

Control Systems

Alessandro Stecchi & Giampiero Di Pirro

LNF • Accelerator Division

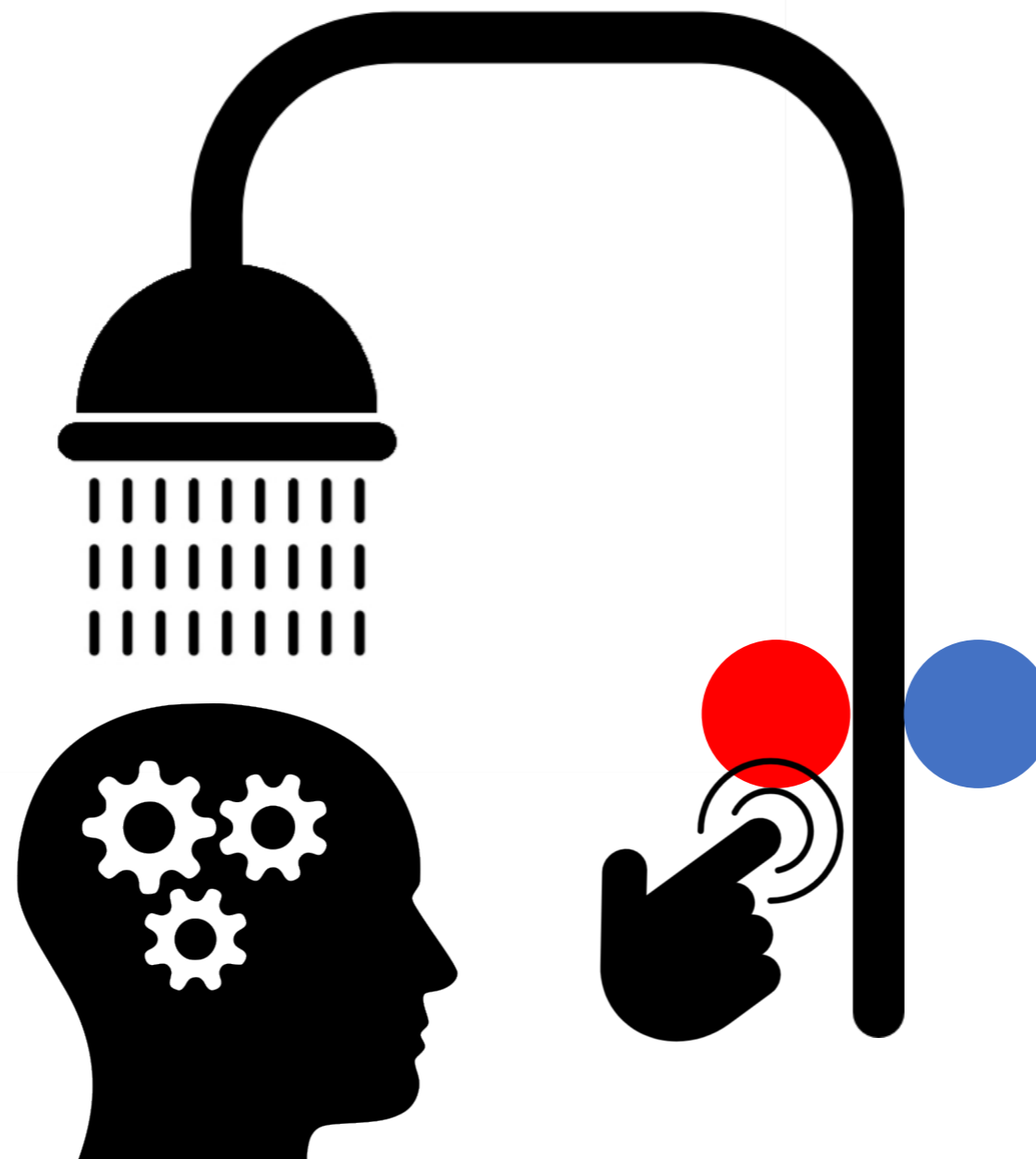




Video courtesy of SpaceX



A Control System is a system that allows the **output variables** of a **process** to be varied (or kept constant) according to **predetermined laws**.



Open loop

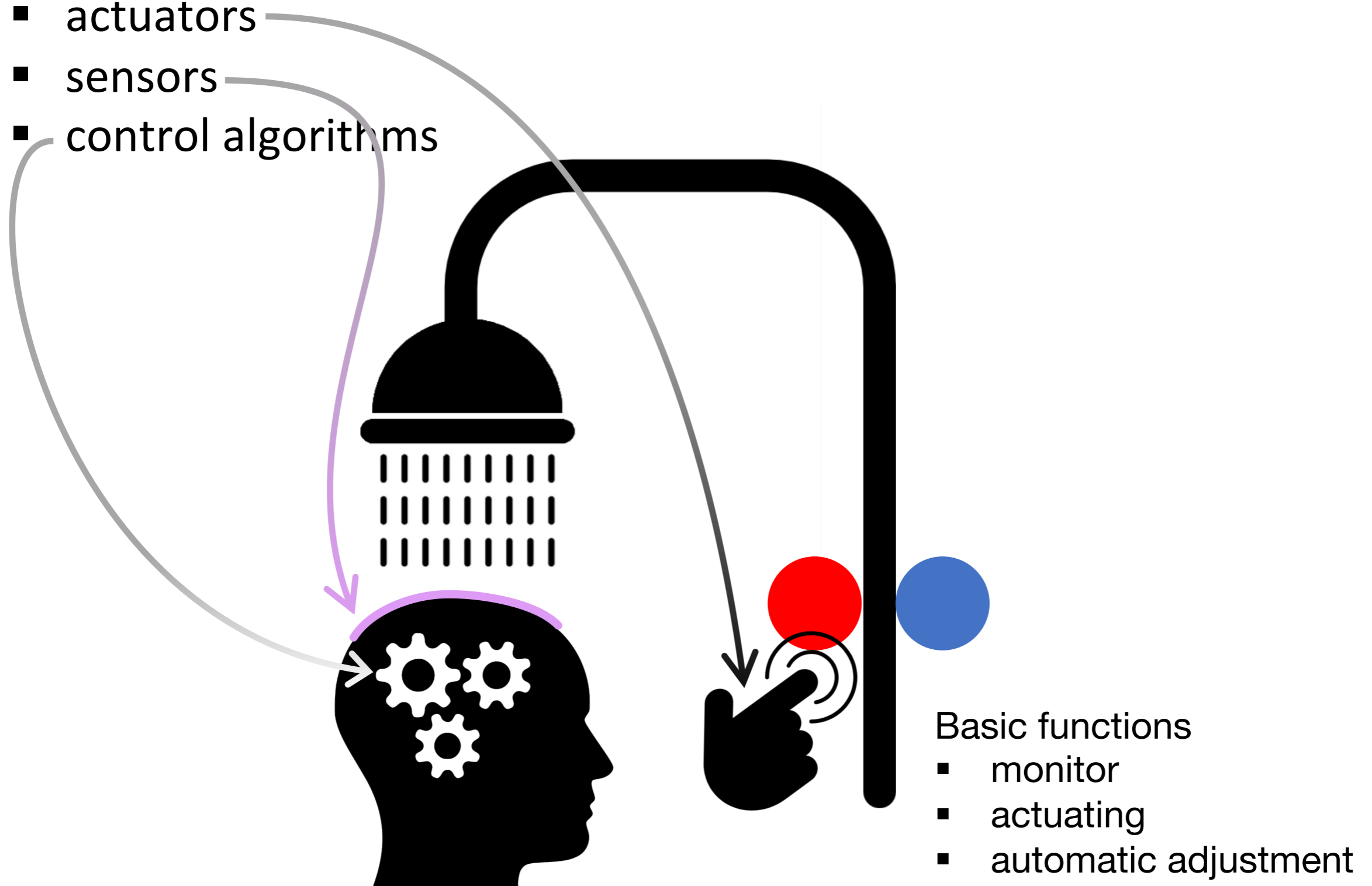
Action is based on an algorithm and initial **static information**.

Closed loop

Action is based on an algorithm and **dynamic information** continuously read back from the controlled system.

Key elements of a Control System:

- actuators
- sensors
- control algorithms



Fields of application of control systems

Civil, Social

- home automation
- automotive, aeronautics
- building climate control (ESCO: Energy Service Company)
- agriculture
- medical equipment, radiotherapy, diagnostics
- telecommunications

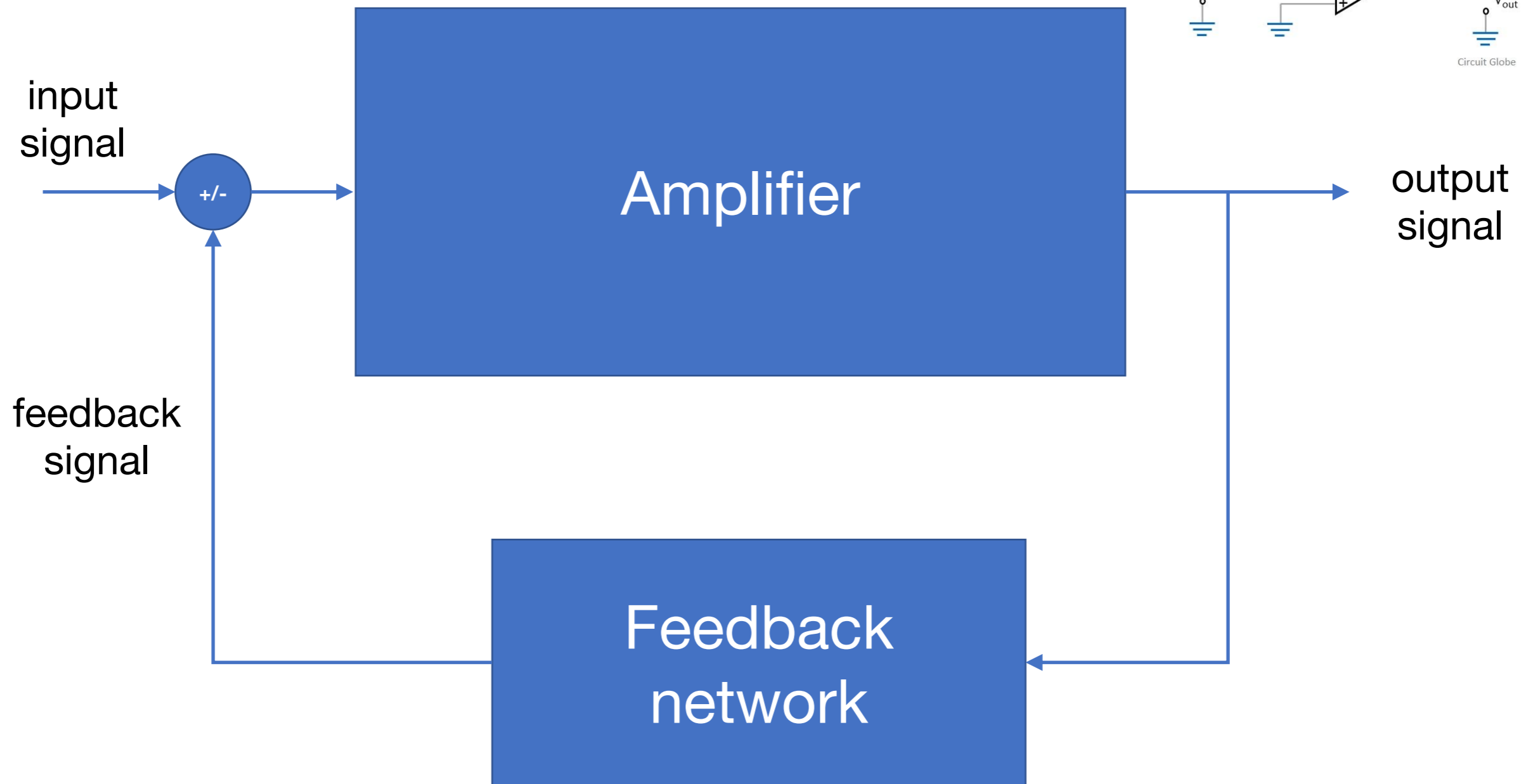
Industrial

- production lines (robots)
- prognostics
- thermal, electrical, nuclear power plants

Scientific

- automatic rovers, space probes
- astronomy
- experimental apparatus, detectors, particle accelerators

What physicists and engineers usually mean by *Control System*



Controls adopting "CPUs"

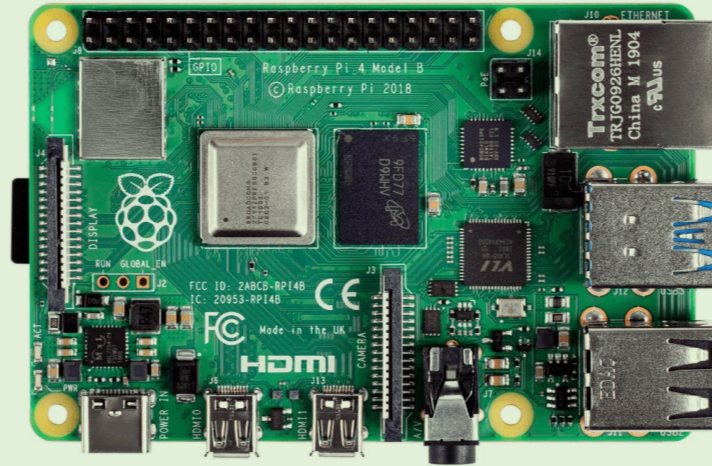


Computers $t \sim 1-10$ ms

PC



SBC: Single Board Computer



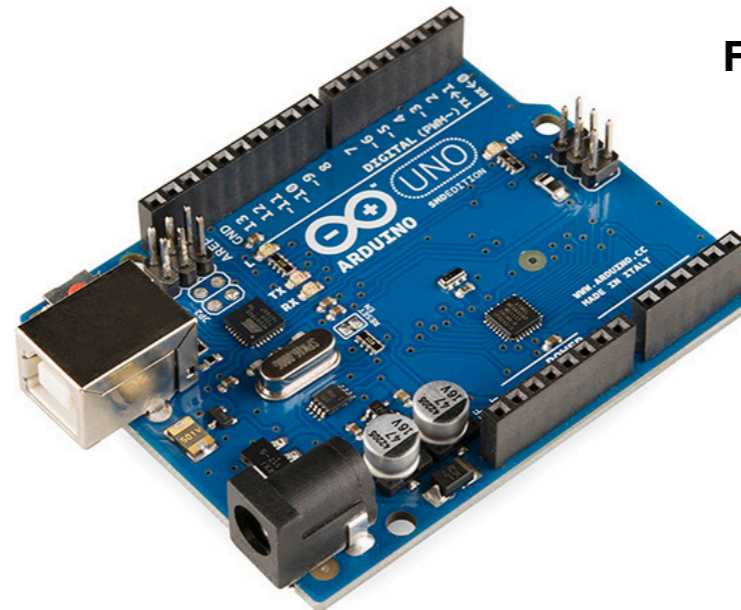
Embedded processor



FPGA: Field Programmable Gate Array
 $t \sim$ ns



Virtual Machines



mP: microprocessor
 $t \sim 1$ ms



ASIC: Application-Specific Integrated Circuit

Fields of application of control systems

Civil, Social

- home automation
- automotive, aeronautics
- building climate control (ESCO: Energy Service Company)
- agriculture
- medical equipment, radiotherapy, diagnostics
- telecommunications

Industrial

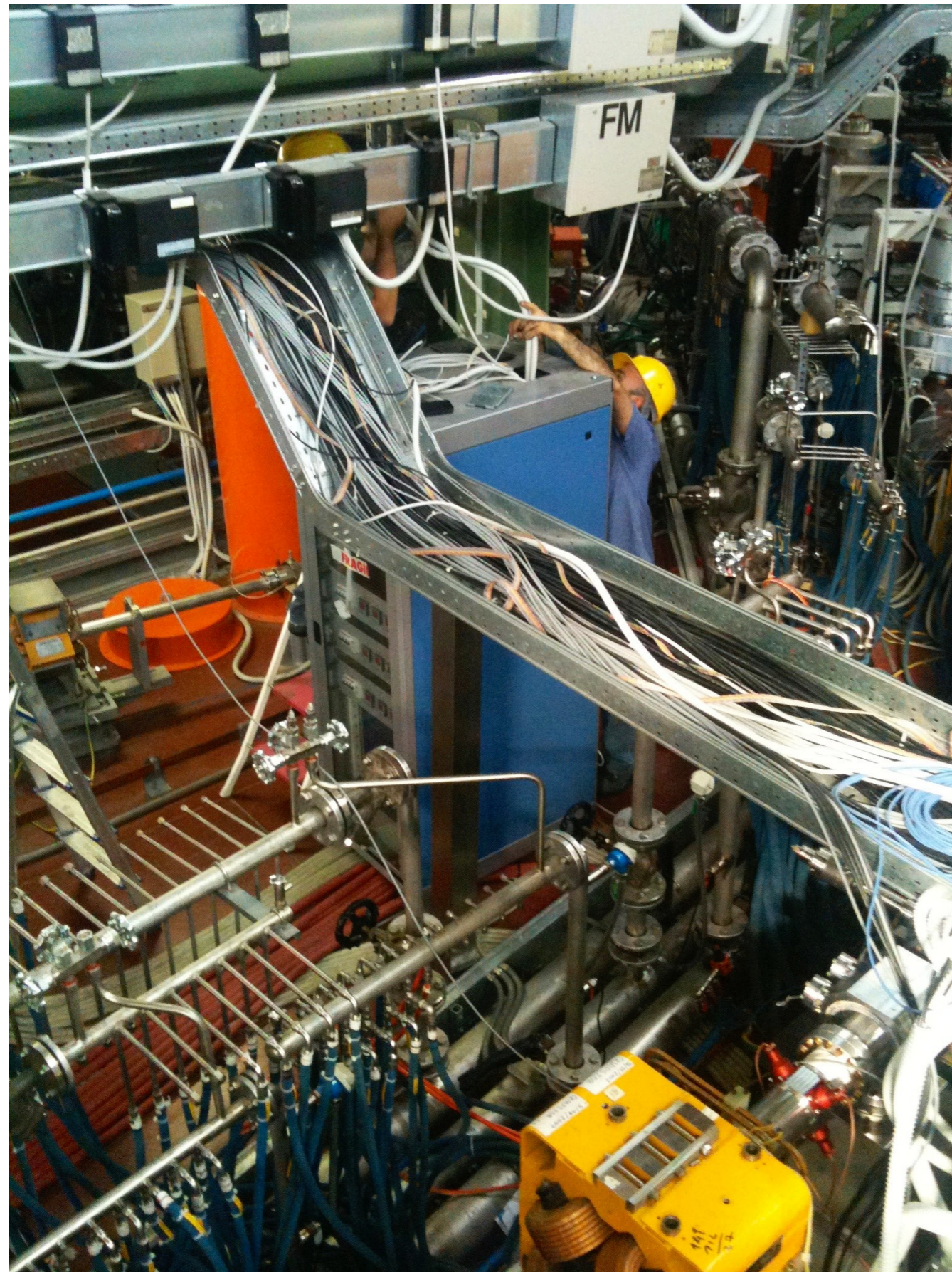
- production lines (robots)
- prognostics
- thermal, electrical, nuclear power plants

Scientific

- automatic rovers, space probes
- astronomy
- **experimental apparatus, detectors, particle accelerators**

Control Systems – of complex plants – employing computers

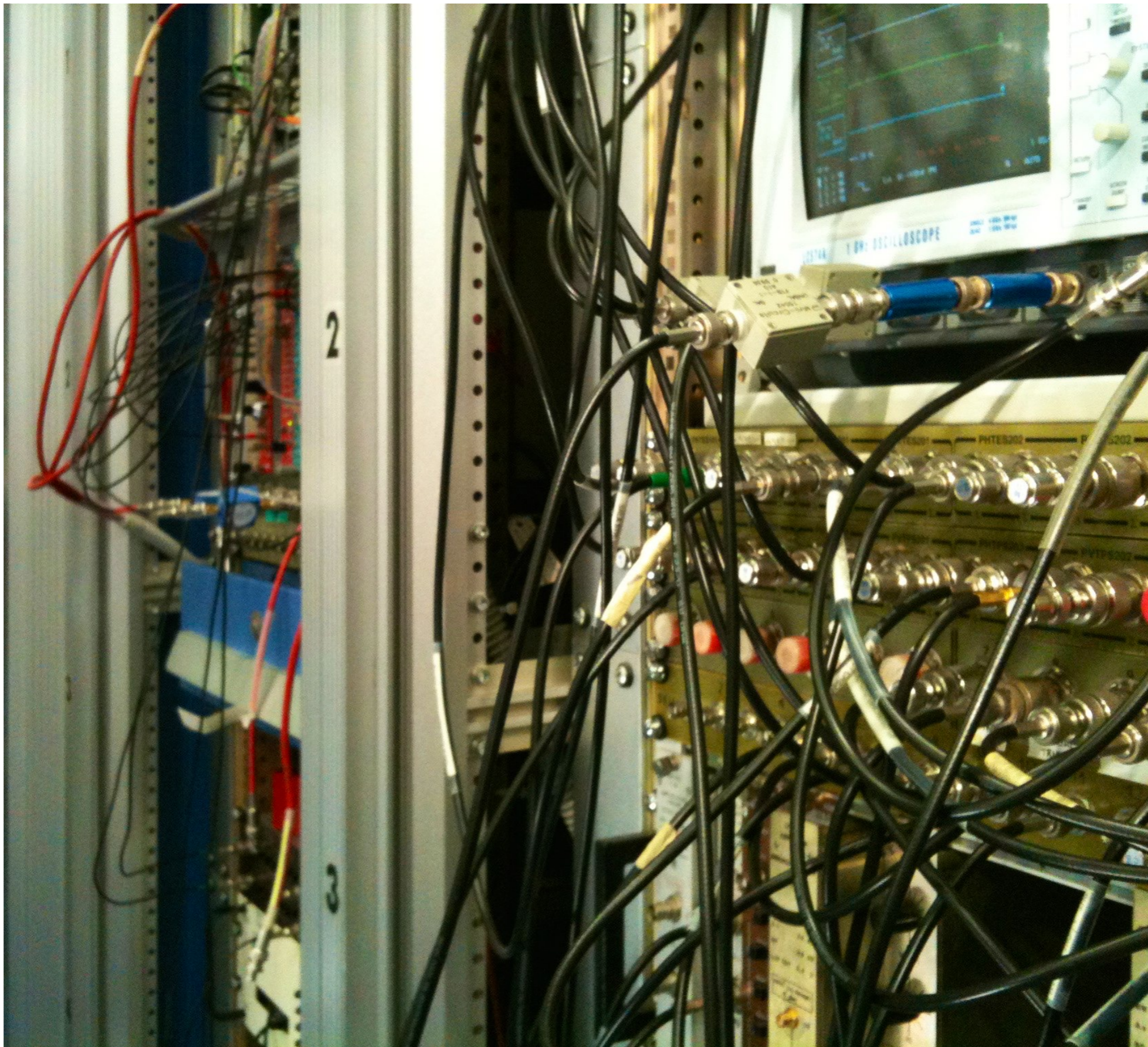
plant



plant → **sensors/actuators**



plant → sensors/actuators → **front-end**



plant → sensors/actuators → front-end → **control room**



Design and Implementation

Definition of specifications and objectives to achieve

- performance
- functionality
- reliability

System modeling

- assessment of complexity and load estimation (processing power, throughput, latency, storage)
- design / adoption of control framework

Development

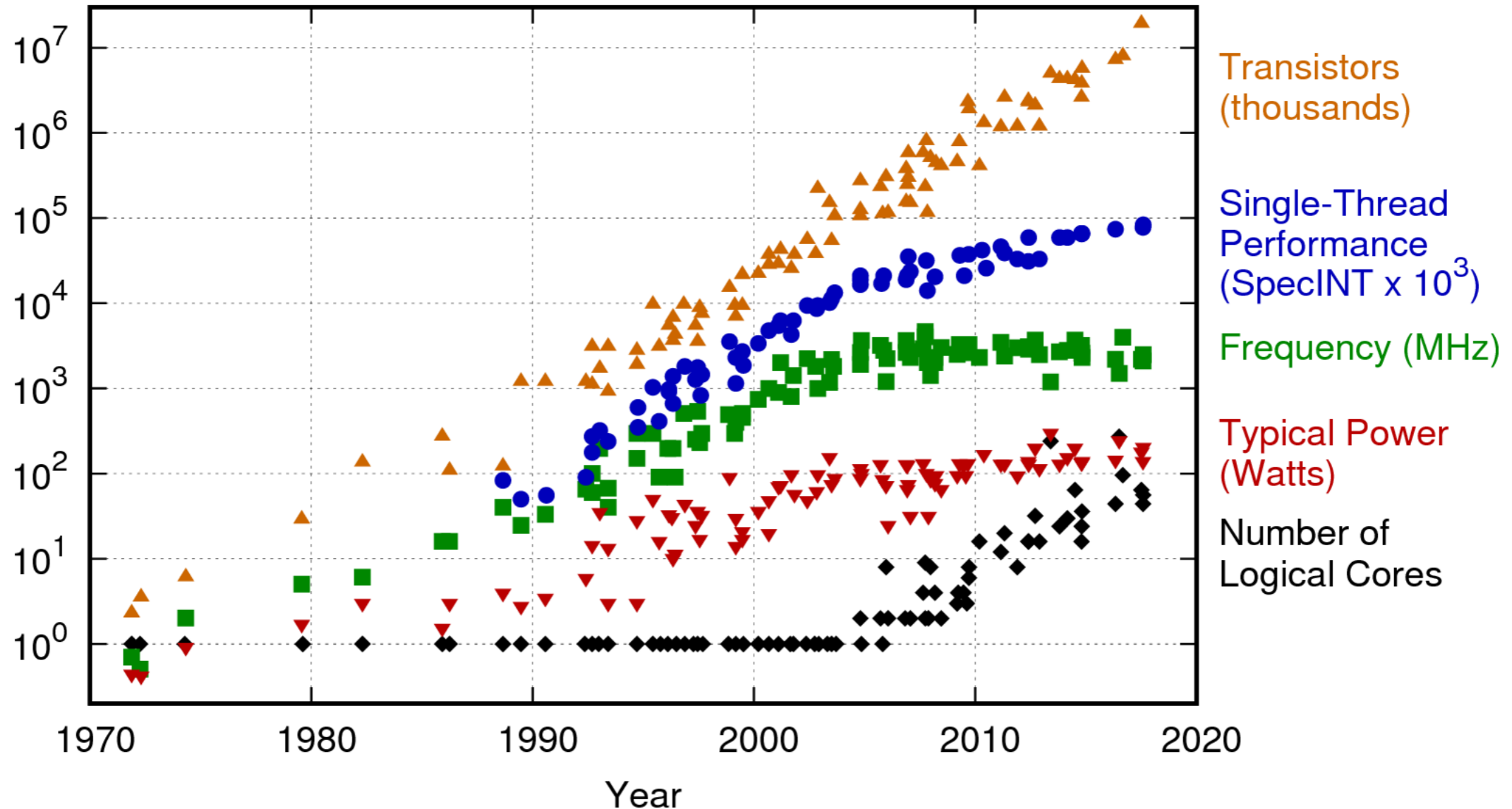
- OOD, OOP (properties, methods)
- software development
- simulation
- engineering
- validation

General characteristics

- **open** to other systems (experiments, DAQ, machine protection, service plants);
- **easy** for the users (advanced UI, data import/export, calculation software interfacing);
- **scalable** according to plant size and required performance;
- **reliable** (no SPF, redundancy);
- **flexible** with respect to technological updates.

Technological growth

42 Years of Microprocessor Trend Data



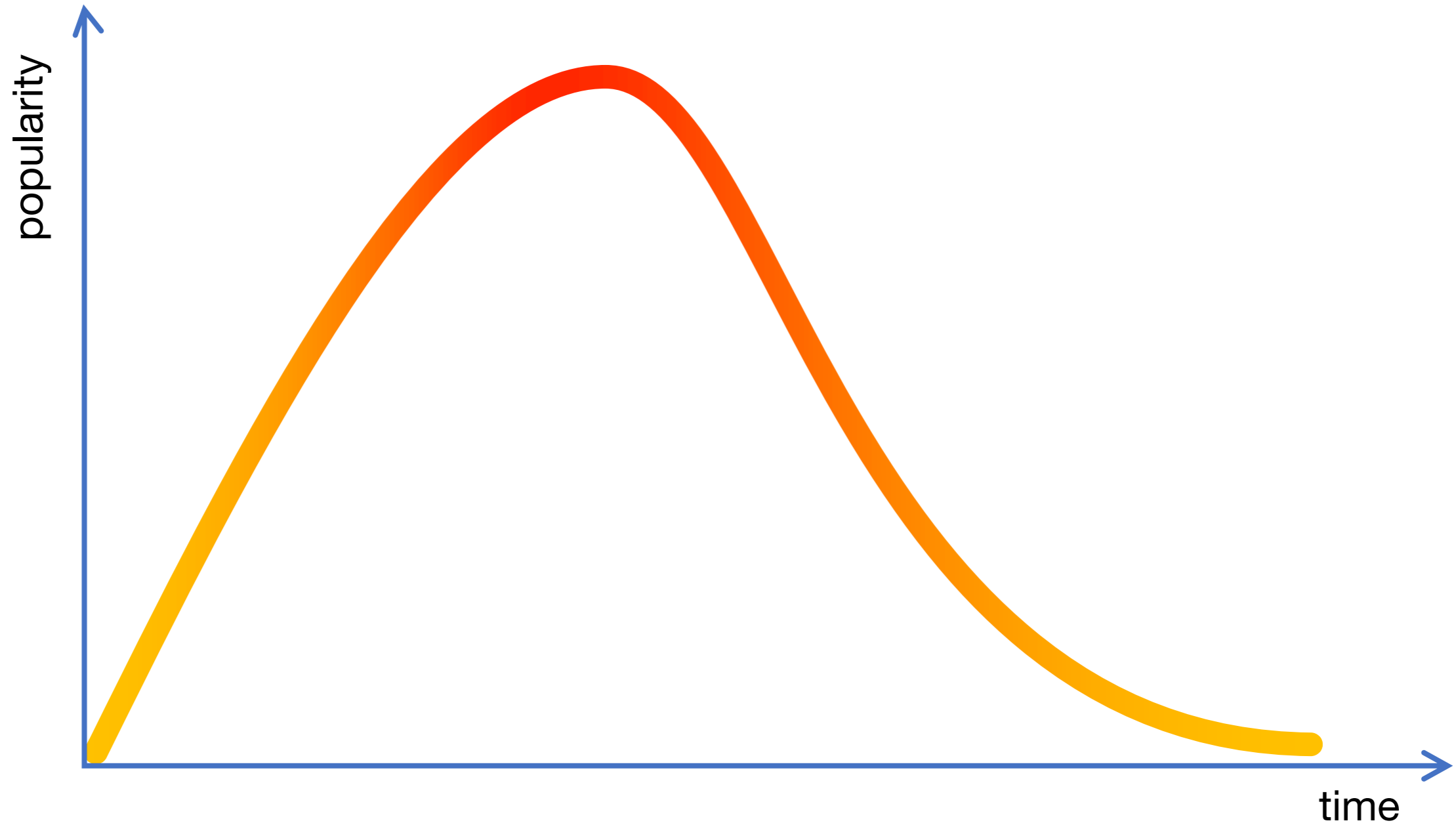
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Moore's law

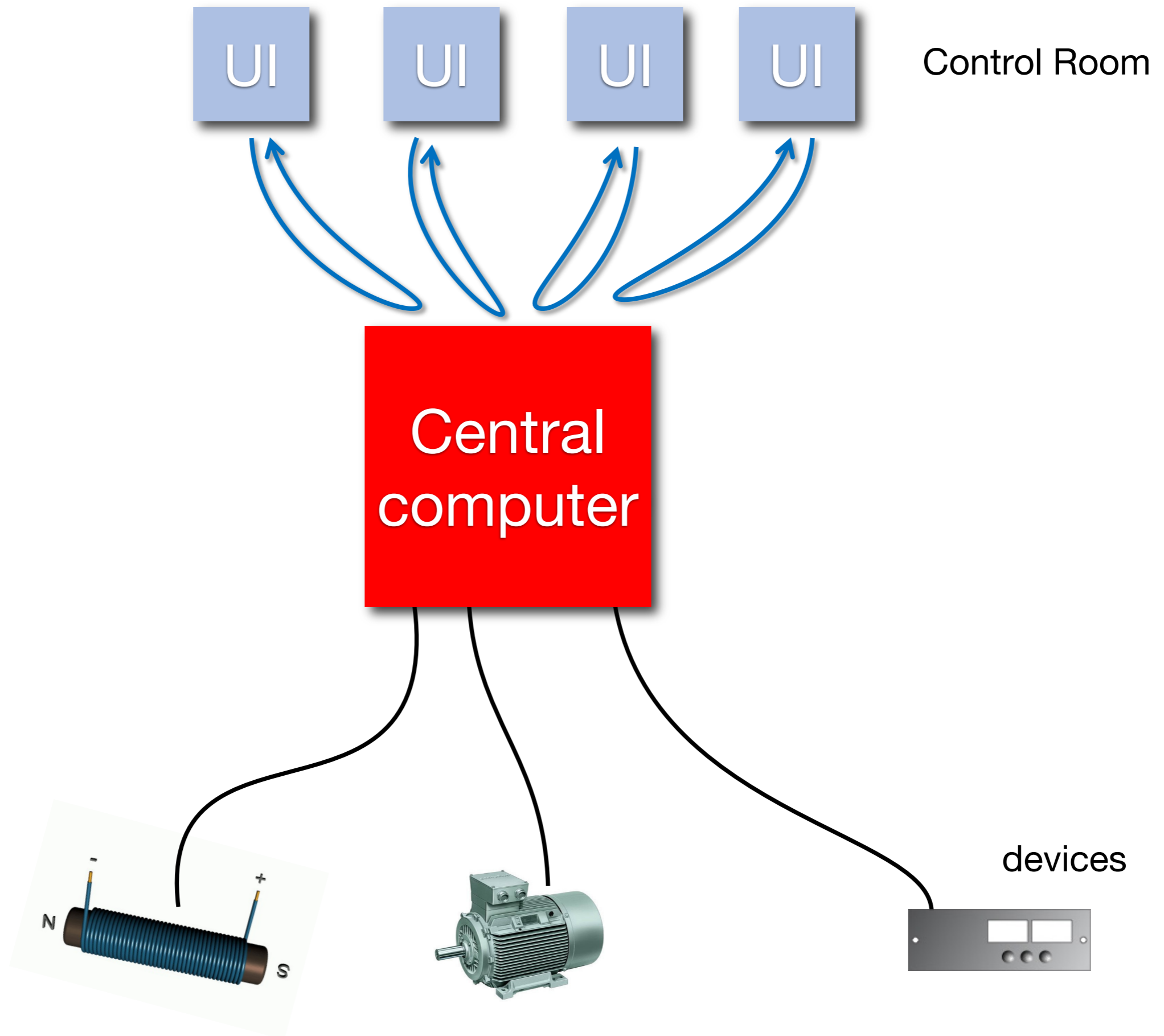
the number of transistors in processors doubles every 18 months

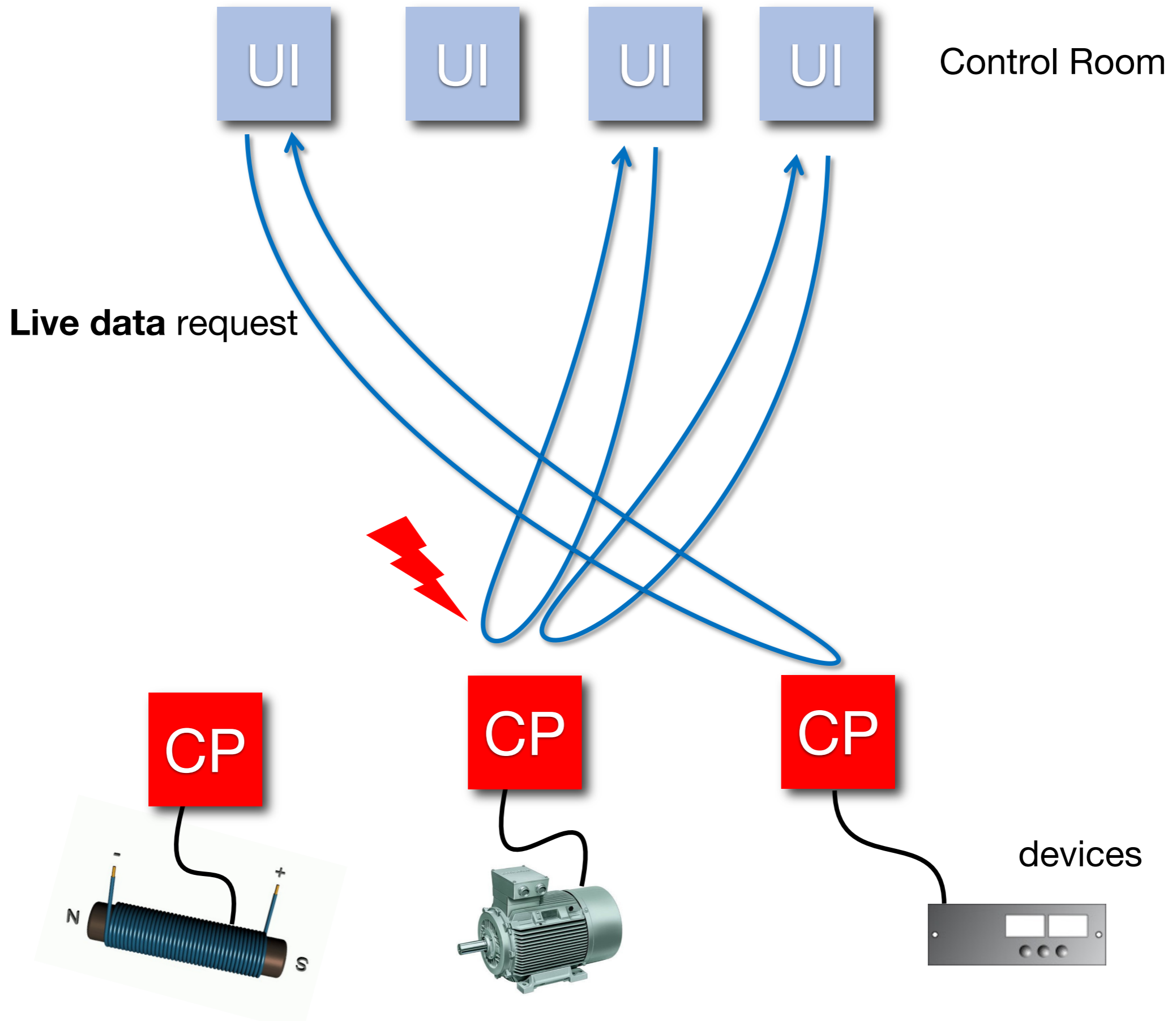
-> limits on reducing the size of transistors below which unwanted 'parasitic' quantum effects would occur in electronic circuits.

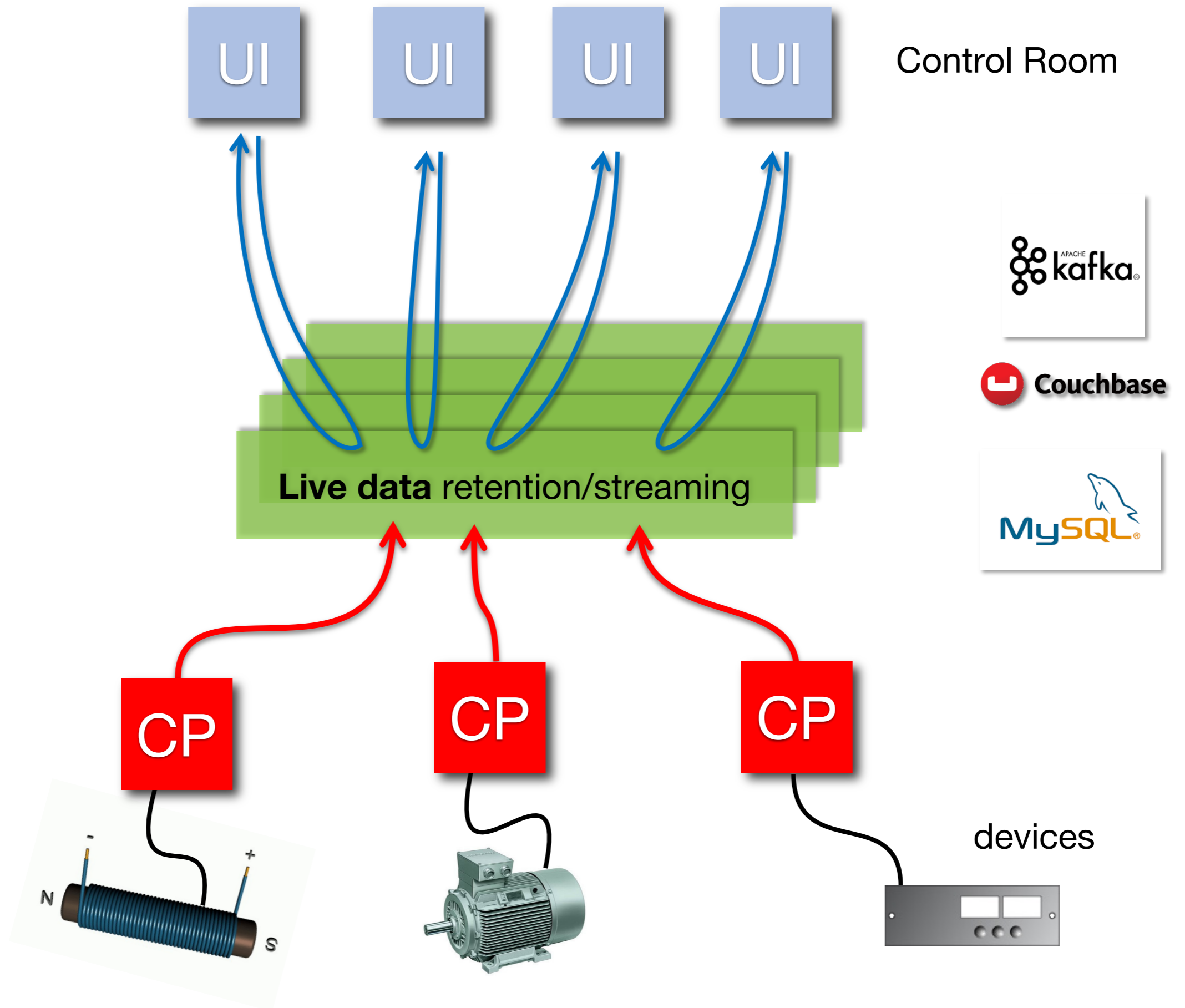
Obsolescence of standards

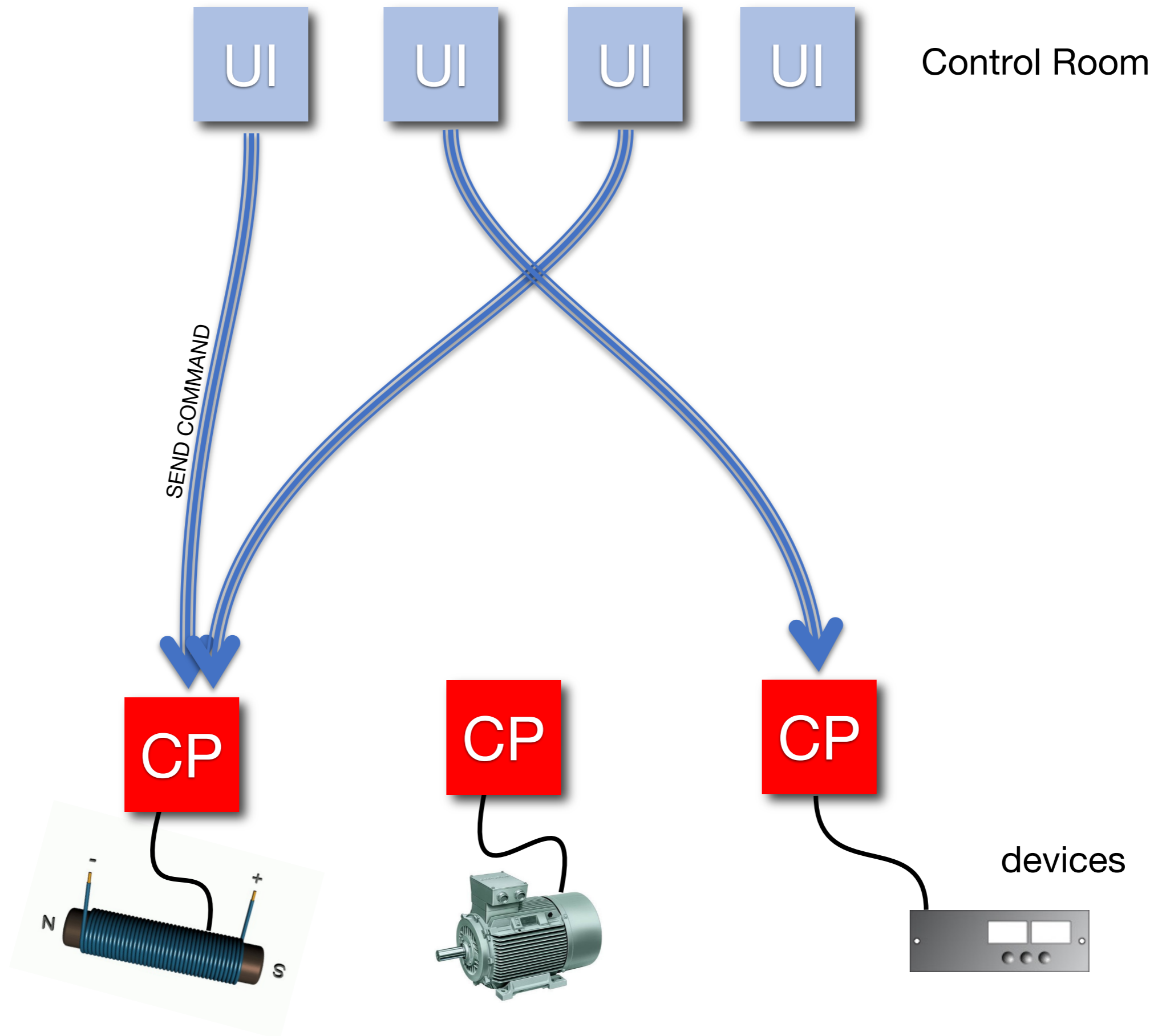


The lesson is: a system has to be as independent as possible from hardware and software standards.



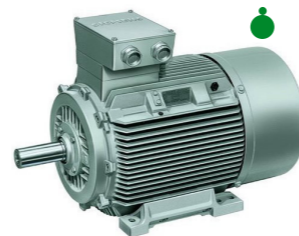
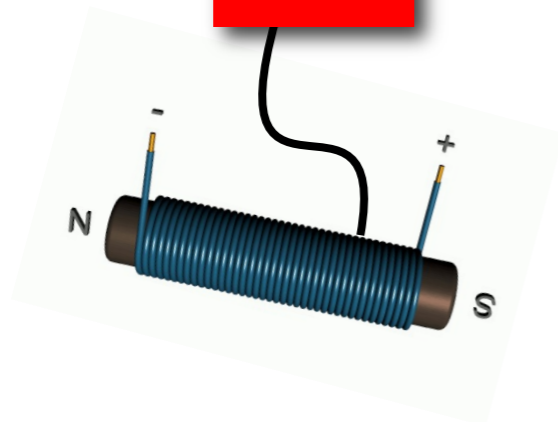
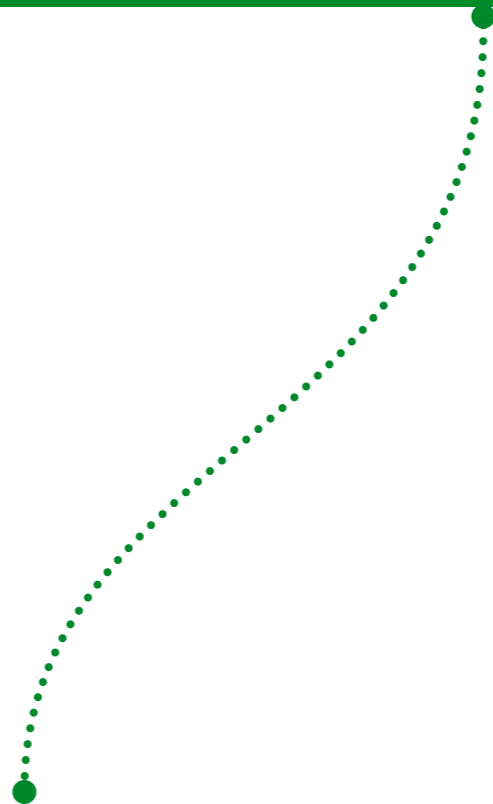








Control Room



dispositivi

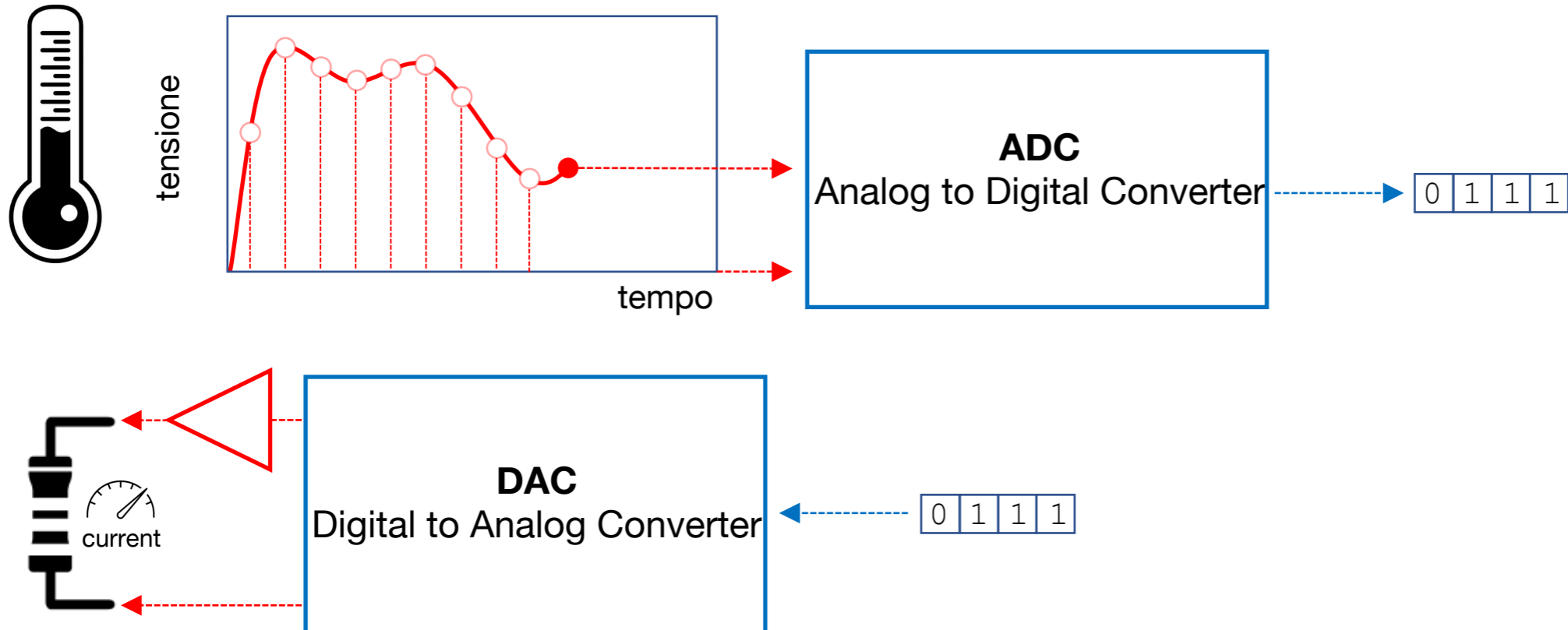


Basic features

- acquisition / implementation
- communication
- storage
- presentation
- analysis / correlation
- historization

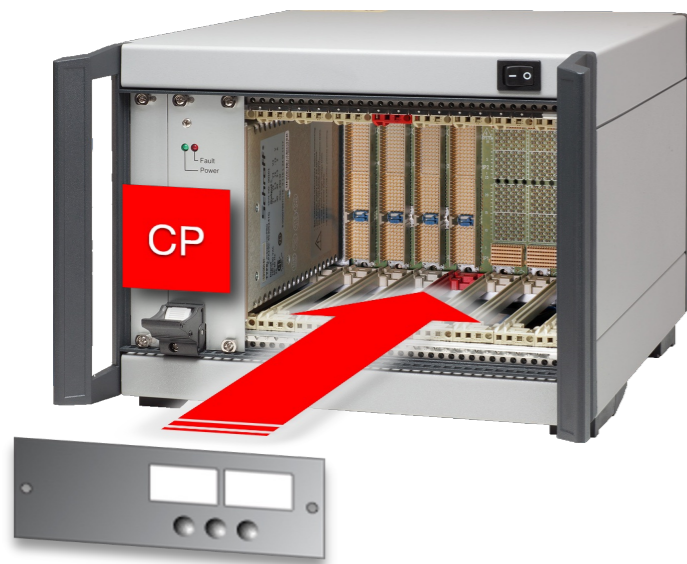
- **acquisition / implementation**

- communication
- storage
- presentation
- analysis / correlation
- historization



- acquisition / implementation
- **communication**
- storage
- presentation
- analysis / correlation
- historization

Embedded devices



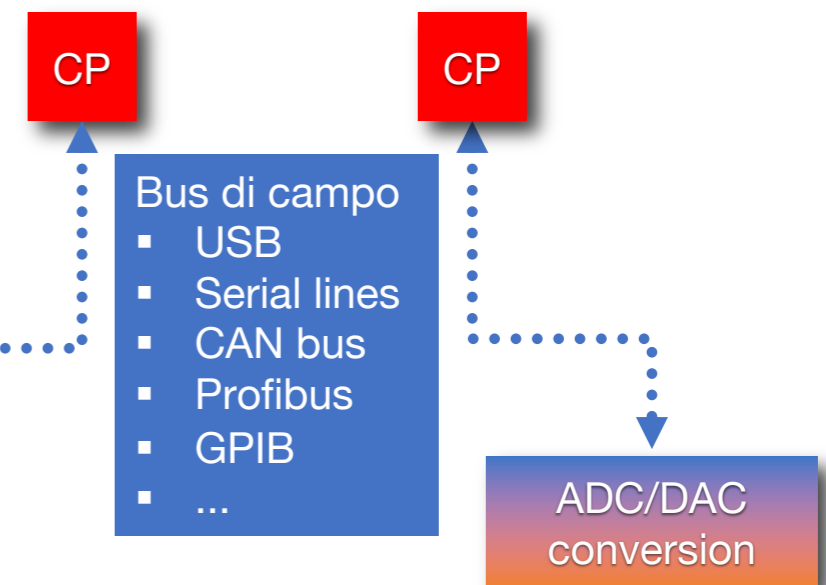
Intelligent devices



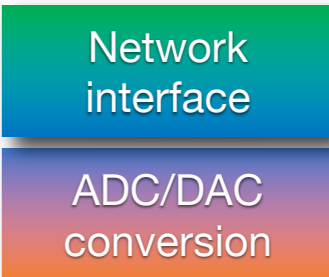
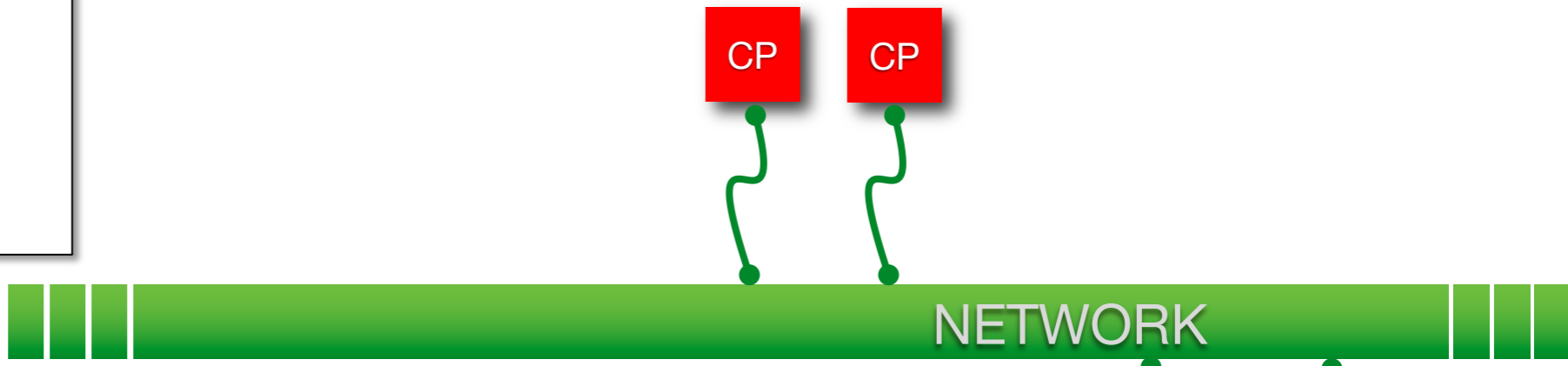
Digital devices



Analog devices



- acquisition / implementation
- **communication**
- storage
- presentation
- analysis / correlation
- historization



Analog devices



Digital devices



Intelligent devices

Embedded devices



- acquisition / implementation
- **communication**
- **storage**
- presentation
- analysis / correlation
- historization

average size (@DAFNE)
~ 650 byte



```
{
float64 temperature_read;
char    units;
boolean status;
....
}
```

```
{
float64 current_read;
float64 current_set;
char    units;
boolean error;
....
}
```

```
{
float64 mag_field_read;
float64 mag_field_set;
int     polarity;
....
}
```

```
{
int     aperture;
int     time;
int     n_pixels_h;
int     n_pixels_v;
boolean status;
....
}
```



650 byte



Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.

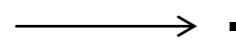
Ahi quanto a dir qual era è cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinova la paura!

Tant'è amara che poco è più morte;
ma per trattar del ben ch'i' vi trovai,
dirò de l'altre cose ch'i' v'ho scorte.

Io non so ben ridir com'i' v'intrai,
tant'era pien di sonno a quel punto
che la verace via abbandonai.

Ma poi ch'i' fui al piè d'un colle giunto,
là dove terminava quella valle
che m'avea di paura il cor compunto

guardai in alto e vidi le sue spalle
vestite già de' raggi del pianeta
che mena dritto altrui per ogni calle.

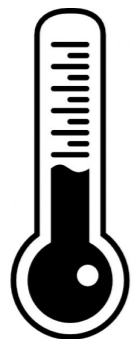


DAFNE ~1.4 Mbyte

Relational database →



```
{  
float64 temperature_read;  
char    units;  
boolean status;  
.....  
}
```



```
{  
float64 current_read;  
float64 current_set;  
char    units;  
boolean error;  
.....  
}
```



```
{  
float64 mag_field_read;  
float64 mag_field_set;  
int     polarity;  
.....  
}
```



```
{  
int     aperture;  
int     time;  
int     n_pixels_h;  
int     n_pixels_v;  
boolean status;  
.....  
}
```



Relational database →



```
{  
float64 temperature_read;  
char    units;  
boolean status;  
.....  
}
```



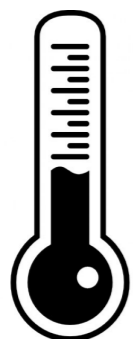
```
{  
float64 current_read;  
float64 current_set;  
char    units;  
boolean error;  
.....  
}
```



```
{  
float64 mag_field_read;  
float64 mag_field_set;  
int     polarity;  
.....  
}
```



```
{  
int     aperture;  
int     time;  
int     n_pixels_h;  
int     n_pixels_v;  
boolean status;  
.....  
}
```



Relational database →



```
{  
float64 temperature_read;  
char    units;  
boolean status;  
.....  
}
```

```
{  
float64 current_read;  
float64 current_set;  
char    units;  
boolean error;  
.....  
}
```

```
{  
float64 magnetic_read;  
float64 magnetic_set;  
int    orientation;  
int    polarity;  
.....  
}
```

```
{  
int    aperture;  
int    time;  
int    n_pixels_h;  
int    n_pixels_v;  
boolean status;  
.....  
}
```



serialized and self-describing data

Key	Value



my_string

`0x16\0x00\0x00\0x00\0x02hello\0x00\0x06\0x00\0x00\0x00world\0x00\0x00`



my_odd_data

`..... another descriptive sequence`

BSON serialization

`\0x16\0x00\0x00\0x00`

`\0x02`

`hello\0x00`

`\0x06\0x00\0x00\0x00world\0x00`

`\0x00`

total document size

0x02 = type string

field name

field value (size of value, value, terminator)

type EOO ('end of object')

- acquisition / implementation
- communication
- storage
- **presentation**
- analysis / correlation
- historization

User Interface

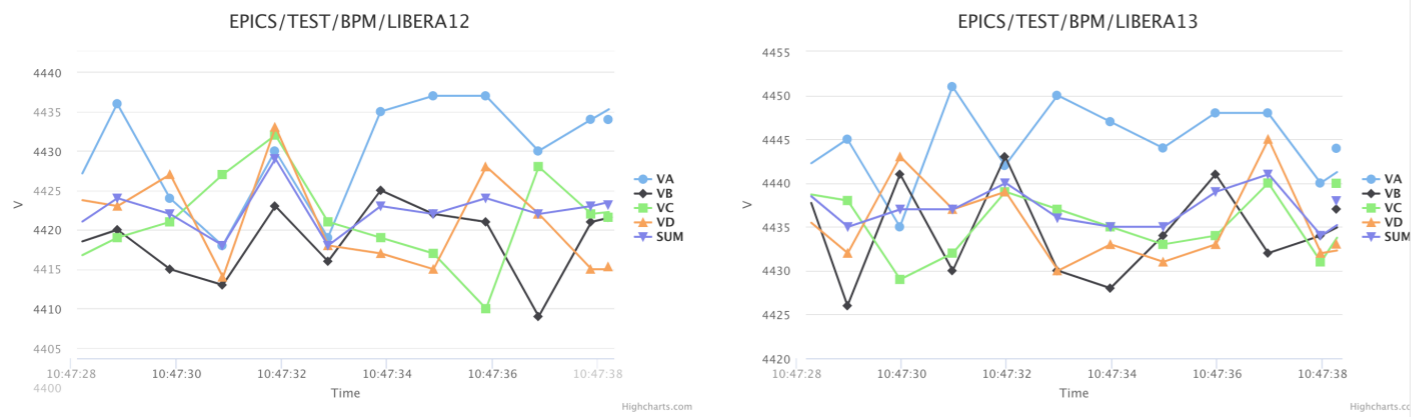
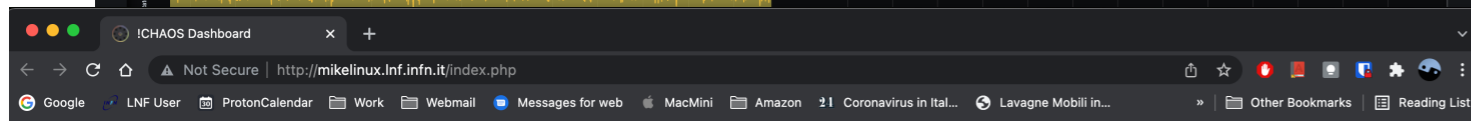
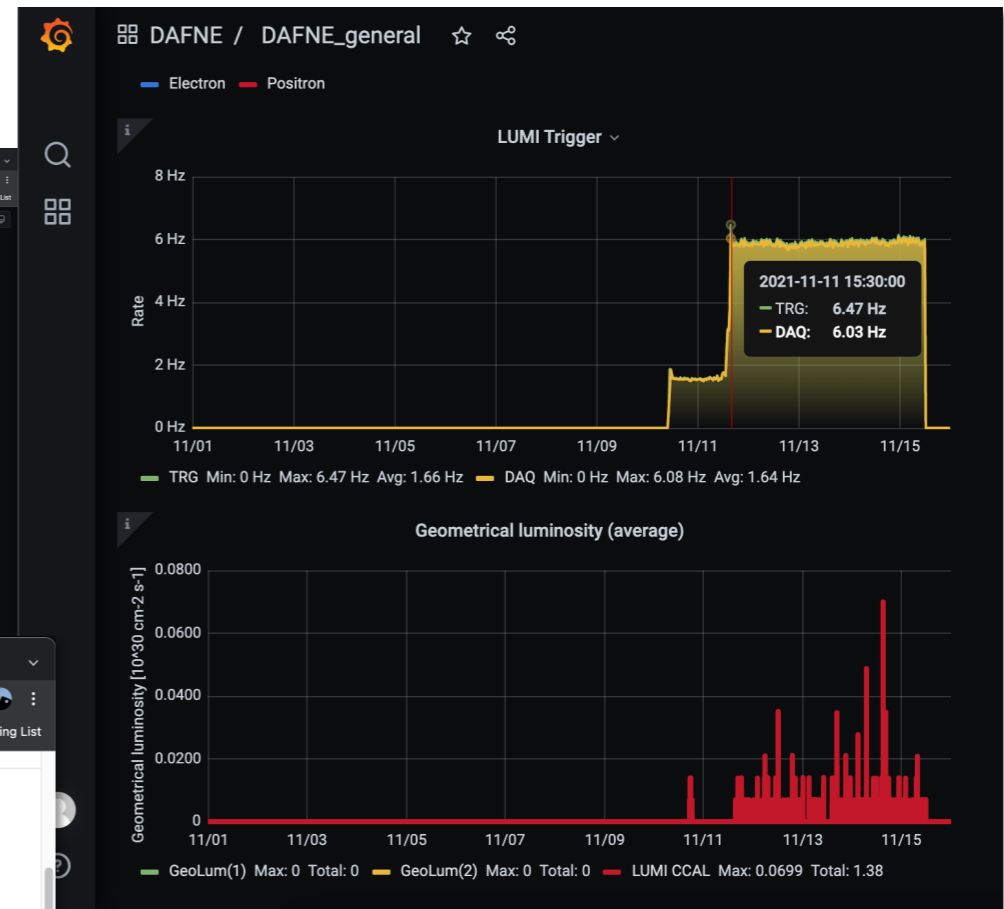
The screenshot displays a complex LabVIEW interface for a particle accelerator control system. It is divided into several functional panels:

- Table Panel (Left):** A table listing magnet parameters for various elements. The columns are ELEMENT, READOUT, SETTING, SAVED, and FLAGS.
- ChargeMonitor.vi (Center):** A real-time monitoring window showing current levels in mA. It includes a 'Pause Acq' button, a 'wait [ms]' control, and a plot of current over time.
- Monitors (Bottom):** Two bar charts showing current levels in nC for different monitors. The left chart shows a significant spike in the 'Monitors' category.
- Diagram (Right):** A schematic diagram of the particle ring, labeled 'Main_Ring_e', with various components like TM, TS, TB, TT, TP, TR, and TE.

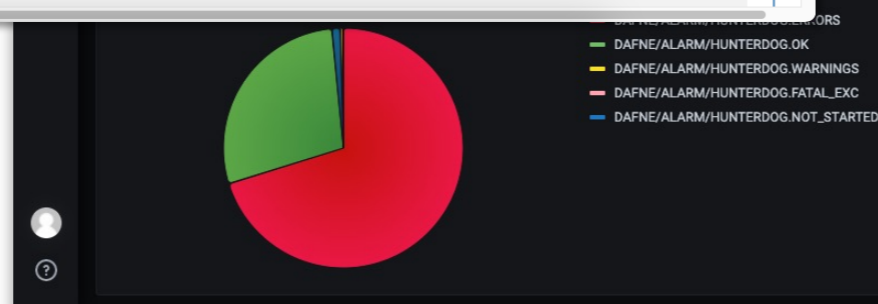
ELEMENT	READOUT	SETTING	SAVED	FLAGS
QUATM001	21.521	21.500		⚡ +
CHHTM001	0.000	-0.004		⚡
CVVTM001	-0.009	-1.900		⚡
QUATM002	67.324	67.500		⚡ -
CHHTM002	0.000	-0.001		⚡
CVVTM002	-0.018	-2.000		⚡
QUATM003	50.964	51.000		⚡ +
CHHTM003	0.001	-0.010		⚡
CVVTM003	-0.005	-0.004		⚡
QUATM004	13.022	13.000		⚡ -
DHPTB101	294.516	295.000		⚡ +
QUATB101	29.813	30.000		⚡ -
QUATB102	29.941	30.000		⚡ +
DHSTB001	320.451	320.400		⚡ +
QUATB001	29.953	30.000		⚡ -
CHHTB001	-0.007	-0.017		⚡
CVVTB001	0.005	0.000		⚡
QUATB002	34.943	35.000		⚡ +
CHHTB002	-0.010	0.010		⚡
CVVTB002	-0.012	0.000		⚡
DHPTB102	0.500	0.500		⚡
CHHTT008	-0.010	-0.005		⚡
CVVTT008	4.990	5.000		⚡
QUATB003	49.966	50.000		⚡ -

- acquisition / implementation
- communication
- storage
- **presentation**
- analysis / correlation

User Interface

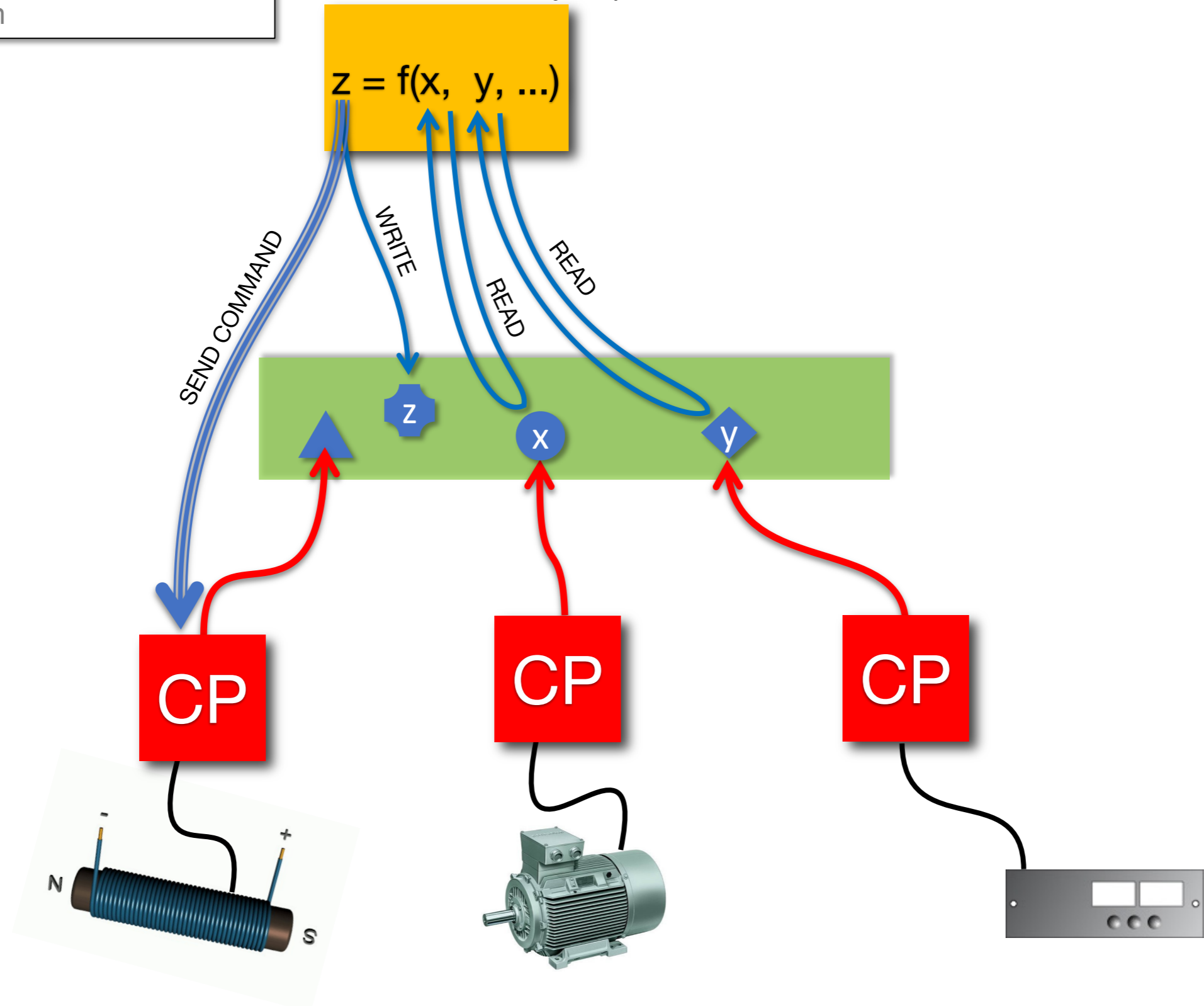


Sel	Element	Status	X	Y	VA	VB	VC	VD	SUM	Samples/Trigger	Alarms dev/cu	
<input type="checkbox"/>	EPICS/TEST/BPM/LIBERA01	▶	ψ	0.000	0.000	667	862	1166	1093	3788	0	No Trigger
<input type="checkbox"/>	EPICS/TEST/BPM/LIBERA02	▶	ψ	0.000	0.000	777	508	950	873	3110	0	No Trigger
<input type="checkbox"/>	EPICS/TEST/BPM/LIBERA03	▶	ψ	0.000	0.000	591	481	705	650	2428	0	No Trigger
<input type="checkbox"/>	EPICS/TEST/BPM/LIBERA05	▶	ψ	0.000	0.000	554	446	640	515	2156	0	No Trigger
<input type="checkbox"/>	EPICS/TEST/BPM/LIBERA06	▶	ψ	0.000	0.000	624	732	762	1920	4039	0	No Trigger

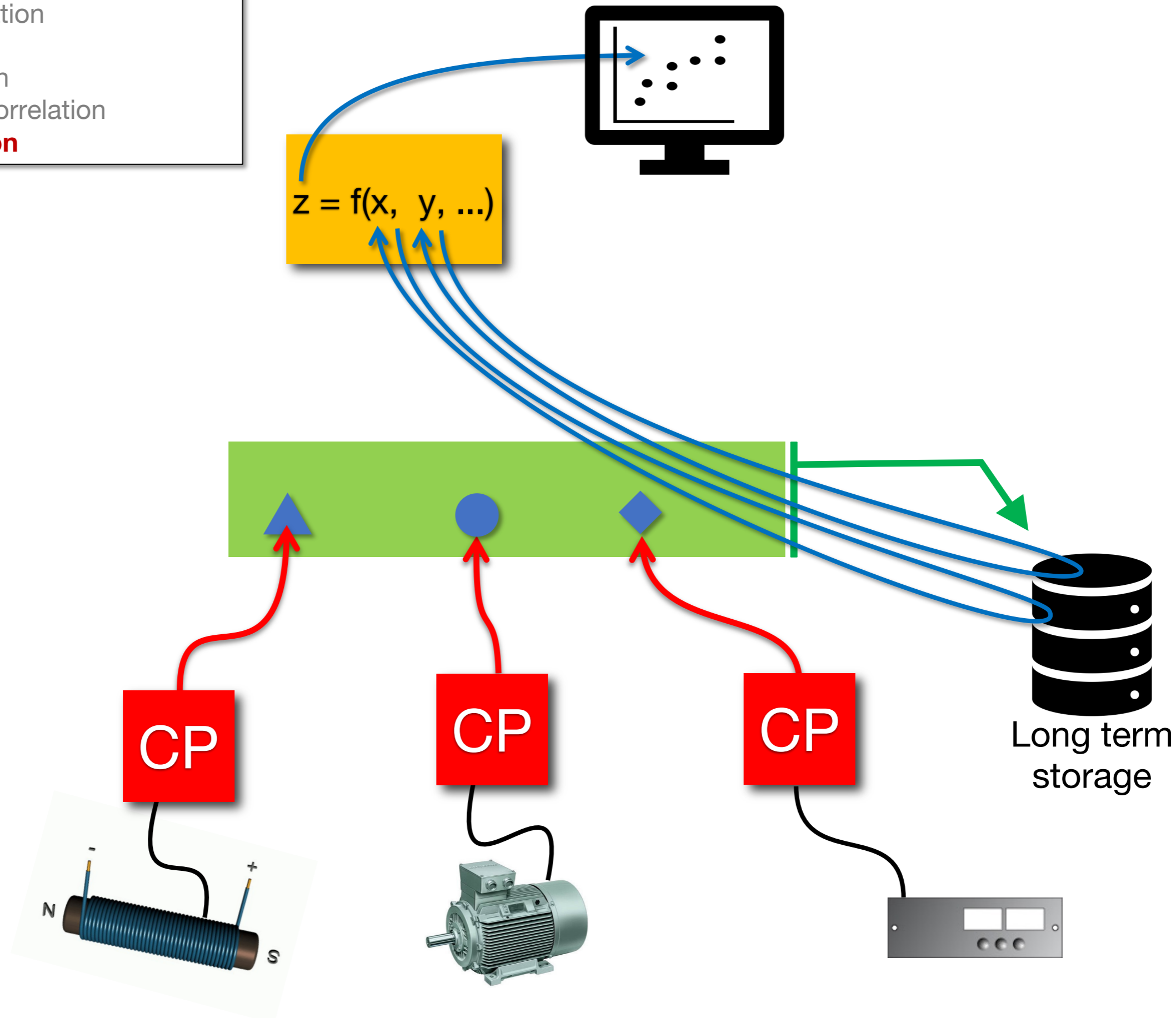


- acquisition / implementation
- communication
- storage
- presentation
- **analysis / correlation**
- historization

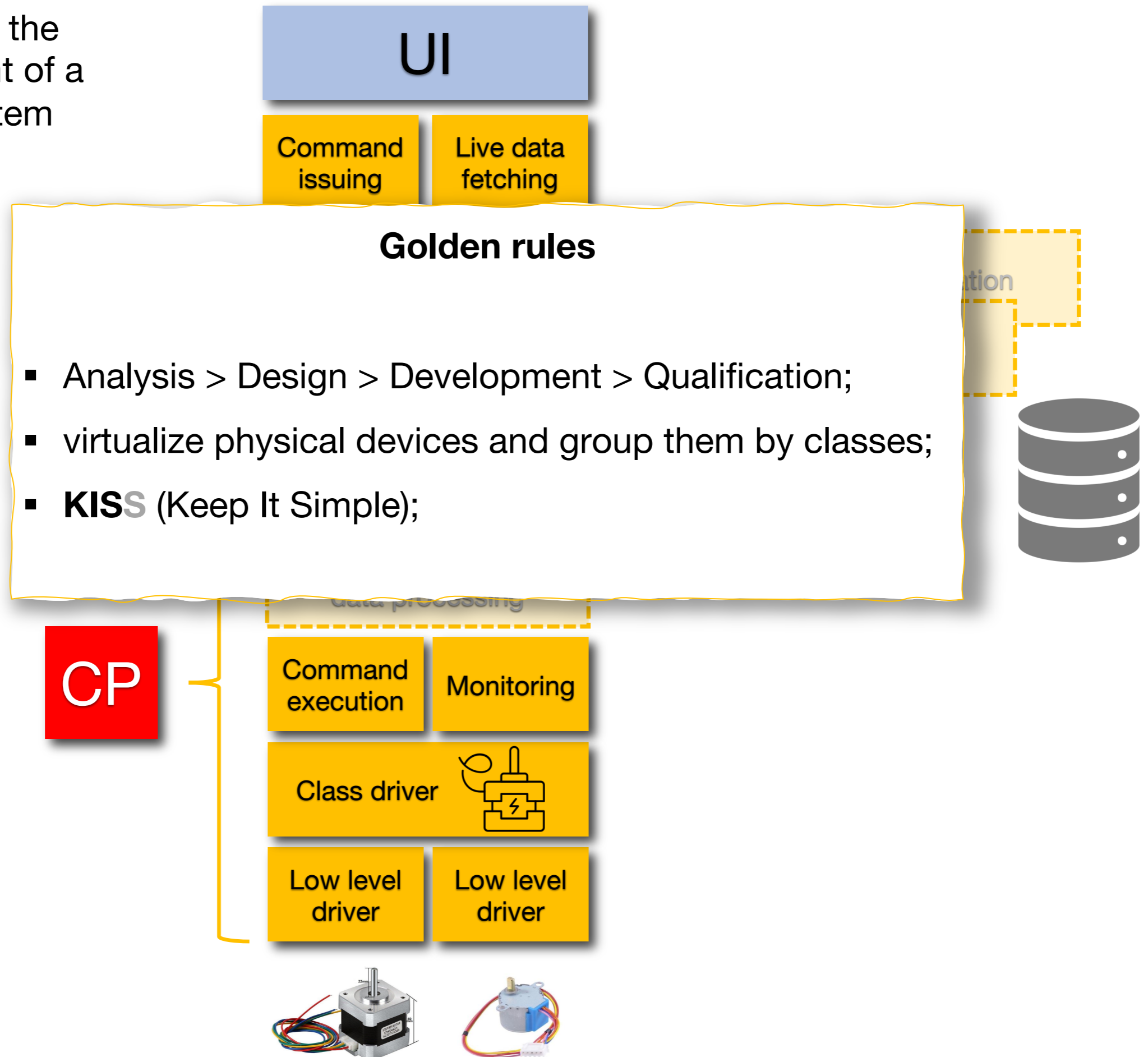
Correlation and analysis processes



- acquisition / implementation
- communication
- storage
- presentation
- analysis / correlation
- **historization**



How to face the development of a Control System



Good methods & practices

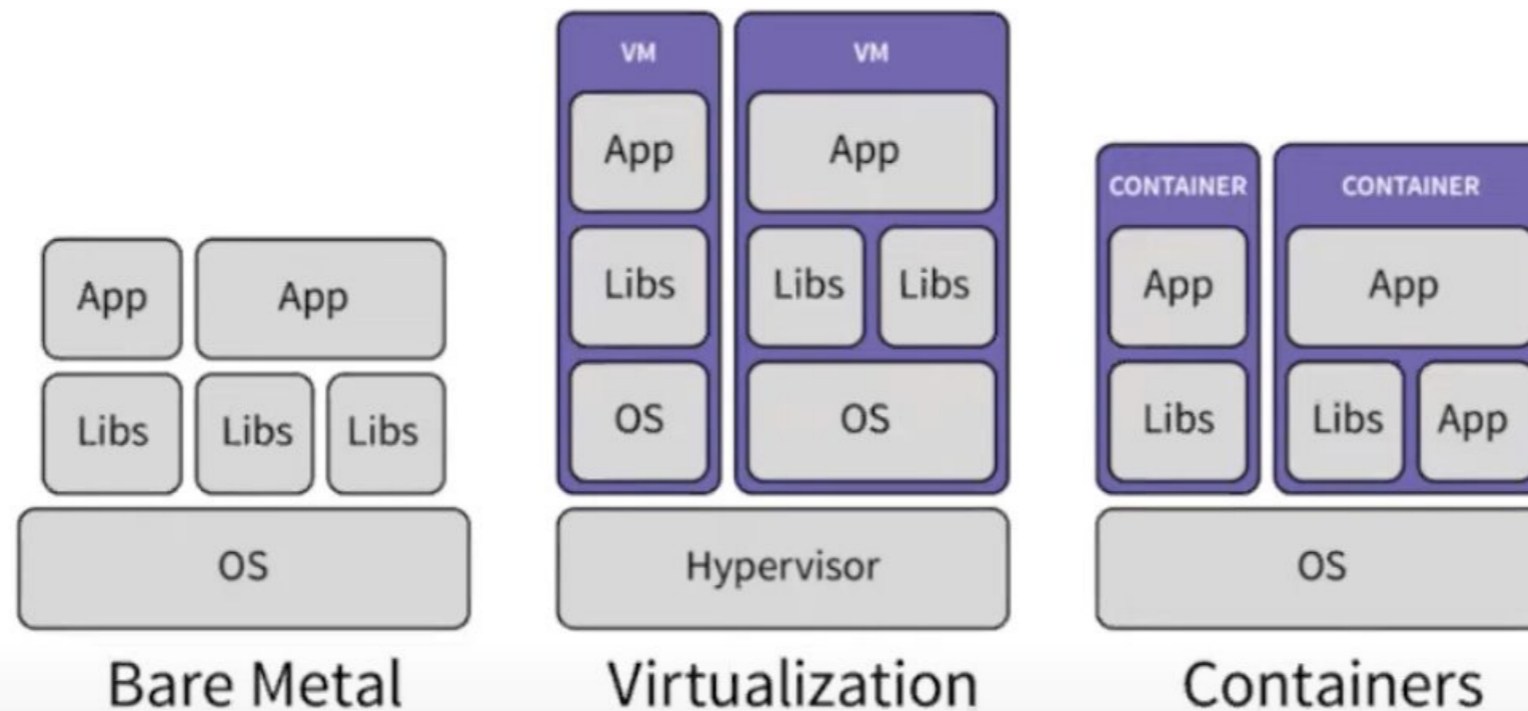
- Development: [Agile approach](#), CI/CD, DevOps
- [Documentation](#), Asset Management
- Project tracking

The screenshot shows a JIRA Scrum Board for the project 'Teams in Space'. The board is organized into three columns: '12 To do', '2 In progress', and '3 Done'. The 'To do' column is further divided into two sections: 'TIS Developer Love' (3 issues) and 'Everything Else' (21 issues). Each issue card includes a title, a description, a priority indicator, a 'SeeSpaceEZ plus' button, and a count of comments or attachments.

Column	Issue ID	Description	Priority	Assignee	Buttons	Count
12 To do	TIS Developer Love (3 issues)					
	TIS-37	Service should return prior trip details and info	High	[Avatar]	SeeSpaceEZ plus	2
	TIS-10	Bad JSON data coming back from hotel API	High	[Avatar]	SeeSpaceEZ plus	
	TIS-8	Requesting flights is now taking > 5 seconds	High	[Avatar]	SeeSpaceEZ plus	
	Everything Else (21 issues)					
	TIS-68	Homepage footer uses an inline style-should use class	High	[Avatar]	Large Team Support	
TIS-20	Engage Saturn Shuttle lines for group tours	High	[Avatar]	Space Travel Partners	3	
TIS-12	Create 90 day plans for all departments in Mars office	High	[Avatar]			
2 In progress	Empty					
	Empty					
3 Done	Empty					

IT trends

- Big data
- AI
- Virtualization, Dockers, Orchestrators



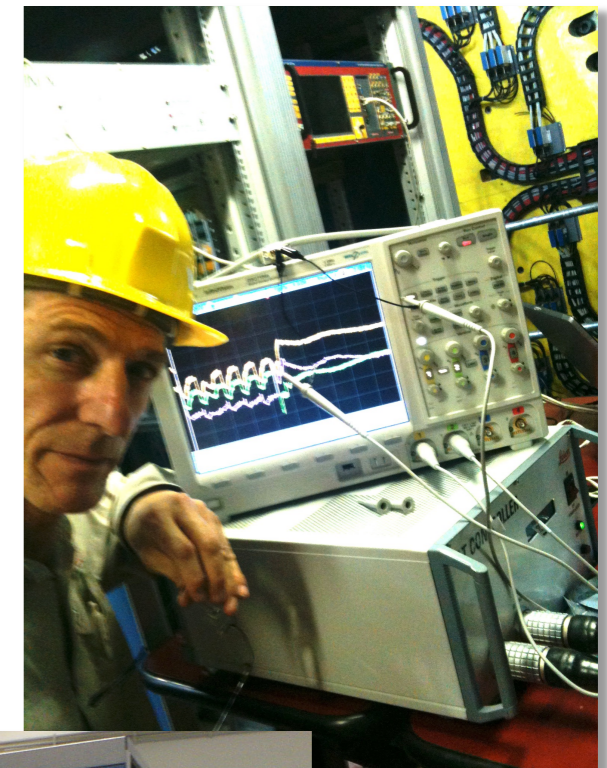
Containers, like VMs, isolate an application and its dependencies into a self-contained unit that can run anywhere.

Courtesy of Giles Knap (Diamond)

Professionals

- Data scientist
- Software Developer, Software Engineer, DevOps
- System Manager
- Network Manager
- Systems Integrator

```
mirror_mod = modifier_ob.  
# set mirror object to mirror.  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
# selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob))  
mirror_ob.select = 0  
= bpy.context.selected_obj  
data.objects[one.name].se  
print("please select exactly  
--- OPERATOR CLASSES ---  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
context):  
context.active_object is not
```



Infrastructure @LNF

PRODUCTION CLUSTER

SERVICE CLUSTER

STORAGE 1/2 PB

SPARES

SERVICE CLUSTER

DEVELOPMENT CLUSTER

PRE-PRODUCTION CLUSTER

- 376 cores
- 824 GB RAM
- 500 TB HD