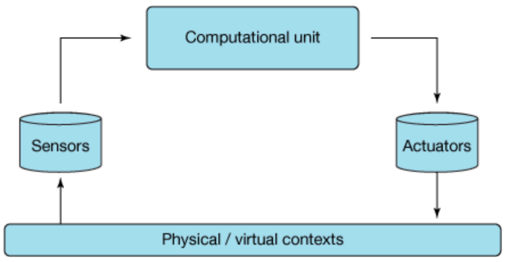
IoT - 1

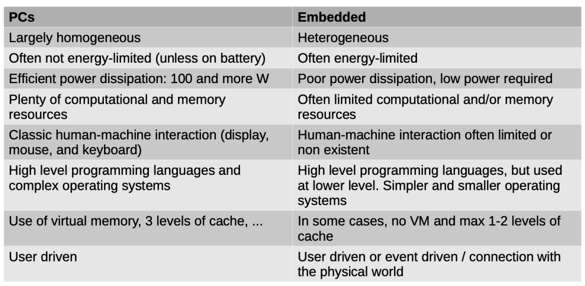
INTRODUCTION

Cyber-Physical System

A system exhibiting synergy of computational and physical components. The CPS combines both hardware (physical components) and software (computational elements) which interact closely with each other. This indicates that a CPS is composed of multiple components that work together, receiving data from the physical world (input), processing it, and then acting upon the physical world (output).A physical system with computational intelligence indicates that the system not only has a physical presence (called **plant**) but also possesses smart computational capabilities that can process data and make decisions. Here the plant of a CPS with computational intelligence.
A **full-fledged CPS** is typically designed as a network of interacting elements with physical input and output, where the embedded systems plays the role of building blocks for CPU, i.e. suppose a CPS is a Car, the "emergency brake" can be seen as an embedded system which servers as a component for a CPS (the car).



Embedded Architectures Vs. PCs



PCs	Embedded
Largely homogeneous	Heterogeneous
Often not energy-limited (unless on battery)	Often energy-limited
Efficient power dissipation: 100 and more W	Poor power dissipation, low power required
Plenty of computational and memory resources	Often limited computational and/or memory resources
Classic human-machine interaction (display, mouse, and keyboard)	Human-machine interaction often limited or non existent
High level programming languages and complex operating systems	High level programming languages, but used at lower level. Simpler and smaller operating systems
Use of virtual memory, 3 levels of cache, ...	In some cases, no VM and max 1-2 levels of cache
User driven	User driven or event driven / connection with the physical world

IoT

The Internet of things (IoT) is the extension of Internet connectivity into physical devices and everyday objects. IoT is a convergence of multiple technologies (is not only embedded systems) [illegible] IoT [illegible] tegra[illegible] also machine l[illegible]rning tech[illegible]ques[illegible]eal-ti[illegible] analytics, net[illegible]orking [illegible]mer[illegible] and so on[illegible] Trad[illegible]ional fields of e[illegible]edded s[illegible]tems[illegible]irele[illegible] sensor networ[illegible] contr[illegible] [illegible]stem[illegible] automati[illegible] (incl[illegible]ing home and bu[illegible]ing au[illegible]nation[illegible] and [illegible]ers all contribu[illegible] to [illegible]nablin[illegible] [illegible]t of th[illegible]
Some [illegible]amples o[illegible]T app[illegible]cations:
- Sma[illegible] Agricultur[illegible] pest [illegible]sect detection)
- Sma[illegible] Industry ([illegible]edict[illegible] maintenance, Goo[illegible]uarantee [illegible] ensu[illegible]g the authenticity[illegible]ality, and delivery)

Interaction with the Physical World

To interact with the physical world we rely on two main components:
- sensors: for **measuring**
- actuators: for "acting upon"

Dealing with the physical world mens handling **accuracy, reliability, safety.**

Sensors

A sensor is a device, module, or subsystem whose purpose is to detect events or changes in the environment. Measurement can be **direct** or **indirect**. With direct measurement, the actual physical quantity is measured as it is, using instruments or sensors that are designed to capture and record that specific parameter. For example, a thermometer directly measures the temperature. In contrast, indirect measurement involves estimating the value of a parameter by measuring other related quantities and applying a known relationship between those quantities and the parameter of interest. Often the indirect measurement is performed by the means of electric current.

Actuators

A component of a machine that is responsible for moving and controlling a mechanism or system. An actuator operates by adjusting a control parameter that then affects a target parameter within a system. For example, by regulating the electric current flowing through a resistor, we can control the amount of heat it produces. This is a practical illustration of how an actuator manipulates one aspect to influence another, achieving the desired outcome. Moreover, actuators can be integrated directly into the system they are controlling or connected through a network, allowing for remote operation and greater flexibility in system design.

There are two types of Systems

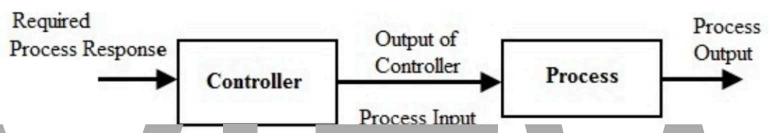
- **Continuous Dynamics:** Continuous dynamics relate to systems where changes occur in a smooth, uninterrupted manner. This is typical for physical systems found in nature and engineering, such as chemical reactions, the growth of living organisms, or the operation of mechanical devices
- **Discrete Dynamics:** Discrete dynamics, on the other hand, describe systems that change in distinct steps. These steps can be regular or irregular but are separate and countable. An example provided is a system that counts arrivals and departures. Each arrival or departure triggers an increase or decrease in a counter, which is then displayed. This process happens in a sequence of individual events rather than continuously.

CONTINUOUS DYNAMICS SYSTEMS

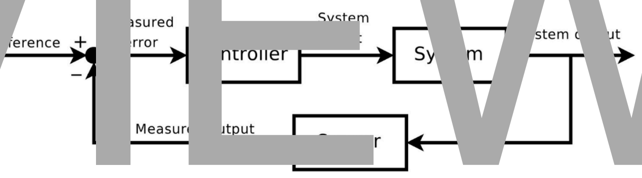
In continuous dynamics systems we require:
- a **model** of the physical system
- a controller suitable for the model: we need to cope a system that is continuous in time with a controller that is discrete in time

There are two types of Control Systems in Continuous Dynamics:

- **Open loop:** In an open-loop control system, the controller executes actions without considering the output of the process it is controlling. Essentially, the controller sends a command to the process, and there is no further check or adjustment based on what happens next. *The image show[illegible] flow f[illegible] to th[illegible] process.*



- **Clos[illegible] loop:** Cl[illegible]ed-lo[illegible] control systems [illegible] more [illegible] they de[illegible]back f[illegible]m the p[illegible]cess to a[illegible]ust th[illegible] control actions. As show[illegible] in the ima[illegible]e, the [illegible]ystem output is mon[illegible]red by a s[illegible]sor [illegible] to a reference value. Any deviation, known as the measured error, is fed back to the controller. The controller then adjusts its output to minimize the error, thus continually correcting the system's performance. *The output of the system is compared with the reference to measure an error, which the controller uses to adjust its inputs to the inputs.*



Examples

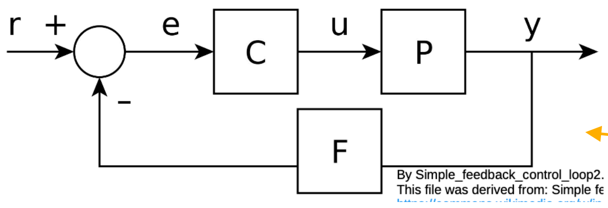
The first example discusses a mechanical arm powered by a step-by-step electric motor, which divides a full rotation into a set number of steps. This type of motor is typically managed using an open-loop control system, where it receives a command to move a certain number of steps that correspond to a specific angle of rotation. However, in such a system, there is no mechanism to ensure the arm actually reaches the intended position; errors can occur due to lack of feedback, initial calibration issues, or accumulated errors from repeated movements.

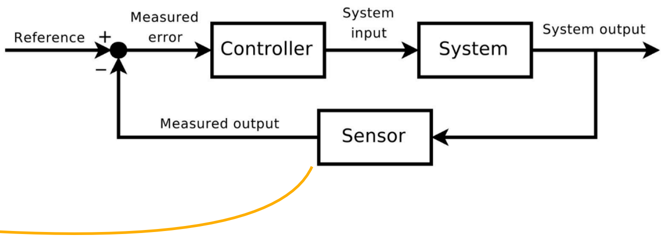
The second example considers an electric water heater, such as a boiler. In an open-loop control, the heater might be turned on for a set duration. The actual temperature of the water can vary widely due to factors like the starting water temperature, environmental conditions, the heater's efficiency, and fluctuations in the electric current. In this system, there's no feedback to correct such deviations.

In contrast, a closed-loop control system for both the mechanical arm and the water heater would include sensors to monitor the actual position or temperature, respectively. The control system would make adjustments based on this feedback to ensure the final position or temperature stays within the desired range, despite any potential errors or external influences. This ensures that the output of the system matches the target outcome with higher precision and reliability.

Closed Loop Transfer Function

This is just explanation for the closed loop diagram:


r
+
e
C
u
P
y
-
F
By Simple_feedback_control_loop2.
This file was derived from: Simple fe



(As explained above) The output of the system y(t) is fed back through a sensor measurement F to a comparison with the reference value r(t). The controller C uses the error e between the reference and the output to change the inputs u to the system under control P. **We always assume C, P and F to be linear** (*the system's output is directly proportional to its input*) **and time invariant** (*the system's behavior and characteristics do not change over time. If you give the system a certain input now or ten minutes later, the output will be the same, assuming the initial conditions are also the same*)**.**
Note there exists **SISO** (Single-input-single-output) and **MIMO** (Multi-Input-Multi-Output).

Laplace Transform

In closed loops, the measure of the error and the consequent adjustments of controller input can be performed using the Laplace Transform on Y(t), the output of the system. In short, Laplace is an integral transformation that transforms a function of a real variable *t* (often time) to a function of a complex variable *s* (complex frequency). The Laplace transform is a mathematical technique used in control systems engineering for simplifying the analysis and design of systems in [illegible]e tim[illegible] domain b[illegible] transfo[illegible]ming them into [illegible]e frequen[illegible] dom[illegible]n. Th[illegible] r(t) output bec[illegible]es Y(s) [illegible]ich [illegible] the output [illegible] the [illegible]stem H(s) multip[illegible]d by the [illegible]put [illegible] the s[illegible]tem R(s). H(s) te[illegible] us h[illegible] [illegible]e sys[illegible]m will [illegible]espon[illegible] input i[illegible]he presen[illegible] of feedba[illegible].

$$Y(s) = \left(\frac{P(s)C(s)}{1 + P(s)C(s)F(s)}\right) R(s) = H(s)R(s).$$

$$H(s) = \frac{[illegible]}{[illegible] F(s)P(s)C(s)}$$

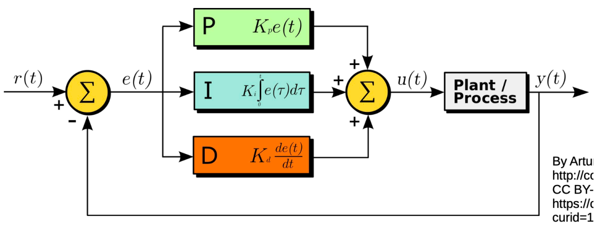
- T[illegible] nume[illegible]tor is the [illegible]ward [illegible]pen-loop) gain from [illegible] y
- The [illegible]en[illegible]inator is [illegible] plus [illegible] gain in going aroun[illegible] the [illegible]eedbac[illegible] the [illegible]c[illegible]ed loop ga[illegible]

ht[illegible]ikipedia.org/wik[illegible] th

PID Controller

The PID controller adjusts the control input to using a system based on three terms:

- proportional **P:** current error
- integral **I:** the accumulation of past errors
- derivative **D:** the prediction of future errors.



The PID controller aims to minimize the error over time by adjusting the control input u(t), which affects the plant or process to produce the desired output y(t). This adjustment process, known as "tuning," involves iterative optimization of the three parameters:

- **Kp** : the proportional term. This is responsible for the **magnitude of correction** applied to the control input based on the error.
- **Ki** : the integral term. This is responsible for the accumulation of the error
- **Kd**: the derivative term. This is responsible for the rate of change of the error.

PID Controller Tuning

As said above, PID tuning is is about setting the K_P,K_I, and K_D parameters to obtain the desired system response.

Tuning Kp (the proportional gain)

Recall that, **proportional gain**, denoted as **Kp**, determines how much the control action is influenced by the current error, which is the difference between the desired and actual output.
Note that:

be removed
Durable connection: when a client disconnects, all his subscriptions will be remain
Will: when a client connects to a broker it may inform the broker that it has a will, i.e. a message that wants to be sent/broadcasted when the same client disconnects unexpectedly.

MQTT QoS

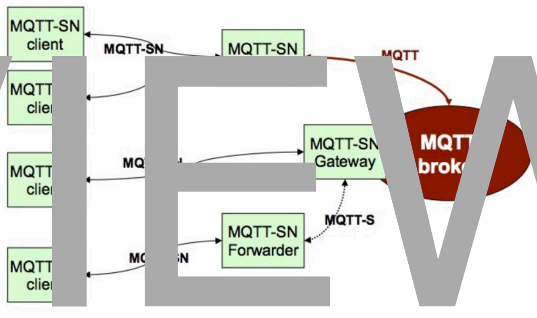
Here the QoS is slightly different than the priority one we defined above. Here MQTT QoS does not define the priority on which a message is sent, but it defines the reliability level of a message. There are different levels of QoS:

- 0: The broker/client will deliver the message once, with no confirmation
- 1: The broker/client will deliver the message at least once, with confirmation required
- 2: The broker/client will deliver the message exactly once by using a four step handshake
- Higher levels are more reliable, but involve higher latency and have higher bandwidth requirements

The quality of each communication depends on:
- the QoS level provided by the broker, which states the maximum QoS available for that specific communication.
- The broker will adapt available QoS to the client request: if the client requests a level 2, but broker can have only level 0 or 1, the broker will label that connection as with reliability 1.

MQTT for Sensor Networks (MQTT-SN) Architecture

The gateways are between MQTT-SN nodes and the MQTT-Broker. The gateway will translate from MQTT-SN [illegible] MQ[illegible]ward.



T[illegible]ere a[illegible] two types [illegible]f gat[illegible]ways for MQTT-S[illegible]
- **tran[illegible]** snaps[illegible]TT-SN [illegible]des t[illegible] the b[illegible]oker. So w[illegible]cheve[illegible]nformation the MQ[illegible]T-SON[illegible] nodes tra[illegible]mit w[illegible] go directly to the b[illegible]ke[illegible]
- **agg[illegible]gating-GW** [illegible]all th[illegible]MQTT-SNO nodes w[illegible]e aggr[illegible]gated in the [illegible]ES. We n[illegible]d aggregating-GW because the MQTT broker may support a limited number of nodes. An thus we aggregate multiple nodes describing them as just one address, sir the broker could be able to handle high number of nodes.

Difference between MQTT and MQTT-SN

(the professor didn't explain that too much)

- The CONNECT message is split into three messages
  - The two additional ones are optional and used to transfer the Will topic and the Will message to the server
- The topic name in the PUBLISH messages is replaced by a short, two-byte long "topic id"
  - A registration procedure is defined to allow clients to register their topic names with the server and obtain the corresponding topic id
- "Pre-defined" topic ids: two-byte long replacement of the topic name, their mapping to the topic names is known in advance by both the client's application and the gateway/server
  - Both sides can start using pre-defined topic ids without registration
- A discovery procedure helps clients without a pre-configured server/gateway's address to discover the actual network address of an operating server/gateway
  - Multiple gateways may be present at the same time within a single wireless network and can co-operate in a load-sharing or stand-by mode
- The semantic of a "clean session" is extended to the Will feature
  - Not only client's subscriptions are persistent, but also Will topic and Will message
  - A client can also modify its Will topic and Will message during a session
- A new offline keep-alive procedure is defined for the support of sleeping clients
  - Battery-operated devices can go to a sleeping state during which all messages destined to them are buffered at the server/gateway and delivered when they wake up

EDGE COMPUTING

(recall what we said in fog computing). In edge computing the big part of the computation happens in edge devices. Which are the pros and cons of moving the computation to the edge?
- we need to consider the latency of the computation w.r.t. the requirements we have. Cloud computing has more latency than edge computing (as explained already)
- Energy and Power of computation vs energy of the networking. In some cases the energy that we need to transmit the data over the network right be more than directly do computation on the devices.
- Is it harder to update software for all the devices in the network or the cloud servers?

- What is the number of devices? If we need to receive data to thousands of die devices, will our network infrastructure (cloud) support that?
- Security and Privacy (already mentioned)

Whenever we have simple control/logic and simple computation, we should rely on edge devices to compute. We do not need to send data to the cloud if the computation are easy and can be directly done locally. Also, cloud computing is not ideal for real-time applications.

Design Space Exploration

First of all you define a metric, i.e. if your goal is to minimise energy, you define a metric connected to energy and that your objective is to minimise that metric. Then you define the design space: what are the possible implementation (all computation on cloud, all on edge, half/half, use fog and so on). By exploring design spaces, i.e. looking at how you can optimise the metric (i.e. how much can you minimise the energy) you then choose. To explore design spaces with high precision you should implement all the possible solutions. In most cases you just rely on preliminaries test and simulations.

Example

- IoT nodes collect images with the purpose of counting dogs present in different dog areas; entrance is closed if no. of dogs is higher than a threshold
  - IoT nodes collect 1 image (approx 2MB) every 30s
  - A cloud infrastructure collects data about dog presence (no. of dogs over time) and shows it on a map
- Let's concentrate on the task of counting dogs
  - Metric: node energy, provided that the computation is completed before the arrival of a new image (i.e., 30s latency)
  - Design space:
    - Sending images to the cloud and running models for counting on the cloud
    - Running models for counting on the nodes and sending only the number of dogs to cloud
    - Pre-processing images (e.g., extract objects that can be dogs) on the node and sending pre-processed images to the cloud for counting

- Suppose that we know the following
  - Netwo [illegible] 0.001 mWh/[illegible]
  - Runni[illegible] full model on edge n[illegible] requir[illegible]
    - 3[illegible] computation
    - 0.[illegible]h
    - 2[illegible]nt
  - Image [illegible]processing requires:
    - 2[illegible]omputation
    - 0.[illegible]Wh
    - A[illegible] of 1MB of data to be se[illegible]he cl[illegible]

- Exploring the design space
  - [illegible] 2MB*1000*0.001mWh/KB=2mWh
    - Computation time: image sent in les[illegible]2s
  - [illegible]) All on node:
    - Energy per image: 0.100002mWh
      - [illegible]*0.001mWh/KB=0.0[illegible]Wh
      - [illegible]1mWh
    - Computation time: 35s + less than 0.5s for communication
  - [illegible]) Pre-processing on node:
    - Energy per image: 1.075mWh
      - Network: 1MB*1000*0.001mWh/KB=1mWh
      - [illegible]mWh
    - [illegible] less than 2s for sending the image

The optimal solution is no. 2, but it does not satisfy the requirement on latency: we opt for solution 3

Intermittent Computation

Often devices are not processing data all the time, in a continuous way. Computation can be periodic (every some time device took a photo and send it to the cloud) or event based (a digital sensor will notify with an interrupt the devices that some event happened).
So, to save energy, we can implement intermittent computation as:

- Wake-up on events (interrupts)
- Walke-up periodically (timer-based events)
- Switch on/off parts of the system depending on the computational needs
  - E.g., in a 2-core setup, one core can be always on, the other can be waken-up only when more computational resources are required
    - Typical case: one core used for sensing/controlling; the other core is used for computations (e.g., image/signal processing or machine learning)
    - One core uses a microphone to detect if there is any sound over a certain volume threshold, if there is any, the second core processes the sound to detect if it corresponds to steps or a broken window

There is also a slightly different approach, called **Intermitted Computing** (which is different from Intermitted Computation). In this approach, the device have a tiny amount of energy (e.g. RFID devices such credit cards) or does not have a battery. But it gets energy trough the communication. With this tiny amount of energy, your computation may stop suddenly -> intermitted computing means that your computation may terminate suddenly, and you want to recover the computations in some way.

Machine Learning and Edge Computing

Moving machine learning to edge devices may be actually useful (embedded devices are becoming better and better, and lot of application would profit from local computation). Note that moving ML computations to the edge nodes may not be possible (due to memory/ performance restrictions).

Workflow of Machine Learning

When we deploy a model we would also like to monitor model performance over time and collect data to further optimise it. This is easy to do if you do everything in the cloud. In edge computing, usually, all the training part is done on the could and we do deployment on edge devices. Whenever we do model development (such as training) it may still include IoT devices to collect data.

Phase	Task	Cloud	Edge device
Model development	Collect data	X	X
	Prepare the data	X	
	Training	x	
Model deployment	Deploy the model	x	x
	Collect data for improving the model	x	x
	Improve the model	x	

- In general, more computationally intensive operations (e.g., training) are performed on cloud
- Data are collected by using edge devices, similar to the final ones (+ available datasets)
- Deployment may be performed on the cloud or on edge devices (partly or fully)

Example (still with counting dogs) and we want to train the model on cloud and deploy it on edge devices. First we want to collect data using devices similar to the final ones of the edge network. Once our model is trained (and we know it works), we can deploy that on edge devices. Acquired image will go to a preprocessing procedure in the mended device and then passed [illegible] (still [illegible] embedded device) which produces an outp[illegible]

- Let's consider the system that counts dogs again and let's suppose that the model is going to run on edge devices
- The ideal model training process is as follows:
  - Edge devices similar to the final ones (or at least with similar cameras) are used to collect images (e.g., by sending them to the cloud, or by saving them on local memories that are later collected)
  - On the cloud
    - Images are preprocessed (e.g., resizing, elimination of duplicates, ...)
    - Once a sufficient number of images has been collected, the training, test, and validation datasets are formed; the model is trained and validated
  - The model is deployed on edge nodes along with required preprocessing modules (e.g., image resizing)
    - Images captured by the camera are fed into the model deployed on edge nodes after suitable preprocessing

[illegible]odel [illegible]aintena[illegible]e

How d[illegible] we collec[illegible] data to [illegible]mprove our model? We c[illegible] have s[illegible]me de[illegible]ces which save d[illegible]ta [illegible] externa[illegible] memory a[illegible] from [illegible]me to time we go t[illegible]re [illegible]nd down[illegible]ad. M[illegible]ybe we have devi[illegible]s [illegible]ith sends [illegible]w images [illegible] the [illegible]oud. The update of t[illegible]odel shou[illegible] be m[illegible]de on cloud. After [illegible] model [illegible]s been up[illegible]ted, [illegible] **[illegible]eploy it ag[illegible]n** (we mig[illegible] just [illegible] the n[illegible] weights and not the entire model, but still, it is not easy).

Model Optimisations for Edge Computing

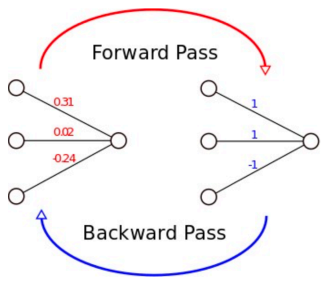
Models can be and often they are too. big to be deployed on edge devices. I need to introduce some optimisations on the model to redusce its memory footprint or computational requirements. This might be done by also giving up some accuracy of the model itself.
Common Methods for reducing memory footprint are:
Model Pruning: (just mentioned)
Knowledge Distillation: (just mentioned)
Factorisation: (not even mentioned).
Model Quantization: you can have that the parameters in your model are **quantised** (instead of using full precision floats you can use reduce precision floats, i.e. float-16 instead of 32, or use integers).

Common methods for reducing memory footprint are
- Model pruning
- Quantization of weights
- Knowledge Distillation
- Factorization
  - The redundancy present within the convolutional filters is exploited to derive approximations that significantly reduce the required computation [*]

- It can be performed:
  - After training
    - It leads to lower accuracy
  - After training + tuning
    - Partly compensates for the loss of accuracy
  - During training

Model Quantization During Training

- The forward pass of the neural network uses a scheme for rounding float-precision parameters to lower precision
- In the backwards pass, the high-precision parameters are updated using gradients calculated during the forward pass
- Only the reduced-precision network is saved for inference



Encryption
Encryption may be not enough: just the fact of sharing something (even if encrypted) gives the attacker an information from which he can assume/infer something useful.

(skipped a bunch of slides)

Secure Communication Protocols

- Secure communication protocols offer security by relying on encryption and authentication algorithms
- They can be used at different levels of the OSI stack
  - IPSec: Internet layer (layer 3)
  - SSL/TLS (SSL is deprecated): application layer
- In embedded devices, hardware support for encryption algorithms is fundamental

Process isolation
Having the ability to isolate different processes (one process cannot access the resources of another one ore cannot access specific resources/network interface) inside the device, may make the system safer.
ARM processors implement **Trustzones** which is a process isolation feature.

Malware Detection (quick read)

- Still an open research problem for IoT, at least in the general case
  - Where to perform detection (edge, fog, cloud)?
  - How to perform detection?
  - How to update the detection methods to detect new malware families?
- [illegible]o (dire[illegible] user interactio[illegible]mplies [illegible]t false positives must [illegible]extremely un[illegible]y, yet, [illegible]etectio[illegible]erformance m[illegible]be goo[illegible]
- [illegible]What to [illegible] after detecti[illegible]
  - Dep[illegible]s on the syste[illegible]d on the [illegible]alware family
  - In I[illegible]eboot of the dev[illegible] usually [illegible]ves the problem
  - Pro[illegible]s isolation can b[illegible]sed to c[illegible]ne the malicious process
  - Process killing is also a possibility

Software vulnerabilities and IoT Devices (quick read)

- Dealing with vulnerabilities (and bugs) usually means updating software
- May be done (provided that devices support it) through Over The Air (OTA) updates
  - Otherwise, it should be done by hand...
- May not be easy/feasible:
  - Devices may be in remote locations with limited network connectivity
    - E.g., extremely limited bandwidth
  - Devices may be equipped with very limited energy
    - OTA can be expensive
- Even when feasible, updates may be risky

Privacy (quick overlook)
By having IoT devices, designing for privacy is complex. (now follows an example, the other slide are almost skipped by professor). Think about devices deployed around a city (example some cameras): there are different problems:
- how do you notify users of the presence of a camera
- consent: how can I provide consent for being filmed

Also it has been shown that anonymising data may not being that effective. Anonymity can be broken easily by knowing few external informations.

Privacy, is there any?

- Most used technique to collect data and protect privacy: data anonymization
- Is anonymization effective?
  - By using known characteristics of a user, data can be de-anonymized, e.g.:
    - By locating four times an user in one year, it was possible to extract complete user location information from an anonymized database of 1.5 million mobile phone users [3]
    - By using data emitted from accelerometers of different devices, it is possible to correlate multiple persons and use the location of one of them to compute the location of the others [4]