Sensors, communication, and interrupts

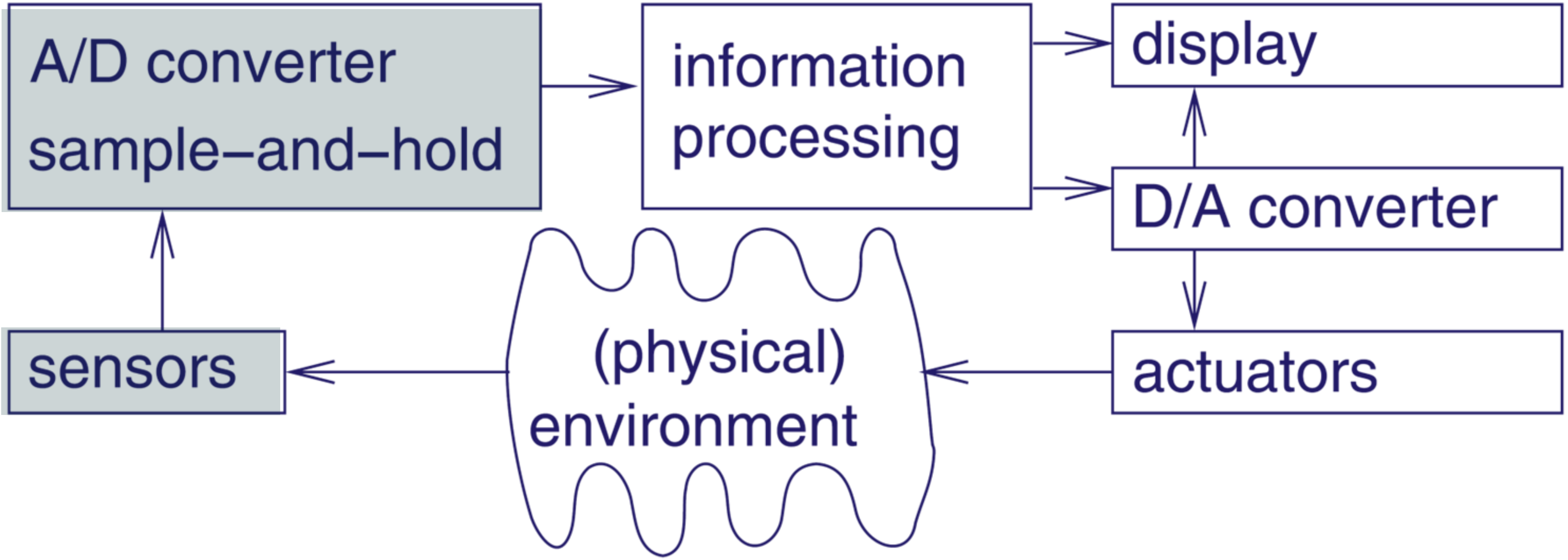
2021 DE2-Gizmo (Physical Computing)
Lecture 6

Dyson School of
Design Engineering

v1.1 – N Rojas

Sensors and Discretization

Hardware in the loop

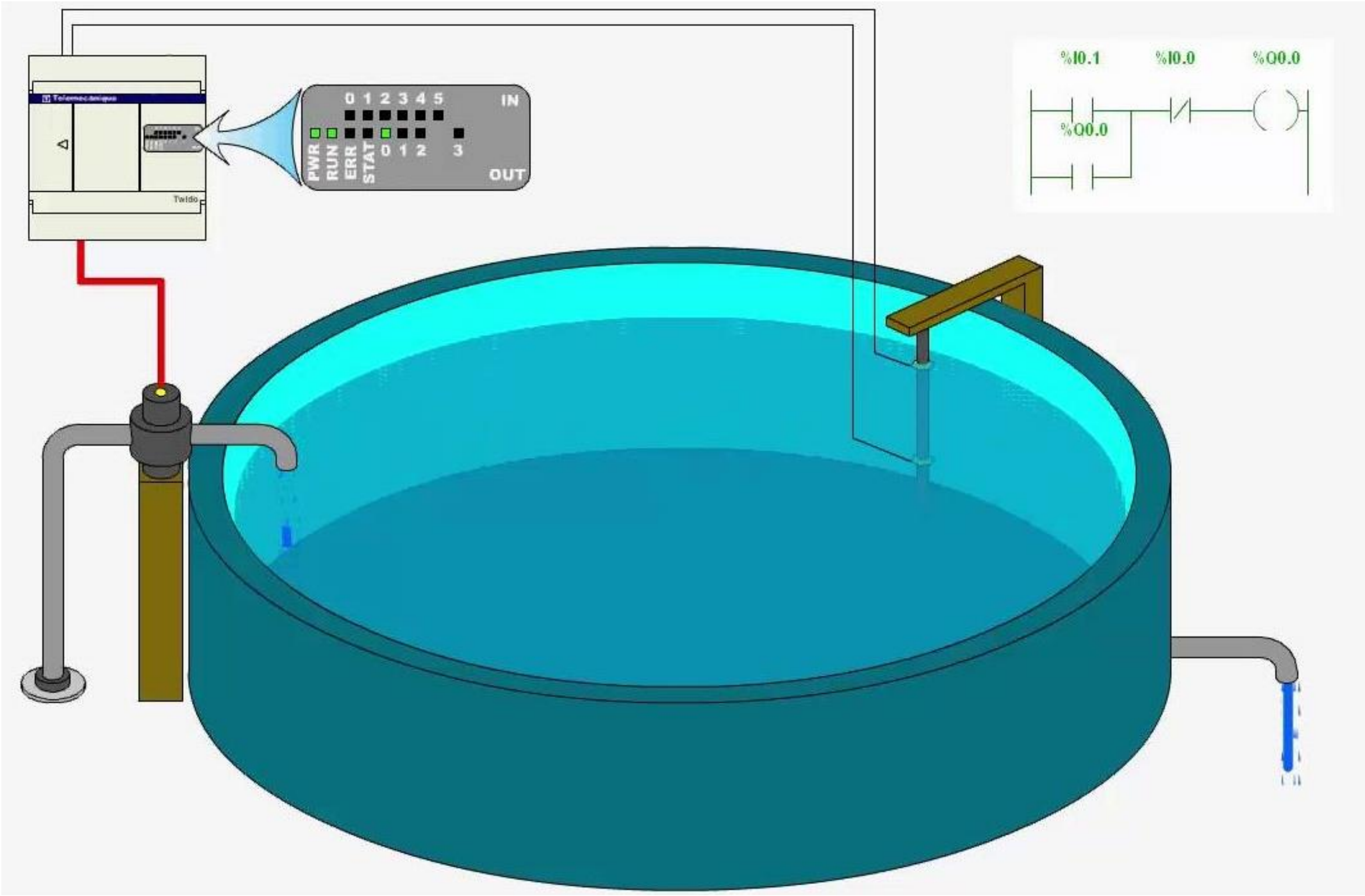


In this loop, information about the physical environment is made available through sensors

Hardware in the Loop

Some examples

- Heating
- Lights
- Engine control
- Power supply
- Tank filling
- …



Sensors

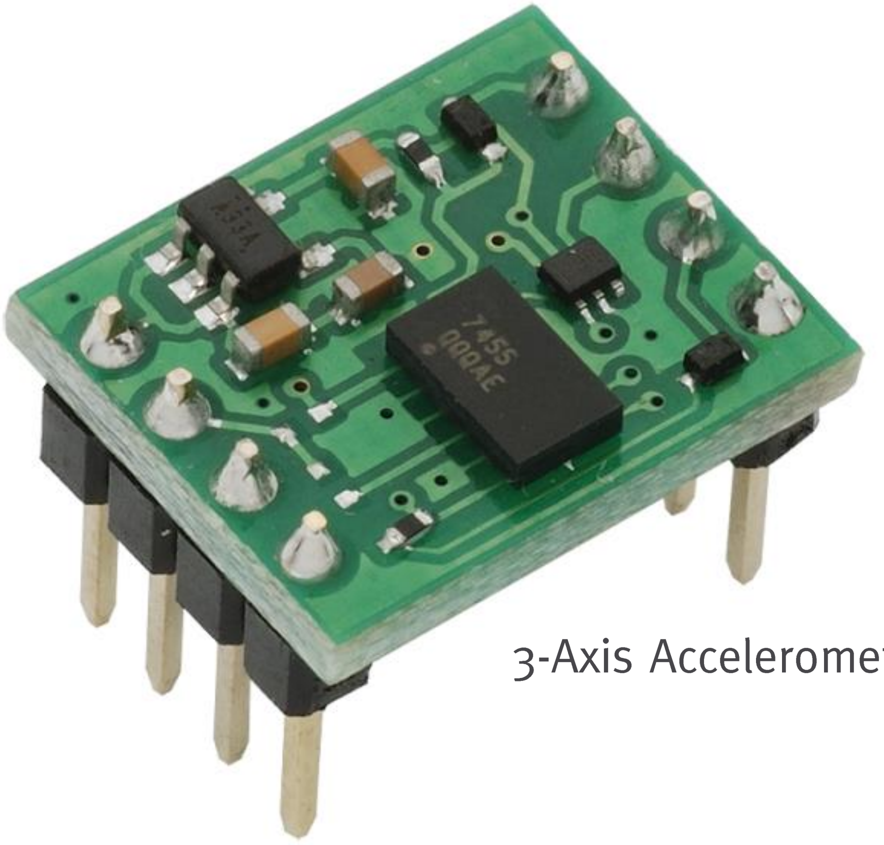
Devices that measure a physical quantity:

- Often produce a voltage that is proportional to the physical quantity
- Conversion to a number by an analogue-to-digital converter (ADC)
  - Sensor with ADC: *Digital sensor*
  - Sensor without ADC: *Analogue sensor*
- Multiple physical and chemical quantities
  - E.g., weight, velocity, acceleration, electrical current, voltage, temperatures, chemical compounds
- Physical effects used for construction
  - E.g., law of induction - generation of voltages in an electric field, light-electric effects)

Sensors

Some examples

- Cameras
- Accelerometers
- Rate gyros
- Strain gauges
- Microphones
- Magnetometers
- Radar/Lidar
- Chemical sensors
- Pressure sensors
- Rain sensors
- Switches
- …



3-Axis Accelerometer

Issues with Sensors

- Calibration/Characterisation
  - Relating measurements to the physical phenomenon
  - Can dramatically increase manufacturing costs
- Nonlinearity
  - Measurements may not be proportional to physical phenomenon
  - Correction may be required
  - Feedback can be used to keep operating point in the linear region
- Sampling
  - Aliasing
  - Missed events
- Noise
  - Analogue signal conditioning
  - Digital filtering
  - Introduces latency

Accelerometer

It measures proper acceleration: *acceleration + gravitational force*. It may be challenging to separate them. Accelerometers can measure tilt relative to gravity

b-bit accelerometer

$$f : (L, H) \to \{0, \ldots, 2^b - 1\}$$

Precision

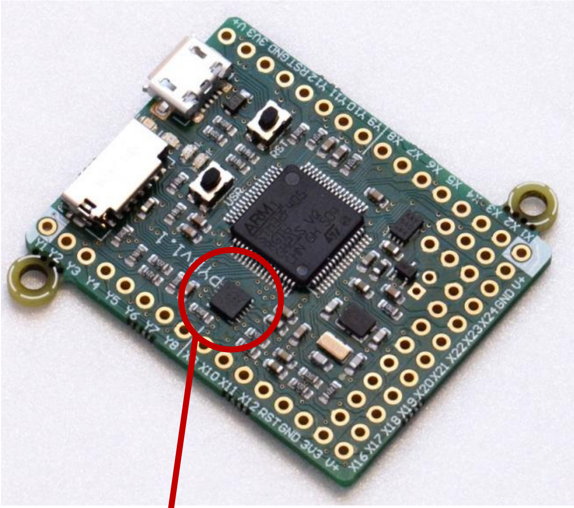
$$p = (H - L)/2^n$$

L: Minimum measurable value

H: Maximum measurable value

Accelerometer

Example: Accelerometer on Pyboard



Pyboard v1.1
Single-board microcontroller
Microcontroller: *STM32F405RG (ARM Cortex-M4)*
Language: *MicroPython (Python 3)*

MMA7660FC
±1.5g 6-bit 3-Axis
Orientation/Motion Detection Sensor

Precision

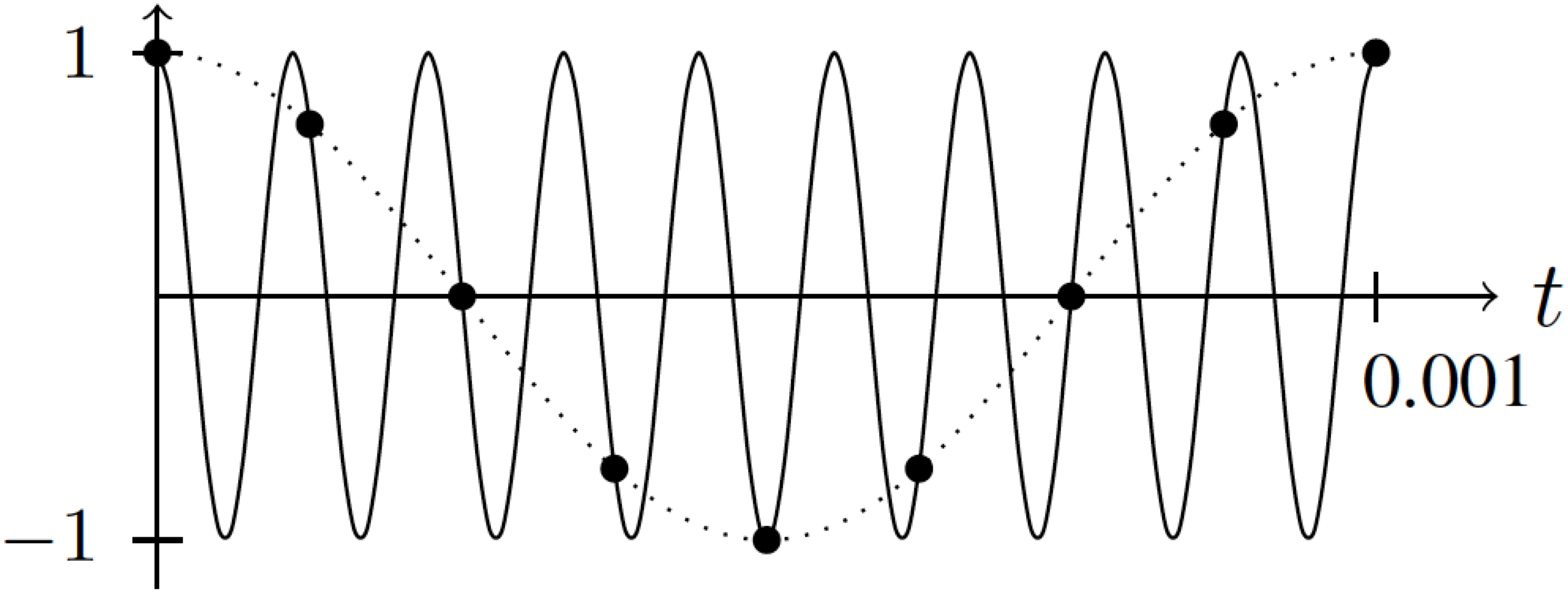
$$p = \frac{H - L}{2^n} = \frac{(1.5\ g - (-1.5\ g))}{2^6} = \frac{3\ g}{64} = 0.047\ g$$

Discretization

From the analogue world to the digital world

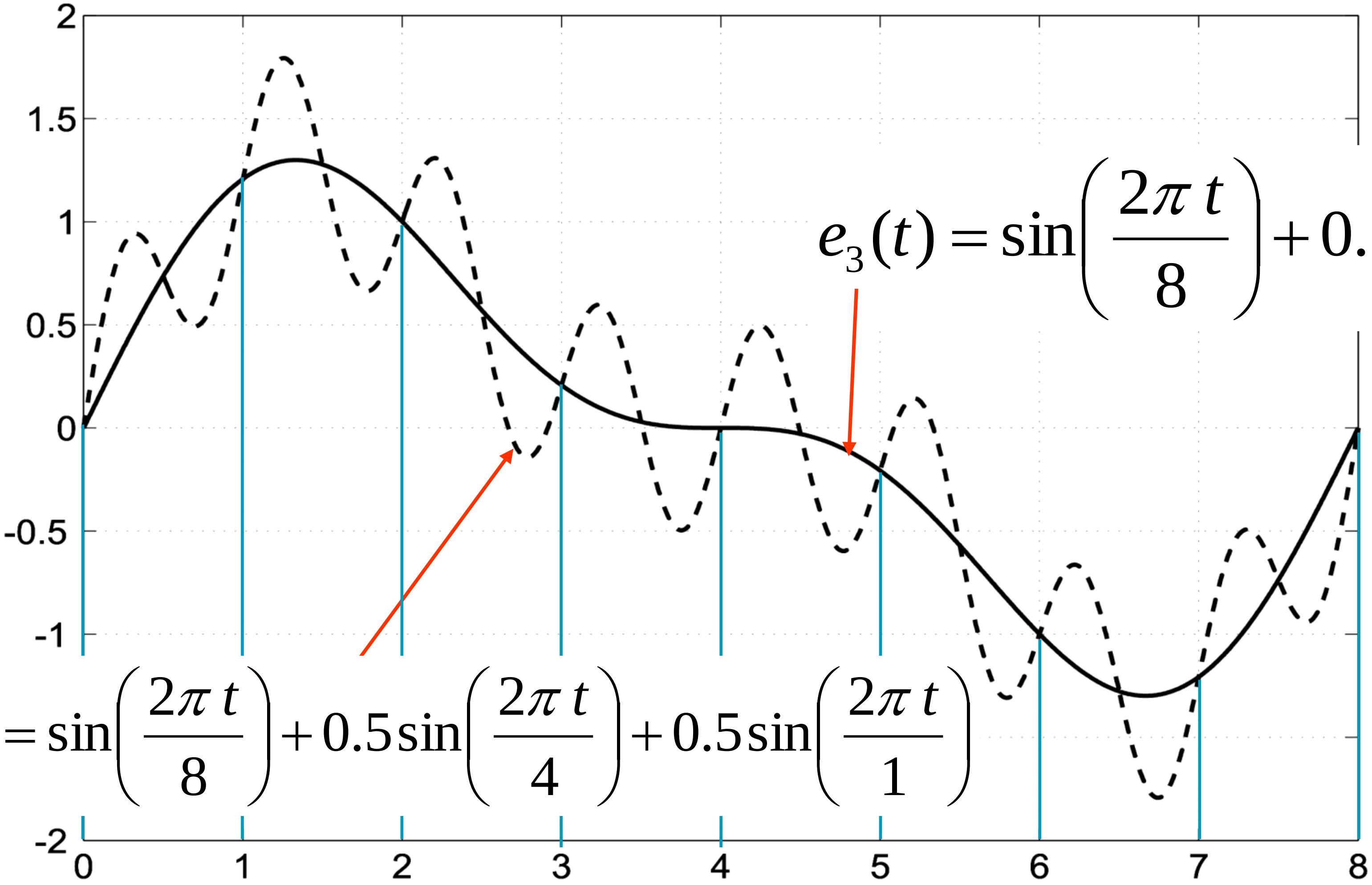
- Discretization of time
- Discretization of values

Discretization of Time



The fact that two or more signals can have the same sampled representation is called **aliasing**

Discretization of Time



$$e_3(t) = \sin\left(\frac{2\pi t}{8}\right) + 0.5\sin\left(\frac{2\pi t}{4}\right)$$

$$e_4(t) = \sin\left(\frac{2\pi t}{8}\right) + 0.5\sin\left(\frac{2\pi t}{4}\right) + 0.5\sin\left(\frac{2\pi t}{1}\right)$$

Indistinguishable if sampled at integer times, p_s=1

Discretization of Time

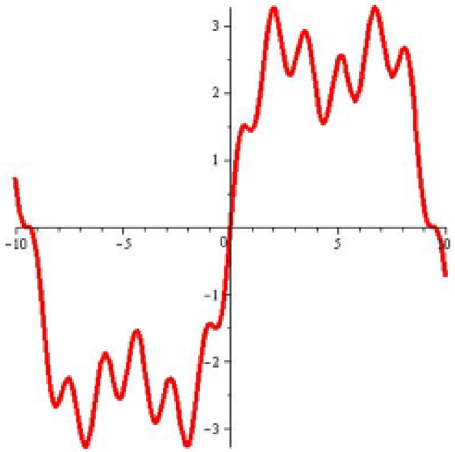
Aliasing is avoided if the frequencies of the incoming signal are restricted to less than half of the sampling frequency f_s

Nyquist sampling criterion

$f_s \quad > \quad 2f_N$ where f_N is the frequency of the "fastest" sine wave

f_N is called the Nyquist frequency
f_s is the sampling rate

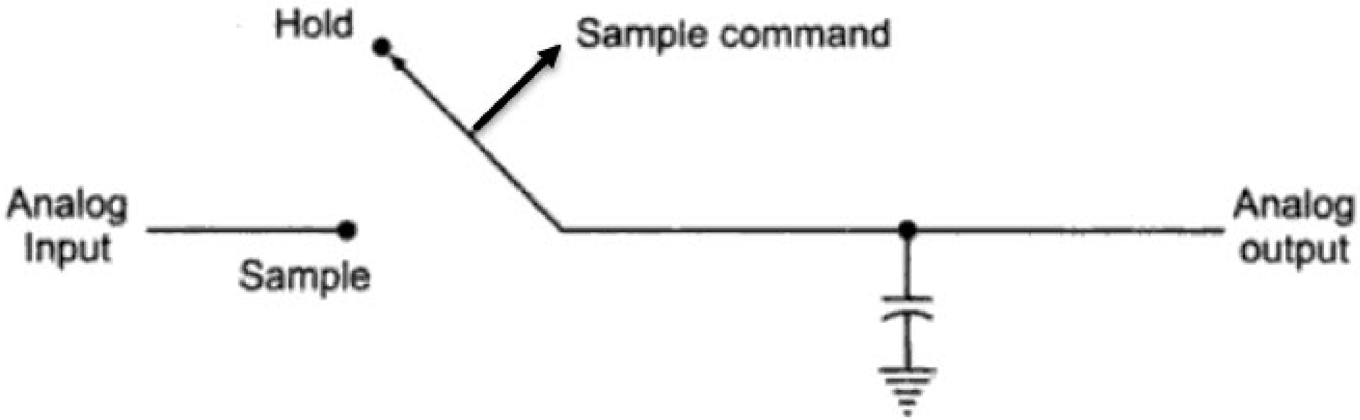
Assume that we have an input signal x consisting of the sum of the sine waves $A_1 \sin t$, $A_2 \sin \frac{1}{3} t$, and $A_3 \sin 4t$. We are sampling x at a rate of 1 Hz. Will we be able to reconstruct the original signal after discretisation of time?

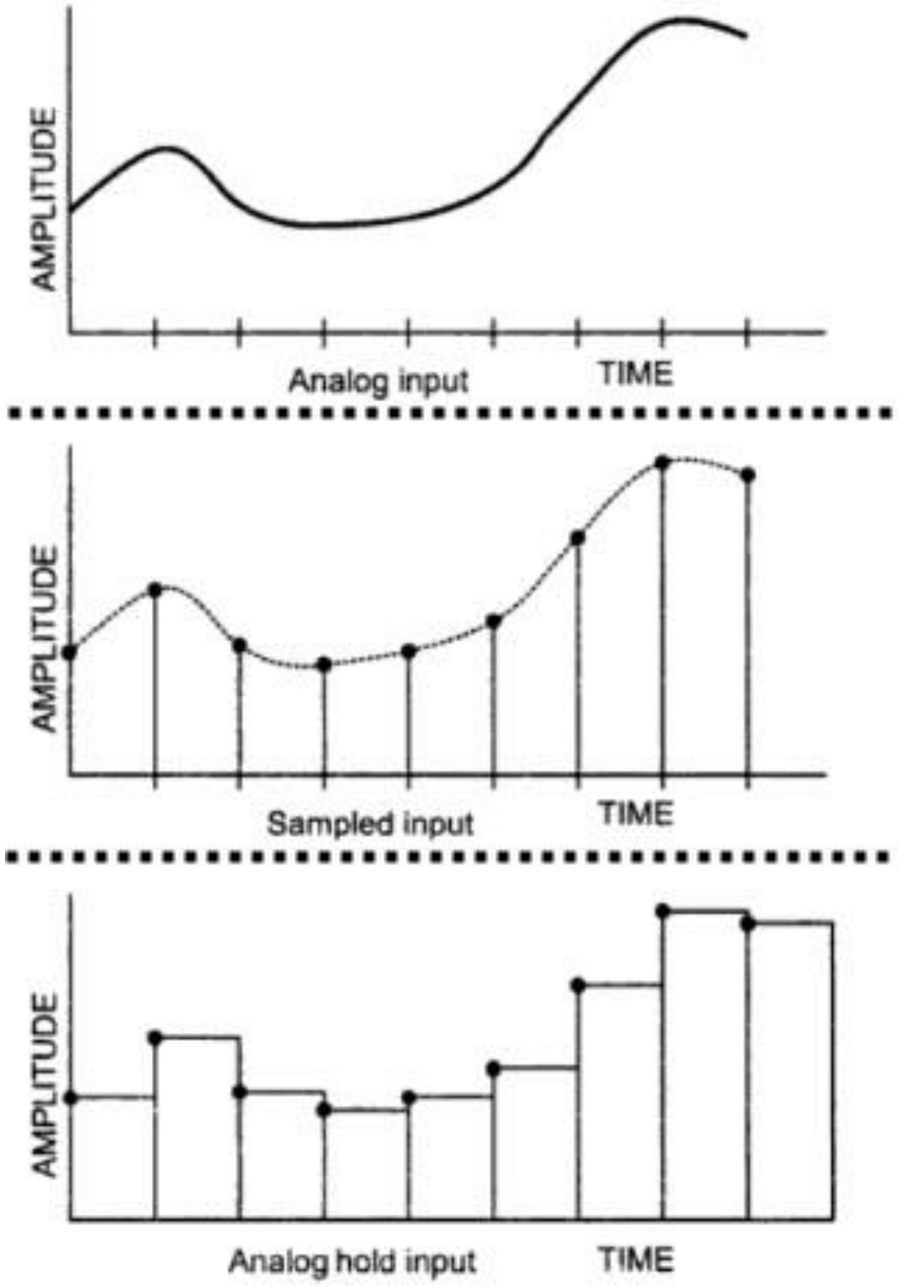


What do you think?

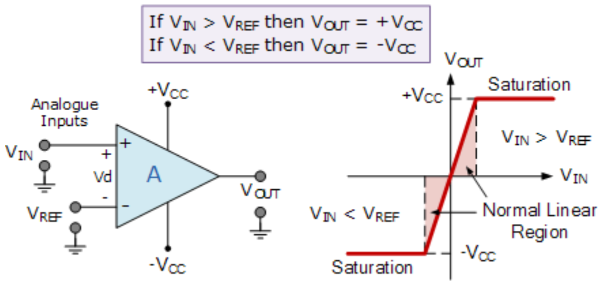
Discretization of Time

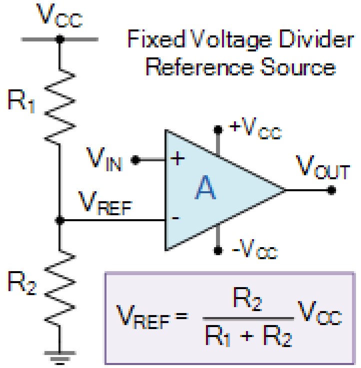
Sample and Hold

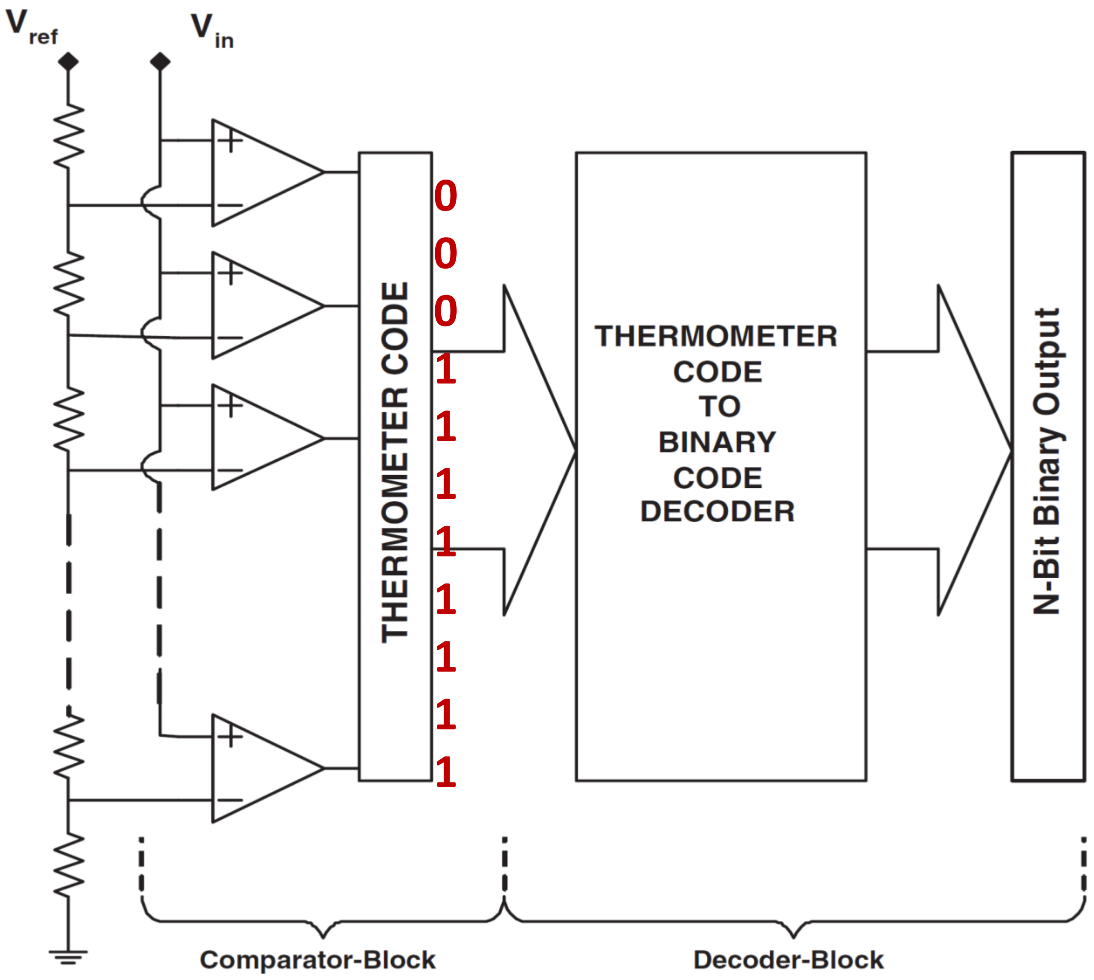




Discretization of Values

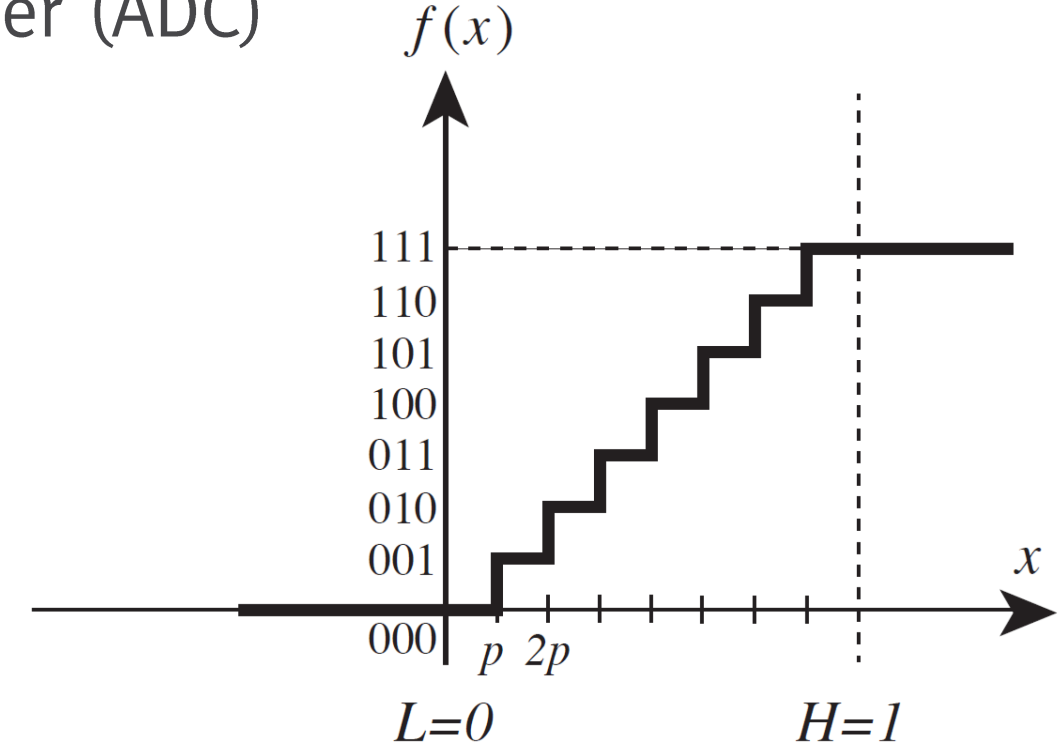




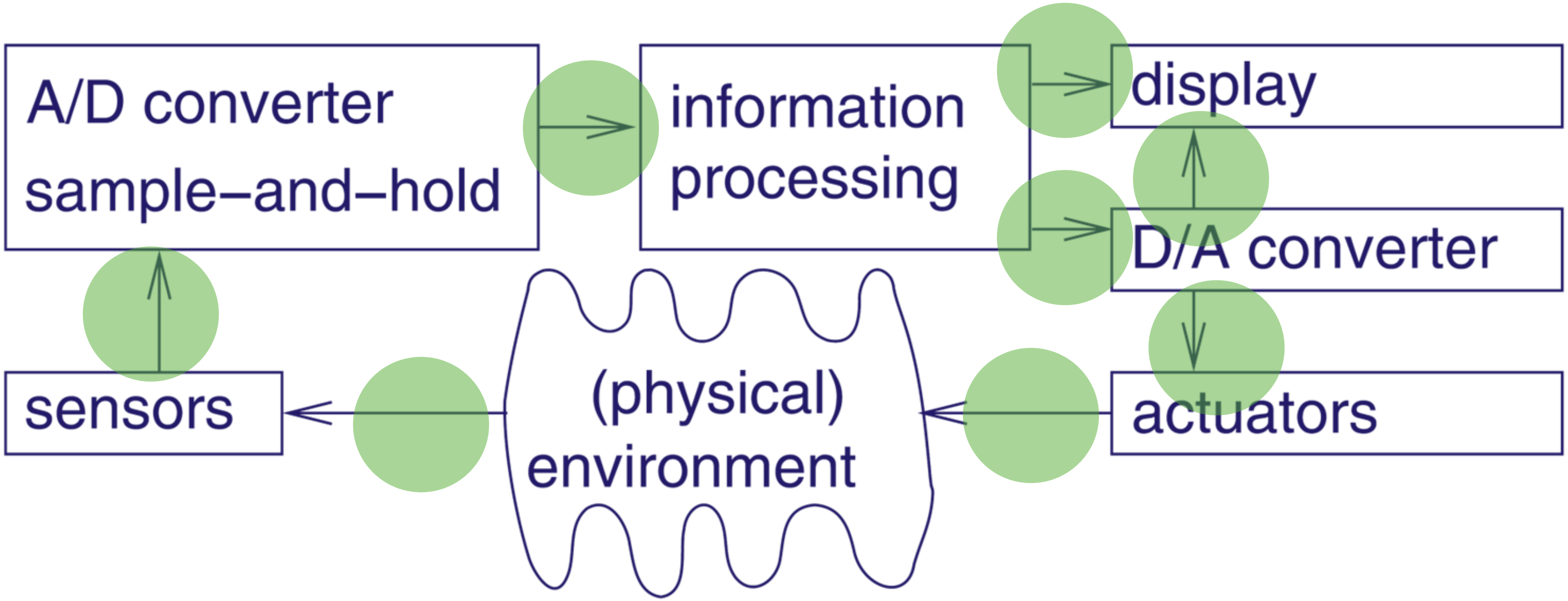


Flash A/D converter

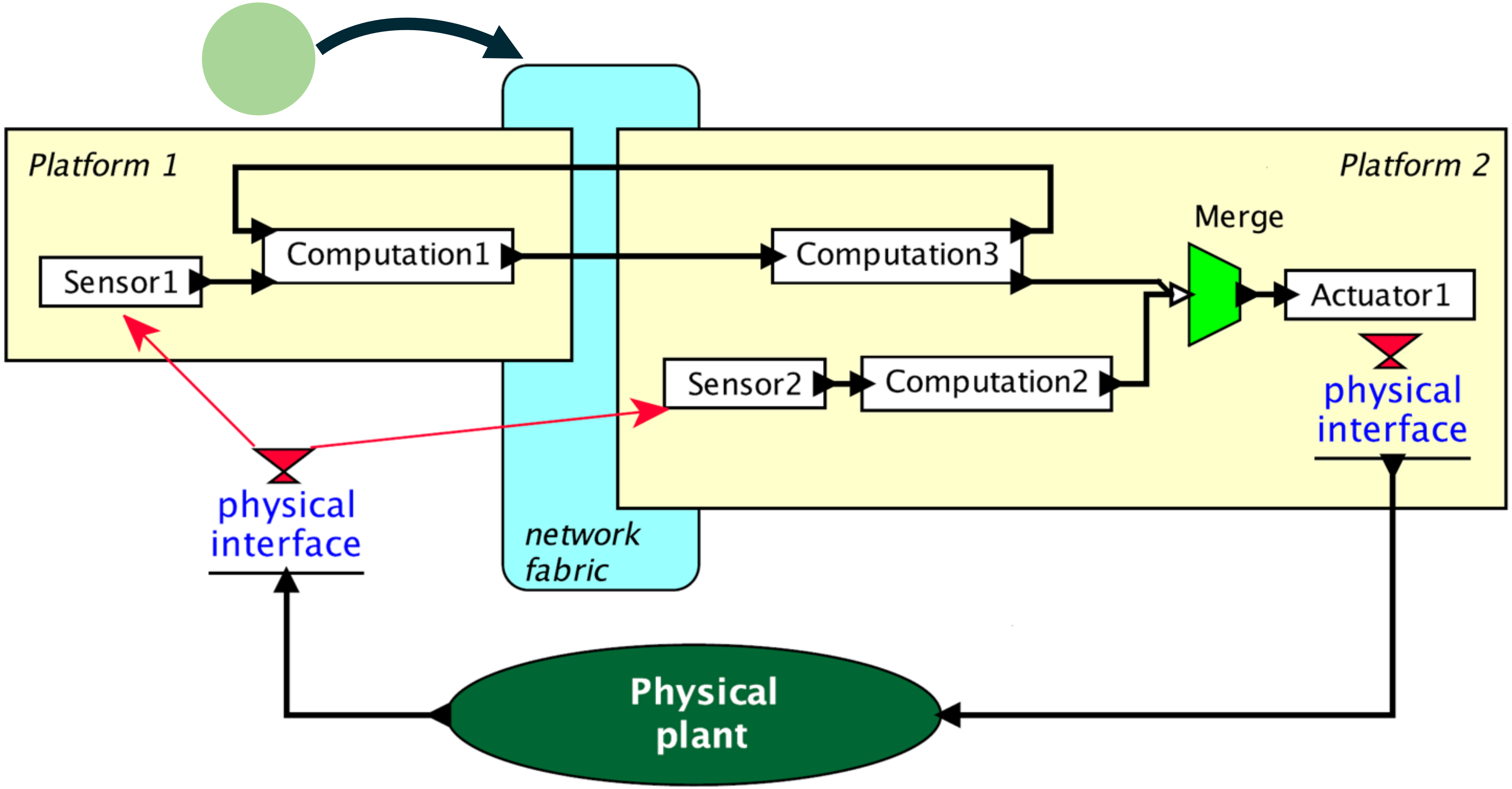
Conversion of a voltage to a number. Analog-to-digital converter (ADC)



Communication Interfaces



Communication Interfaces



Basics

Channels

Abstract entities characterized by the essential properties of communication (e.g., maximum information transfer capacity, noise parameters)

Media

The physical entities enabling communication (e.g., wireless media: radio frequency media, infrared, optical media: fibers, wires)

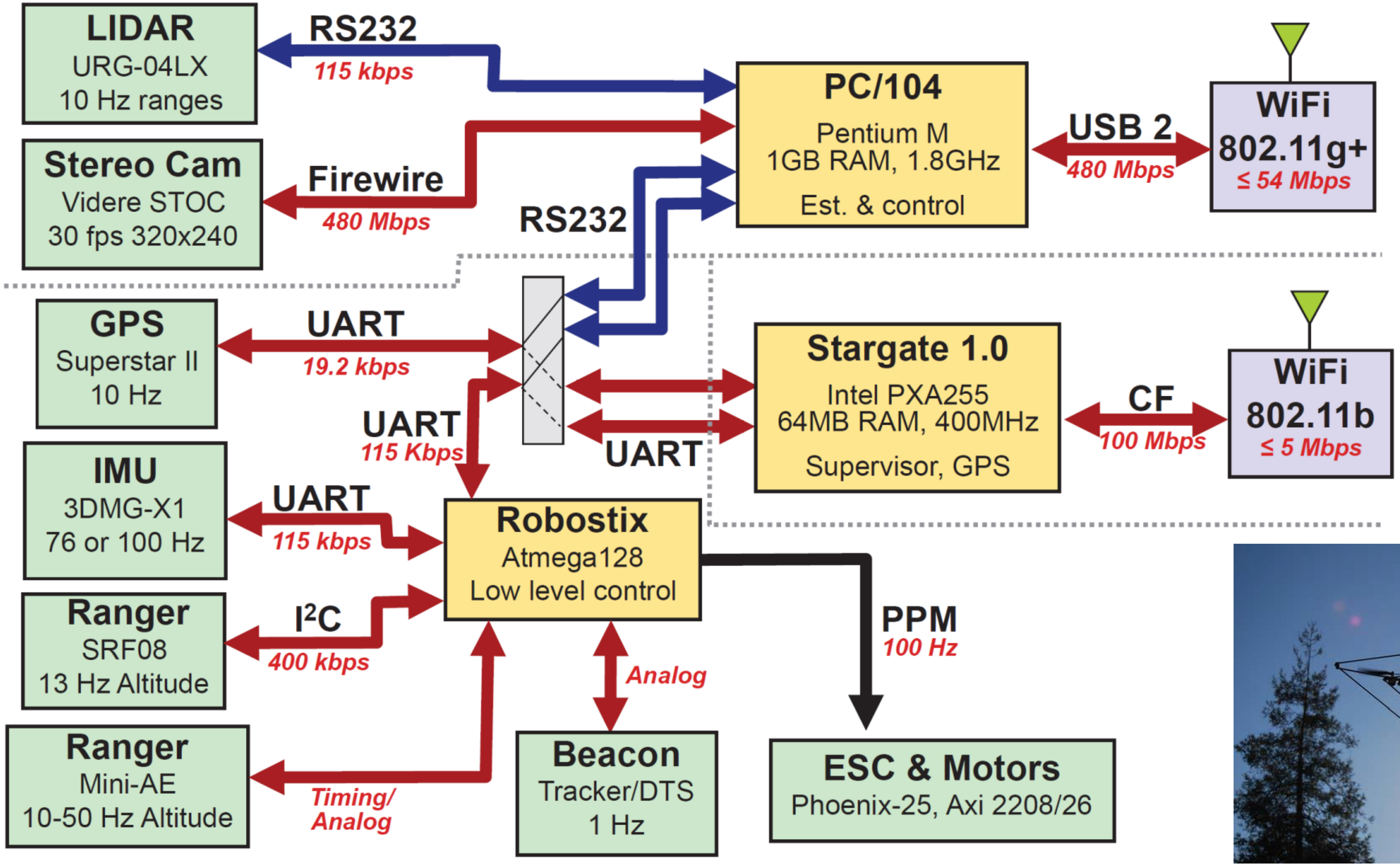
Common Requirements in Cyber-Physical Systems

- Real-time behaviour
- Efficiency (cost-efficient designs)
- Appropriate bandwidth and communication delay
- Support for event-driven communication
- Robustness (e.g. extreme temperatures)
- Fault tolerance
- Maintainability
- Privacy

Some Relevant Examples

- Sensor/actuator buses and point-to-point: for attaching peripherals to processors and microcontrollers, (in general) low rates and short distances
  - UART, I^2C, USB
- Wireless communication
  - Bluetooth, IEEE 802.11 (Wi-Fi), ZigBee

Structure of a Cyber-Physical System

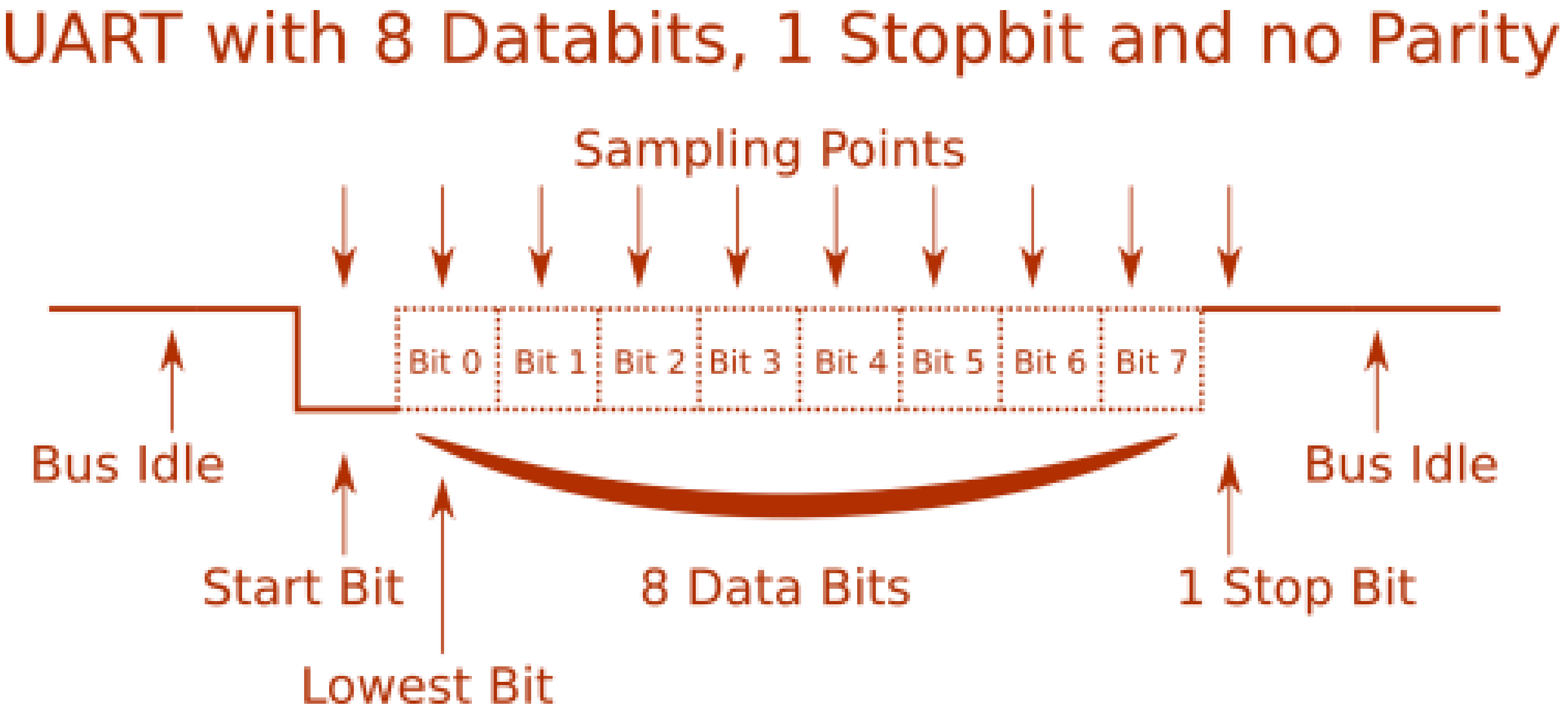


STARMAC

UART

Universal asynchronous receiver/transmitter

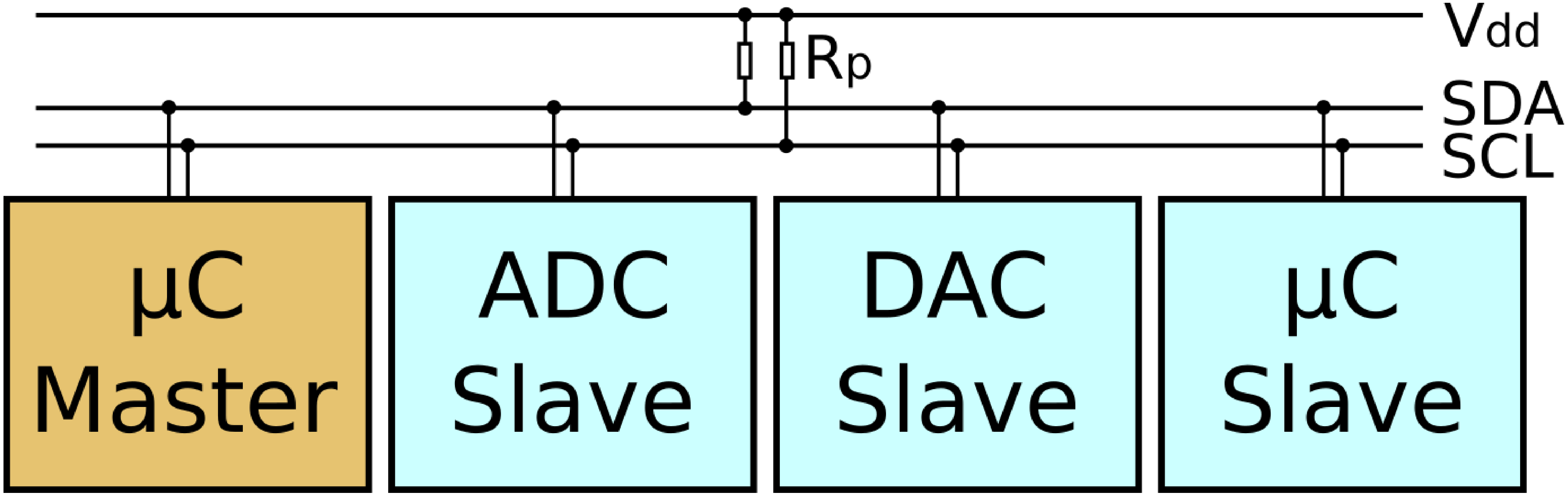
- Bit-serial protocol
- Asynchronous (no clock signal)
- Point-to-point



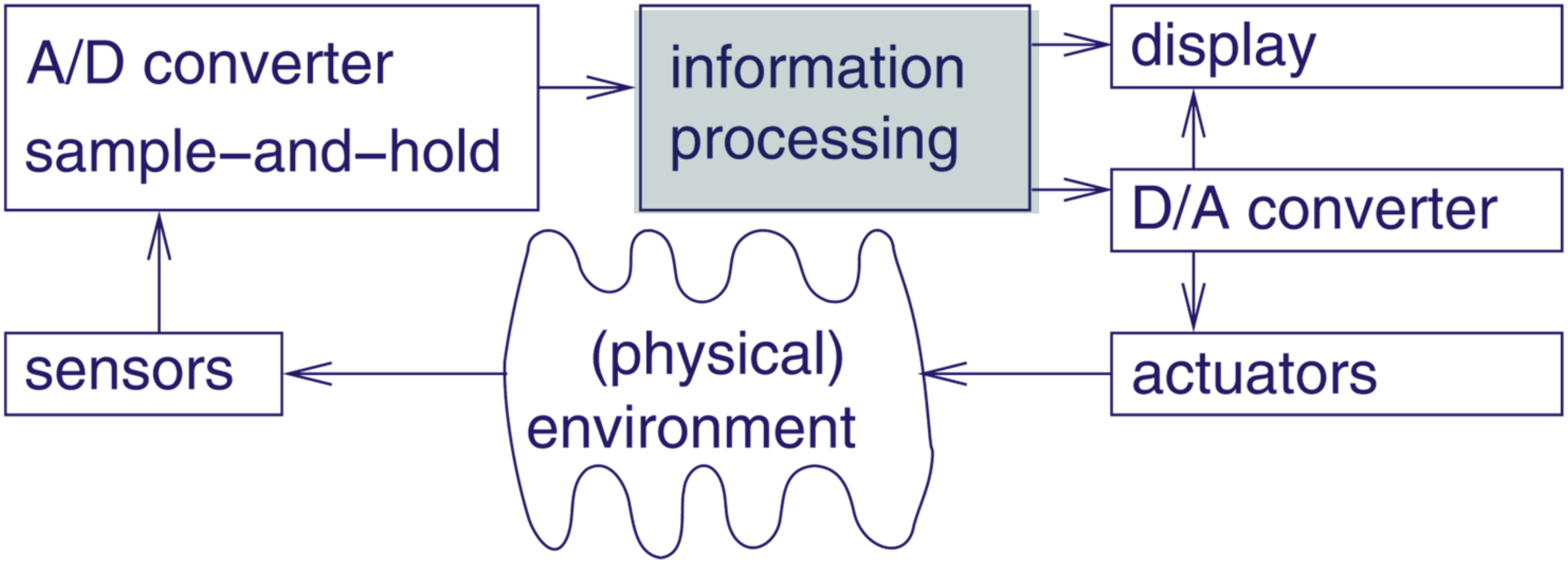
I²C

Inter-integrated circuit

- Bit-serial protocol
- Synchronous (clock signal)
- Multi-master, multi-slave



Interrupts



A/D converter
sample–and–hold
information
processing
display
D/A converter
sensors
(physical)
environment
actuators

Concurrency and Parallelism

A parallel program is concurrent, but a concurrent program need not be parallel

- A program is said to be concurrent if different parts of the program **conceptually execute simultaneously**
- A program is said to be parallel if different parts of the program **physically execute simultaneously on distinct hardware**

Concurrency in Computing

- Interrupt Handling
  - Reacting to external events (interrupts)
  - Exception handling (software interrupts)
- Processes
  - Creating the illusion of simultaneously running different programs (multitasking)
- Threads
  - How is a thread different from a process?
- Multiple processors (multi-cores)

Interrupts

Sequential software in a concurrent world

Semantic mismatch

Cyber:
- Digital
- Discrete in time
- Sequential

Physical:
- Continuum
- Continuous in time
- Concurrent

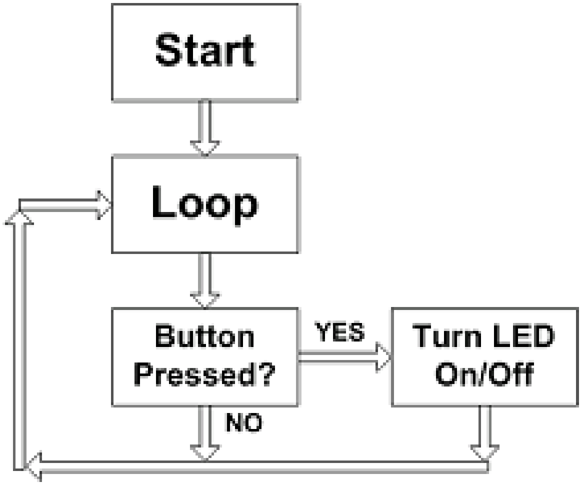
Interrupts

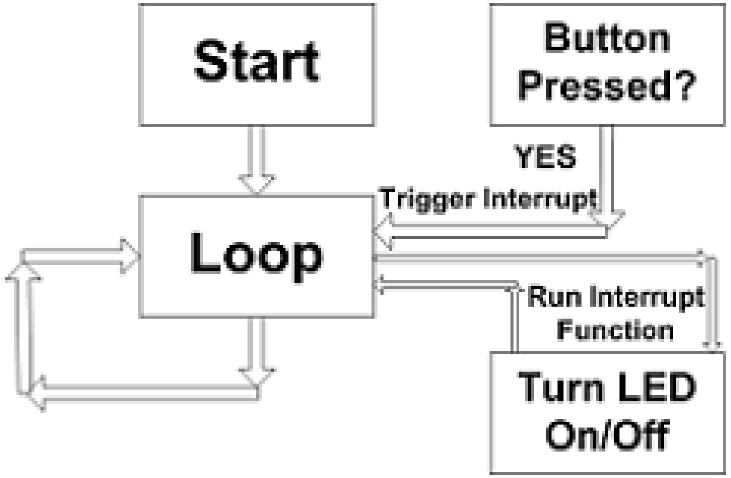
Mechanism for pausing execution of whatever a processor is currently doing and executing a ***pre-defined code sequence***

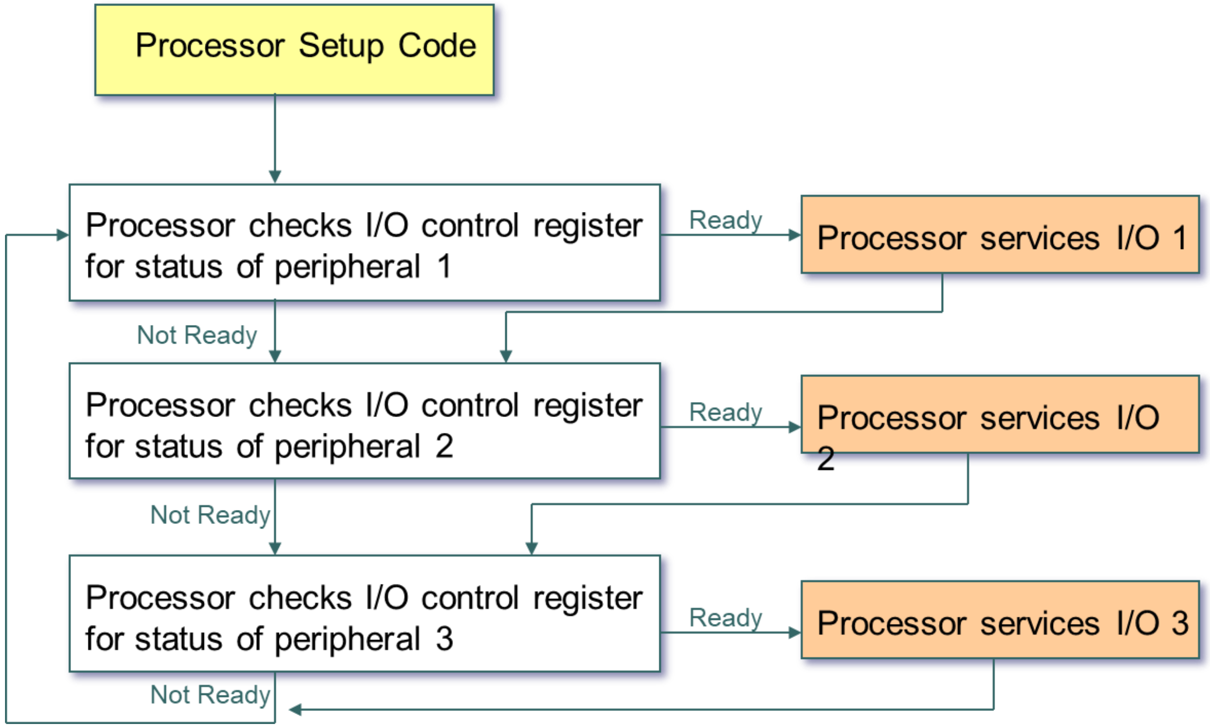
Code sequence is called **Interrupt Service Routine (ISR)** or interrupt handler

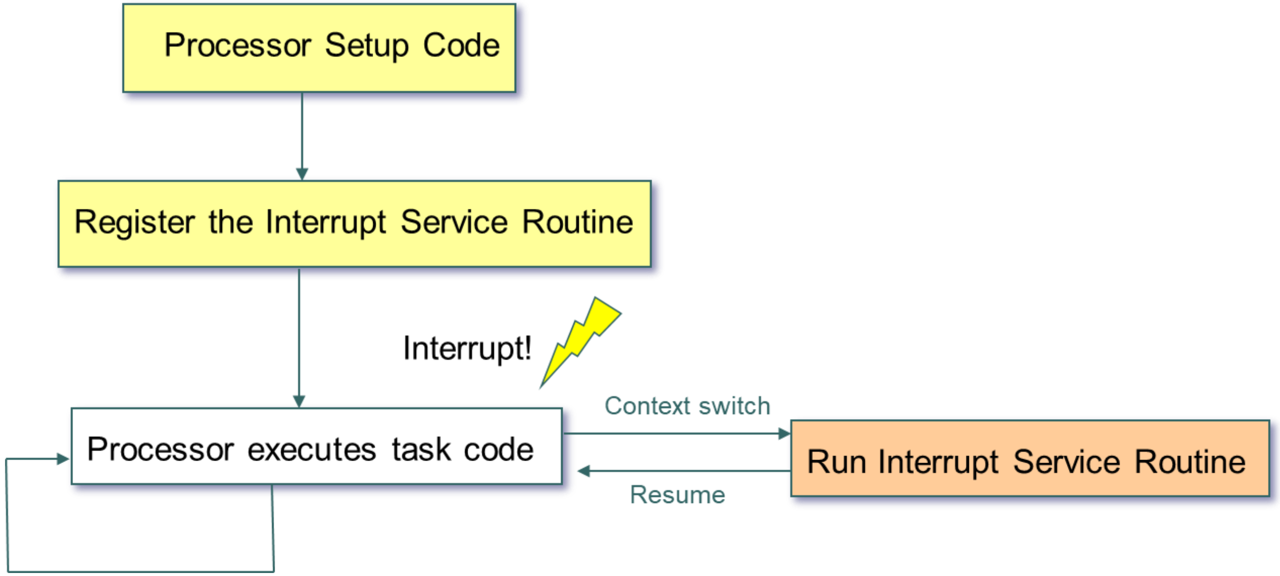
Interrupts != Polling

Polling vs Interrupts









Interrupts

Three kinds of events may trigger an interrupt:

- <u>Hardware interrupt: *external hardware*</u>
- Software interrupt: *programme*
- Exception: *internal hardware (faults)*

Digital pins usable for interrupts in Mega 2560: 2, 3, 18, 19, 20, 21

HARDWARE SOFTWARE DOCUMENTATION COMMUNITY BLOG ABOUT

Reference > Language > Functions > External interrupts > Attachinterrupt

attachInterrupt()

[External Interrupts]

Description

Digital Pins With Interrupts

Arduino

Sensors, communication, and interrupts

2021 DE2-Gizmo (Physical Computing)
Lecture 6

Dyson School of
Design Engineering

v1.1 – N Rojas