

This is usually the place where most of software engineering work is concentrated. In order to listen to data send by either microcomputer or microcontroller, a web server must be up, running and connected to internet. Simple web application can be written with a high-level framework using REST (e.g. Flask + Python) and deployed using docker container. Creating a web-application is a common task for a backend developer and there are many guides on it, for example:

Docker for Flask Developers

Last fall I taught myself how to use Docker and wrote a piece titled Docker for Rails Developers. Since then, I have...

medium.com

If the decision making cannot be simply described using conditional logic and requires some **data analysis** and **pattern recognition** (e.g. maintenance prediction using data from multiple sensors connected to single machine or person identification and liveness detection on video

stream using computer vision), AI tools are used. My favorite tools for data science are: Python, Pandas, Keras + Tensorflow, Sklearn, Numpy but there are multiple other tools. Several example projects will be added in case this area gains enough interest.

4. Applying decision

In case the process is **fully automated** (it is sometimes referred to as the Internet of Robotic Things or IoRT), in order to apply decision, a signal has to be send to some device. It can be an **actuator**, which basically passes on controlling signal to connected piece of equipment (e.g. motor or valve, see different examples in Wikipedia), a **relay**, which acts as a controlled switch for the equipment it is connected to. Both of these devices are typically controlled via code present in microcontroller (or microcomputer), which, in turn, receives controlling input via internet. The software therefore is similar to paragraph “sensors and devices”.

Actuator — Wikipedia

An actuator requires a control signal and a source of energy. The control signal is relatively low energy and may be…

en.wikipedia.org

Magnetic latching relays can have either single or dual coils. On a single coil device, the relay will operate in one…

en.wikipedia.org

Automation also can happen via scripted logic of a process (e.g. automatically creating a ticket for maintenance of a specific part of machinery on a dedicated maintenance exchange service). This can be implemented using integration with API of 3rd party service provider.

In case the process requires human input, the software to implement it might take all shapes and forms. Straight to examples:

- Android / iOS app notifying a user that an old relative of his is experiencing stress / requires medical attention
- A Telegram chatbot notifying a user about an unrecognized person near his/her home, along with a picture taken by hidden camera
- A complex dashboard for a teacher containing information of student's attentiveness and focus based on camera data, temperature, CO2 level, humidity, light and sound sensors

- Custom dashboard containing KPIs of distributed self-driving AI, along with A/B testing results of various modifications of the algorithm, deployed on different autonomous vehicles of a company developing autonomous vehicles, based on daily report collected from all the vehicles
- A dashboard containing information on irrigation of different fields and deployment status of irrigation systems across the fields of an agricultural firm with options of manual override of automated decisions
- A mobile app with map of free parking spots in a given area
- A daily email newsletter that informs manager about equipment likely to break in the next few months along with interactive map of the equipment

And there are many more examples of user interfaces accepting human feedback / notifying the user about current state of affairs

Different software with various complexity corresponds to many possible modules applying the decision of a system. Email notifications are easy and quick to setup and require some familiarity with backend programming on a language of choice, chatbots are not a lot more difficult in terms of time

and effort (I enjoyed using Chatfuel + Django + Python for a simple FB messenger chatbot, introductory article using Python + Flask is posted below)

How to build and deploy a Facebook Messenger bot with Python and Flask, a tutorial

This is my log of how I built a simple Facebook Messenger bot. The functionality is really simple, it's an echo bot...

tsaprailis.com

Custom dashboards are more difficult to develop, but it's not a problem for a team that does it on regular basis. Mobile apps, in my opinion, are a bit trickier and also take significant time to develop, unless developed by experienced team. But, perhaps, the worst option is integration of user interface of IoT system (e.g. notifications or dashboard) into existing ERP / CRM / BPM solution. Companies developing proprietary ERP… software do not leave many opportunities to do it smoothly (they care for their profits after all) and it might be even more time- and money-consuming to go with that option.

That is enough of talking and hands waving :) **Next** we will dive into assembling a simple IoT system ourselves (with code and description of

components). If you are interested, thumbs up!

Internet of Things Sensors Embedded Systems Hardware Software