OVER TECHNOLOGIES
The smart spaces makers

Introduction to home and building automation systems

In collaboration with:

CIS SAPIENZA
Cyber Intelligence and Information Security

DOMOTICA.it
il portale italiano dell'automazione domestica

Mariano Leva
leva@dis.uniroma1.it

Seminar Schedule

- Lecture 1: introducing intelligence in a houses
  - what is an home automation system
  - an example of home automation systems: SM4All & GreenerBuilding
  - layers of an home automation system
    - the actuation devices(relays, dimmer and serial port)
    - the communication layer (I2C, CAN, Ethernet, KNX, EDS)
    - the control layer (choosing the right hardware)
    - the interface layer (polling, piggyback polling and comet)

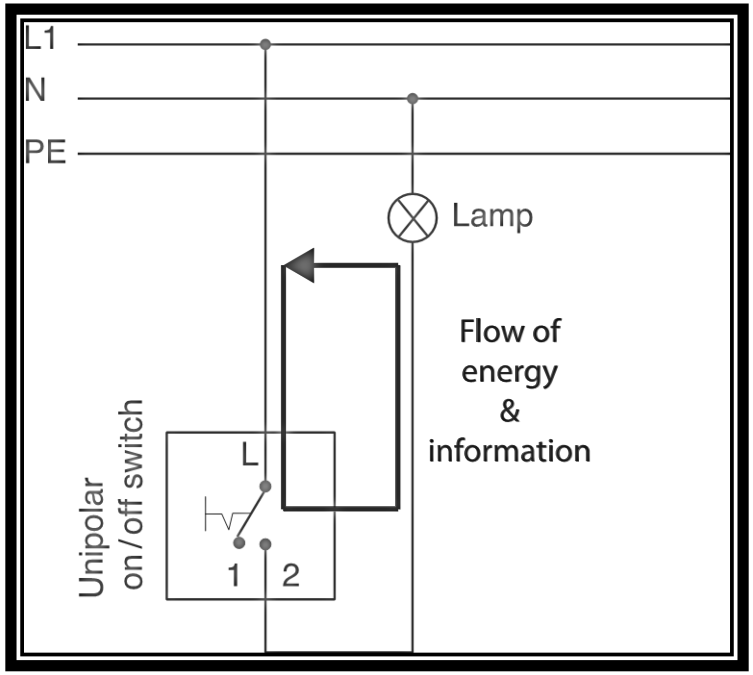
What is an home automation system

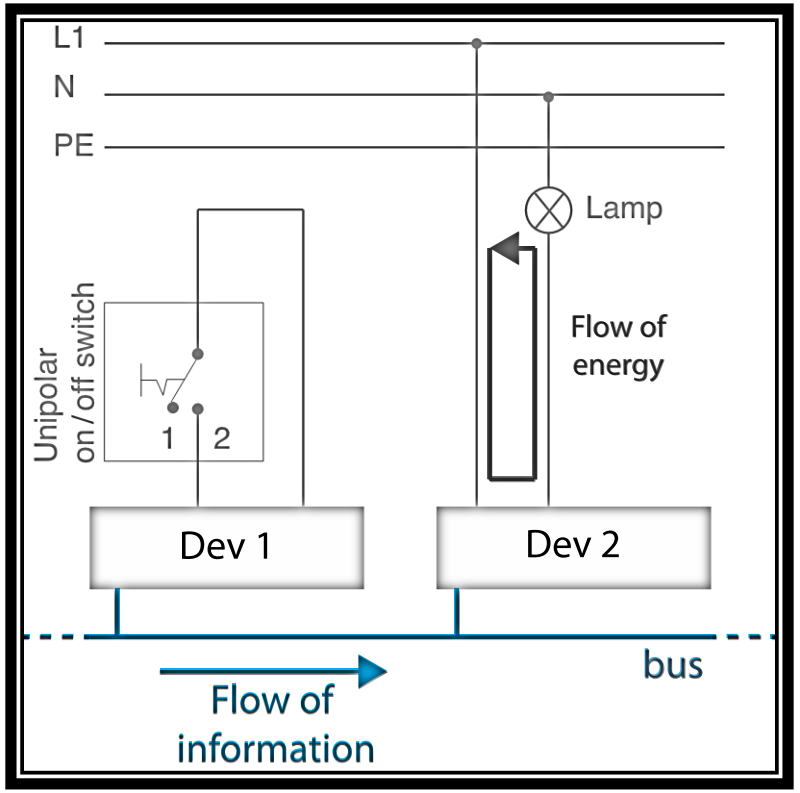
- for home automation system we mean each technology voted to the automation of the house
- in this sense...
  - ...the garden irrigation clock...
  - ...the thermostat...
  - ...the electrical shutters…
- ...are all example of easy home automation
- **question: are all of the houses equipped with home automation system?**

What is an home automation system

- **answer: no**
- in general for **home automation system** we mean an environment in which the actuation and the control are decoupled **at least** by a firmware layer
- a classic electric circuit has a wired (**hardware**) connection between control and actuation, then it **cannot** be classified as “home automation”
- in the following we will use the term “home automation system” or “domotic” in compliance to the definition above

Conventionl electrical circuit VS home automation

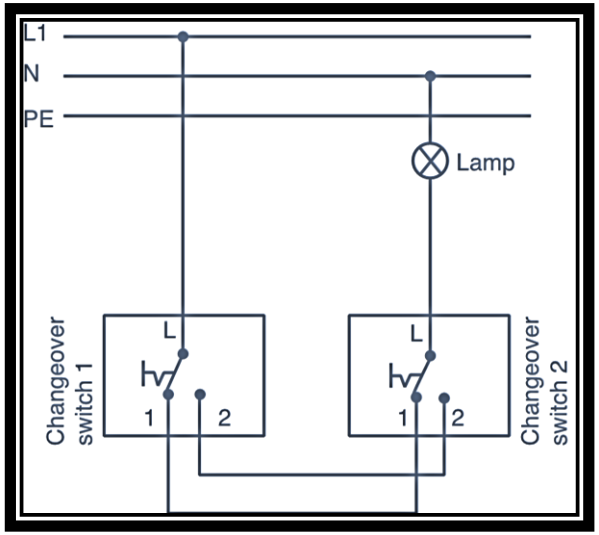




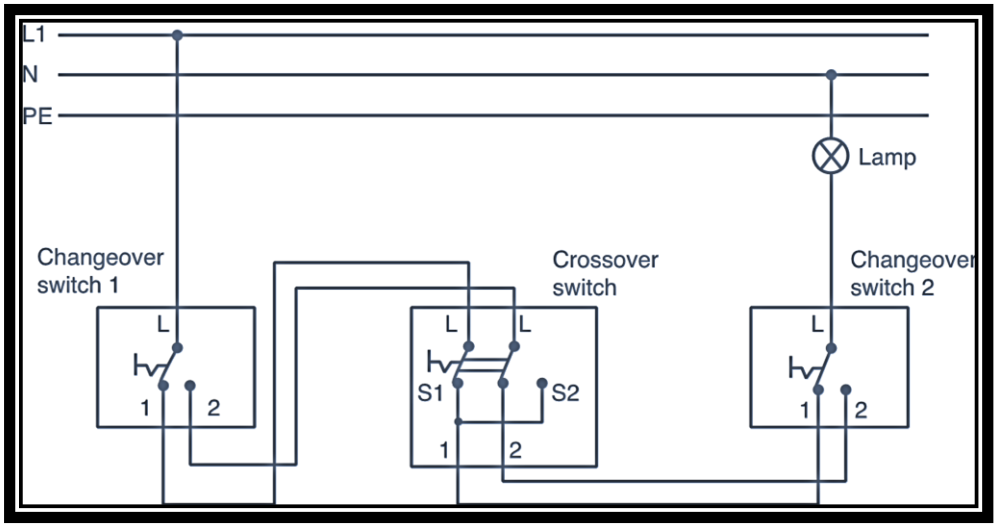
Flow of information and flow of energy
are not separated

Conventionl electrical circuit VS home automation

Turn on/off a lamp
from 2 points

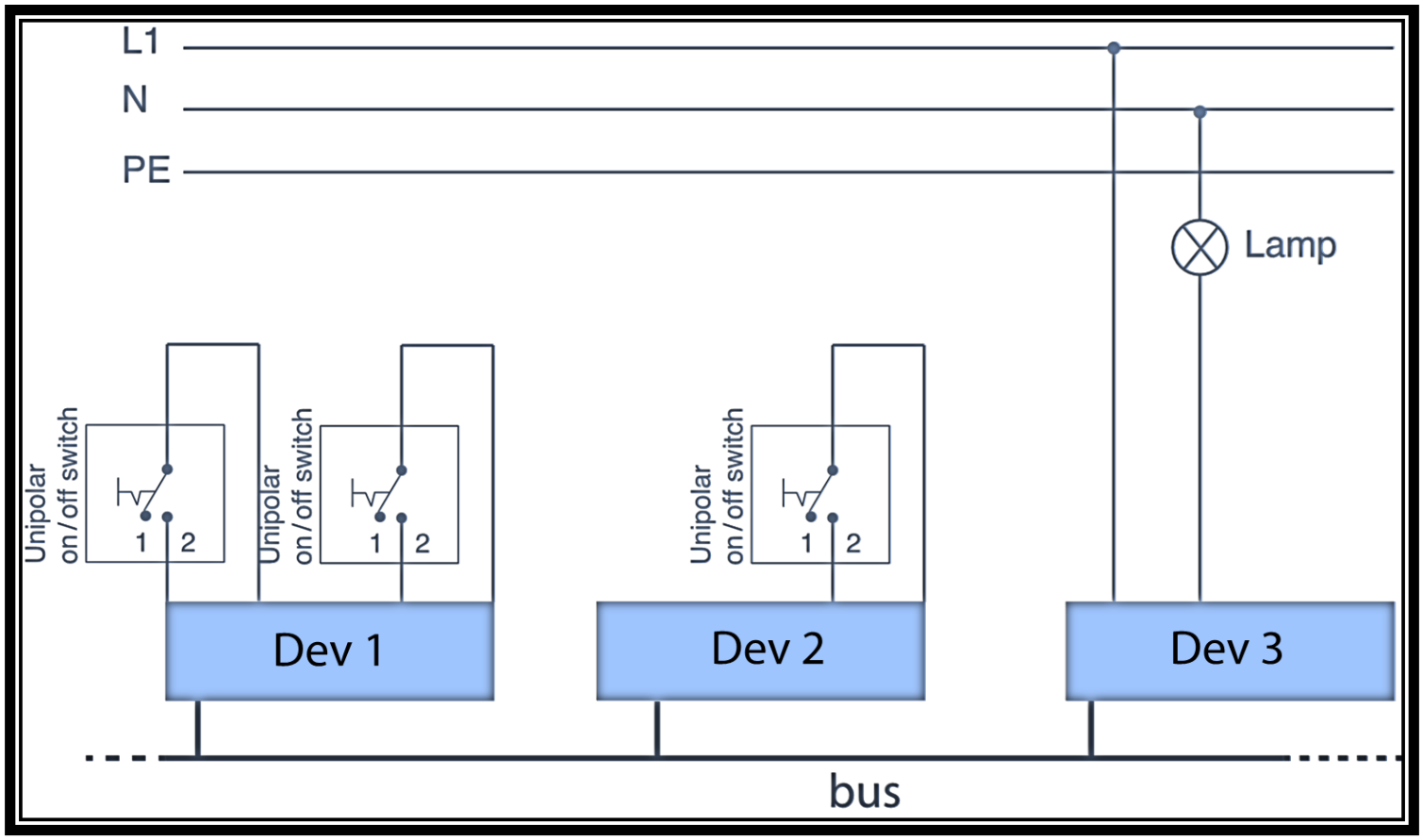


Turn on/off a lamp
from3 points



Again flow of information and flow of energy are not separated

Conventionl electrical circuit VS home automation

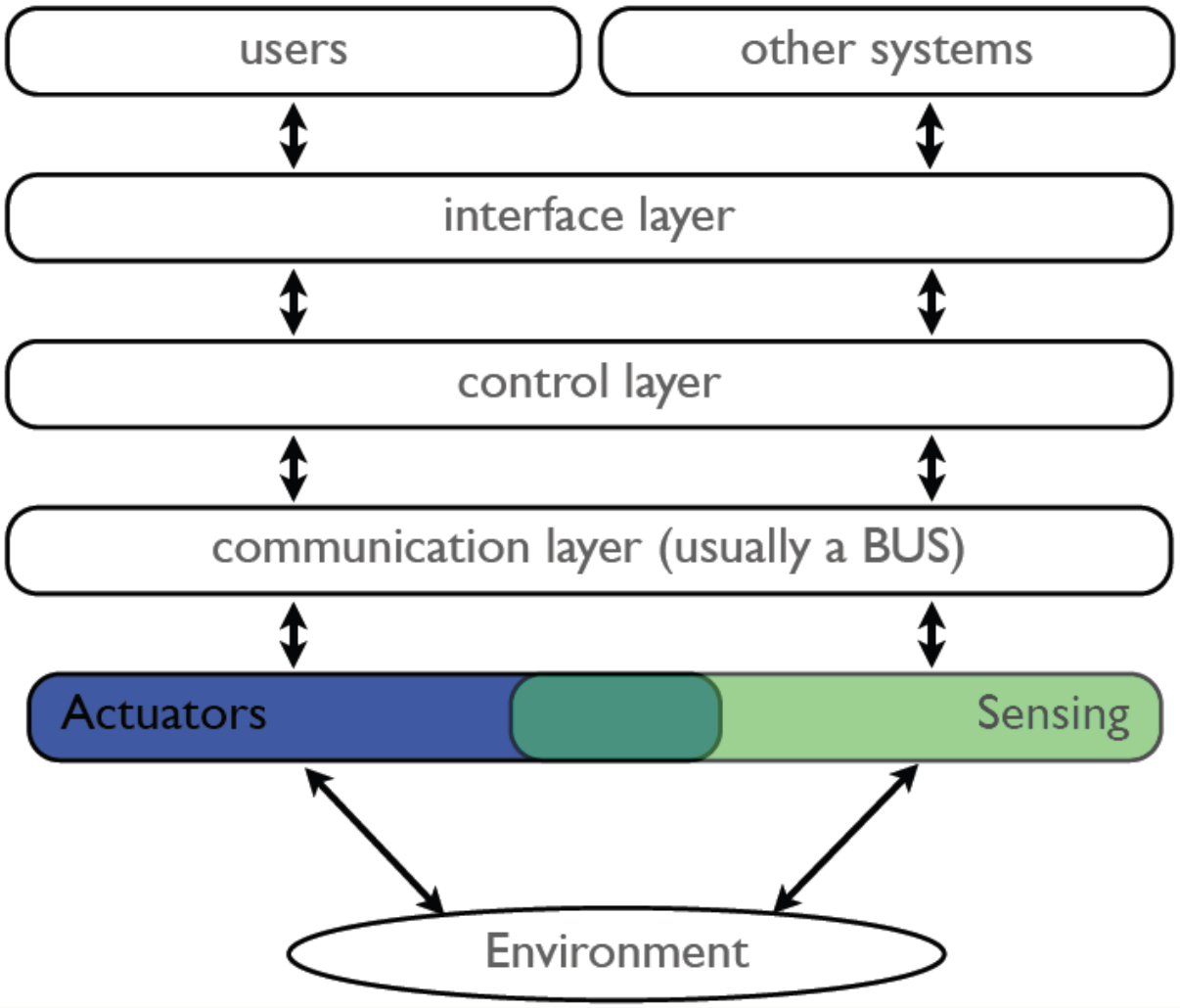


Uniform connection principle

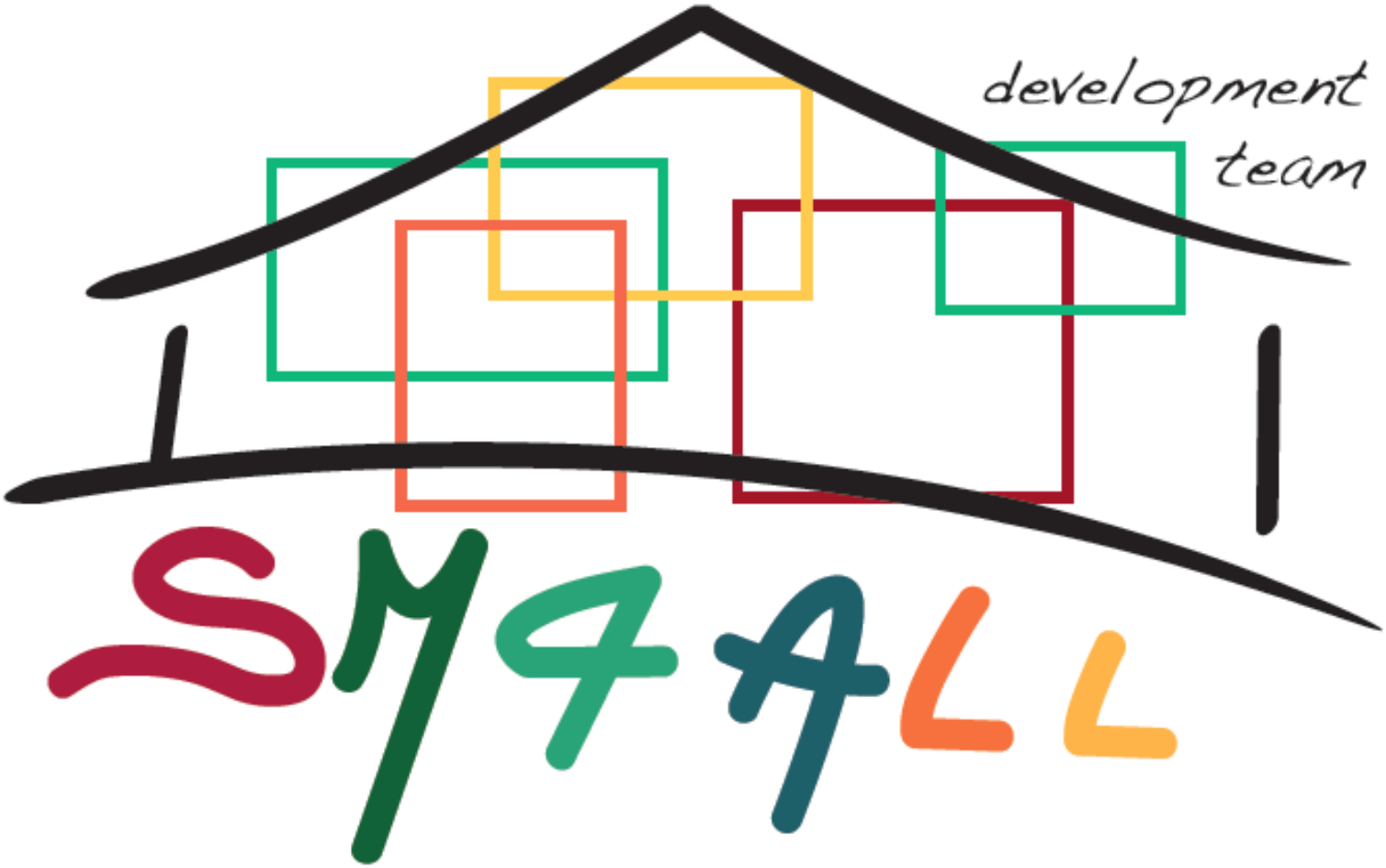
Challenge and future direction

- Yesterday:
  - interfaces: buttons and switches
  - integrated systems: ---
- Today:
  - interfaces: buttons, switches, touch panel (wired and wireless)
  - integrated systems: audio/video (sometimes), thermal control, windows and shutters, ad-hoc interconnection
- Tomorrow (challenges):
  - interfaces: reduce as possible the needs of buttons, switches and touch interfaces, the house must predict what the user wants
  - integrated systems: potentially everything
  - **...the house accessible as-a-resource by using a standard library e.g. “import world.myhouse”**

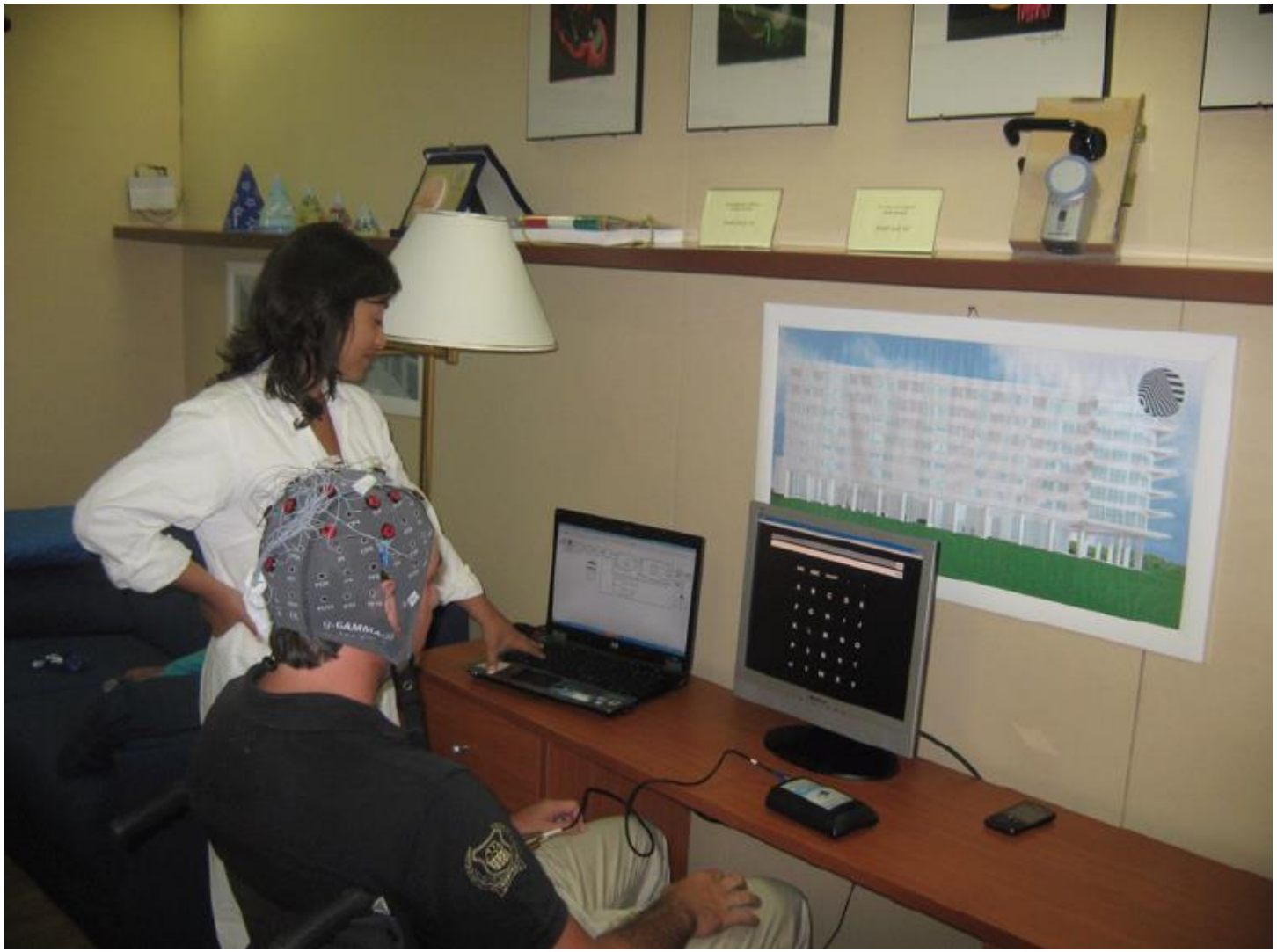
Layers of an home automation system



An example of an home automation system: SM4All

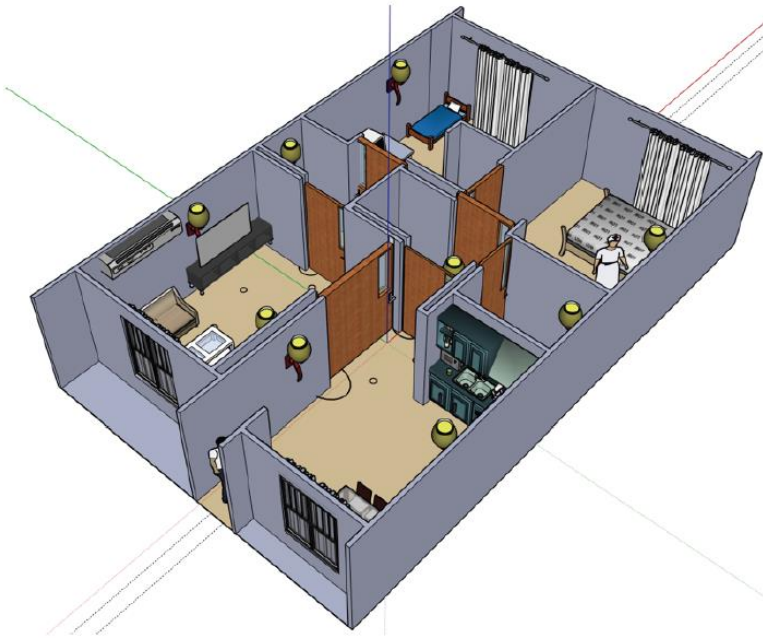


An example of an home automation system: SM4All

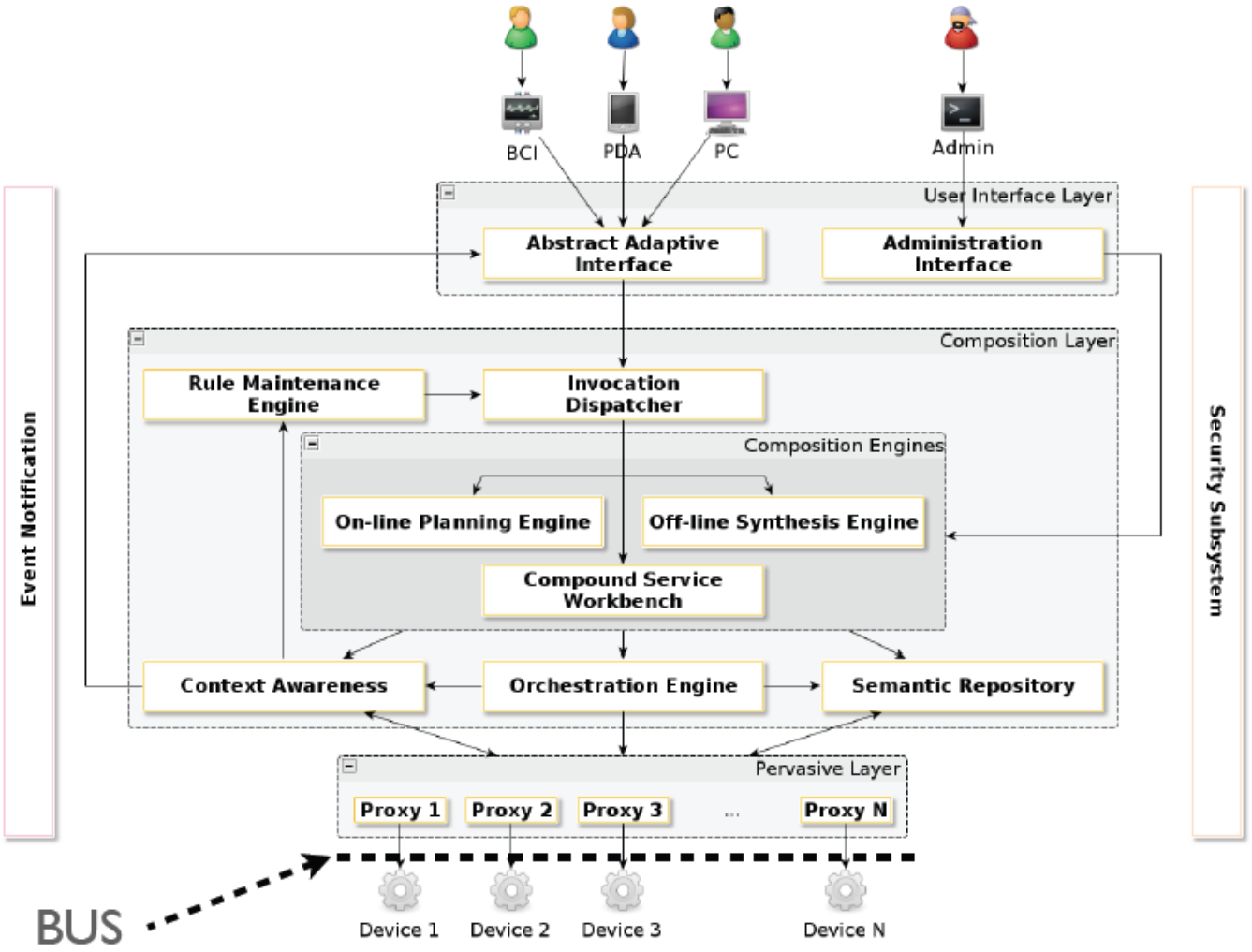


An example of an home automation system: SM4All

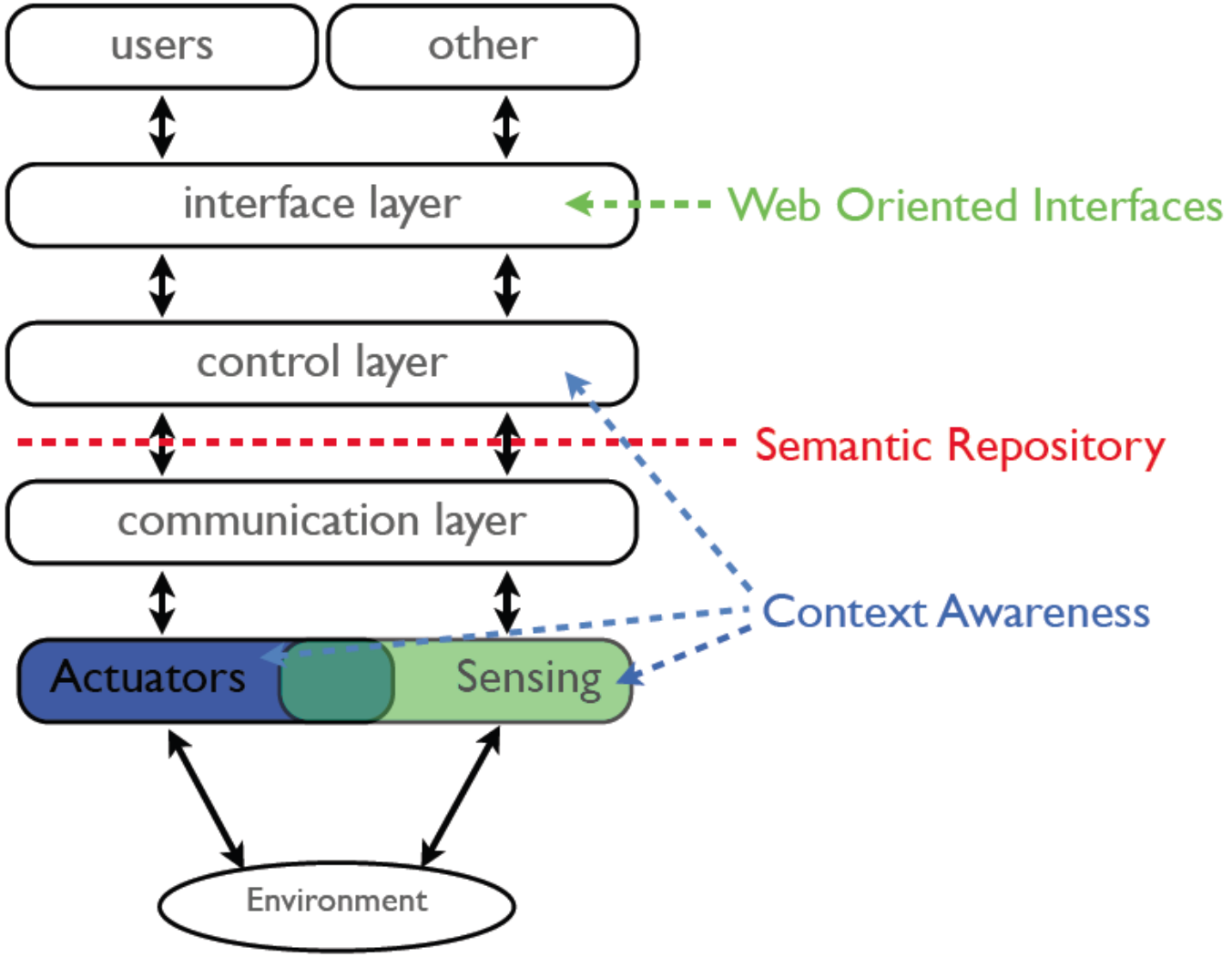
An example of an home automation system: SM4All



An example of an home automation system: SM4All



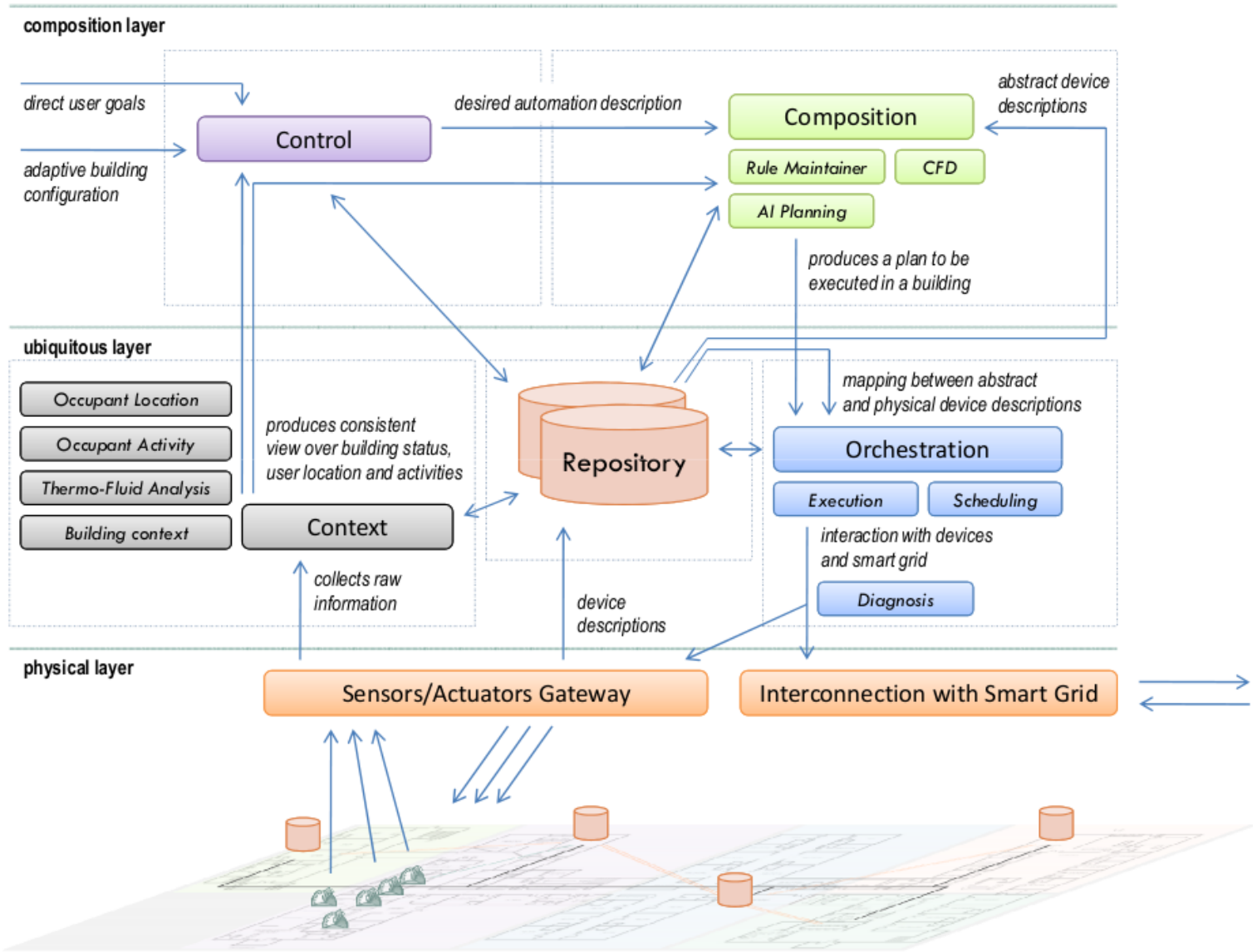
Fundamental concepts of SM4All



An example of an home automation system: GreenerBuildings

GREENERBUILDINGS

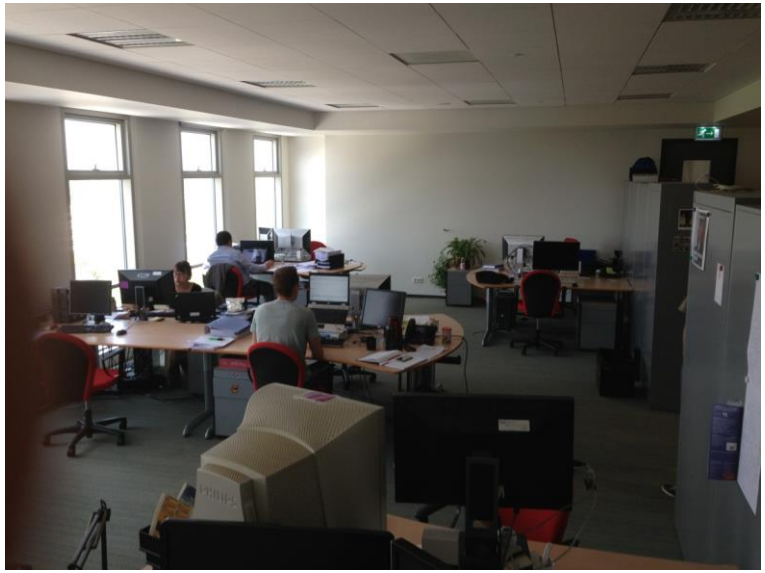
GreenerBuildings architecture



GreenerBuildings deployment & showcases

- Two living labs:
  - an office room and a meeting room in the Potentiaal building in TU/e campus
  - an open space environment in the Metaforum building in TU/e campus
- Fine-grained activity monitoring and different showcases
  - *Adaptive lighting* depending on user activities
  - *Adaptive meeting room* (brainstorming or presentations) adjustment of HVAC and light requirements
  - *Ubiquitous personalization*. User is able to see control possibilities of their environment from computer, smartphone and tablet
  - *Background energy saving service.* The framework enables autonomous lighting system to address occupant comfort and facility manager saving needs

GreenerBuildings deployment



OVEL TECHNOLOGIES
The smart spaces makers

Challenge and future direction

- Research interests:
  - location component
    - follow-me function (open issue...)
  - find the right trade-off between the buttons and the touch interfaces
  - open source solution for house interconnection
  - efficient energy-sharing and energy-management
  - space management for office-oriented house

- Drivers:
  - the users are scared about the technology: they don’t want to be controlled by a computer
  - the users don’t want “something more”, they prefer “something known” with some “intelligence”

Challenge and future direction

- Nice fast test: which is your favourite functionality?
  - a. open/close with a single button all the shutters
  - b. switch-off all the lights
  - c. control the house status from outside

- The 70% of the users answered “b”...so: the more the functionalities are simple and the more the users will appreciate them

A generic home automation architecture

the actuation devices

The actuation devices

- an actuator is a device that is **able to modify the environment**

- some examples:
  - a bulb light connected to a switch is an actuator
  - an automatic door is an actuator
  - an electrical shutter connected to a switch is an actuator

- question: **a relay is an actuator?**

The actuation devices

- answer: potentially yes...
  - if the relay is connected to a device (ex. a bulb light) it can be intended as an actuator
  - if the relay is not connected to a device...it is not an actuator

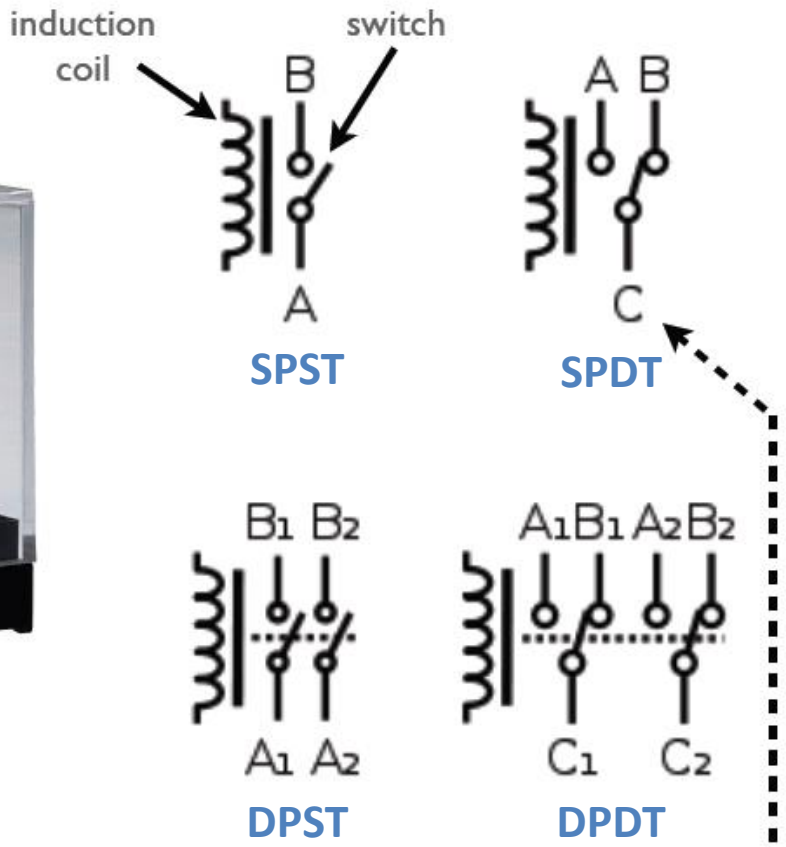
- in general when we talk about relays we intend them connected to devices

The actuation devices

- from a low-level point of view an actuator always contains:
  - one or more relays
  - one or more dimmers
  - a serial port (more generally a communication device)

- relays, dimmers and serial ports represents all of the devices that can be controlled by an home automation system in order to implement actuators

The actuation devices: latching relays

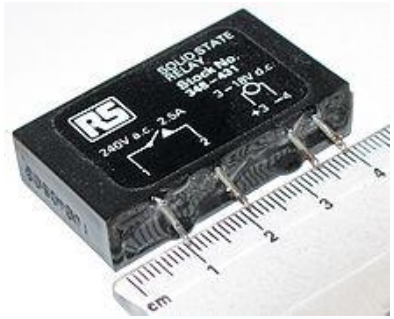


relays usually work with a normally-opened/normally-closed logic but we also have magnetic latching relays

The actuation devices: state solid relays

ADVANTAGES

- No moving parts, they use semiconductor to perform the switch operation
- Faster than electromagnetic latching relays
- Totally silent



SHORTCOMING

- High susceptibility to damage (a relatively high vulnerability to overloads in comparison to electromechanical relays)
- limited switching arrangements (SPST switching)
- when closed, higher resistance (generating heat), and increased electrical noise
- Possibility of spurious switching due to voltage transients

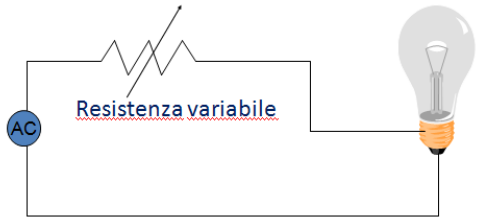
The actuation devices: relays

- a relay is required in order to switch on/off an electrical line
  - relays are useful because they use a low-voltage controller (tipically 12V) to interrupt an high voltage one (220V or higher)

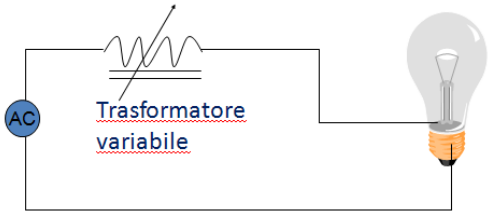
- relays are the most common device for automation due to several reasons:
  - they are inexpensive
  - they are reliable (large MTTF, the first patent of a relay is of **1840**, they born with the telegraph)
  - they can have a very reduced size
  - the energy consumption of each relay is practically zero (and really zero in the case of magnetic latching relays)

The actuation device: dimmers

- a dimmer is an electric regulator used to change the lux level
- it was invented in the 1961 by Joel Spira founder of Lutron Electronics
- a dimmer can be implemented in different ways:
  - rheostats-based dimmer
    - Increasing the resistance, the voltage on the lamp decrease
    - Not efficient, dissipate power of the load as heat



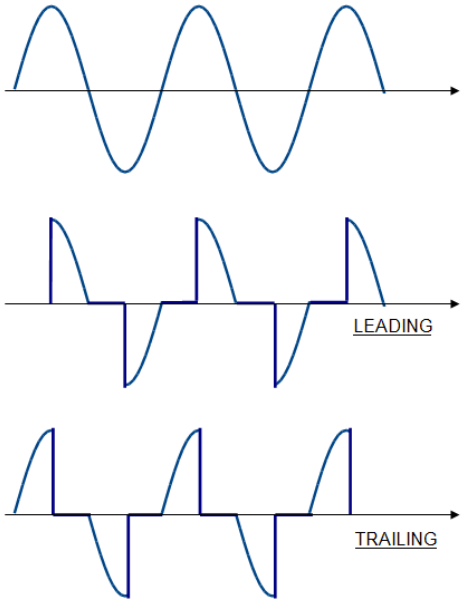
  - autotransformer-based (variac)
    - dimming by adjusting the output voltage for a steady AC input voltage
    - Heavy and unwieldy

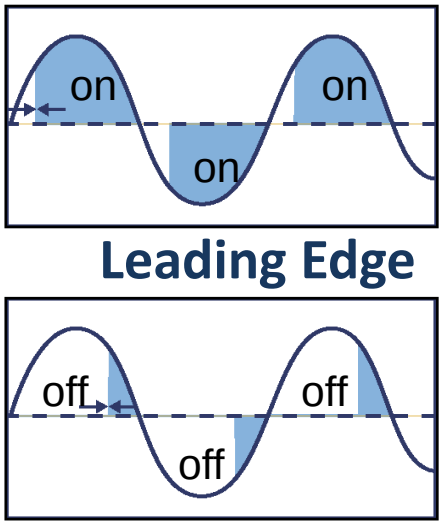


The actuation devices: a very easy variac-based dimmer

The actuation devices: dimmers

- a dimmer can be implemented in different ways:
  - Triac based dimmer -They operate stopping the voltage to the lamp that reduce the available voltage, realizing the dimming
    - more efficient – the voltage is interrupted and not absorbed
    - Different modalities for the dimming

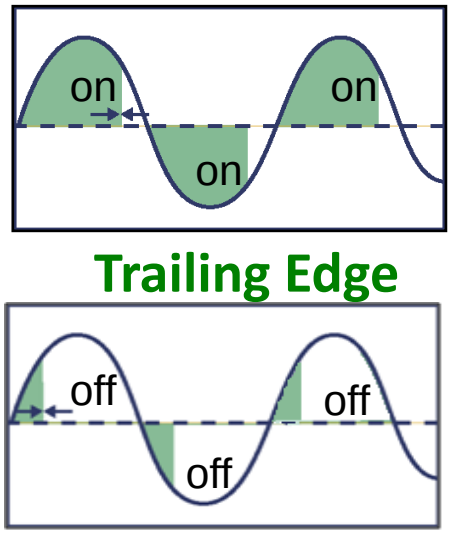




Leading Edge

80%

20%

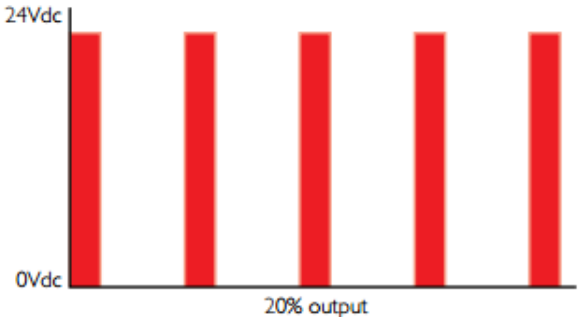


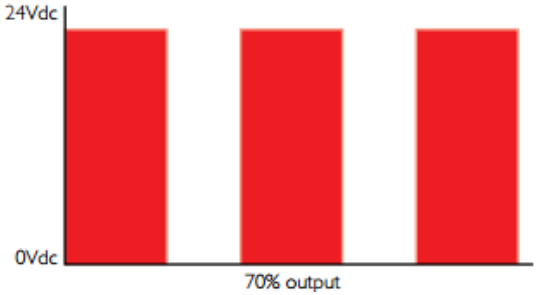
Trailing Edge

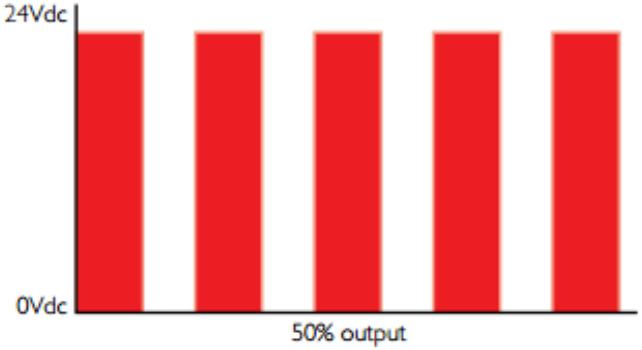
  - These three different modalities are used for different types of loads

The actuation devices: dimmers

- PWM: is a rapid succession of switching on and off the lamp power. Used in DC voltage





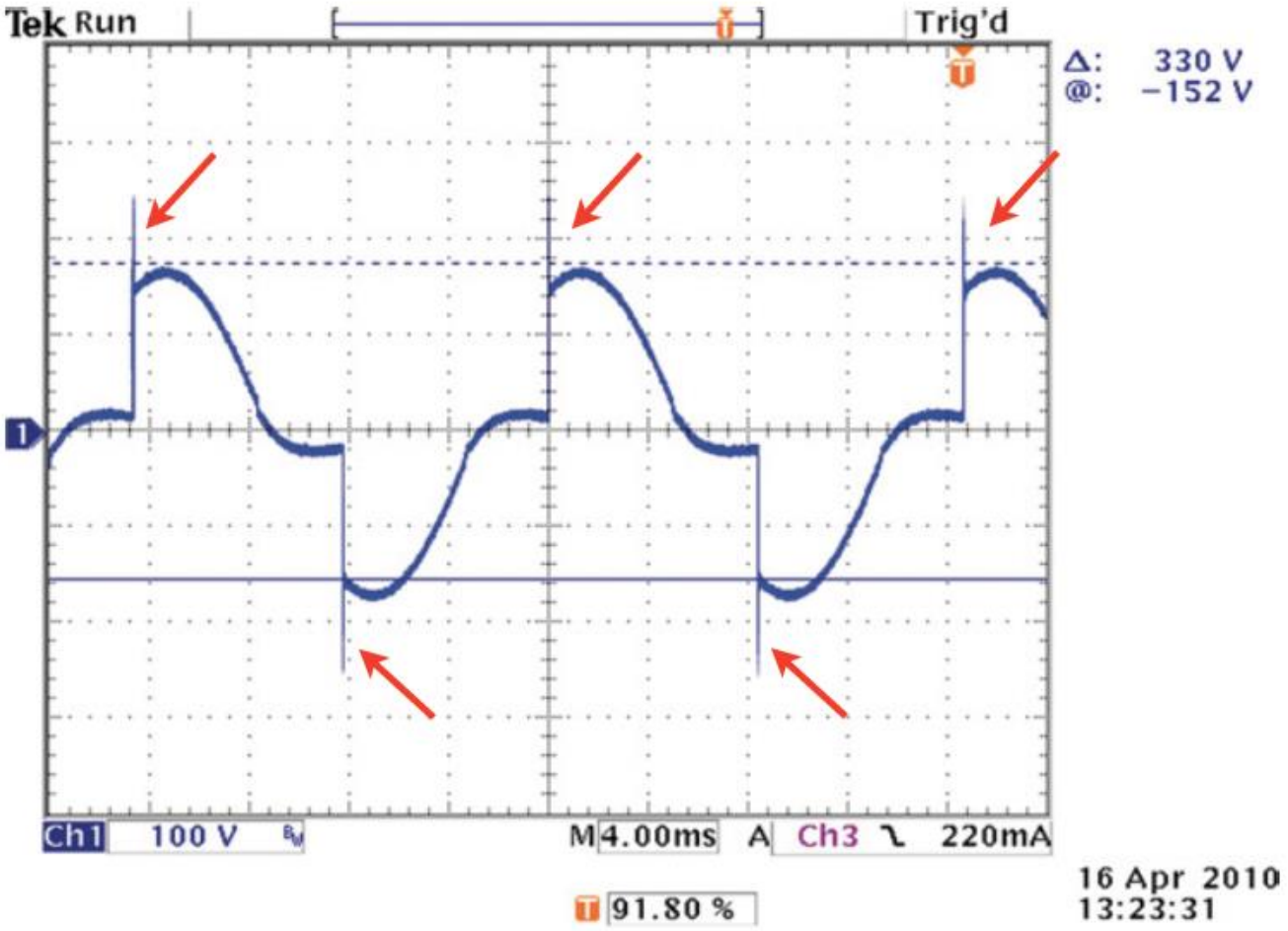


The actuation devices: dimmer issues

- **variac-based dimmers** offer best performance but unfortunately they cannot be used for home automation due to:
  - the large size (and weight...)
  - the mechanic regulation

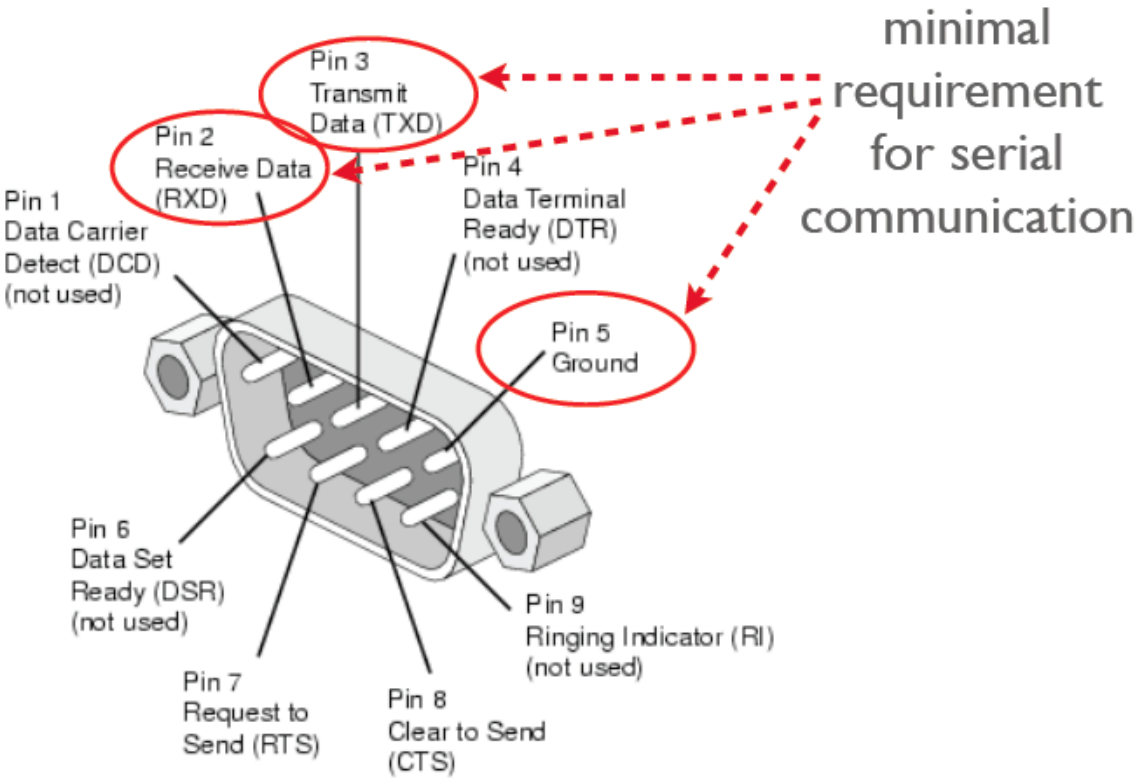
- **triac-based dimmers** can be used for home automation but...
  - they introduce electrical noise (see next slide) - LC filtering circuits are required, they are usually larger part of the dimmer
  - they are usually designed for specific load (if the user change it they can have faults)
  - they can introduce problems with the electricity meters (because they introduce some peaks during the triac commutation) - **this is so rare**

The actuation devices: dimmer electrical noise



The actuation devices: serial port

- is a standard
- widely supported by programming language
- Commonly, its parameters indicated by means of a conventional notation (e.g. 8/N/1 meaning 8 data bits, no parity, 1 stop bit)



OVER TECHNOLOGIES
The smart spaces makers

A generic home automation architecture

The Communication Layer

The communication layer

- the communication layer is one of the most important part of the automation system because it decides the bandwidth for the data exchange:
  - a **low-frequency** (i.e. limited bandwidth) communication **allows** (i) to use simple wires and (ii) to interconnect the device with a non-fixed architecture **but** it makes impossible to transmit audio/video and complex events
  - a **high-frequency** (i.e. large bandwidth) communication **allows** to transmit whatever (also full-hd streams) **but** it imposes (i) to use a cat5/cat6 cable (€++) and (ii) a fixed interconnection schema

- the potential of each home automation system is always related to the communication layer

The communication layer

- in home automation systems the communication layer is always (often) represented by a BUS
- a BUS is the preferred solution because it dramatically reduces the amount of cable
  - ...even if a large amount of companies do not exploit this advantage..
- a bad bus design example: 3 independent busses for a single home automation system

The communication layer:
a brief overview on some famous bus standards:

- I^2C
- CAN
- Ethernet
- KNX
- EDS

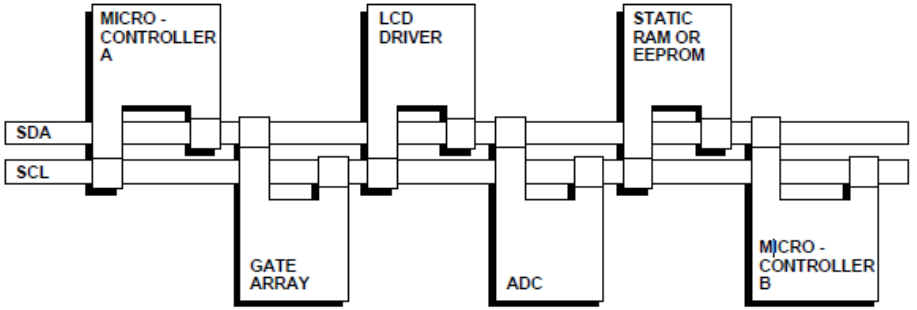
- question: we presents five “wired” protocols...why don’t we care about wi-fi protocols?

- possible answer**S**: reliability? ...security? ...or both?

The communication layer: I²C (Inter Integrated Circuit)

- I²C was developed by Philips in 1982 (it becomes free from 2006)
- it was designed for micro-controllers interconnection, so it works for very **limited extensions** (less than a meter)
- it was originally developed for low-cost application
- speed: from **10 Kb/s to 400 Kb/s,** 5 or 3.3 V, 7-10 address space
- I²C is characterised by two bidirectional lines:
  - **SDL**: serial data line
  - **SCL**: serial clock line
- Nodes are divided in **masters** and **slaves**
  - masters nodes can write on the Serial Clock Line (SCL) and also on SDL
  - slaves can only write on SDL

    only after a master invocation



The communication layer: I^2C

- the protocol:
  - a start bit followed by the 7-bit address of the slave it wishes to communicate with plus a single bit representing whether it wishes to write(0) to or read(1) from the slave.
  - If the slave exists on the bus then it will respond with an ACK bit for that address (master must receive 0 on SDA for a clock round)
  - If the master wishes to write to the slave then it repeatedly sends a byte with the slave sending an ACK bit.
  - If the master wishes to read from the slave then it repeatedly receives a byte from the slave, the master sending an ACK bit after every byte but the last one.
  - The master then either ends transmission with a stop bit, or it may send another START bit if it wishes to retain control of the bus for another transfer

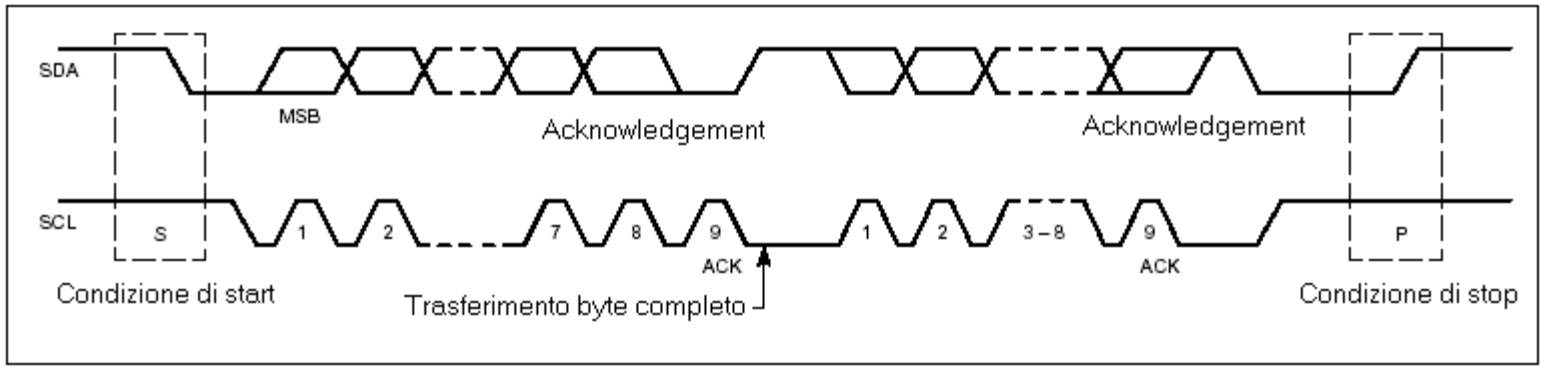


Fig.1

The communication layer: I^2C arbitration

- Arbitration occurs very rarely, but is necessary for proper multi-master support.
- Each transmitter checks the level of the data line (SDA) and compares it with the levels it expects; if they do not match, that transmitter has lost arbitration, and drops out of this protocol interaction.
- if two masters are writing exactly the same message to the same address the collision will be not discovered, but it doesn't represent a problem, because slave will only see a message

The communication layer: CAN

- CAN bus was originally developed by **Bosh** in **1983**
- CAN bus was designed for **automotive** electronics in order to allows micro-controllers to communicate without an host computer
- CAN bus runs on a **single twisted pair** cable (up to 40 meters)
- speed: 1Mb/s
- CAN is data-frame oriented, if a message is divided in multiple data-frames the devices must arbitrate the BUS for each dataframe
- CAN BUS **implements priority based bus arbitration**: high-priority devices has an address close to zero, low-priority devices the opposite
  - A message consists primarily of an ID (identifier), which represents the priority of the message, and up to eight data bytes.

The communication layer: CAN

- The devices that are connected by a CAN network are typically sensors, actuators and other control devices

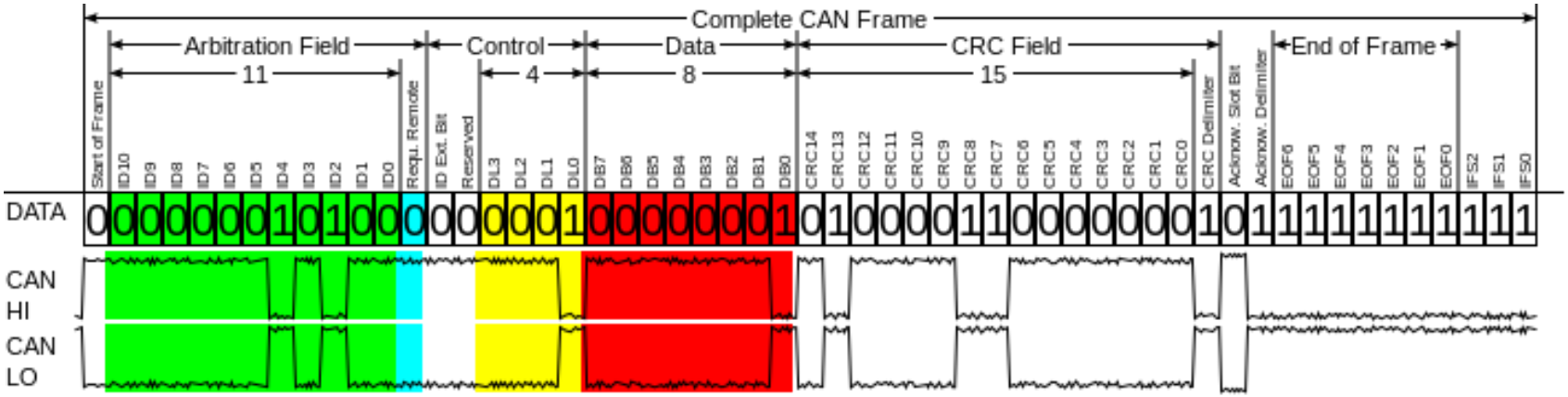
- These devices are not connected directly to the bus, but through an *host processor* and a *CAN controller*
  - The host processor decides what received messages mean and which messages it wants to transmit itself.
  - The CAN controller stores received bits serially from the bus until an entire message is available, which can then be fetched by the host processor
  - The host processor stores its transmit messages to a CAN controller, which transmits the bits serially onto the bus.

- Each node also requires a *Transceiver*
  - Receiving: it adapts signal levels from the bus to levels that the CAN controller expects and has protective circuitry that protects the CAN controller.
  - Transmitting: it converts the transmit-bit signal received from the CAN controller into a signal that is sent onto the bus.

The communication layer: CAN

- the protocol:
  - **transmission**: once a device hears the bus clear it can start to transmit the message.
  - **reception**: once a device hears the start of a transmission it starts to receive it if the message contains its address, the **clock** is "**extracted**" from the message. All the message always starts with a logic 1 and the address of the recipient device
    - Each node in a CAN network has its own clock, and no clock is sent during data transmission
  - **arbitration**: CAN - like I2C - is a CSMA/CA protocol, each node reads and writes the bus at the same time, once it writes 1 and reads 0 it immediately stops to transmit. **This ensure that the device with a lower address obtains the precedence**.
    - in CAN we talk about 0 as "**dominant**" and 1 as "**recessive**" states

CAN: base frame format



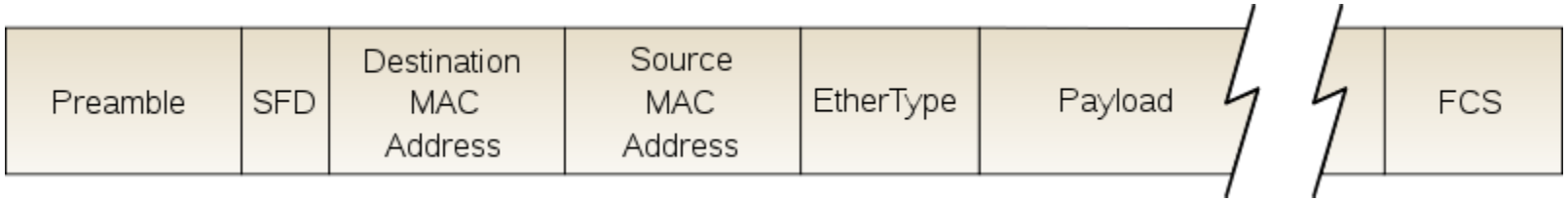
Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier for the data which also represents the message priority
Remote transmission request (RTR)	1	Dominant (0) (see Remote Frame below)
Identifier extension bit (IDE)	1	Declaring if 11 bit message ID or 29 bit message ID is used. Dominant (0) indicate 11 bit message ID while Recessive (1) indicate 29 bit message.
Reserved bit (r0)	1	Reserved bit (it must be set to dominant (0), but accepted as either dominant or recessive)
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

The communication layer: Ethernet [ETH]

- ETH bus was developed by Xerox Parc between the 1973 and the 1974 and it becomes a IEEE standard (802.11) in **1980**
- ETH bus was designed for Local Area Network
- ETH was originally developed for coaxial cable, then the project moved to twisted pairs (tx+/- and rx+/-). A CAT5/6 cable with ETH can cover a distance of **about ~100 meters**
- speed: 10/100 Mb/s (also 1 Gb/s), usually **10Mb/s**
- ETH is a **CSMA/CD** protocol: if a collision is detected both the devices stop and retransmit after a random timeout.
- In order to avoid starvation the random timeout is set up with an exponential increment

The communication layer: Ethernet [ETH]

- Ethernet frame is called *packet,* used to describe the overall transmission unit and includes
  - Preambles
  - Start frame delimiter (SFD)
  - Source and Destination Mac Address
  - Ethernet Type indicating the type of frame. Different frame type have different formats and different payloads
  - The payload section includes any header for other protocol (e.g. Internet Protocol)
  - Frame check sequence is a 32-bit cyclic redundancy check, which is used to detect corruption of data in transit



The communication layer: Ethernet [ETH]

- **question**: why does ETH become so popular?
- **answers**: it was presented two years before I^2C and three years before CAN, moreover it is an IEEE standard and it can run up to 100 meters

- **question**: why does the **automotive prefer CAN** even if ETH is faster?
- **answers**: because CAN is CSMA/C**A** and it implements priority
  - what-if the air-bags are connected using ETH? [...]

The communication layer: Konnex [KNX]

- The standard, starting to be developed in **1990,** is based on the communication stack of EIB but enlarged with the physical layers, configuration modes and application experience of BatiBus and EHS
- KNX is designed to run on a single twisted pair, even if are available extensions for:
  - powerline
  - radio (KNX-RF)
  - infrared
  - ETH (known as KNXnet/IP)
- KNX bus wire can be up to **1.000 meters** (using repeater it can reach 4.000 meters)
- KNX is a **CSMA/CA** bus
- speed: **9.600 bit/s, packets** have a **variable length**

The communication layer: Konnex [KNX]

- There are three categories of KNX devices
  - **A-mode** or "Automatic mode“, devices automatically configure themselves, and are intended to be sold to and installed by the end user.
  - **E-mode** or "Easy mode“, devices require basic training to install. Their behavior is pre-programmed, but has configuration parameters that need to be tailored to the user's requirements.
  - **S-mode** or "System mode“, devices have no default behavior, and must be programmed and installed by specialist technicians.

- KNX uses the *Client/Server* model, dividing the involved entities in:
  - **AR** (*Application Resource*), the device providing the service
  - **AC** (*Application Control*), the entity asking for the service

- A KNX application is composed by *functional block* (FB) operating on shared variable called *data point* (DPT)
  - DPT are standardized (e.g. DPT5 is a 2byte float used to represent temperature values in floating point

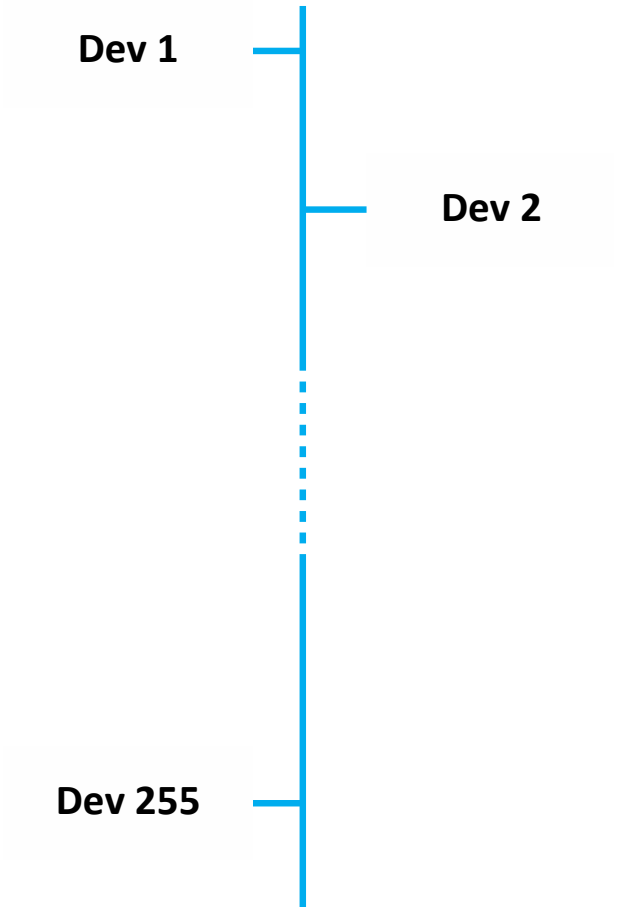
The communication layer: Konnex [KNX]

- in order to develop KNX devices the KNX consortium requires a fee
- KNX consortium is composed by:
  - GIRA
  - **ABB**
  - **AMX** LLC
  - Berker GmbH Co. KG
  - **Bosch** Thermotechnik
  - **Cisco Systems**
  - Control4 EMEA
  - **Creston** International
  - **Daikin** Industries
  - Embedded Automation
  - Jung
  - **Legrand**
  - **Miele** & Cie KG
  - ON Semiconductor
  - Hager
  - **Schneider** Electric Industries S.A.
  - **Somfy**
  - Radiocrafts
  - **Bosch**
  - Russound/FMP Inc.
  - **Siemens**
  - **Toshiba**
  - Uponor corporation

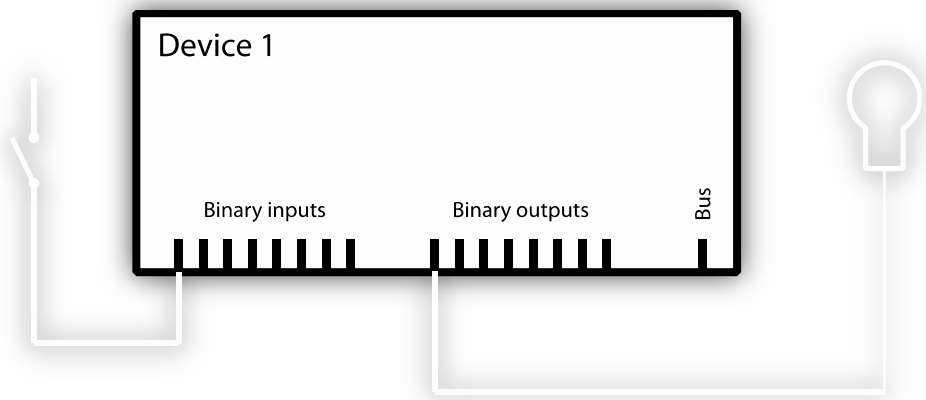
The communication layer: EDS

- EDS was originally developed in **1999** and it is owned by World Data Bus from 2004
- EDS is a **9.600** bit/s **CSMA/CD** protocol
- in EDS the collision detection is implemented via **ACK** messages: the transmitter element does not listen to the bus during the transmission phase
- EDS is designed to work on a single wire, but usually it runs using 2+1 wires such that:
  - 2 wires are dedicated to **Vcc** and **Data**
  - 1 wire is the **ground** reference
    - the house ground system can be used...usually it is better to use a dedicated line
- **twisted pair is not required**, it is possible to use standard - **inexpensive** - wires up to **1.200 meters** (experimental uses show that the protocol works till 1.600 meters without repeaters)

EDS bus system: an overview



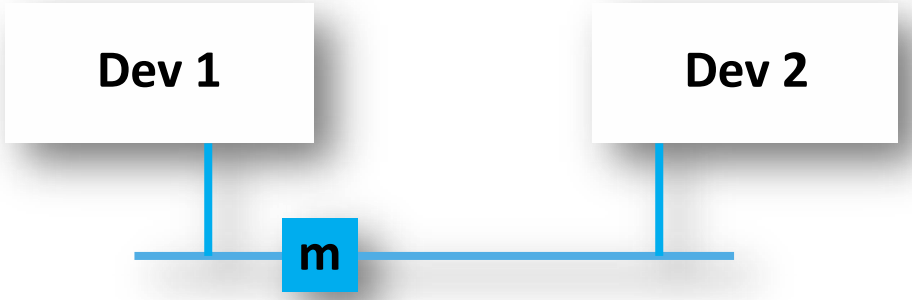
- An EDS bus system may have up to **255** devices connected to the same bus.
- The bus is both a **power supply** and a **data** exchange medium.
- Each device is identified by a unique physical address number, each channel of the devices is identified by a number



- 8-8 devices have
  - 8 binary inputs
  - 8 binary outp

EDS bus system: an overview

Devices interact by exchanging messages on the bus.



Each message is exactly 8 bytes long and has a precise structure.

Byte	Content
Byte 1	Stx
Byte 2	Receiver
Byte 3	Sender
Byte 4	Message type
Byte 5	Info 1
Byte 6	Info2
Byte 7	Checksum
Byte 8	Etx

The communication layer: global recap

	I²C	CAN	ETH	KNX	EDS
max length	1 m	40 m	100 m	1.000 m	1.200 m
speed	up to 400 kb/s	1.000 Kb/s	10.000 Kb/s	9,6 Kb/s	9,6 Kb/s
wire type	on chip	twisted pair	twisted pairs	twisted pair	single wires
arbitration	CSMA/CA	CSMA/CA	CSMA/CD	CSMA/CA	CSMA/CD
costs	medium	medium	medium/low	medium/low	very low

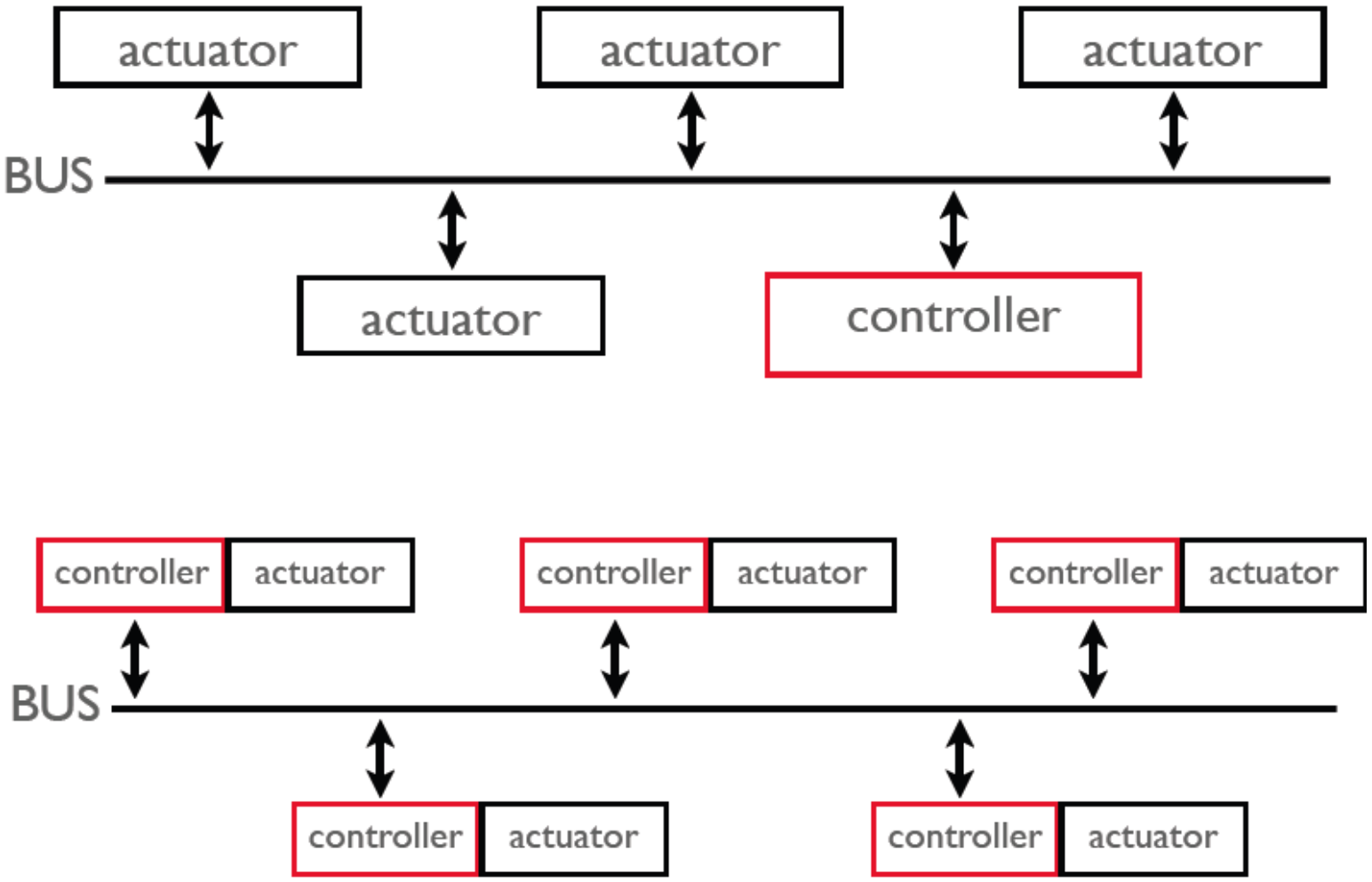
A generic home automation architecture

The Control Layer

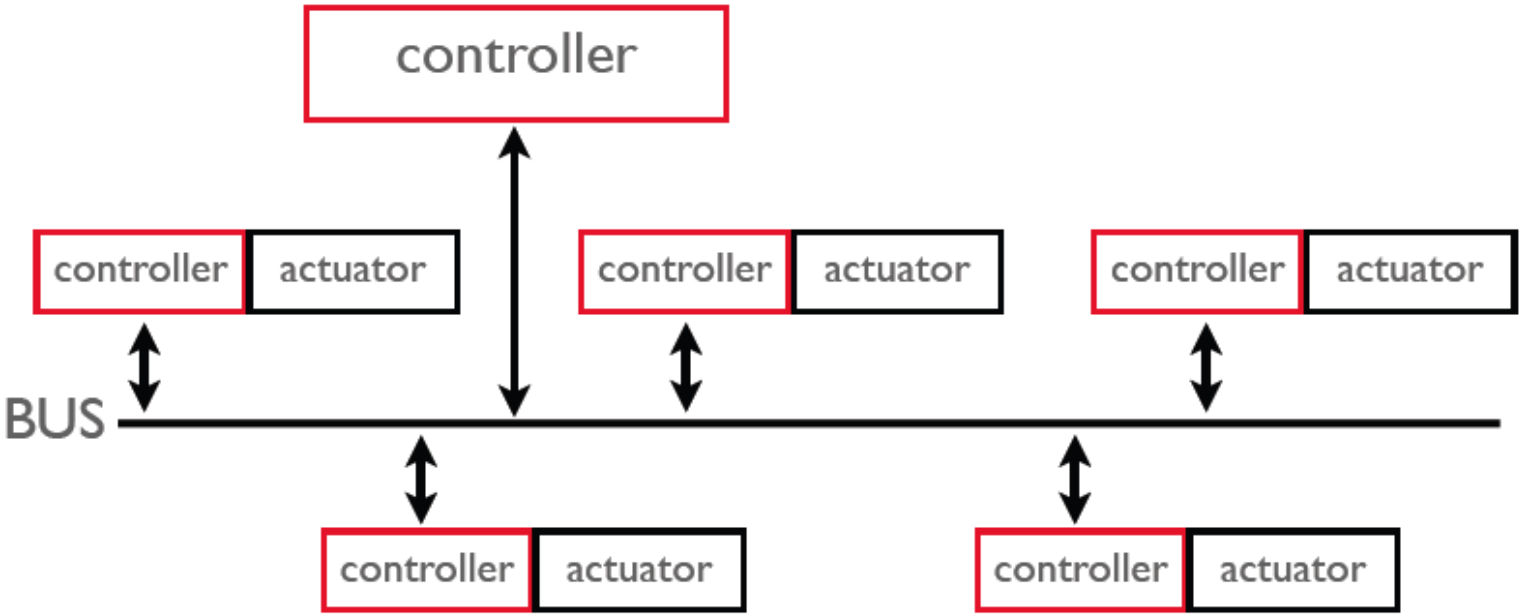
The control layer

- the control layer is the core of an home automation system: it is the responsible of the decoupling between the **control** and the **actuation**
- it can be both **firmware** and both **software**
- it can be **distributed** or **decentralised**

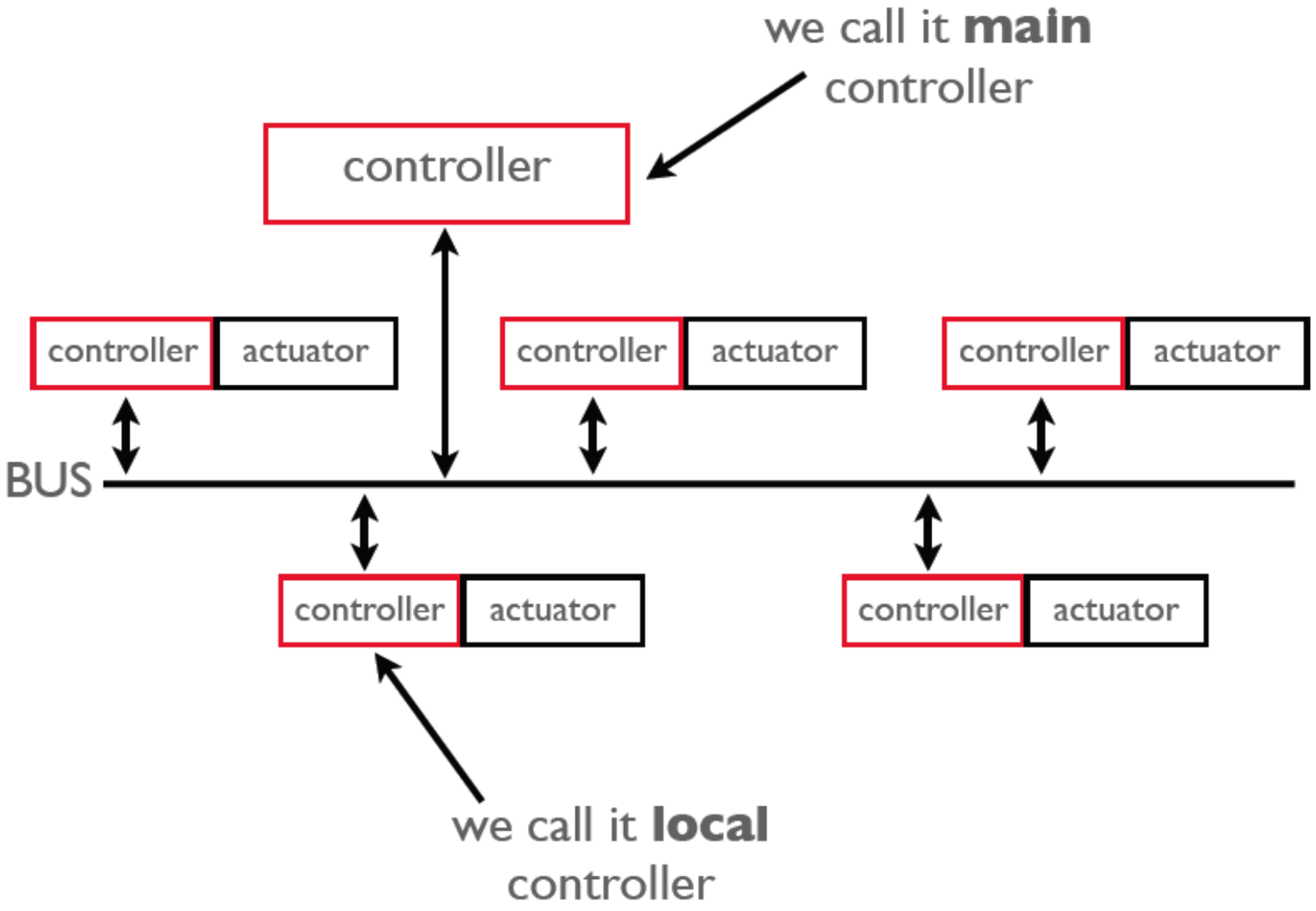
The control layer: possible architectures



The control layer: possible architectures

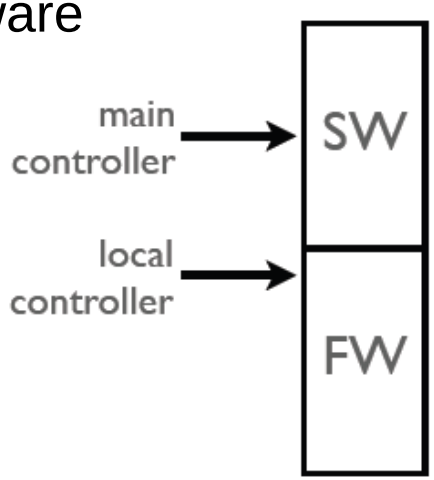


The control layer: possible architectures



The control layer: focus on the main controller

- the main controller is usually a software component
- the local controller can be firmware or software



- How to choose the right hardware?
  - an example driven analysis...
  - Alix 3D3
  - Raspberry Pi
  - Jetway NG74
  - Cubieboard a20

The control layer: PC Engines Alix 3D3

- i86 compatible CPU@500Mhz (AMD Geode)
- 256 Mb RAM
- Flash HDD slot
- fan less
- Connectivity:
  - 1 VGA
  - 2 USB
  - 1 jack audio
  - 1 jack microphone
  - 1 serial
  - 1 ETH
- < 100€ each

The control layer: Raspberry Pi model B

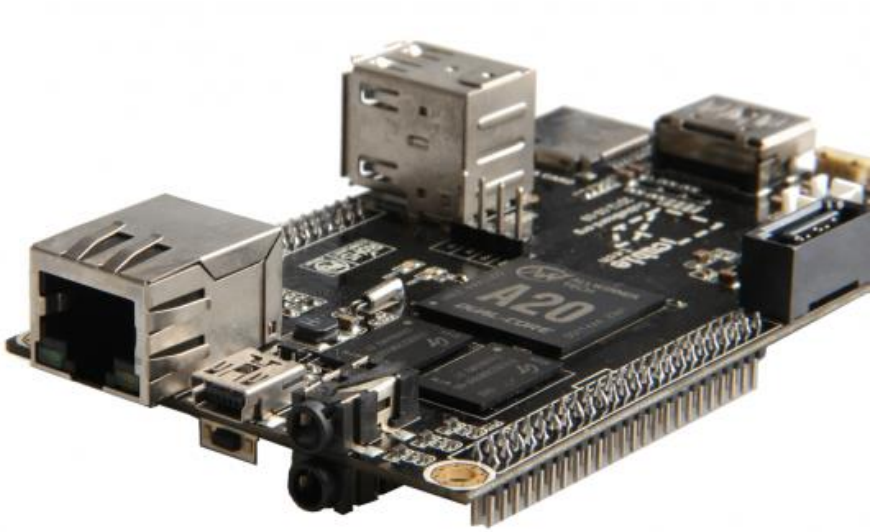
- ARM CPU@700Mhz (ARM 11 family)
- 256 Mb RAM
- SD HDD slot
- fan less
- Connectivity:
  - 1 mini HDMI
  - 2 USB
  - 1 jack audio
  - 1 ETH
  - 1 GPIO (general purpose I/O)
- < 40€ each

The control layer: Jetway NG74-2007

- VIA Nano L2007 CPU@1.6GHz
- up to 8Gb (2 DD3 slots)
- up to TeraBytes (2 serial ATA2)
- Connectivity:
  - 1 HDMI
  - 1 VGA
  - 4 USB
  - 1 ETH
  - 1 serial
  - 3 audio I/O port
  - 1 GPIO
- < 200€ each

The control layer: Cubieboard

- ARM® Cortex™-A7 Dual-Core ARM® Mali400 MP2 Complies with OpenGL ES 2.0/1.1
- 1GB DDR3 @480M
- 3.4GB internal NAND flash, up to 64GB on SD slot, up to 2T on 2.5 SATA disk
- 1x 10/100 ethernet, support usb wifi
- 2x USB 2.0 HOST, 1x mini USB 2.0 OTG, 1x micro sd
- 1x HDMI 1080P display output
- 1x IR, 1x line in, 1x line out
- 96 extend pin interface
- < 70 € each



The communication layer: global recap

	Alix	Raspberry	Jetway	Cubieboard
CPU	500 MHz (AMD Geode)	700 Mhz (ARM 11)	1.6 GHz	Dual-Core 1.2GHz (ARM 7)
RAM	256 Mb	256 Mb	Up to 8 Gb	1 Gb
HDD	FLASH card	SD card	2 serial ATA	4Gb Nand Flash
Serial Port	yes	no	Yes	no
GPIO	no	yes	Yes	yes
Cooling	Fan less	Fan less	Fan based	Fan less
Cost	< 100 €	< 40 €	< 200 €	<70 €

The control layer: how to choose the right software

- the goal is to implement the main controller, the local controllers should be firmware
- the main controller have to:
  - deal with the communication layer
  - deal with the interfaces: both buttons, touch and BCI: a full-OS is required
  - include more complex functionalities (e.g. fault detection and isolation module, energy saving functions etc.)
- Linux is the favourite choice for the operating system
- C or Java as programming languages
  - C it is platform-dependent and it is not so easy to integrate with other software
  - Java has a large variety of libraries and software ready-to-use but it has worse performance than C

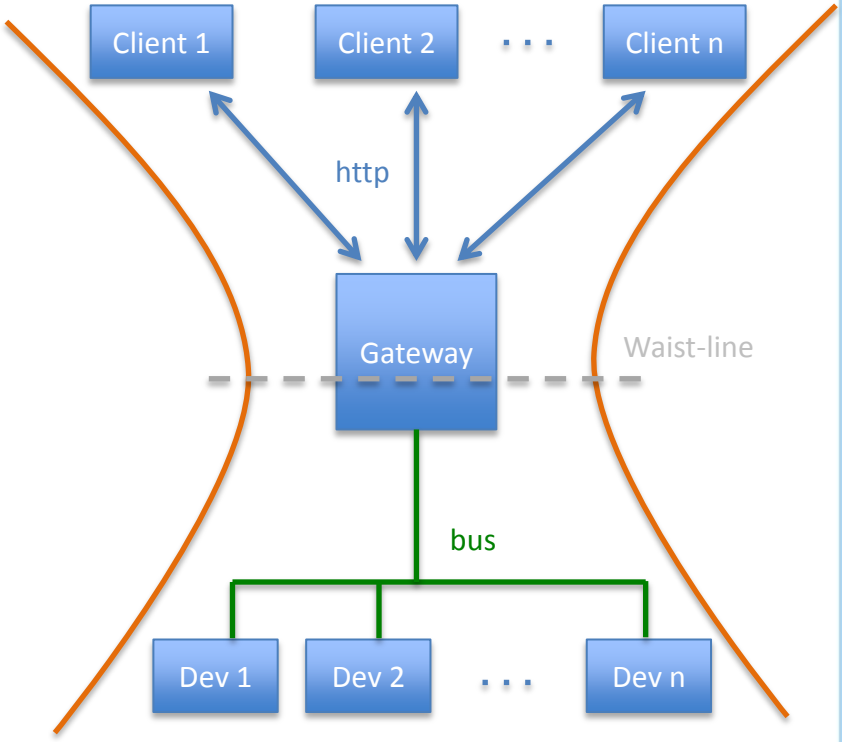
OVER TECHNOLOGIES
The smart spaces makers

A generic home automation architecture

The Interface Layer

The Interface layer

- We want to offer to the user the opportunity to command his house from smartphone, tablet, smart-tv and so on
  - We need to develop a web application and we need a lightweight web server where deploy it
- Different modalities to offer real time interaction
  - *Polling*
  - *Piggyback polling*
  - *Comet*

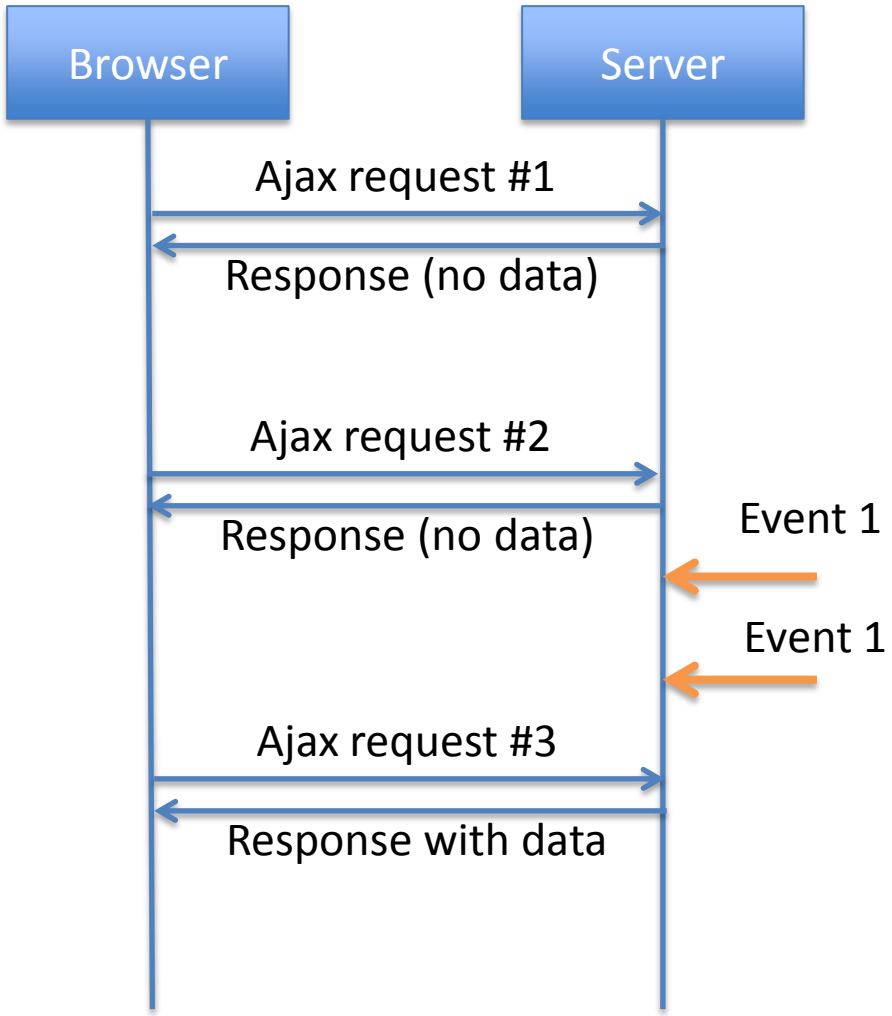


The Interface layer: asynchronous notification to the client

Polling

The browser makes a request of the server at regular and frequent intervals to see if there has been an update. It's like a 5 year old in the back of the car shouting 'are we there yet?' every few seconds.

To get the server events as soon as possible, the polling interval must be as low as possible. *Drawback*: if this interval is reduced, the client browser is going to issue many more requests, many of which won't return any useful data, and will consume bandwidth and processing resources for nothing.



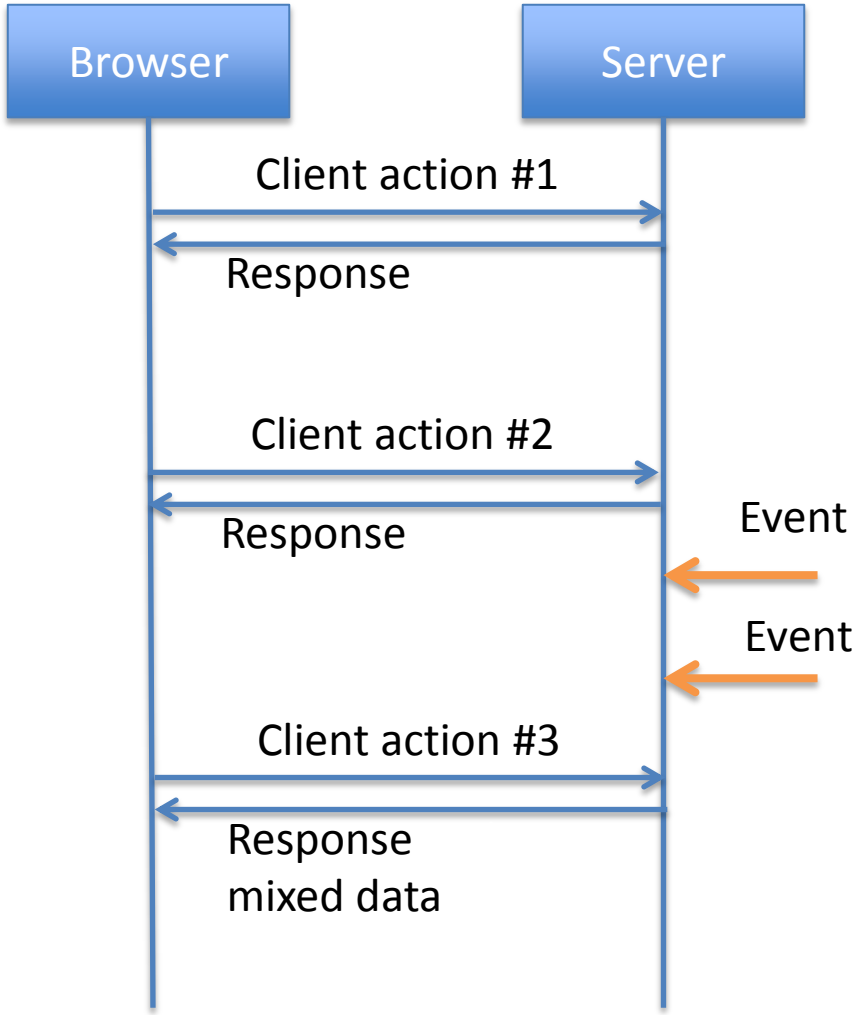
The Interface layer: asynchronous notification to the client

Piggyback polling

The server, having an update to send, waits for the next time the browser makes a connection and then sends it's update along with the response that the browser was expecting.

- Is a much more clever method than polling since it tends to remove all non-needed requests (those returning no data).

- There is no interval; requests are sent when the client needs to send a request to the server



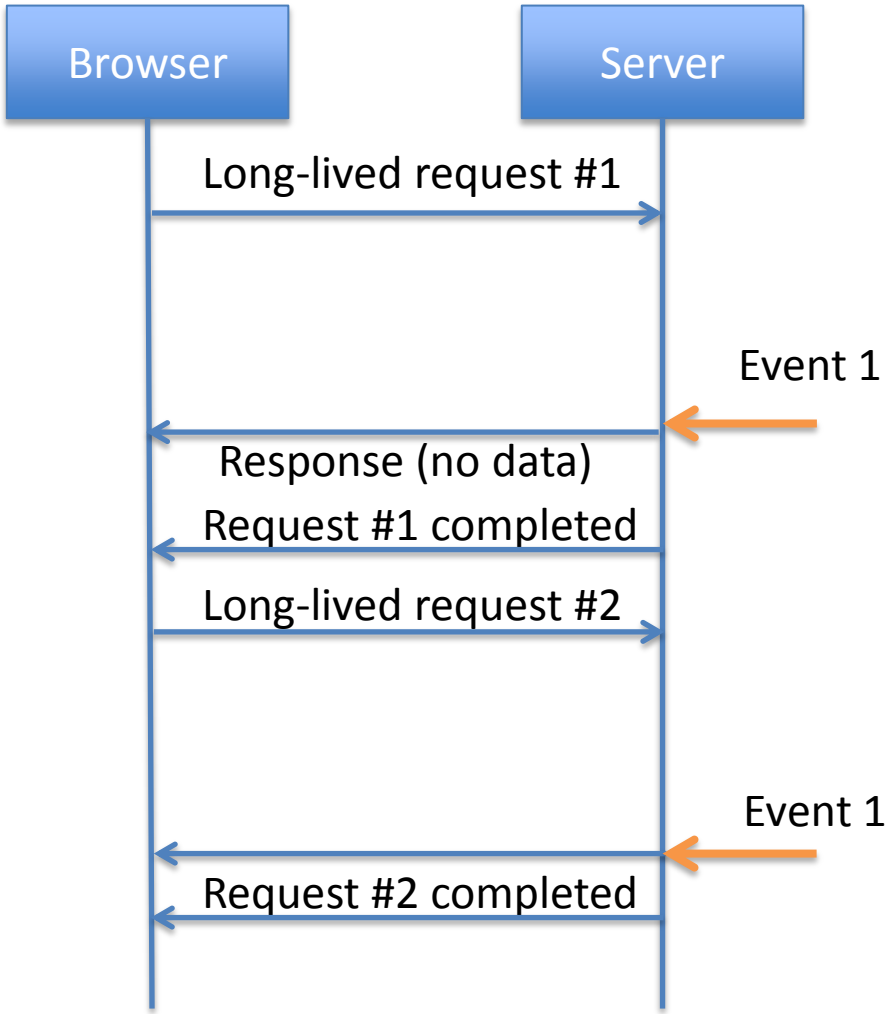
The Interface layer: asynchronous notification to the client

Comet

It is a web application model, encompassing multiple techniques for achieving server push notification

Specific methods of implementing Comet fall into two major categories

- *Streaming*: a single persistent connection from the client browser to the server for all Comet events
- *Long Polling:* The browser makes an Ajax-style request to the server, which is kept open until the server has new data to send to the browser, which is sent to the browser in a complete response. The browser initiates a new long polling request in order to obtain subsequent events.



The Interface layer: asynchronous notification to the client

- *Comet* is also known by several other names, including *Ajax Push , Reverse Ajax, Two-way-web, HTTP Streaming* and *HTTP Server Push*
- Browser-native technologies are inherent in the term Comet. Attempts to improve non-polling HTTP communication have come from multiple sides:
  - *HTML 5 WebSocket API* working draft specifies a method to create a persistent connection with a server and receiving messages via an *onmessage* callback
  - The Bayeux protocol by the Dojo Foundation, based on a pub/sub model with the aim of re-use of client-side JavaScript code with multiple Comet Servers
  - The JSONRequest object as an alternative to the XHR object
  - Use of plugins, such as Java applets or the proprietary Adobe Flash
    - Work identically across all browser
    - You need to install the plugin

The control interface layer: from Ajax to Comet

- Server Side:

```
@Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        Continuation cc = ContinuationSupport.getContinuation(req, null);
        CometSessionObject r = (CometSessionObject) cc.getObject();
        if (r == null) {
            CometSessionObject appo = new CometSessionObject();
            cc.setObject(appo);
        }
        add(cc);
        cc.suspend(TIMEOUT);
        r = (CometSessionObject) cc.getObject();
        resp.getWriter().write(r.getModule() + "-" + r.getState());
        resp.getWriter().flush();
        resp.getWriter().close();
    }
```

The control interface layer: from Ajax to Comet

- Server Side:

```
private void add (Continuation c) {
        this. continuationSet.add(c);
}

public void uponEvent(event e) {
     itc = continuationSet.iterator();
     while (itc.hasNext()) {
           Continuation c = itc.next();
           CometSessionObject appo = (CometSessionObject) c.getObject();
           appo.setModule(e.getName());
           appo.setState(e.getValue());
           c.setObject(appo);
           c.resume();
     }
}
```

The control interface layer: from Ajax to Comet

- Client Side:

```
function startPushNotification() {
   handlerRA = new XMLHttpRequest();
   handlerRA.open("GET","/comet/pushNotification", true);
   handlerRA.onreadystatechange = pushNotification;
   handlerRA.send();
}

function pushNotification() {
   if (handlerRA.readyState == 4) {
        var rText = handlerRA.responseText;
        handlerRA = new XMLHttpRequest();
        handlerRA.open("GET","/comet/pushNotification", true);
        handlerRA.onreadystatechange = pushNotification;
        handlerRA.send();
        //rText contains the push notification
   }
 }
```

- **Project 1: a smart SMS gateway**

  the project has the aim to implement a multi-provider sms gateway able to send and receive sms via REST/COMET interface selecting for every message the most convenient (€) provider

- **Project 2: 3-lines LCD display java driver**

  the project has the aim to implement a Java driver for a simple 3-line LCD display (Arduino style) for Cubieboard 2 board

- **Project 3 (thesis oriented): voice interface for smart spaces**

  the project has the aim to implement a smart device able to deal with the house inhabitants providing information, storing plans and programming the environment. The project must be based on Android voice recognition API

OVER TECHNOLOGIES
The smart spaces makers

Thank you for the attention

In collaboration with:

CIS SAPIENZA
Cyber Intelligence and Information Security

DOMOTICA.it
il portale italiano dell'automazione domestica

Mariano Leva
leva@dis.uniroma1.it