Design and development of embedded systems for the Internet of Things (IoT)

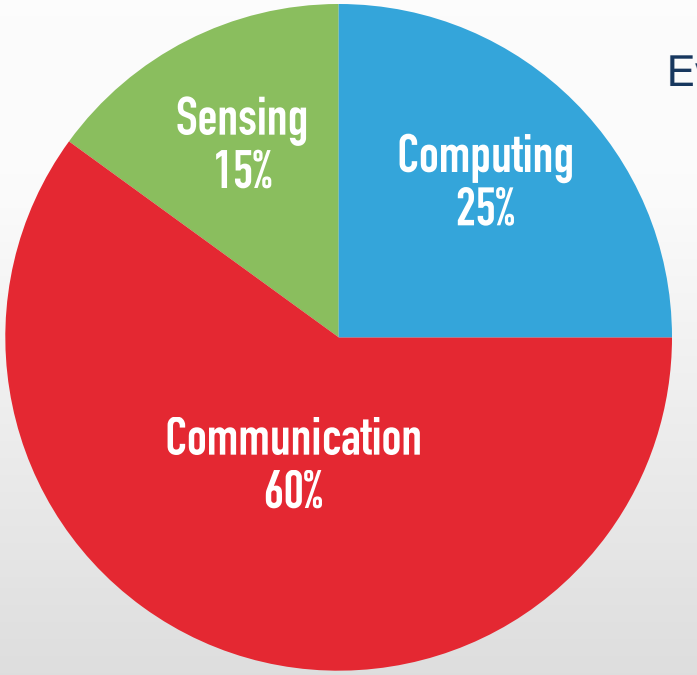
Fabio Angeletti
Fabrizio Gattuso

SAPIENZA
UNIVERSITÀ DI ROMA

W • SENSE
INTEGRATED CABLELESS SOLUTIONS

Node energy consumption

The batteries are limited and usually they can't support long term tasks (months, sometime years). For this reason it is important to save energy in every node component.



Sensing system

Every sensors and actuators connected to the nodes plus the software handler to retrieve the data

Computing system

Part of this system are the core functions: memory, micro-controller and operating system

Communication system

The network operations in order to communicate with the other nodes (MAC, ROUTING, SYNC)

Sensing

Sensors

Sensor Type	Power Consumption
Temperature - Humidity	0.5mW - 5mW
Acceleration	3mW
Pressure	10mW - 15mW
Image	150mW
Gas	500mW - 800mW

How to sample

We can't sample a sensor whenever we want (especially the most expensive ones).
It's important to find the right tradeoff between **accuracy** and **energy consumption**.

IS IT BETTER TO SAMPLE EVERY SECOND (SHORTER, AND MORE ACCURATE) OR EVERY MINUTE (LONGER, LESS ACCURATE)?

Sensing (2)

USE THE INTERRUPTS INSTEAD OF THE POLLING SYSTEM

(when you can)

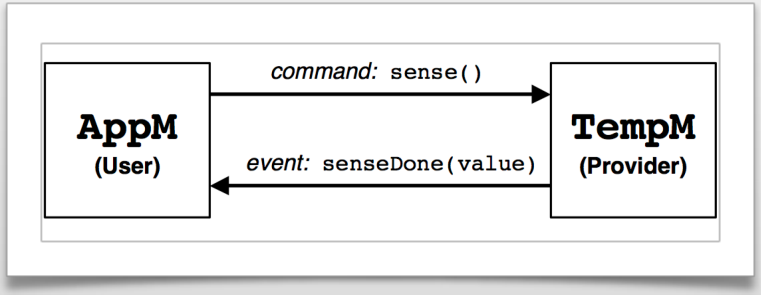
Computing

During the computing phase, the energy task is handled by the **hardware,** that is designed and developed

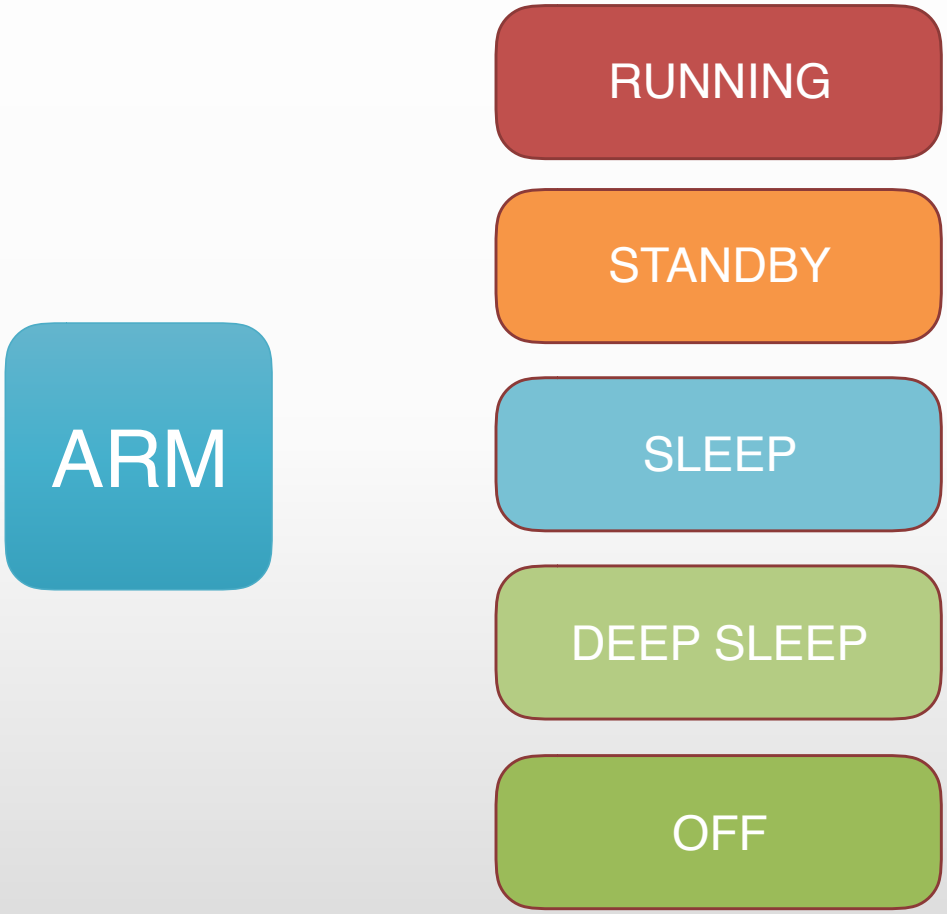
for these requirement, and by the **operating systems**.

An interesting example is how **TinyOS** and the programming language **NesC** force the developer to use better design models studied for the **Wireless Sensor Networks** and the **embedded systems** in general. For example:

- Using NesC it's not possibile to allocate **dynamic memory** and **time-consuming loop cycle** are unsuggested.
- The OS uses the event paradigm and the **Split-Phase** patter to design software based on the interrupt logic.



ARM Cortex Power States



Computing - FreeRTOS

Low Power on FreeRTOS is supported by two strategies:

- Idle Hook function
- Tickless idle mode

Not all functionalities are supported by all the boards (hardware)

FreeRTOS - Idle Hook Function

The idle task can optionally call a task defined as the idle hook.

The **idle hook** runs at:

- **very lowest priority**, so such an idle hook function will only get executed when there are no tasks of higher priority that are able to run and
- **it is an ideal place to put the processor into a low power state** (hardware feature).

You have to define ***configUSE_IDLE_HOOK = 1*** within **FreeRTOSConfig.h**.

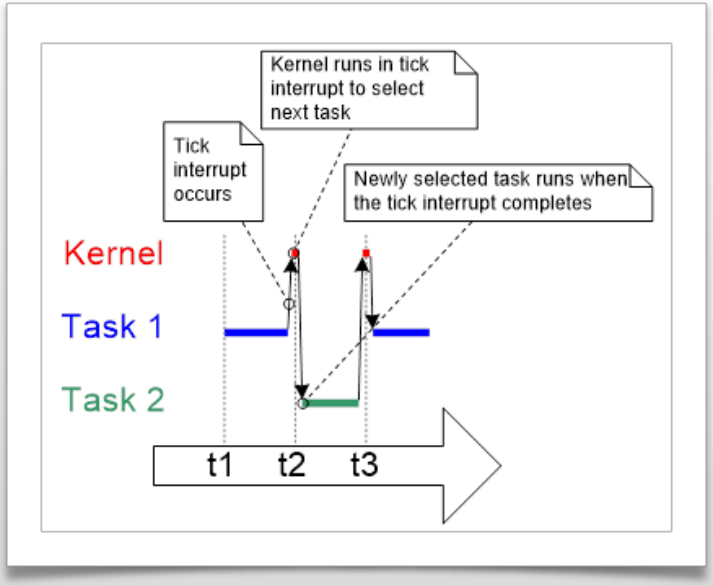
When this is set the application must provide the hook function with the following prototype:

void vApplicationIdleHook(void);

FreeRTOS - Tickless Idle Mode

The power saving that can be achieved by the previous method is limited by the necessity to periodically exit and then re-enter the low power state to process tick interrupts.

If the frequency of the tick interrupt is too high, the energy and time consumed entering and then exiting a low power state is more than or equal to the energy saved.



FreeRTOS - Tickless Idle Mode (2)

The tickless idle mode stops the periodic tick interrupt during idle periods (no application tasks are able to execute), then makes a correcting adjustment to the RTOS tick count value when the tick interrupt is restarted.

Stopping the tick interrupt **allows the micro-controller to remain in a deep power saving state until either an interrupt occurs**, or it is time for the RTOS kernel to transition a task into the Ready state.

Built in tickless idle functionality is enabled by defining ***configUSE_TICKLESS_IDLE = 1*** in **FreeRTOSConfig.h**

The main problem is that it may be waken up only by interrupts!

FreeRTOS - Tickless Idle Mode (3)

If you want to wake up at a specific time you can set ***configUSE_TICKLESS_IDLE = 2*** in **FreeRTOSConfig.h.**

To use this functionality you have to use an **external clock** (low power timer) or you can use the **Real Time Clock** usually onboard (you can't use anymore the functionalities of this hardware).

The application must provide their own implementation by defining ***portSUPPRESS_TICKS_AND_SLEEP()*** in **FreeRTOSConfig.h.**

FreeRTOS - Low power mode

You have to use:

1. **The idle hook function** when you have a light use of the controller and you don't need a high level of power saving

2. **The tickless idle mode level 1** when you are able to wake up your device by an interrupt

3. **The tickless idle mode level 2** when you need an high level of energy power saving and you want to wake up at a specific time

Radio communication

Transmit and receive states

The transceiver is transmitting or receiving a packet.

Idle State

The transceiver is ready to receive but it is not active.

Less energy consuming than TX/RX state.

Radio State	Power Consumption
Sleep	1.1 µA
Idle	4.1 mA
RX	14.5 mA
TX	27.7 mA

Magonode consumption

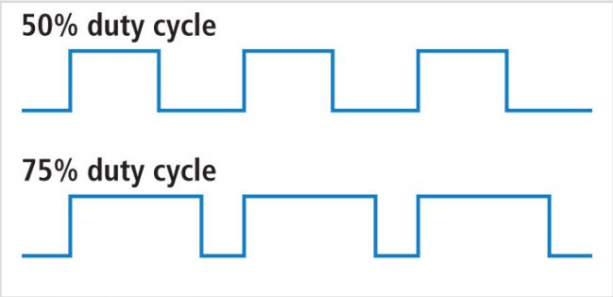
Sleep State

In this state the transceiver can't receive any packet because it is in sleep mode.

A **wake-up time** is required to turn on the radio and make it ready. The energy consume is lesser than the other states.

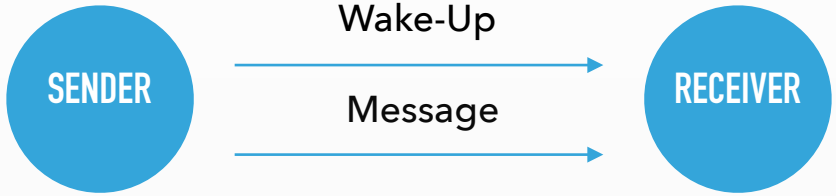
Radio communication (2)

Duty Cycle



- ✓ Ready to use with the standard radio
- ✓ State of the art solution with high-level of power energy saving
- ✓ Implemented in most OS
- ✓ Introduces latency in the network
- ✓ Node are not synchronized: long face of idle listening needed

Wake-Up Radio



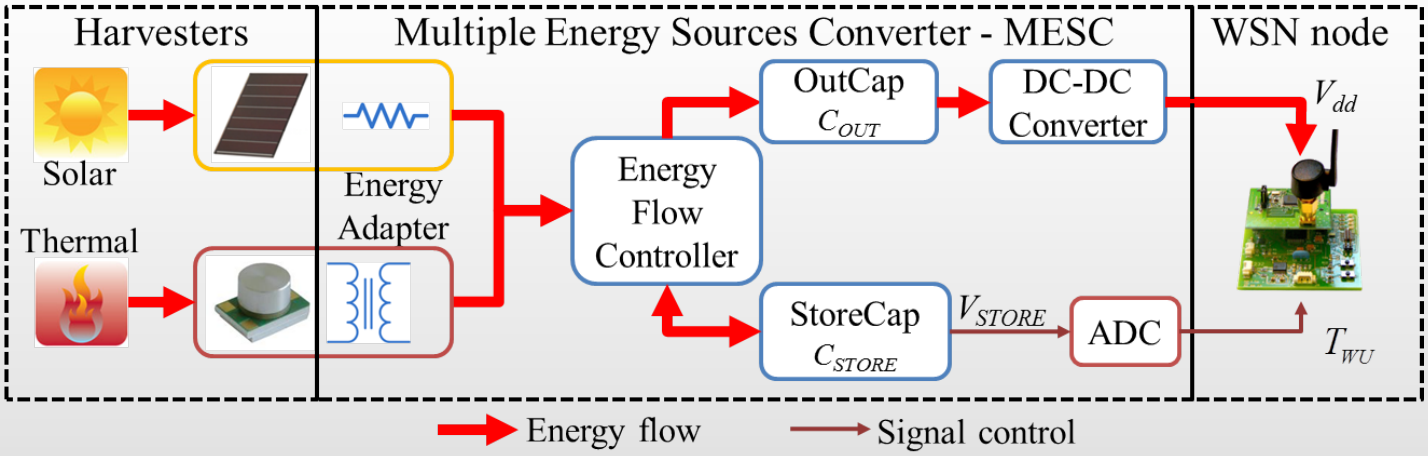
- ✓ No idle listening
- ✓ Introduces the semantic addressing concept
- ✓ Permits a long network lifetime (until months/years)
- ✓ New hardware
- ✓ The topology inducted by the WUR is shorter than the original one

Energy Harvesting

"**Energy harvesting** is the process by which energy is derived from **green external sources**, captured and stored for small devices, like those used in **wearable electronics** and **wireless sensor networks**."

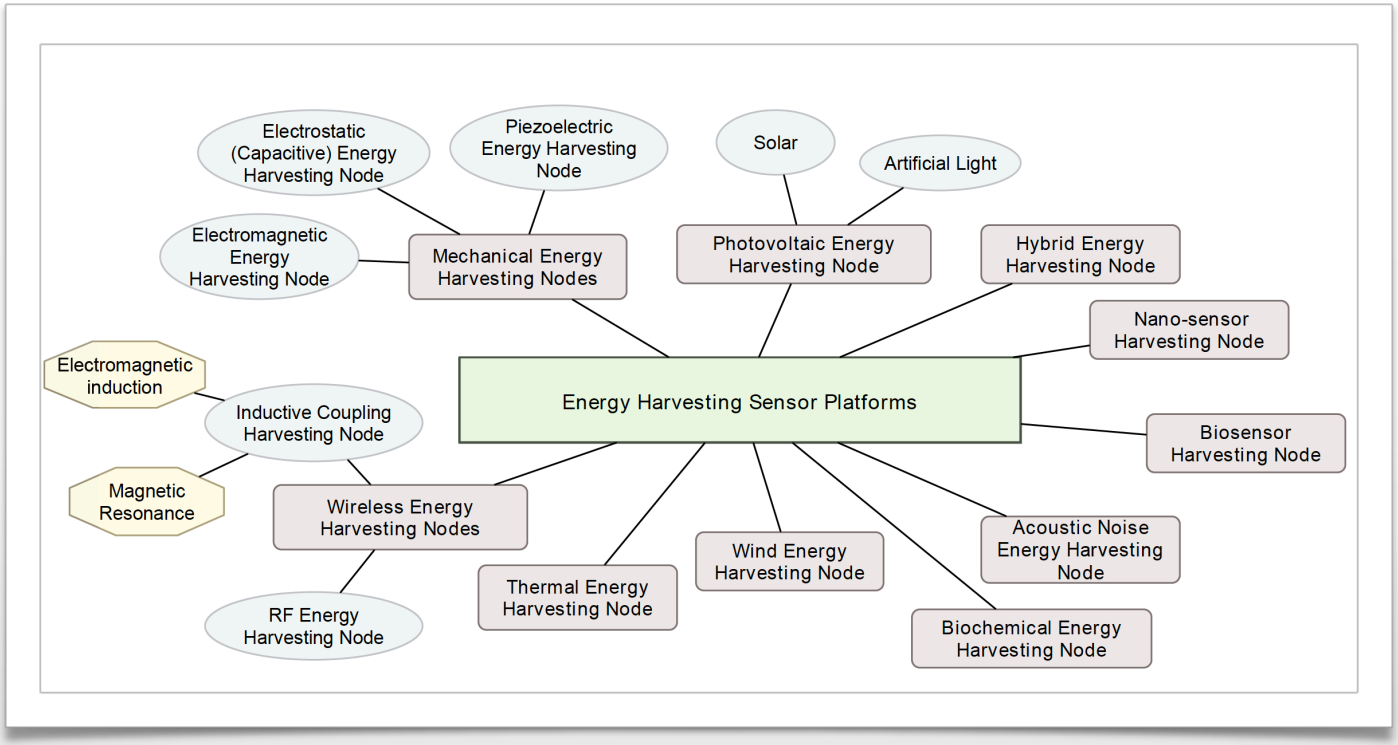
It's possibile to use one or more external sources to sustain the node life.

The energy is either stored in super capacitors and in secondary rechargeable batteries, or it is immediately used.



WHICH ARE THE BEST EXTERNAL SOURCES?

Energy Harvesting (2)



A galaxy of energy resources but

not all of them are able to sustain the node life time

External Sources

Sensor Type	Power Consumption
Photovoltaic	Outdoor: 15 mW/cm^2 Cloudy Outdoor: 0.15 mW/cm^2 Indoor: <10 μW/cm^2
Thermoelectric	30 μW/cm^2
Pyroelectric	8.64 μW/cm^2
Piezoelectric	250 μW/cm^3 - Inside the shoes: 330 μW/cm^3
Electromagnetic	Industrial: 306 μW/cm^3 - 800 μW/cm^3 Human: 1-4 μW/cm^3
Electrostatic	50-100 μW/cm^2
RF	GSM: 0.1 μW/cm^2 WiFi: 0.01 μW/cm^2
Wind	380 μW/cm^3
Acoustic Noise	100 dB: 0.96 μW/cm^3 75 dB: 0.003 μW/cm^3

It is important to choose the right hardware

Capturing the same energy in a indoor scenario with
an indoor optimized solar cell versus a standard solar cell

Indoor cell	Outdoor cell
3,401 mW	3,993 μW

External Sources (2)

Solar Energy

- ✓ One of the most used energy resource
- ✓ High-level of energy acquired
- ✓ Works in **outdoor** and **indoor** scenario
- ✗ The weather conditions are variable
- ✗ In indoor sometimes the energy captured is not enough to support a task

Wind Energy

- ✓ One of the most used energy resource
- ✓ Good level of energy acquired
- ✓ Not so variable in specific scenarios
- ✗ In some areas there is no wind most of the time
- ✗ It's not possibile to use in indoor scenario

Storage problems

Batteries

A battery capacity is assumed to decrease by the amount of energy required by an operation only when the operation is performed. Real batteries **suffer from self-discharge** and **can be recharged only in the order of 1000 cycles**.

Super-capacitors

Energy density is one order of magnitude lower than electrochemical battery, but they suffer from higher self-discharge. The **super-capacitor leakage** is strongly variable and depends on several factors, including the **capacitance value** of the super-capacitor, the **amount of energy stored**, the **operating temperature**, the **charge duration**.

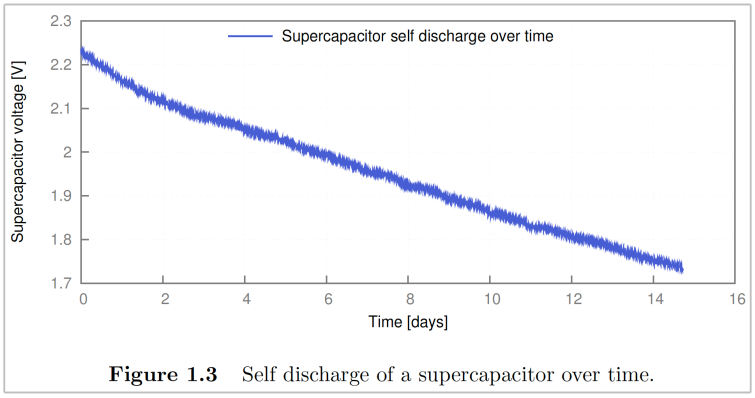


Figure 1.3 Self discharge of a supercapacitor over time.

Prediction models

The major problem when using these external resources is the **uncertainty of the availability**.

- **Can we harvest some energy?**
- **Is the amount of energy enough to support a task?**

For example the solar energy depends on the **weather conditions** and the same holds for the wind energy. It is also really **tricky to understand** the behavior of other sources like the **piezoelectric systems** or the **RF energy**.

Underground case study

220 m of tunnel with **six Telos B** equipped with **wind micro-turbines**, which collected air-flow data generated by passing trains for **33 days**.

The energy harvested from the micro-turbine was then stored in a super-capacitor.

- **Temperature**
- **Humidity**
- **Strain gauges**

Underground case study (2)



MEAN ENERGY HARVESTED BY EACH NODE PER TRAIN PASSAGE & NUMBER OF OPERATIONS THAT CAN BE PERFORMED WITH THE GATHERED ENERGY.

Node id	Mean energy harvested per train passage (mJ)	Rx [KB]	Tx [KB]	Flash read [KB]	Flash write [KB]	Humidity	Temperature	ADC reads	Strain measurements per day
7	5.09	2.25	3.32	1.88	1.23	49	16	802	1
4	47.80	21.14	31.12	17.77	11.56	463	158	7530	13
2	51.72	22.88	33.67	19.24	12.51	501	171	8148	14
5	75.68	33.47	49.27	28.15	18.31	734	250	11924	21
6	66.60	29.46	43.36	24.78	16.11	646	220	10493	18
3	132.95	58.80	86.56	49.45	32.17	1290	439	20946	36