Lecture 1 - Introduction

CSE 456: Embedded Systems

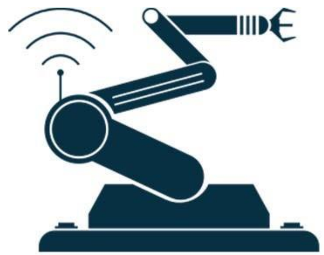
Embedded Systems

Embedded systems (ES) = **information processing systems embedded into a larger product**

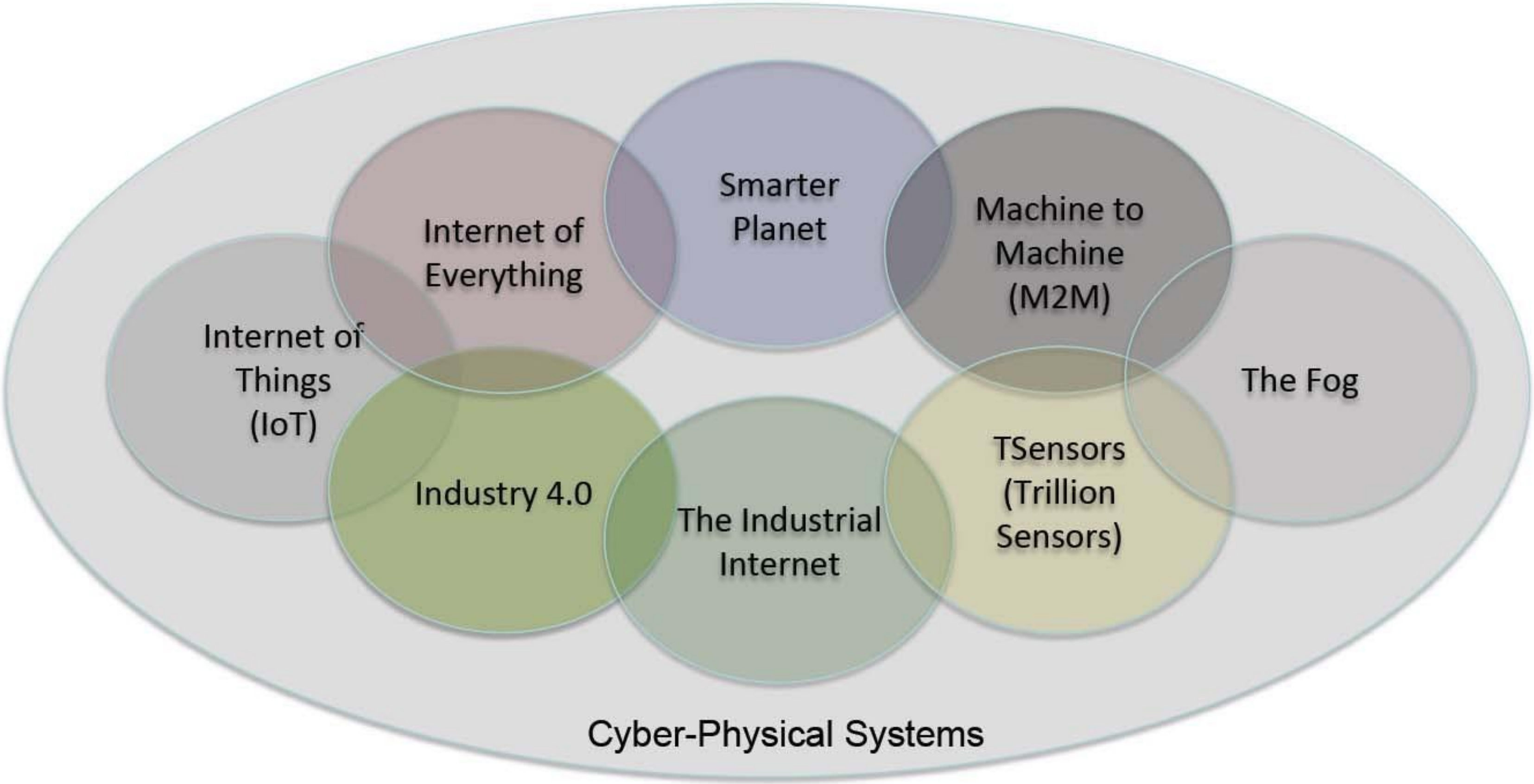
Embedded Systems

Embedded systems (ES) = **information processing systems embedded into a larger product**

Examples:



Many Names – Similar Meanings



Embedded Systems are an essential component

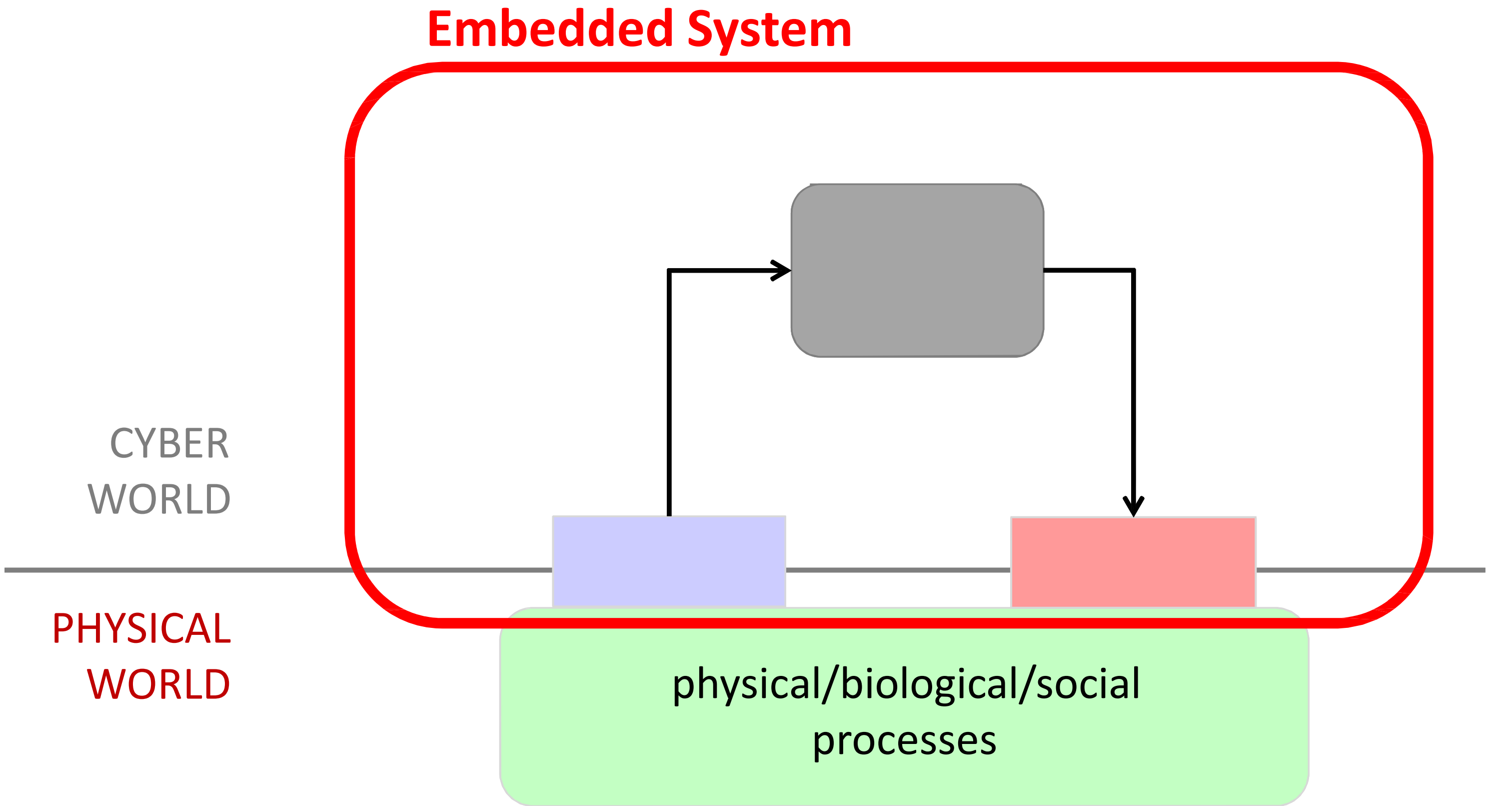
Twelve potentially economically disruptive technologies

Mobile Internet	Increasingly inexpensive and capable mobile computing devices and Internet connectivity
Automation of knowledge work	Intelligent software systems that can perform knowledge work tasks involving unstructured commands and subtle judgments
The Internet of Things	Networks of low-cost sensors and actuators for data collection, monitoring, decision making, and process optimization
Cloud technology	Use of computer hardware and software resources delivered over a network or the Internet, often as a service
Advanced robotics	Increasingly capable robots with enhanced senses, dexterity, and intelligence used to automate tasks or augment humans
Autonomous and near-autonomous vehicles	Vehicles that can navigate and operate with reduced or no human intervention

…

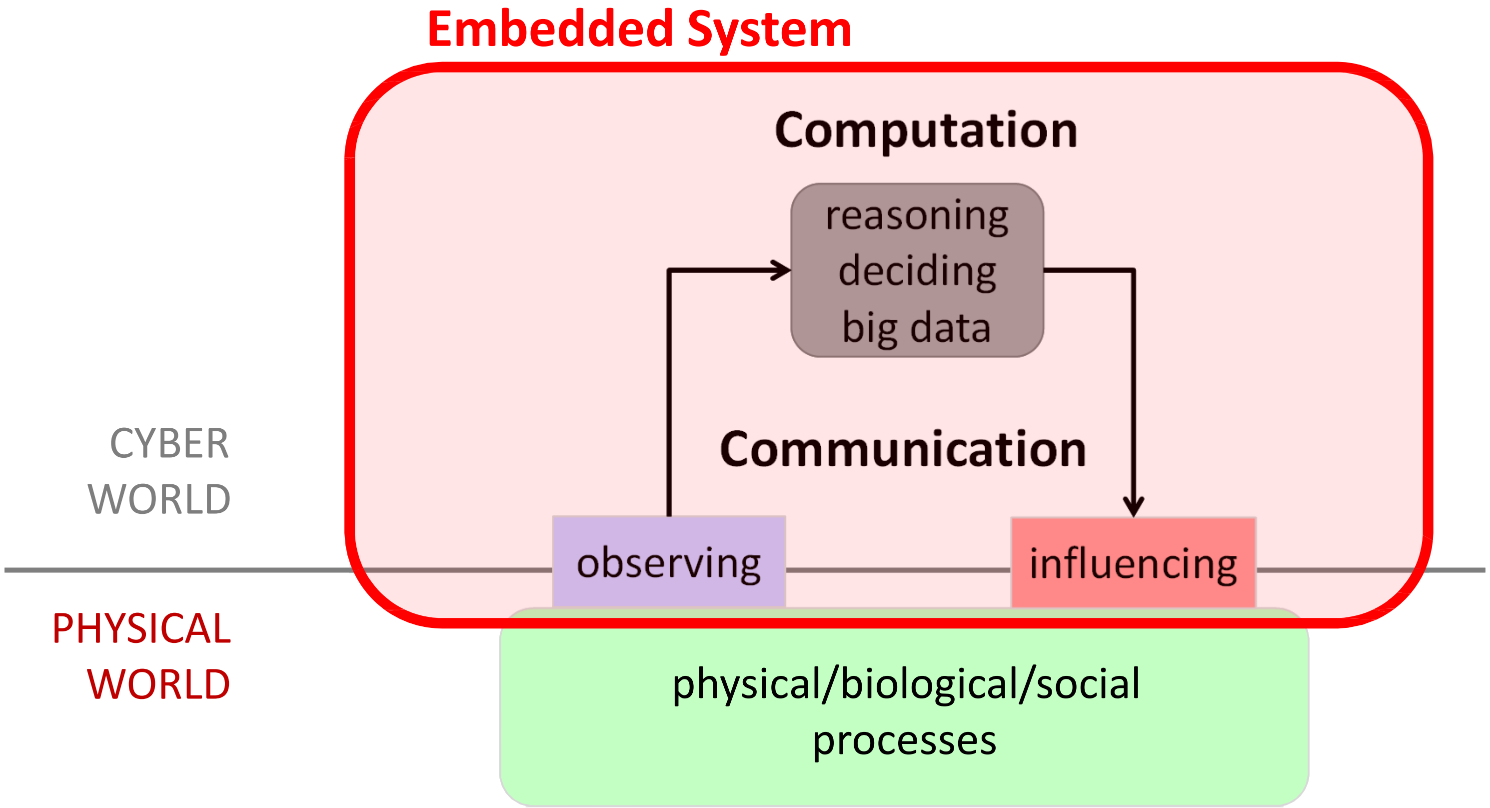
Next-generation genomics	Fast, low-cost gene sequencing, advanced big data analytics, and synthetic biology ("writing" DNA)
Energy storage	Devices or systems that store energy for later use, including batteries
3D printing	Additive manufacturing techniques to create objects by printing layers of material based on digital models
Advanced materials	Materials designed to have superior characteristics (e.g., strength, weight, conductivity) or functionality
Advanced oil and gas exploration and recovery	Exploration and recovery techniques that make extraction of unconventional oil and gas economical
Renewable energy	Generation of electricity from renewable sources with reduced harmful climate impact

Embedded System



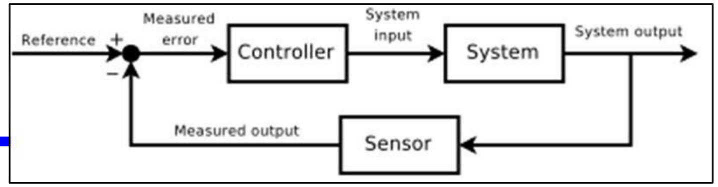
Use feedback to influence the dynamics of the physical world by taking smart decisions in the cyber world

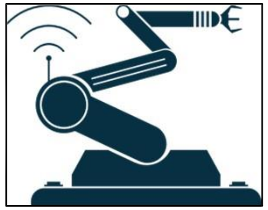
Embedded System



Use feedback to influence the dynamics of the physical world by taking smart decisions in the cyber world

Predictability & Dependability





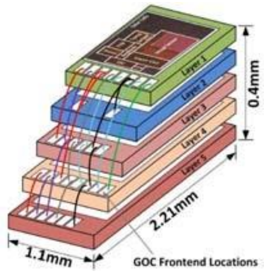
CPS = cyber-physical system

“It is essential to *predict* how a CPS is going to behave under any circumstances […] *before* it is deployed.

“CPS must *operate dependably*, safely, securely, efficiently and in real-time.”

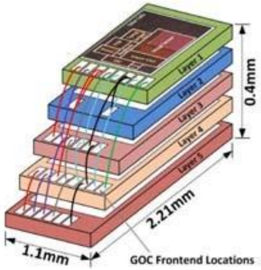
Efficiency & Specialization

- Embedded systems must be *efficient:*
  - *Energy* efficient
  - *Code-size* and *data memory* efficient
  - *Run-time* efficient
  - *Weight* efficient
  - *Cost* efficient



Efficiency & Specialization

- Embedded systems must be *efficient:*
  - *Energy* efficient
  - *Code-size* and *data memory* efficient
  - *Run-time* efficient
  - *Weight* efficient
  - *Cost* efficient



Embedded Systems are often *specialized* towards a certain application or application domain:

- Knowledge about the expected behavior and the system environment at design time is exploited to *minimize resource usage* and to *maximize predictability and reliability*.

Reactivity & Timing

Embedded systems are often reactive:

- Reactive systems must react to stimuli from the system environment
  :

> *"A reactive system is one which is in continual interaction with its environment and executes at a pace determined by that environment"*

Embedded systems often must meet *real-time constraints:*

- For hard real-time systems, right answers arriving too late are wrong. All other time-constraints are called soft. A *guaranteed system response* has to be explained without statistical arguments.

> *"A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe"*

Comparison

Embedded Systems:

General Purpose Computing

Comparison

Embedded Systems:

General Purpose Computing

- Broad class of applications.
- Programmable by end user.
- Faster is better.
- Typical criteria:
  - cost
  - power consumption
  - average speed

Comparison

Embedded Systems:

- Few applications that are known at design-time.
- Not programmable by end user.
- Fixed run-time requirements (additional computing power often not useful).
- Typical criteria:
  - cost
  - power consumption
  - size and weight
  - dependability
  - worst-case speed

General Purpose Computing

- Broad class of applications.
- Programmable by end user.
- Faster is better.
- Typical criteria:
  - cost
  - power consumption
  - average speed

Components and Requirements by Example

Components and Requirements by Example
- Hardware System Architecture -

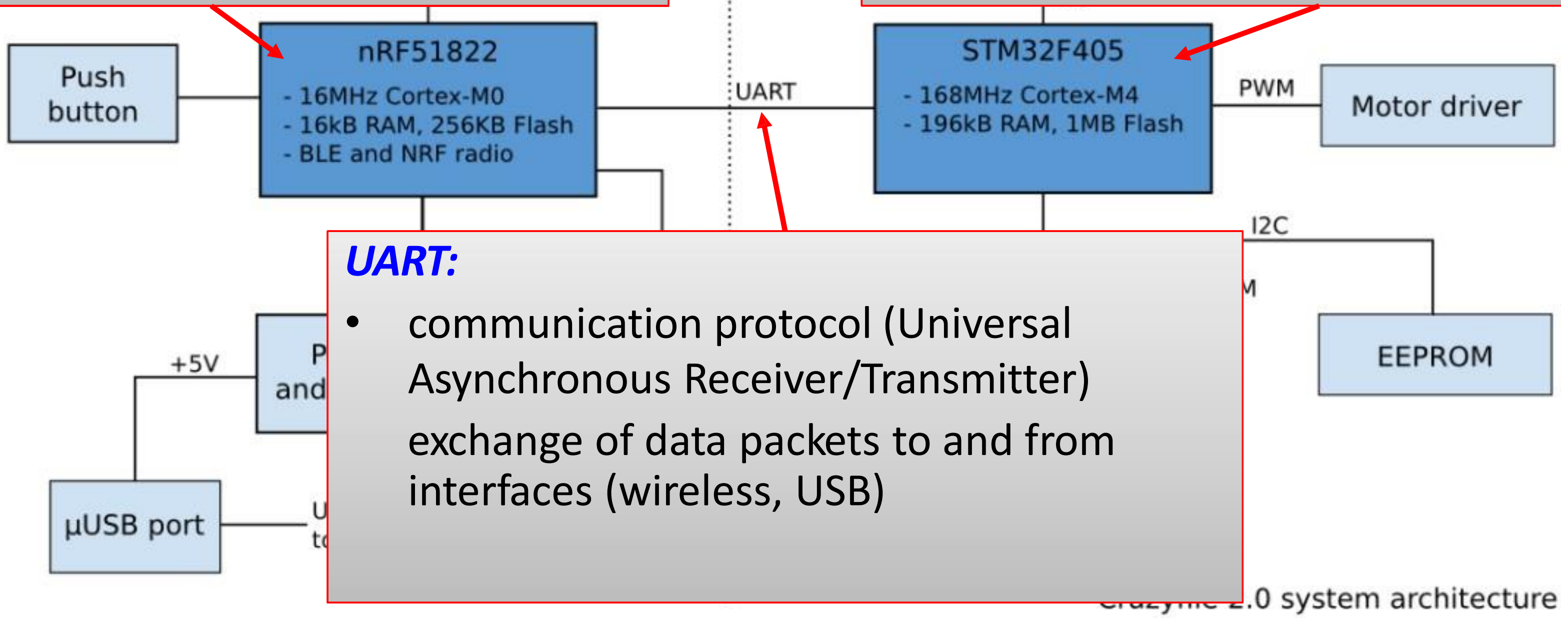
High-Level Block Diagram View

low power CPU

- enabling power to the rest of the system
- battery charging and voltage measurement
- wireless radio (boot and operate)
- detect and check expansion boards

higher performance CPU

- sensor reading and motor control
- flight control
- telemetry (including the battery voltage)
- additional user development
- USB connection



UART:

- communication protocol (Universal Asynchronous Receiver/Transmitter)

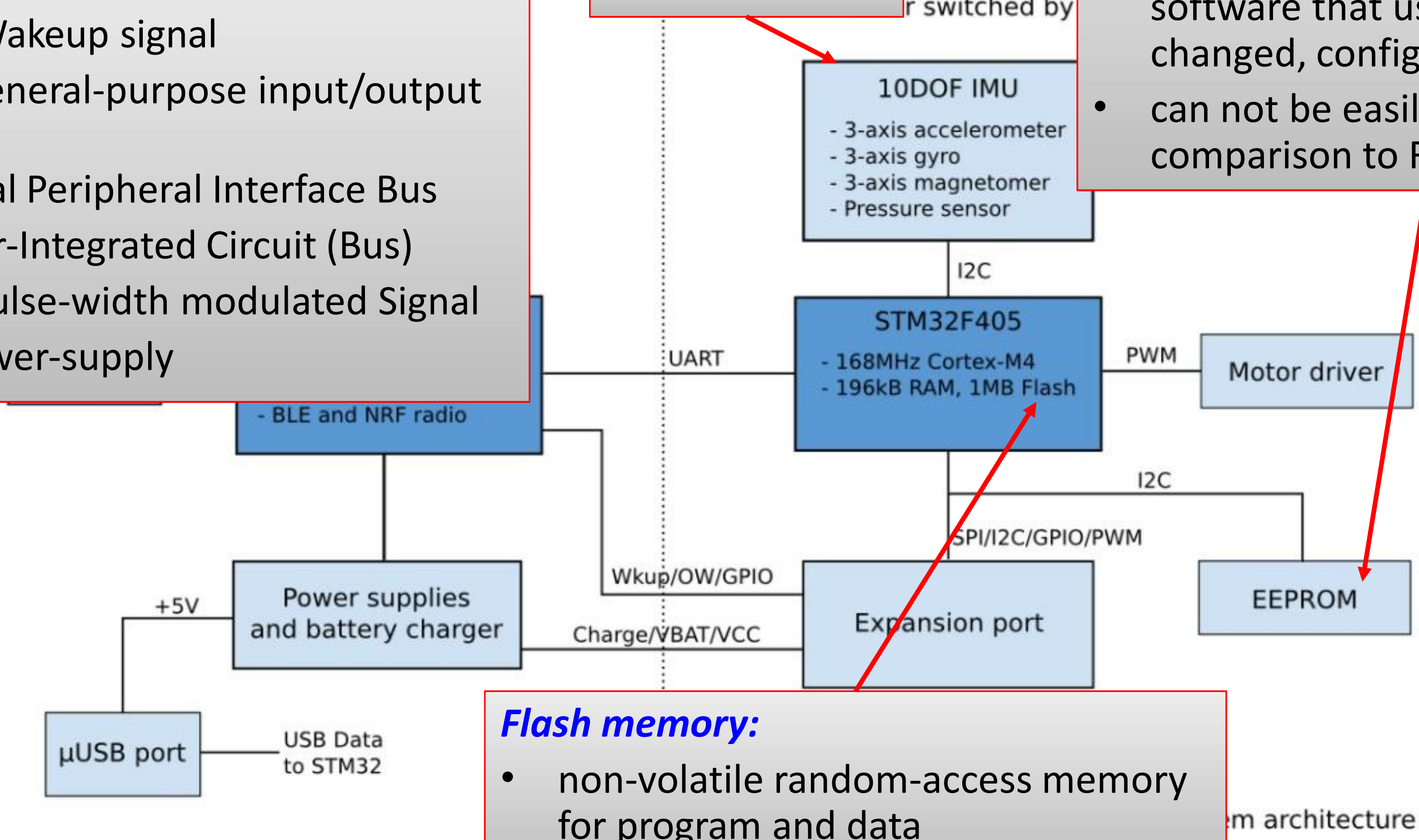
  exchange of data packets to and from interfaces (wireless, USB)

High-Level Block Diagram View

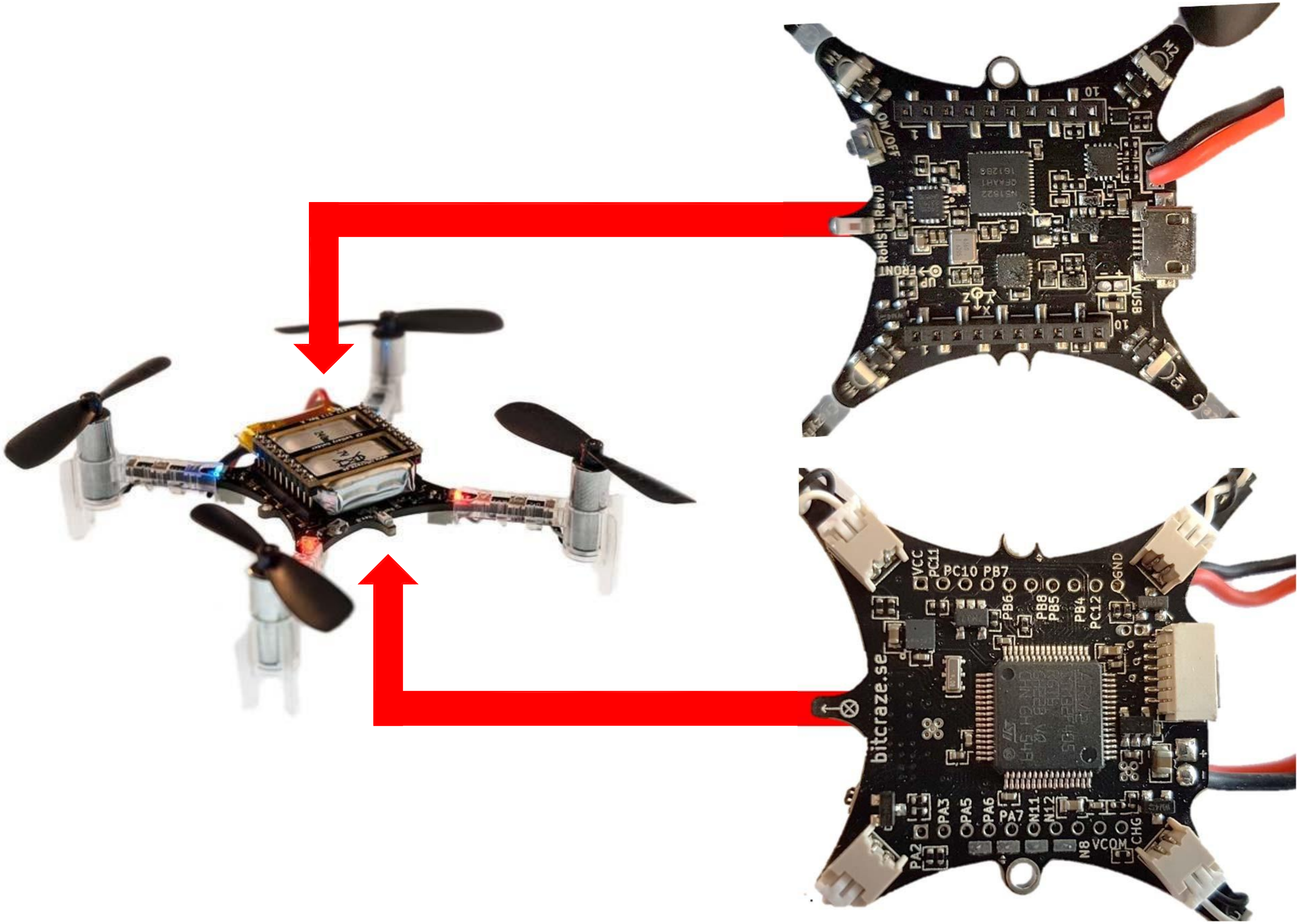
Acronyms:
- Wkup: Wakeup signal
- GPIO: General-purpose input/output signal
- SPI: Serial Peripheral Interface Bus
- I2C: Inter-Integrated Circuit (Bus)
- PWM: Pulse-width modulated Signal
- VCC: power-supply

sensor board

EEPROM:
- electrically erasable programmable read-only memory
- used for firmware (part of data and software that usually is not changed, configuration data)
- can not be easily overwritten in comparison to Flash

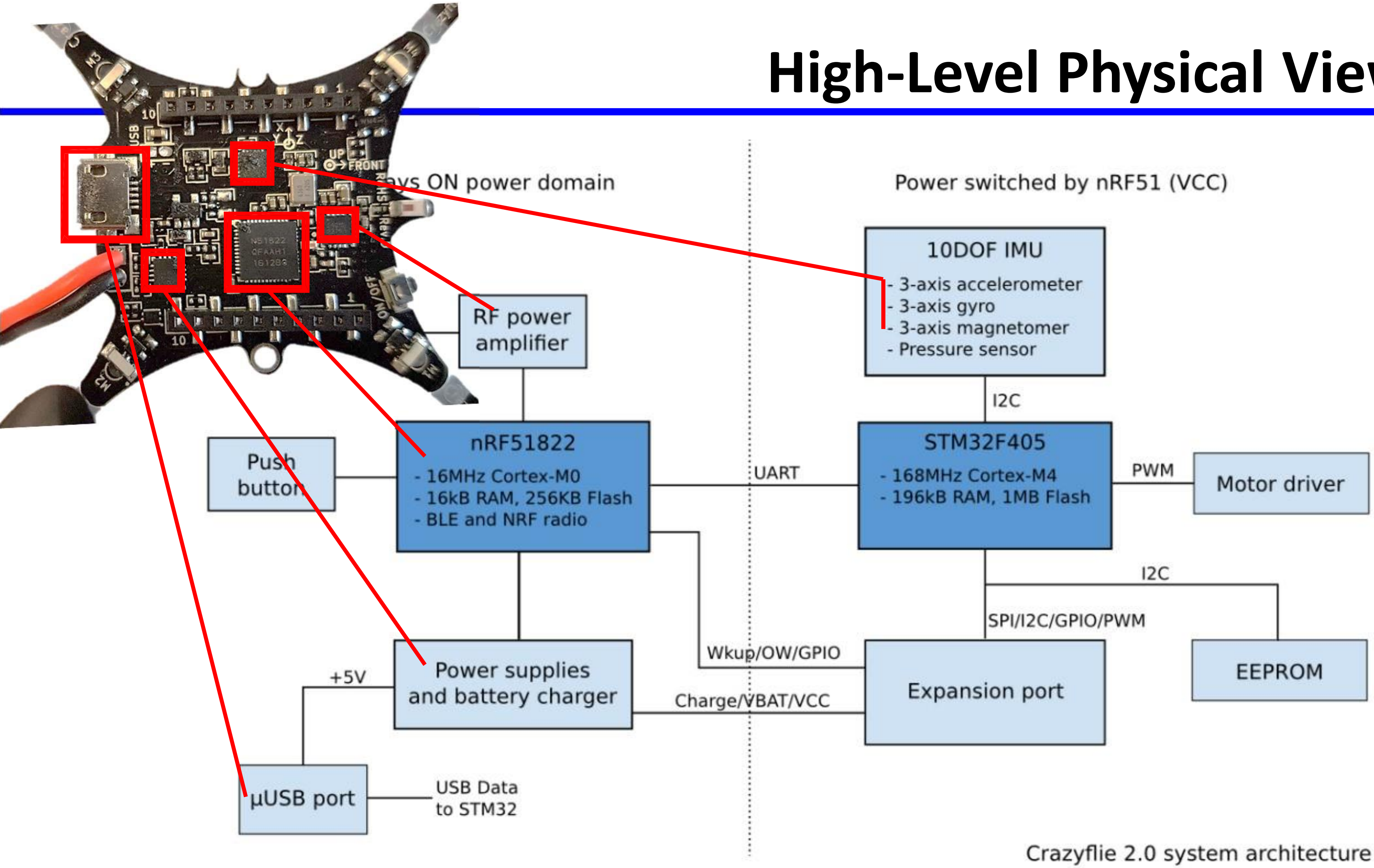


Flash memory:
- non-volatile random-access memory for program and data

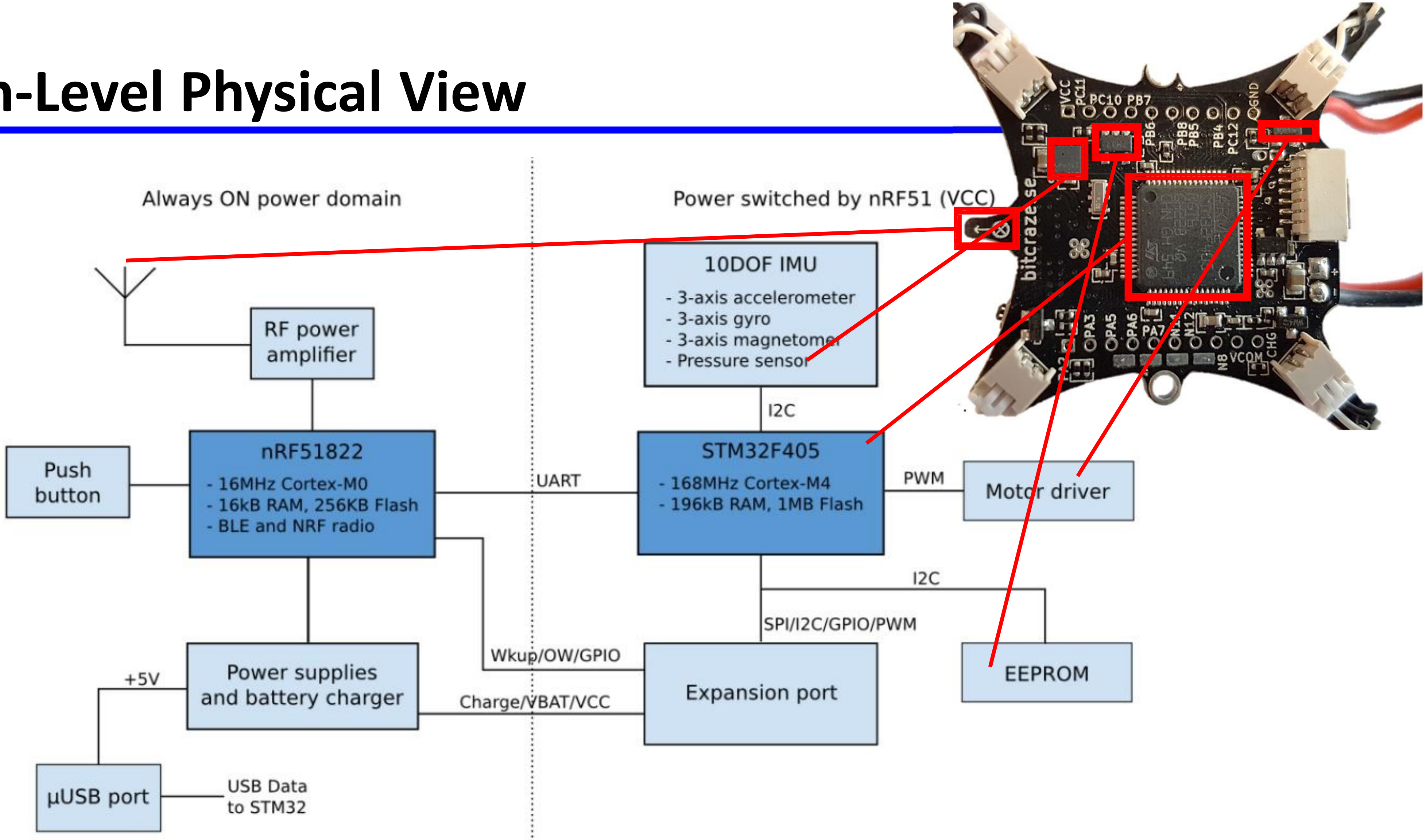


VCC
PC11
PC10 PB7
PB6
PB8
PB5
PB4
PC12
GND
bitcraze.se
PA2
PA3
PA5
PA6
PA7
N11
N12
N8
VCOM
CHG
VUSB
UP
FRONT

High-Level Physical View



High-Level Physical View

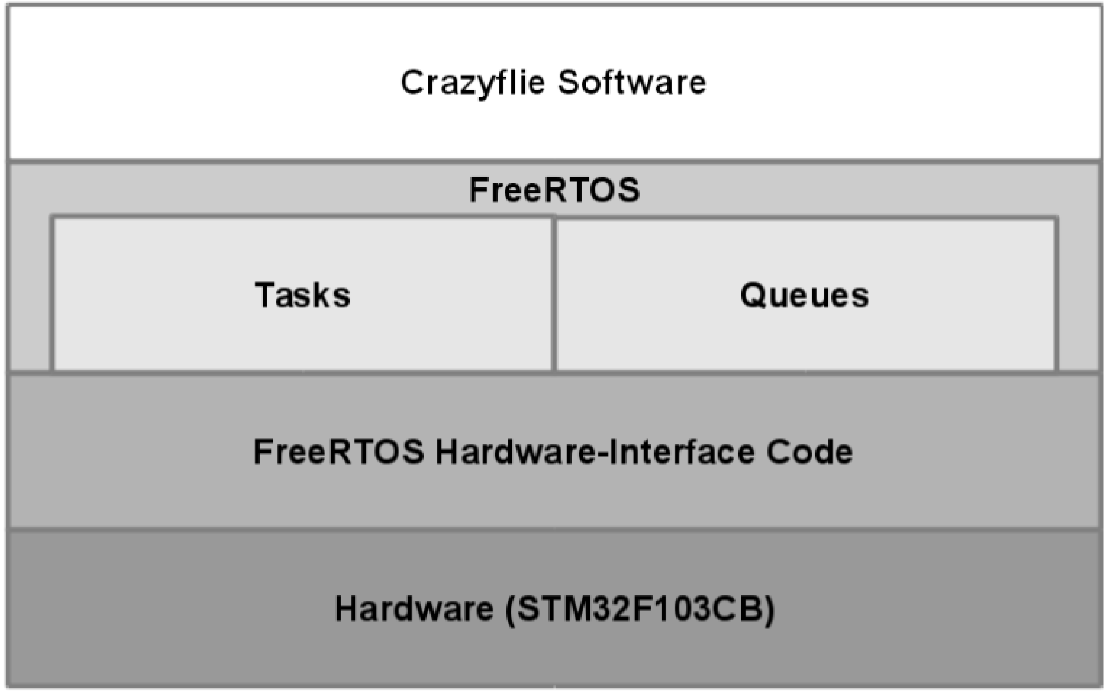


Crazyflie 2.0 system architecture

Components and Requirements by Example
- Processing Elements -

High-Level Software View

- The software is built on top of a *real-time operating system* “FreeRTOS”.
- We will use the same operating system in the Lab.



High-Level Software View

The *software architecture* supports

- *real-time tasks* for motor control (gathering sensor values and pilot commands, sensor fusion, automatic control, driving motors using PWM (pulse width modulation)

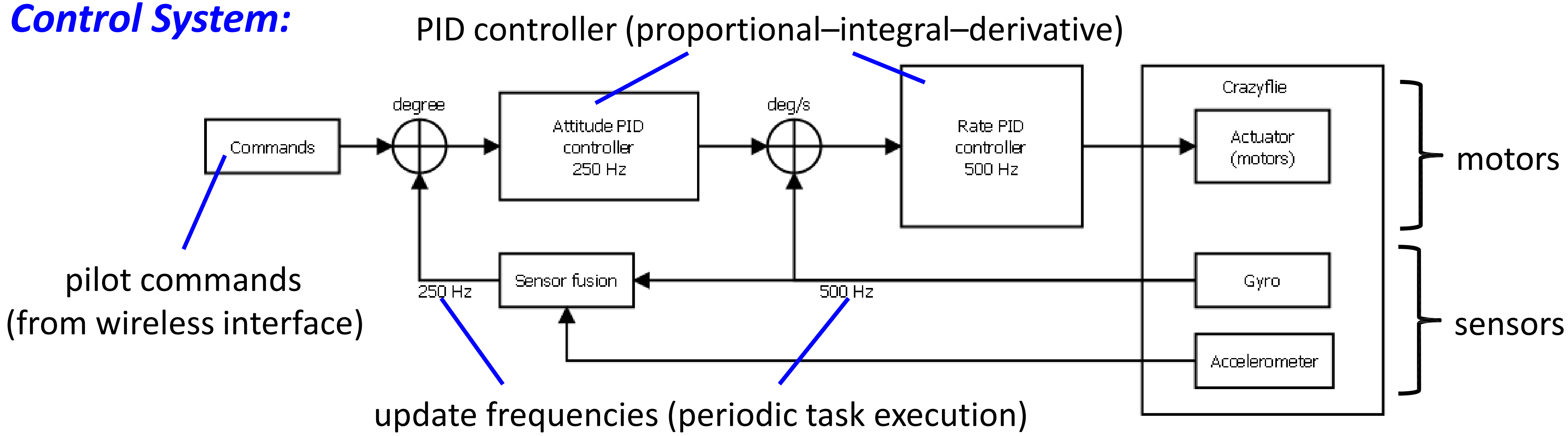
- *non-real-time tasks* maintenance and test, handling external events, pilot commands

High-Level Software View

The *software architecture* supports

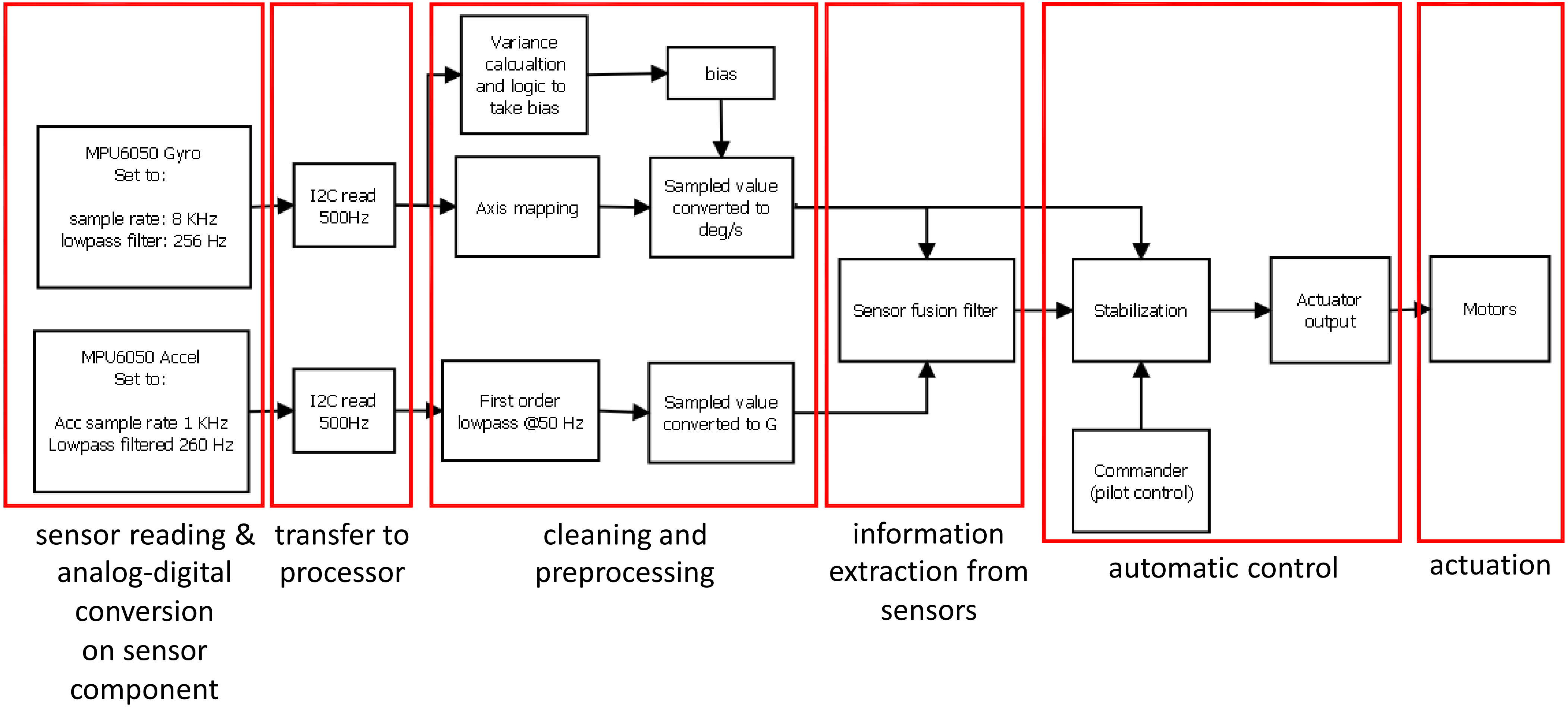
- *real-time tasks* for motor control (gathering sensor values and pilot commands, sensor fusion, automatic control, driving motors using PWM (pulse width modulation).
- *non-real-time tasks* (maintenance and test, handling external events, pilot commands, …).

Control System:



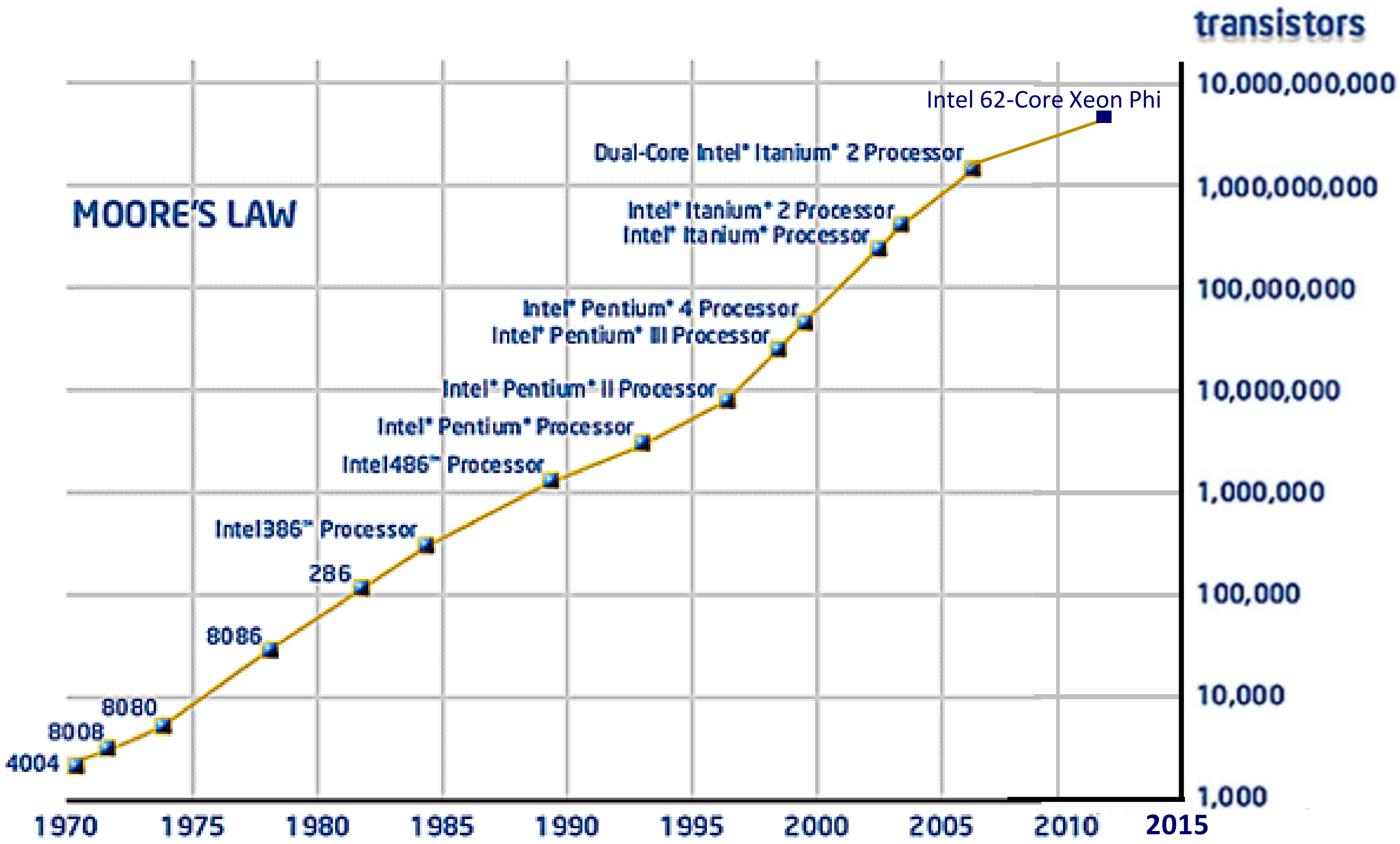
High-Level Software View

Block diagram of the stabilization system:

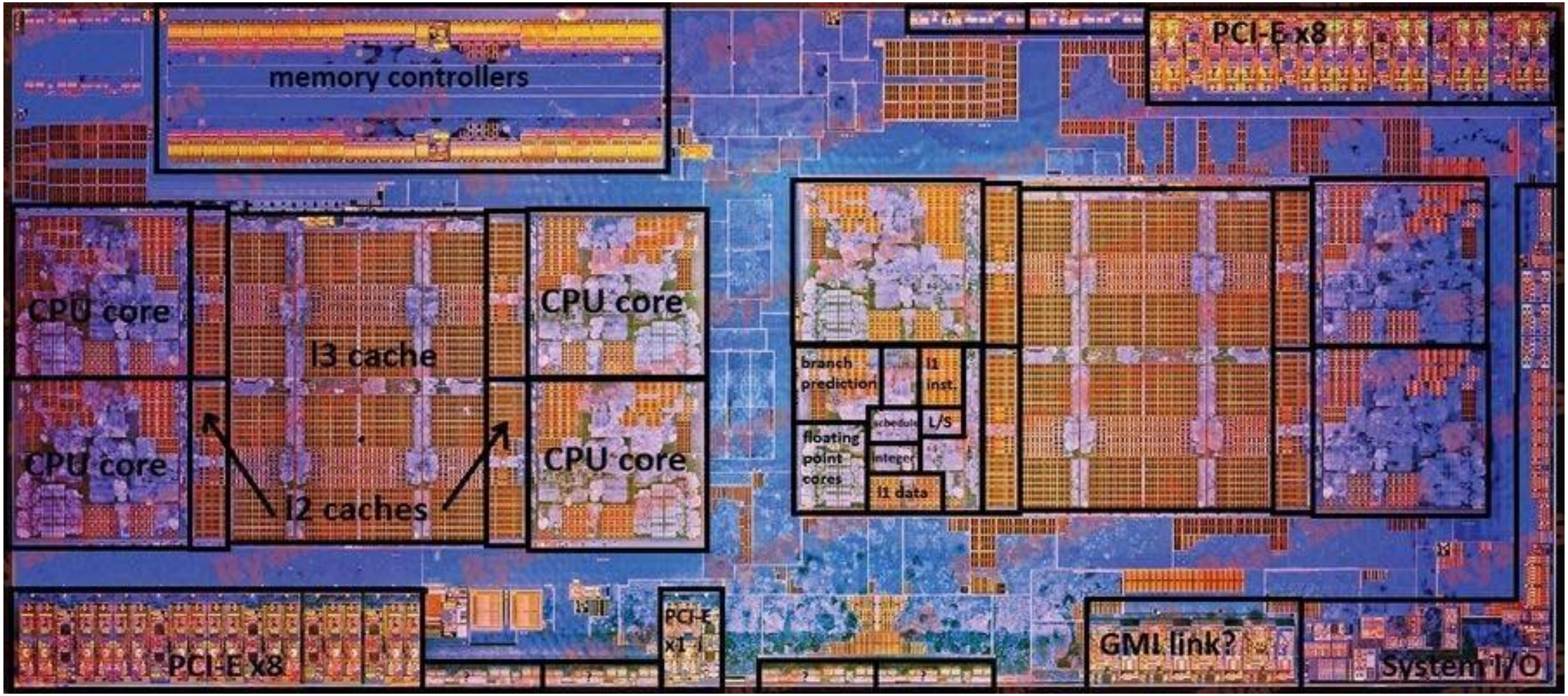


What can you do to increase performance?

From Computer Engineering 1



From Computer Engineering 1:



AMD multicore RYZEN

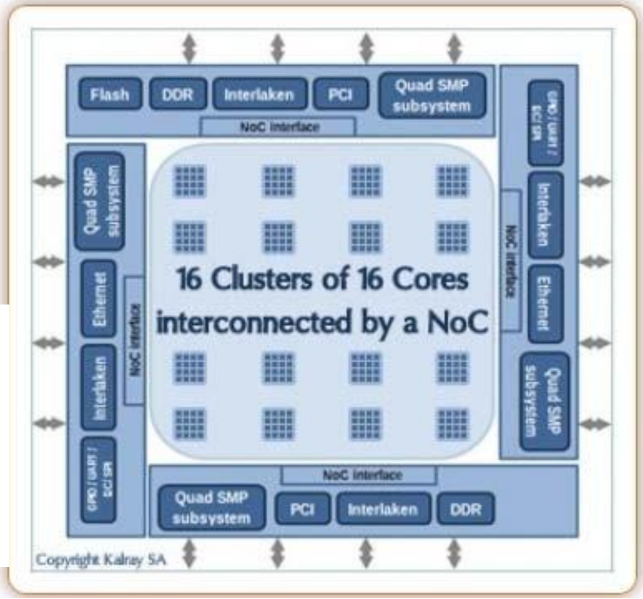
What can you do to decrease power consumption?

Embedded Multicore Example

Recent developments:

- Specialize multicore processors towards real-time processing and low power consumption
- Target domains:

Core Generation	Number of Processing Cores	GFLOPS/W	GOPS/W
Andey	256	25	75
Bostan (2014)	256	50	80
Coolidge (2015)	64/256/1024	75	115

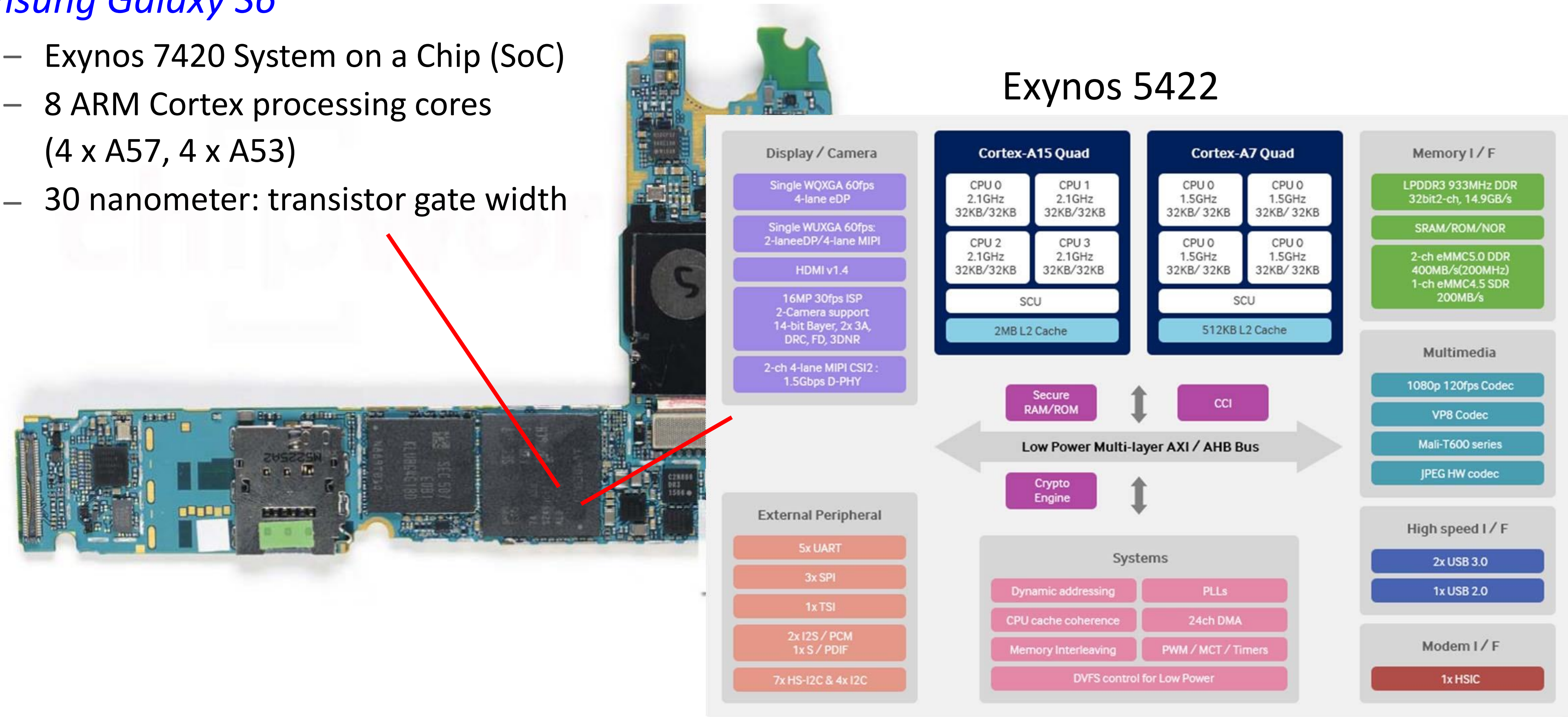


Why does higher parallelism help in reducing power?

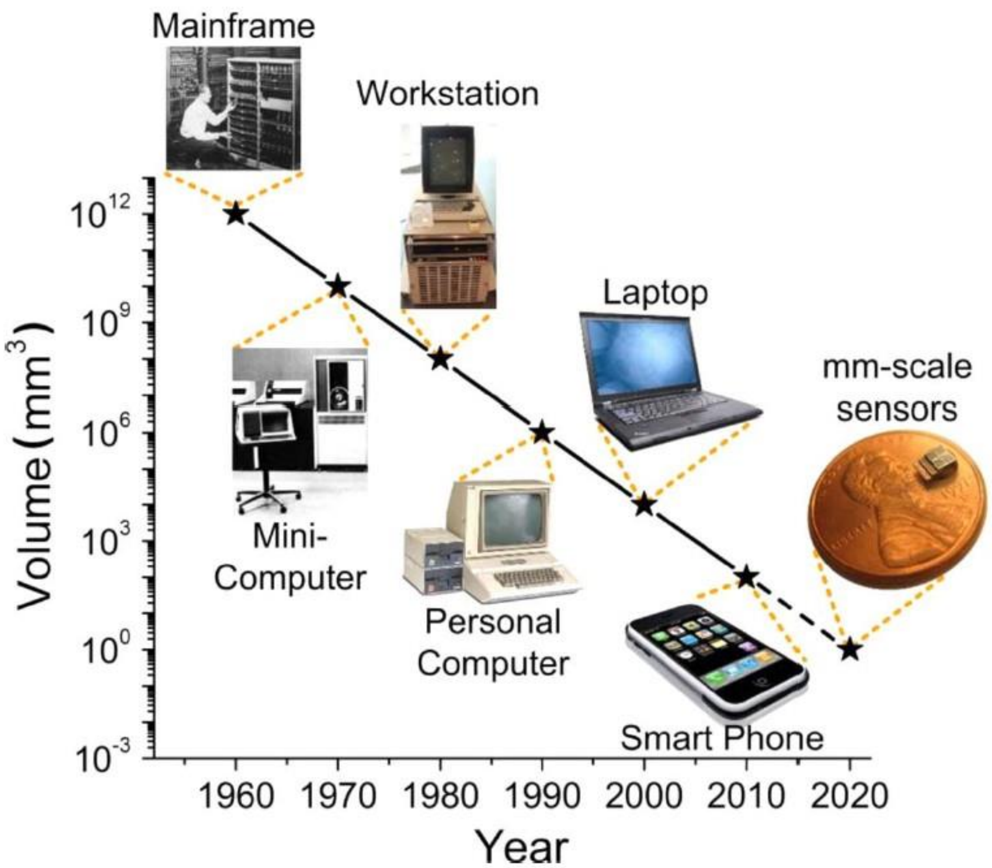
System-on-Chip

Samsung Galaxy S6

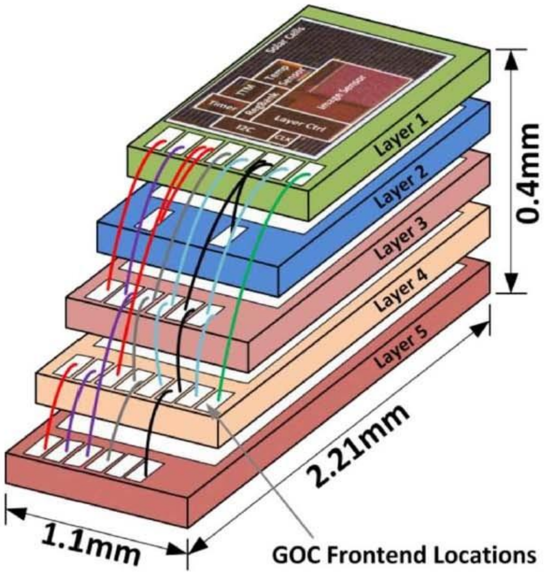
- Exynos 7420 System on a Chip (SoC)
- 8 ARM Cortex processing cores
  (4 x A57, 4 x A53)
- 30 nanometer: transistor gate width

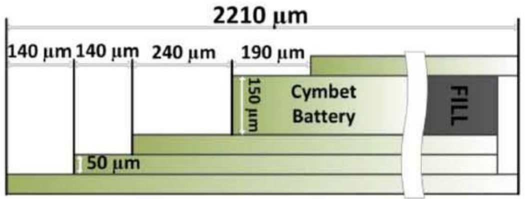


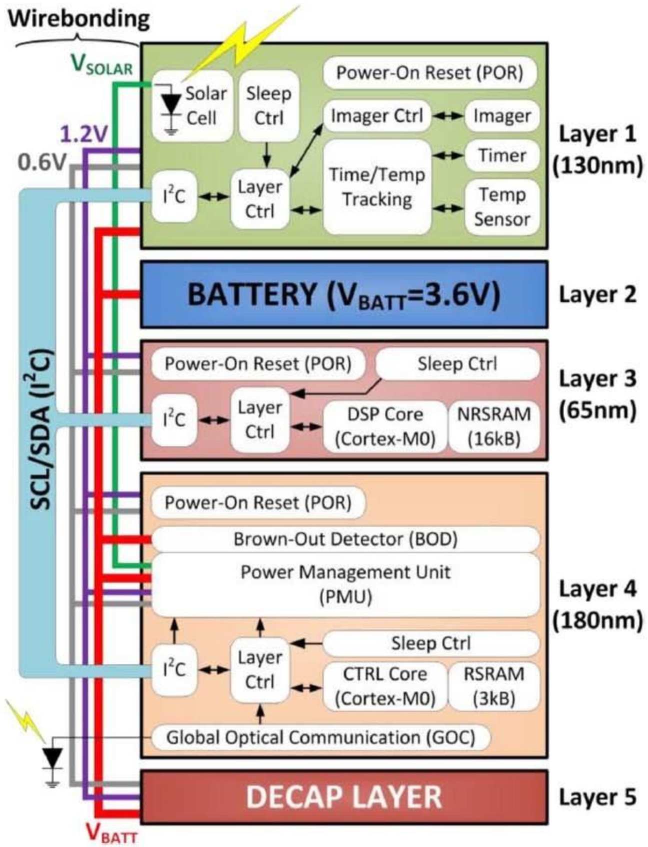
Zero Power Systems and Sensors

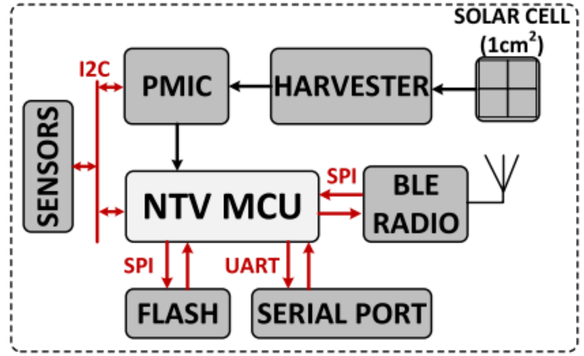


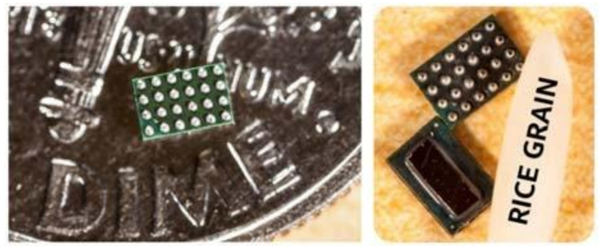
Zero Power Systems and Sensors











Trends …

- *Embedded systems are communicating with each other,* with servers or with the cloud. Communication is increasingly wireless.

- *Higher degree of integration* on a single chip or integrated components:
  - Memory + processor + I/O-units + (wireless) communication.
  - Use of networks-on-chip for communication between units.
  - Use of homogeneous or heterogeneous multiprocessor systems on a chip (MPSoC).
  - Use of integrated microsystems that contain energy harvesting, energy storage, sensing, processing and communication ("zero power systems").
  - The complexity and amount of software is increasing.

- *Low power and energy constraints* (portable or unattended devices) are increasingly important, as well as temperature constraints (overheating).
- There is increasing interest in *energy harvesting* to achieve long term autonomous operation.