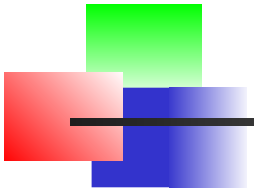


“IoT – Hardware aspects”

“Software does not exist ***per se***; it may be **instantiated statically** in the **memory** or **dynamically** during the execution on hardware **processors**”

“Real Men Have Fabs” Jerry Sanders, AMD founder

IoT – global picture

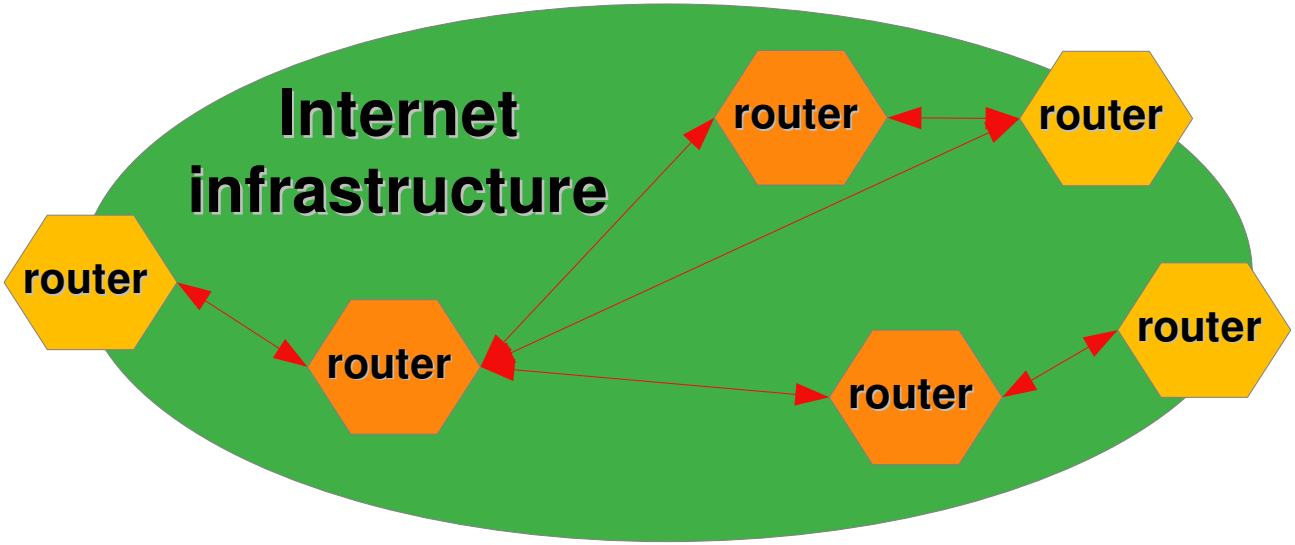
- IoT – Internet of Things
- Things
- Internet

Things: Embedded Software/Hardware

Internet: Communication means

Terminology, terminology, terminology , ..

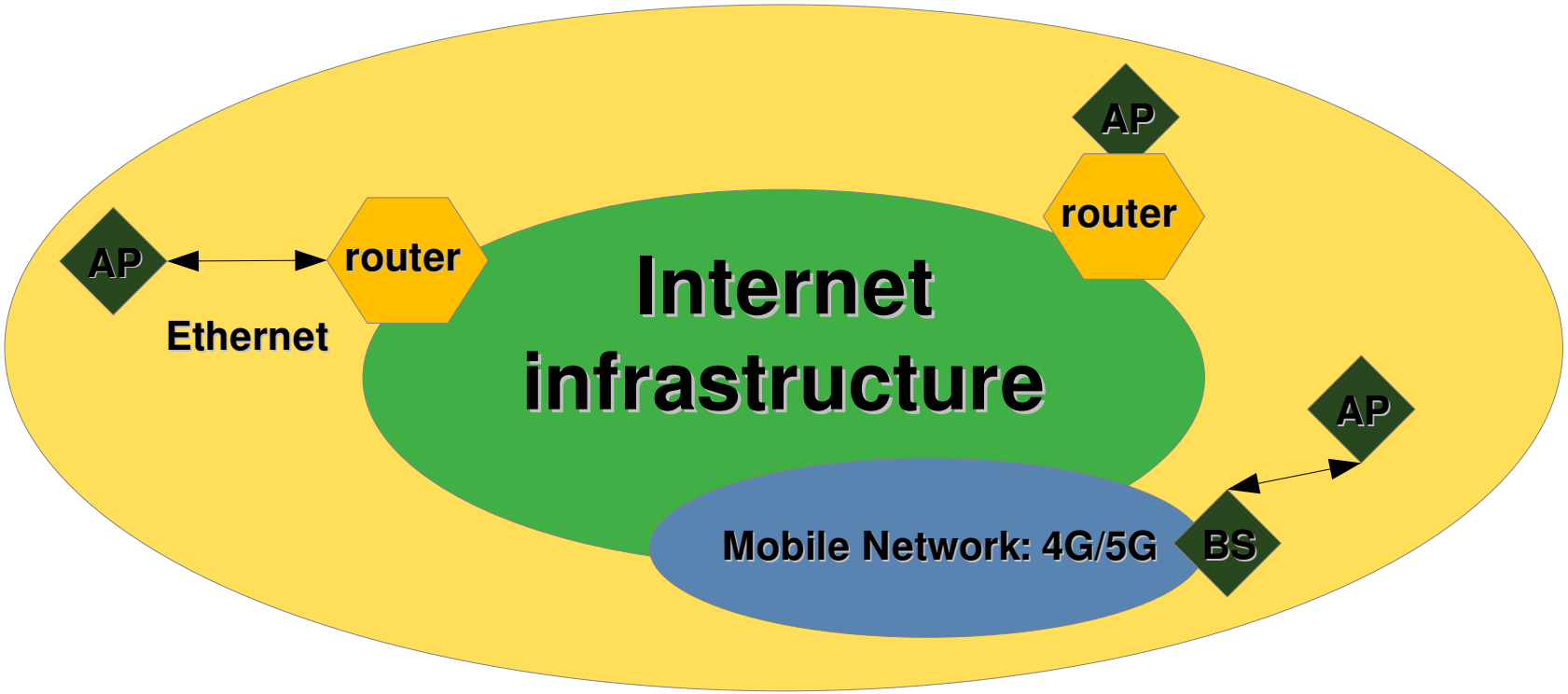
Internet Infrastructure



Router is internal device – IP packets: Packets Per Second, Packet Loss, ..

Long distance links – fiber : Bits Per Second – 10^6, 10^9, 10^{12}

Internet – Access Points

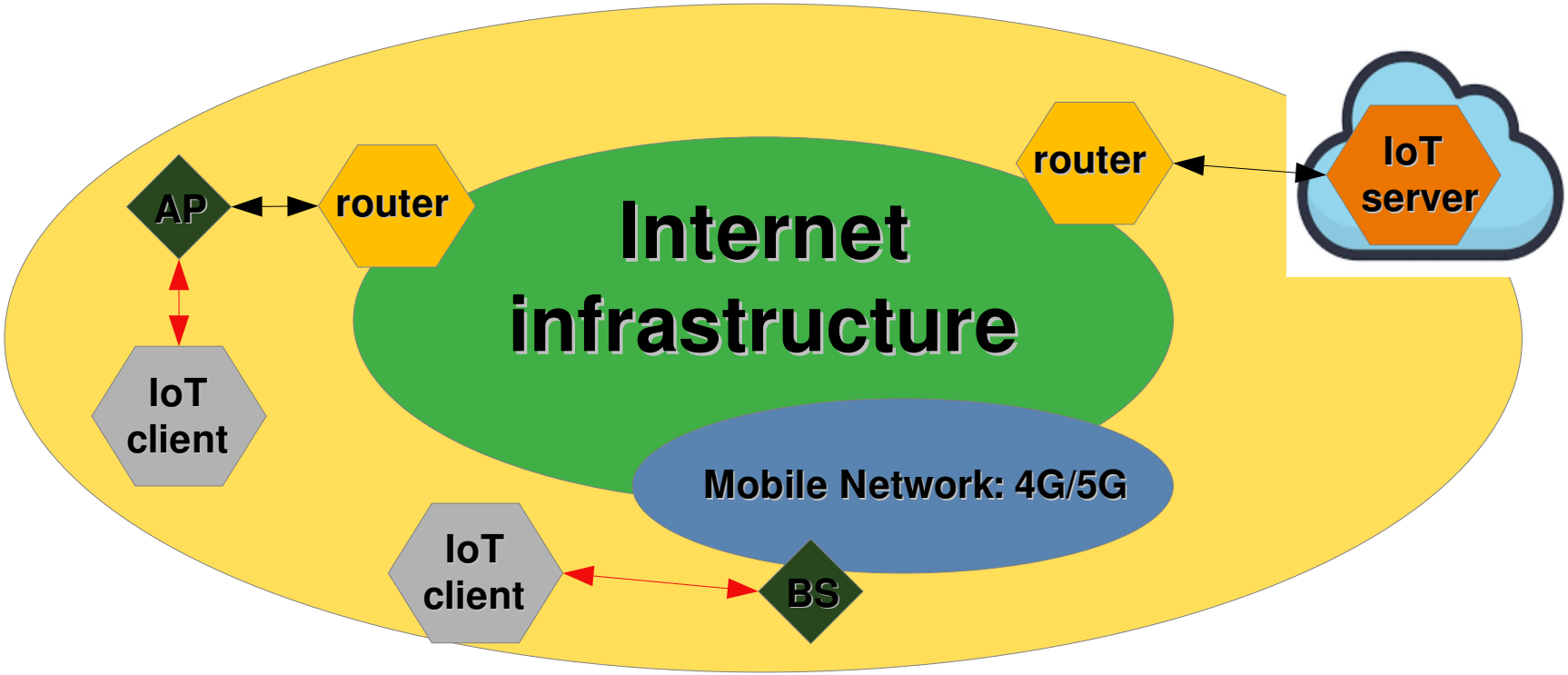


Ethernet – is local wired access to Internet for device, AP, switch, ..

AP – Access Point is a wireless entry (to Internet) for device (WiFi)

BS – Base Station is a wireless entry (to Internet) for device, AP

IoT – Clients and Servers

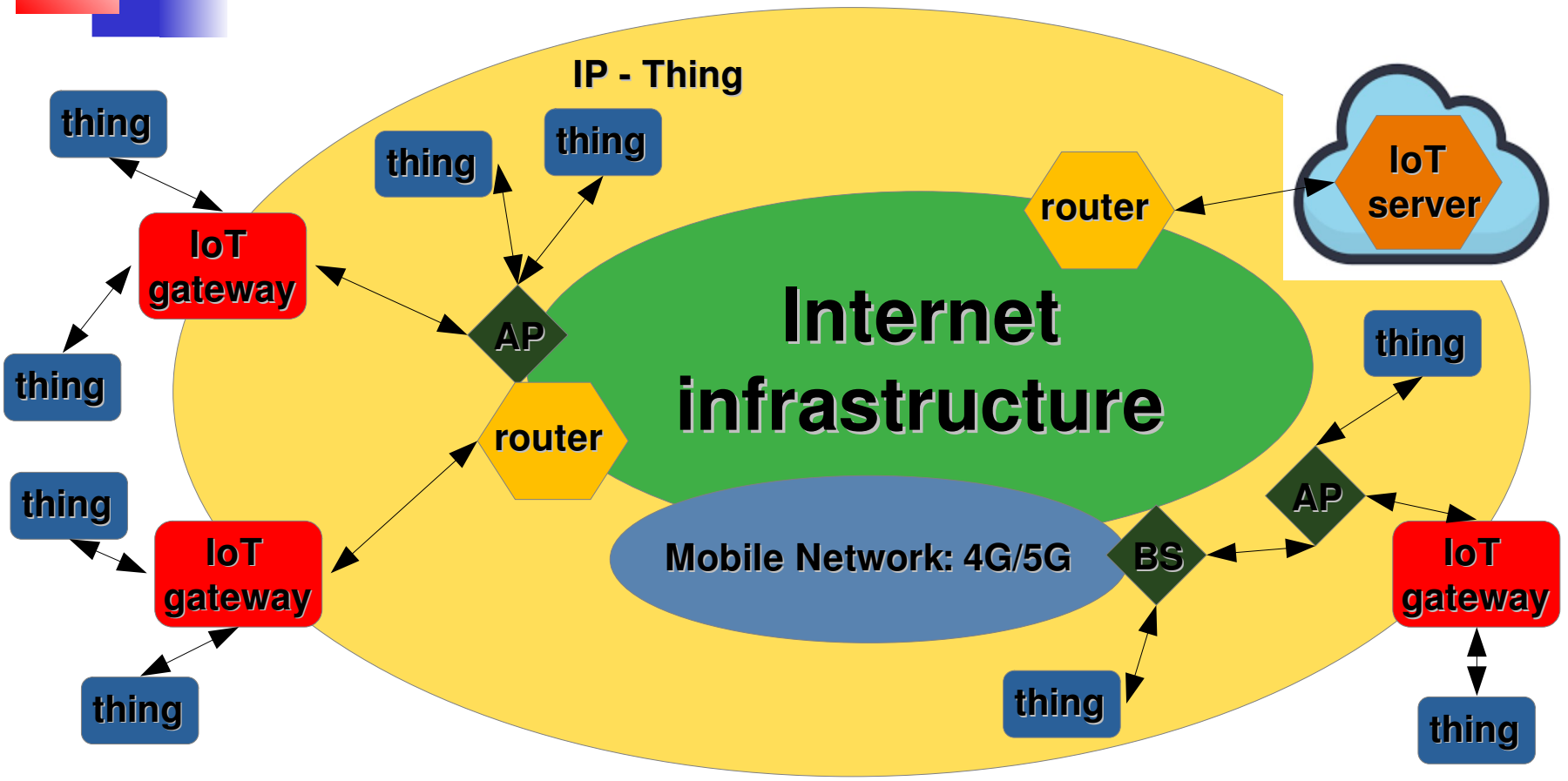


IoT Server and IoT Client are external devices

Client : PC, laptop, tablet, smartphone, IoT device, AIoT device, ..

Server : PC, SBC, HPC with data center, HPC with AI center, (Cloud: UP,DOWN)

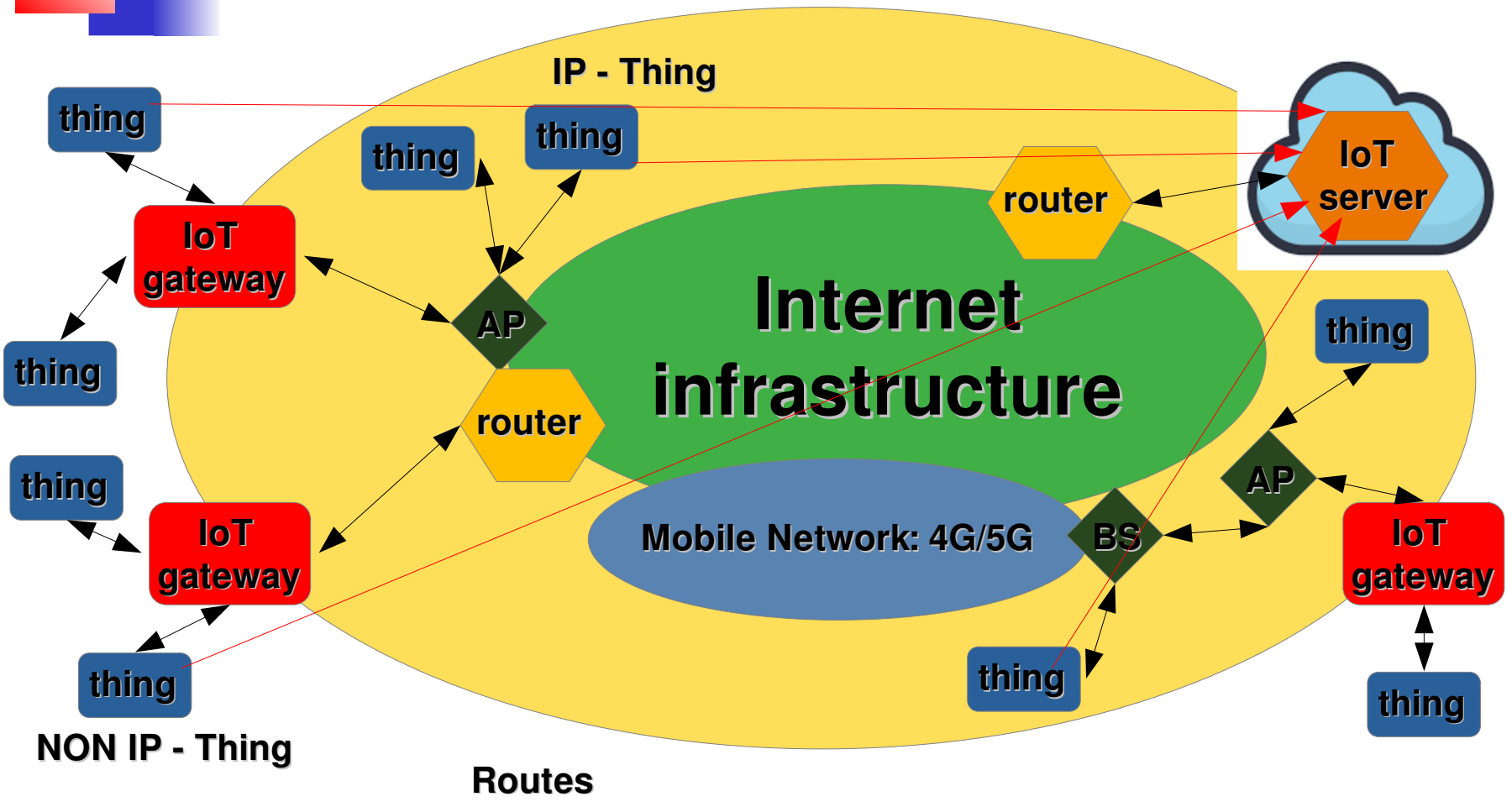
IP Things and Non-IP Things



Thing is a Terminal device

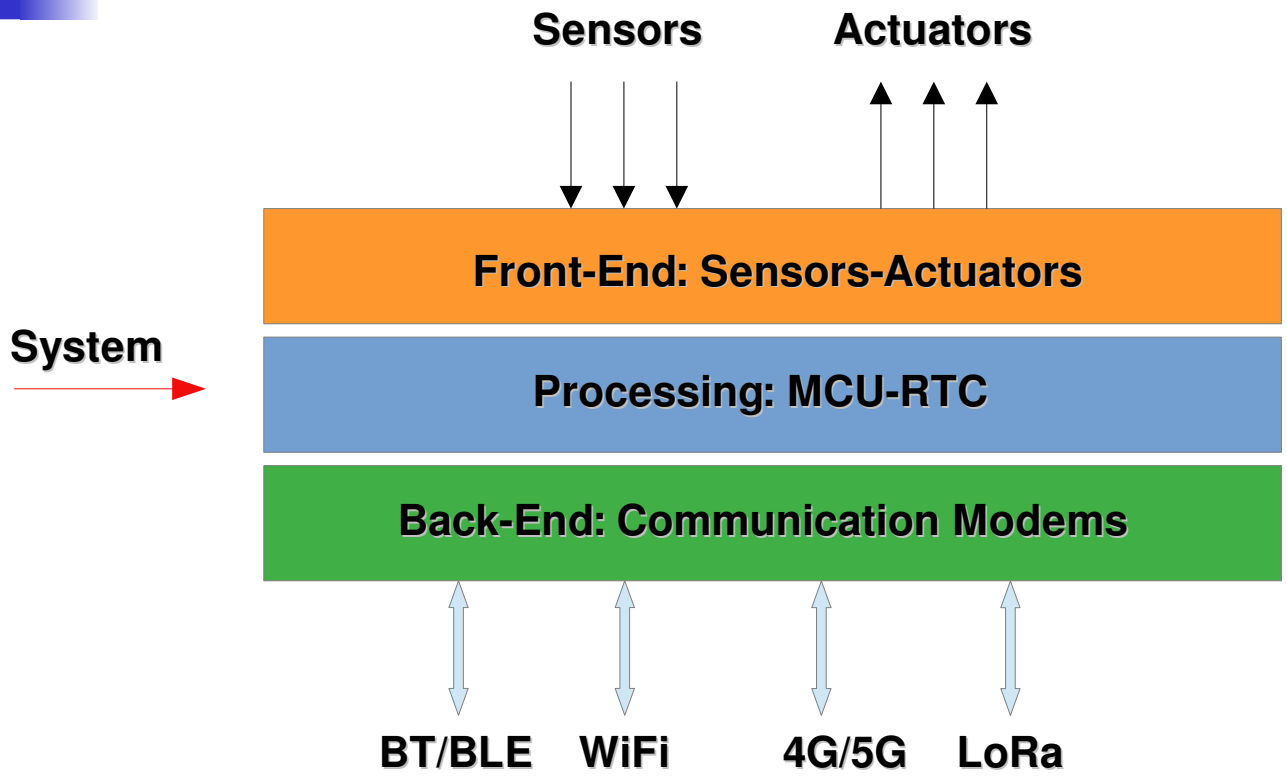
IoT gateway is an intermediate device

Routes and Examples - Discussion



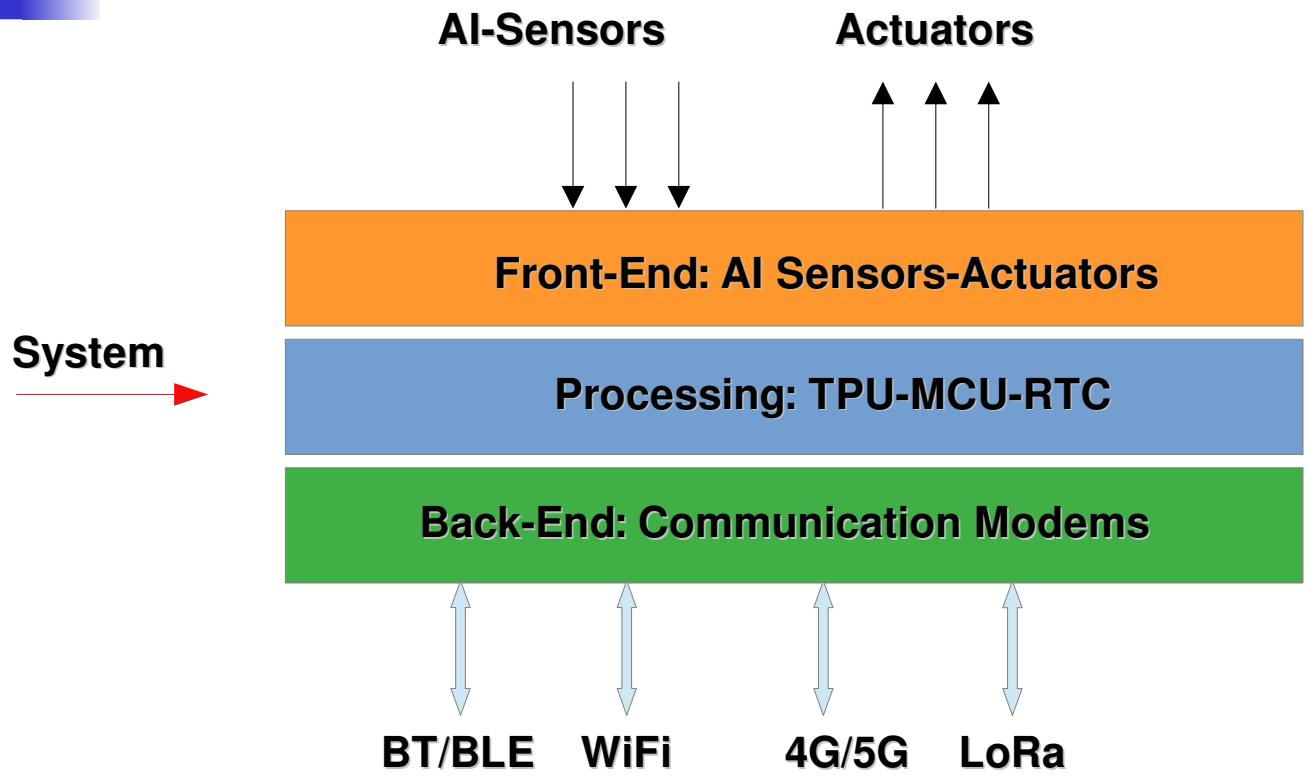
Examples, examples, examples - Discussion

Simple and Intelligent Things



Simple Thing – basic processing of physical data and display and activation of physical devices

Simple and Intelligent Things



Intelligent Thing – AI processing of physical data and display and activation of physical devices

Real example – System on Chip

ESP32 is a series of **low-cost**, **low-power system on a chip - SoC** micro-controllers with **integrated Wi-Fi** and dual-mode **Bluetooth**.
The ESP32 series S (S1,S2,S3 employs either a **Tensilica Xtensa LX6/LX7** microprocessor in both dual-core and single-core variations, The ESP32 series C (C3,C6, ..) are based **single-core RISC-V** microprocessor.
All ESP32 SoCs and include built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules.

ESP32 SoCs have been created and developed by **Espressif Systems**, a **Shanghai**-based Chinese company, and is manufactured by **TSMC** using their **40 nm** process. There are

low-cost	**Wi-Fi**	**Xtensa LX6/LX7**
low-power	**BT-BLE**	**RISC-V**

ESP32 (S1) – System on Chip

Low-cost < $5

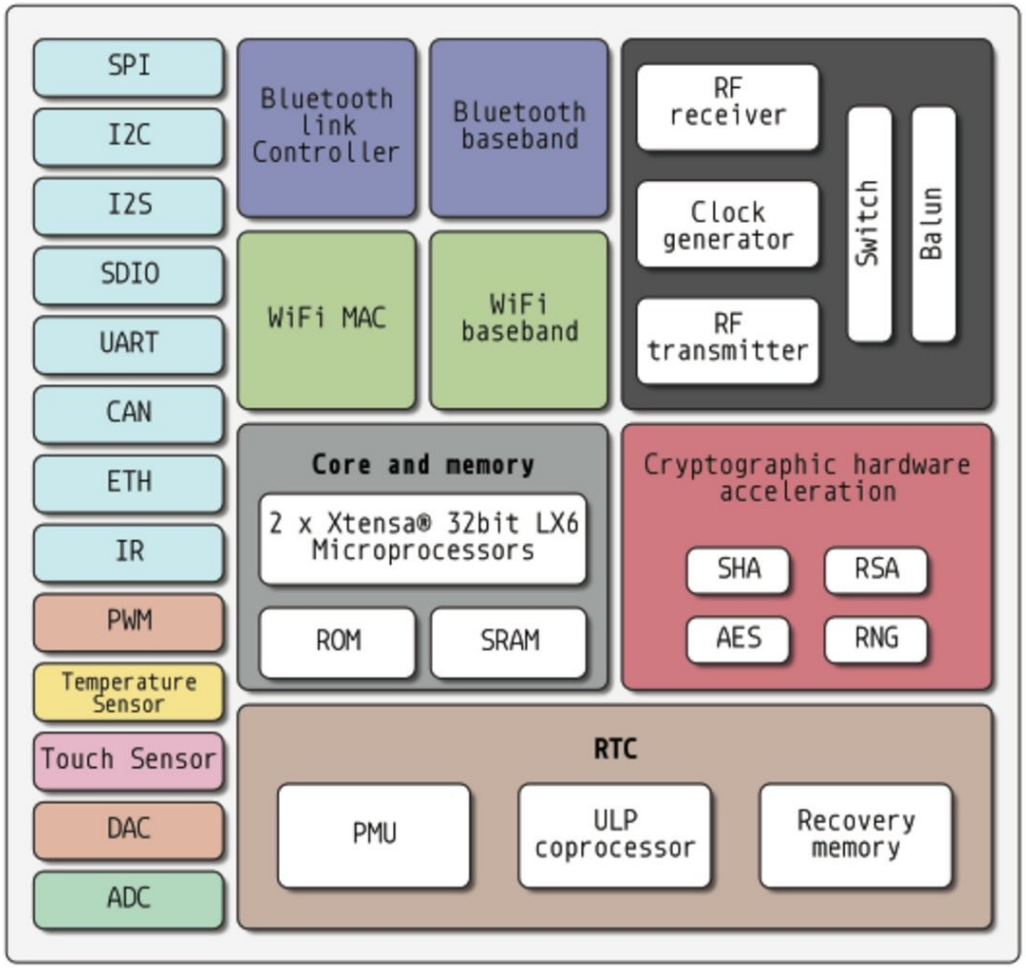
Low-power < 1-100mA (5V)

Power (W) = Current(A) * Voltage(V)

rich-interfaces

Wi-Fi

BT-BLE



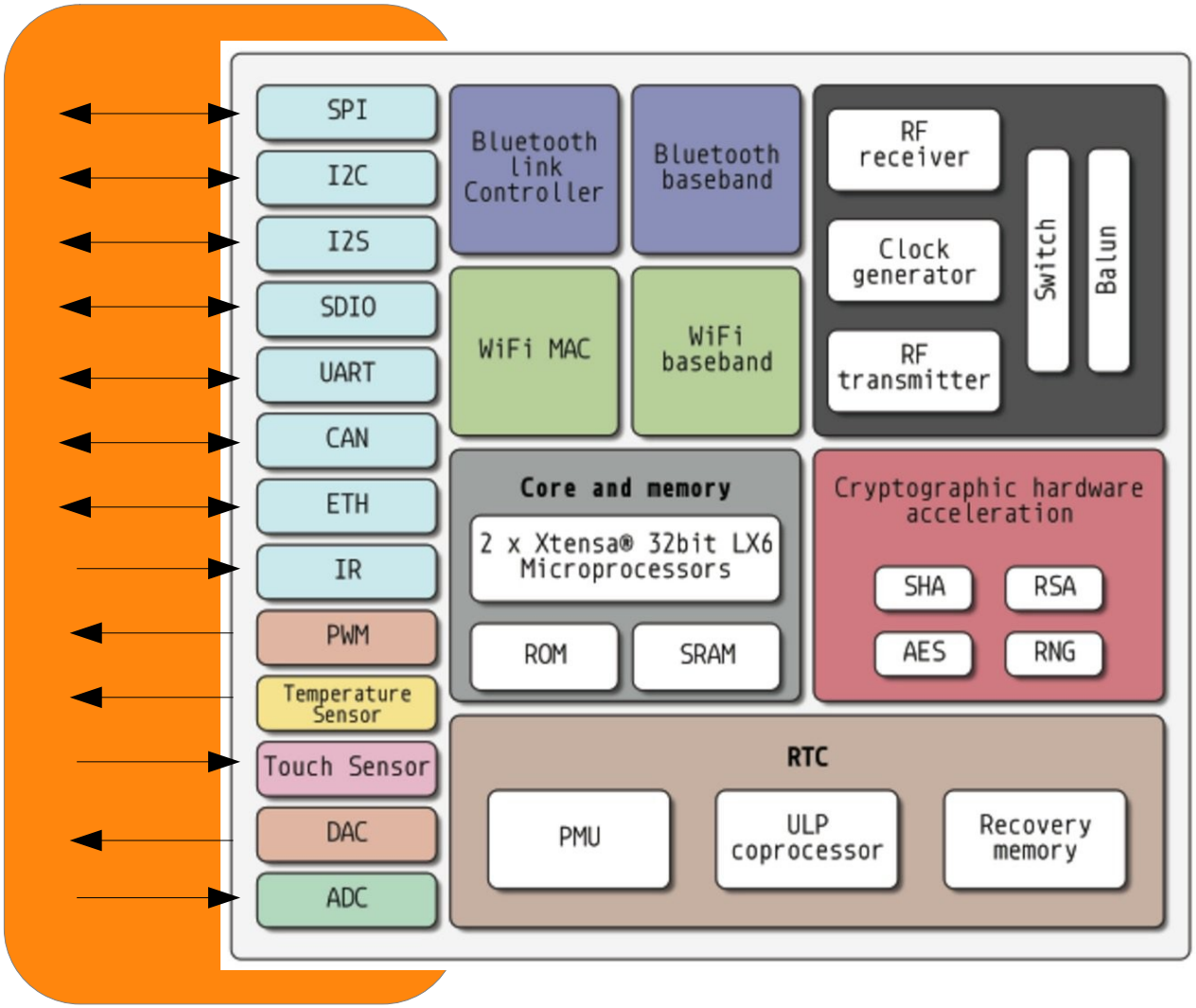
ESP32 (S1) – System on Chip

interfaces to sensors and actuators

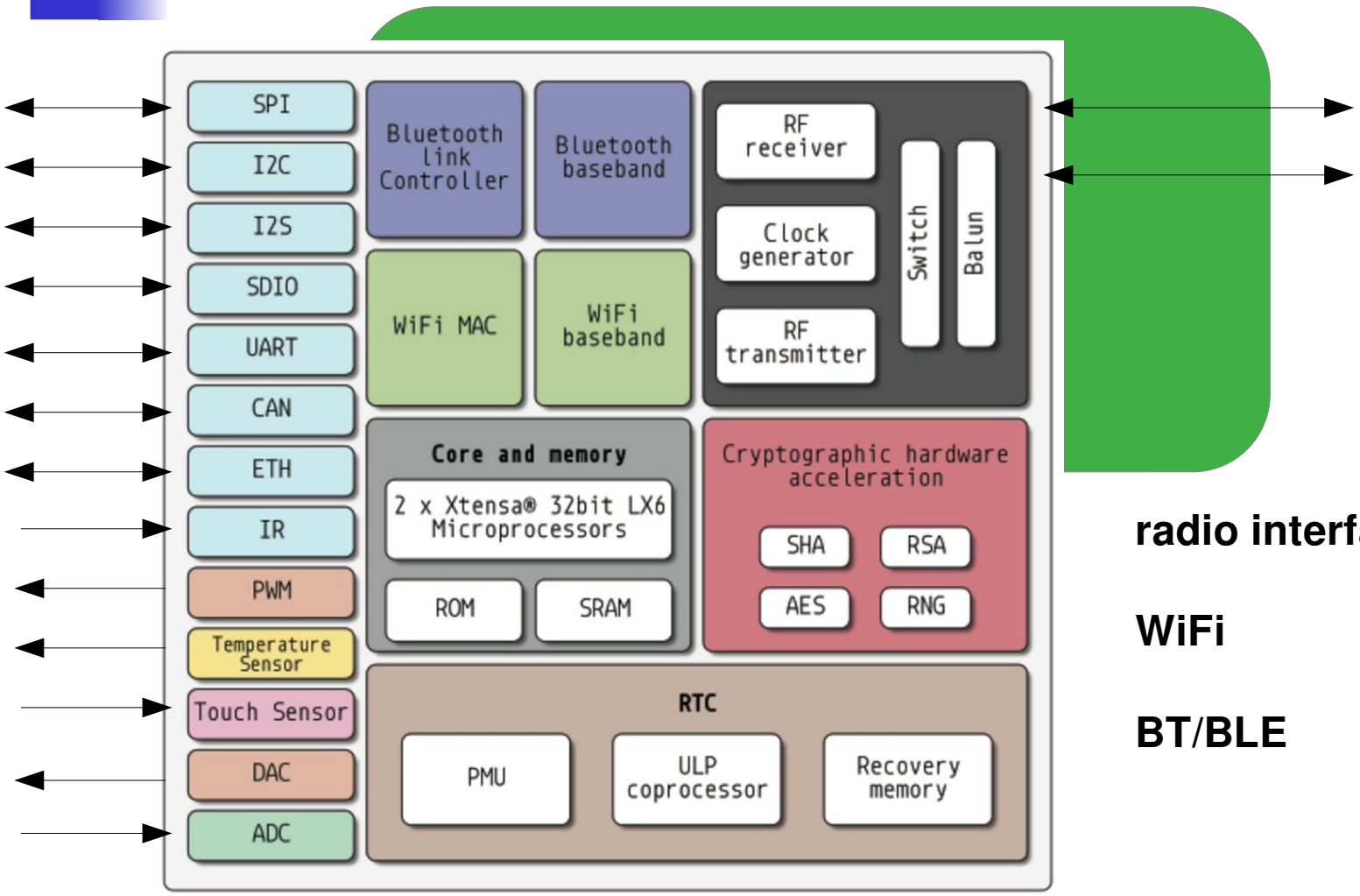
UART

I2C

SPI



ESP32 (S1) – System on Chip

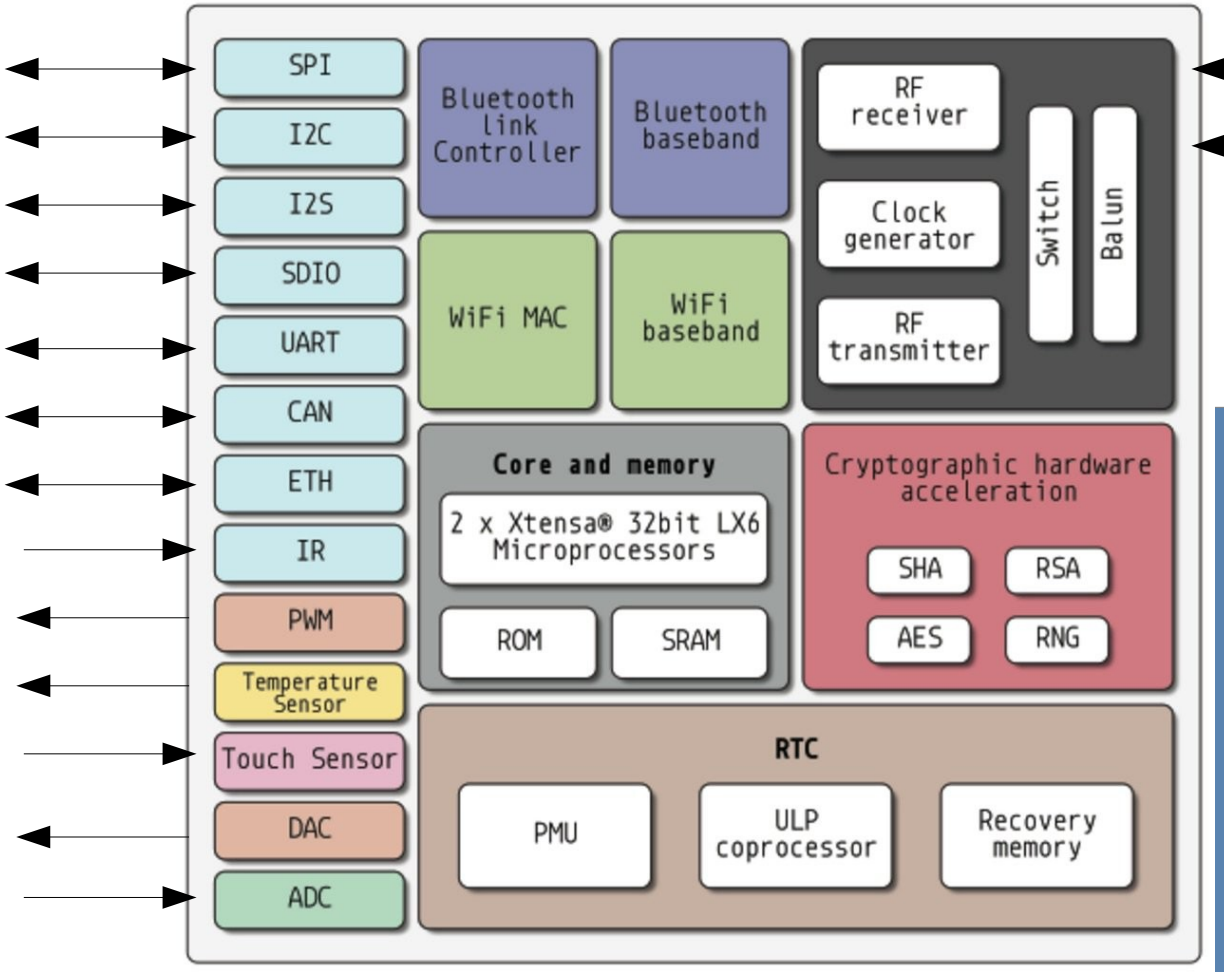


radio interfaces

WiFi

BT/BLE

ESP32 (S1) – System on Chip



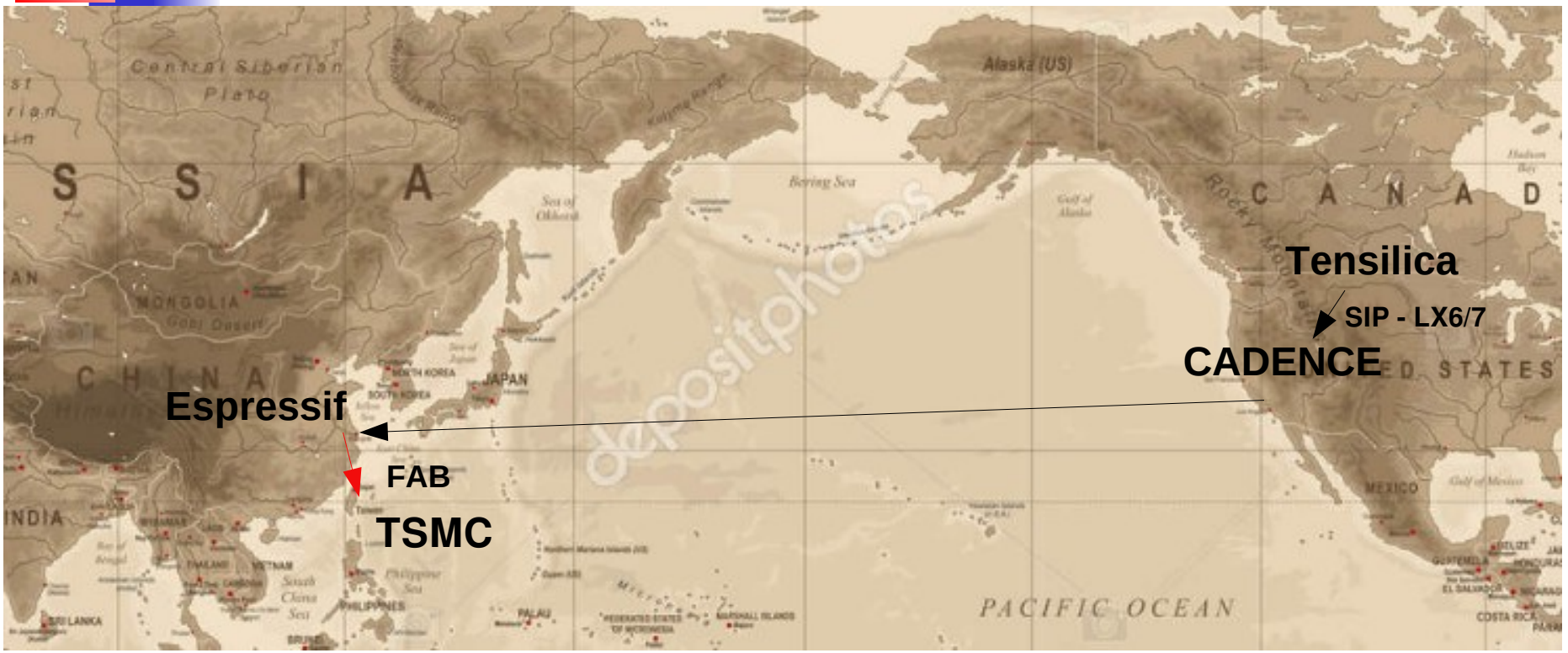
ROM : read only

SRAM : read_write

processors
Xtensa LX6
ULP

cryptographic accelerators

IoT – technology transfer aspects



Cadence Design Systems, Inc., headquartered in San Jose, California, is an American multinational computational software company, founded in 1988. The company produces software, hardware and silicon structures for designing integrated circuits and systems on chips (SoCs) .

IoT – economic aspects

Tensilica is known for its customizable **Xtensa** (LX6/7) microprocessor core.

Tensilica was a company based in Silicon Valley in the **semiconductor intellectual property** (**SIP**) core **business**. It is now a part of Cadence Design Systems.
On March 11, **2013**, Cadence Design Systems bought **Tensilica** for approximately **$380 million in cash**.

Espressif bought **eXtensa LX6/7** license (**SIP**) to design ESP32 SoCs. It went public on Shanghai Stock Exchange in **2019** with **2 billion US dollars**.

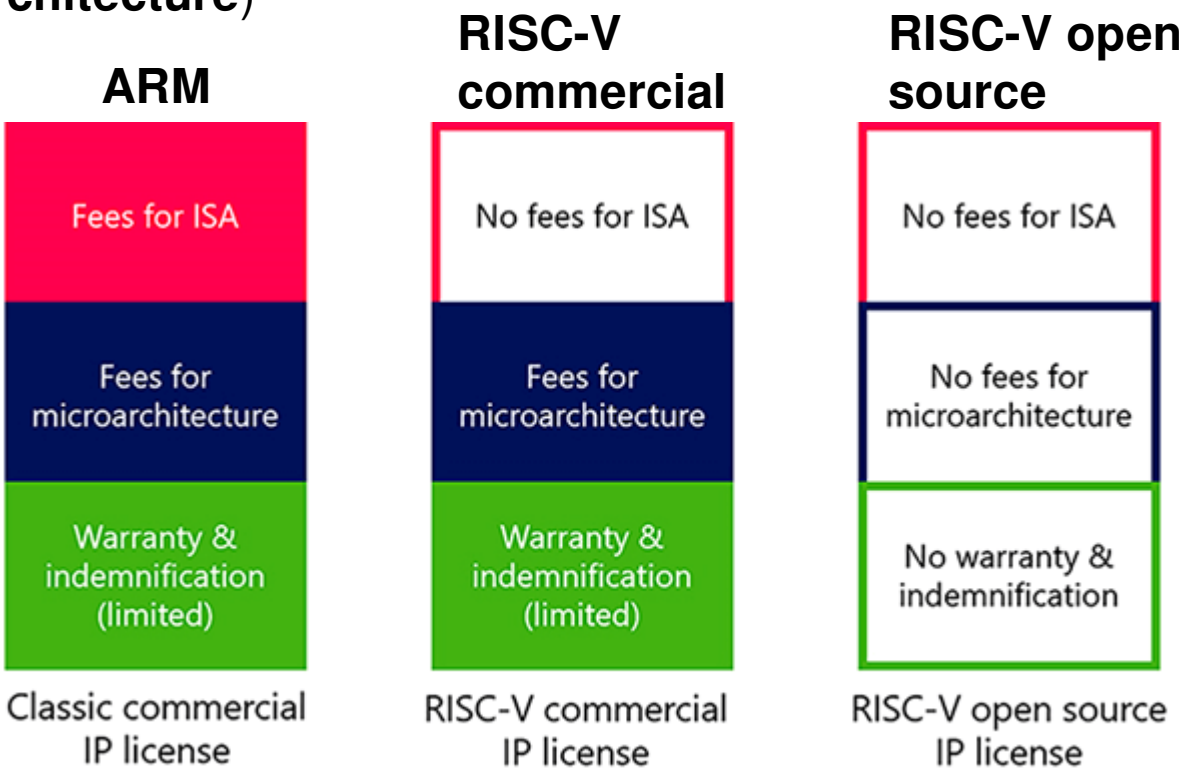
Remark: European and US investors were not allowed to buy the shares !

SIP - Silicon Intellectual Propriety

ARM vs RISC-V business model

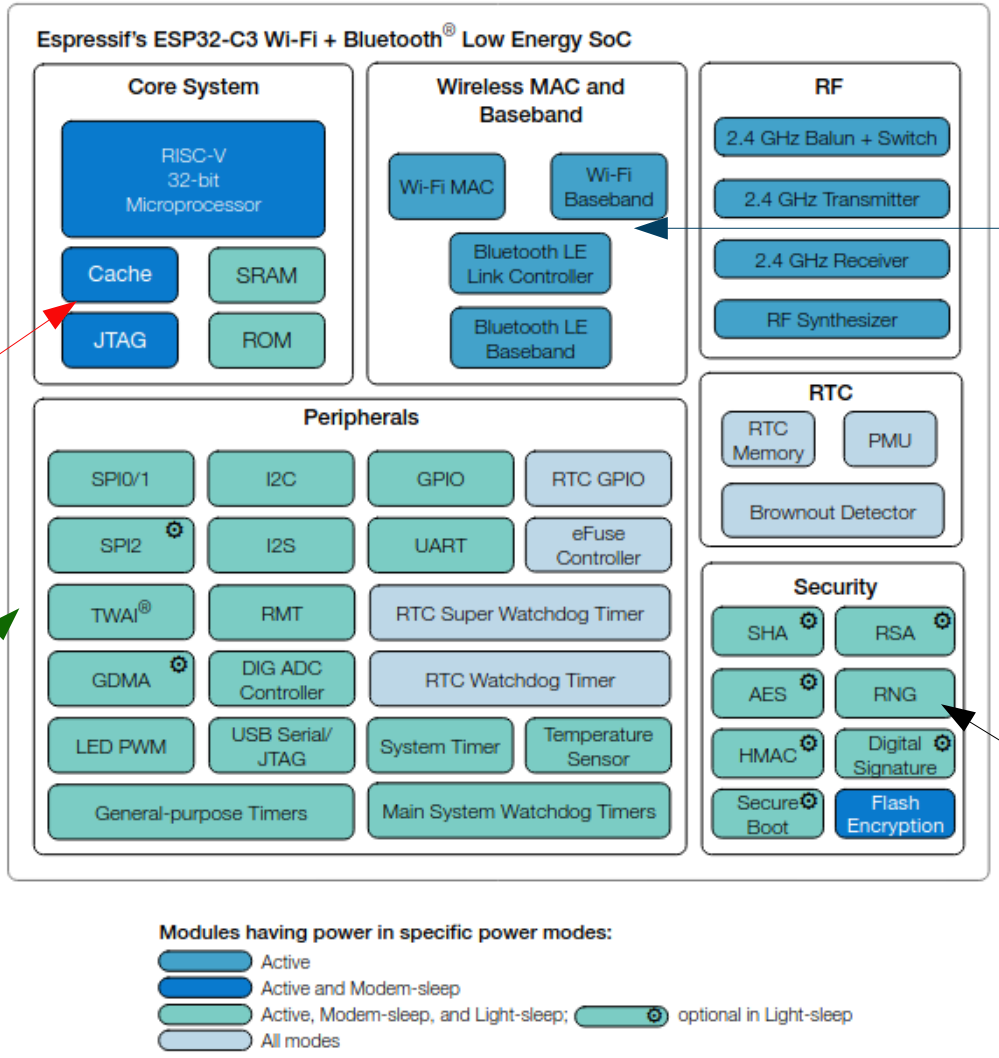
ARM's revenue comes **entirely from IP licensing including ISA**.

RISC-V is a **standard and open architecture** with **no fees** for **ISA** (**Instruction Set Architecture**)



ESP32 (C3) – RISC-V (SoC)

RISC-V is a **standard and open architecture** with no fees for **ISA** (**Instruction Set Architecture**)



Radio: WiFi, BLE

CPU (4-stage)

I/O interfaces

Security accelerators

ESP32 (C3) – RISC-V (SoC)

Main Features

• A complete WiFi subsystem : **Station** mode, **SoftAP** mode, **SoftAP + Station** mode, and **promiscuous** mode

• A **Bluetooth LE** subsystem that supports Bluetooth 5 and Bluetooth mesh

• 32 bit RISCV singlecore processor with a **four-stage pipeline** that operates at up to **160 MHz**

• Storage capacities ensured by **400 KB** of **SRAM** (16 KB for cache) and **384 KB of ROM** on the chip, and **SPI**, Dual SPI, Quad SPI, and QPI interfaces that allow connection to **external flash**

• Reliable security features ensured by

– Cryptographic hardware accelerators that support **AES**-128/256, Hash, RSA, HMAC, digital signature and secure boot

– Random number generator

– Permission control on accessing internal/external memory, and peripherals

– External memory encryption and decryption

ASR6501/2 – ARM+SX1262

The **ASR6501** is a general LoRa Wireless Communication **Chipset**, with integrated LoRa Radio Transceiver, LoRa Modem and a 32-Bit RISC MCU.

The MCU uses **ARM Cortex M0+**, with **48MHz** operation frequency.

The **LoRa Radio Transceiver** has **continuous frequency coverage from 150MHz to 960MHz**. The LoRa Modem supports LoRa modulation for **LPWAN** use cases and (G)FSK modulation for legacy use cases.

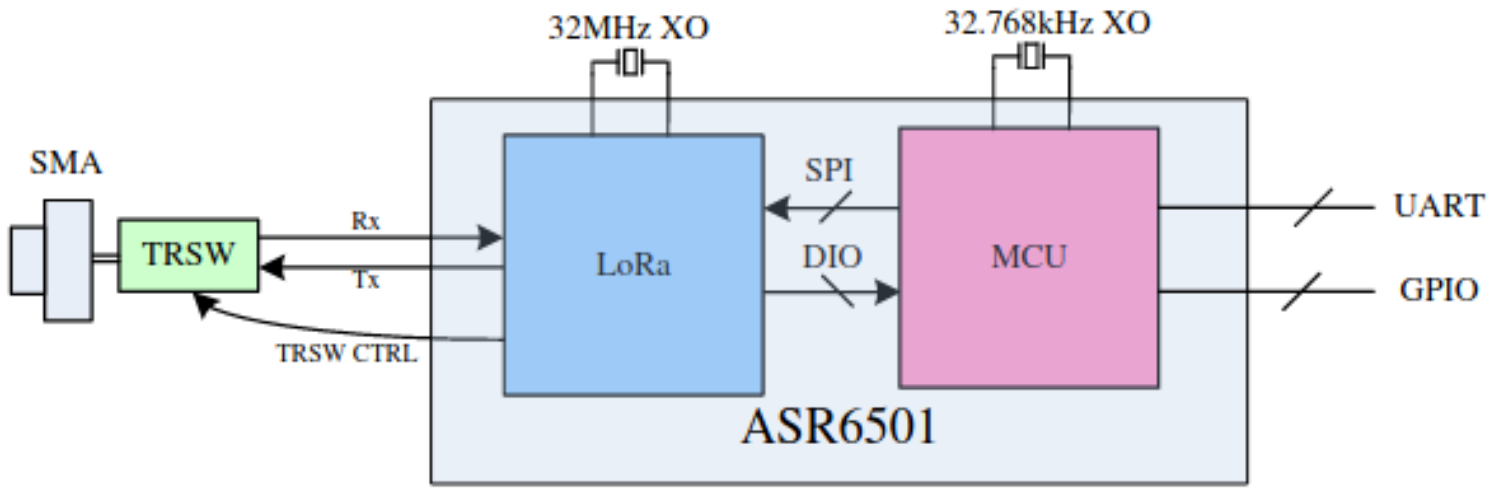
The LoRa Wireless Communication module designed by **ASR6501** provides **ultra long range**, **ultra low power communication** for **LPWAN** application.

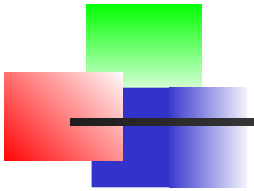
The **ASR6501** can achieve a high sensitivity of over **-140dBm** and the maximum transmit power is higher than **+21dBm**.

This makes it suitable to be used in long range **LPWAN** and have high efficiency.

ASR6501/2 – ARM+SX1262

Figure below shows the block diagram of **ASR6501** with ARM Cortex-M0 MCU and LoRa Communication Module (**SX1262**).





IoT – software aspects

ESP32 (S1,C3) SoCs are **powerful micro-controllers**.

Basically they operate under the control of **FreeRTOS**.

The programming may be carried out with:

- **C/C++** or
- **MicroPython**

C/C++ are **source languages** that must be **compiled** into **binary code** before the execution on the processor.

MicroPython (Python) is source language that is **interpretable**.After loading to the SoC memory the (Python-byte-code) may be directly (executed) interpreted. This solution requires an interpreter to be loaded and ready in the SoC **flash memory**.

IoT - Programming IDE

ESP32 SoCs programming is carried out via an **IDE** – Integrated Development Environment.

For C/C++ the IDE tools perform:

→ **Editing** of the source code
→ **Compilation** the source code to binary code
→ **Loading** (to flash memory)

For MicroPython the IDE tools perform:

→ **Editing** of the source code
→ **Loading** (to flash memory)

→ **C/C++: complete and efficient, 3 phases development cycle**

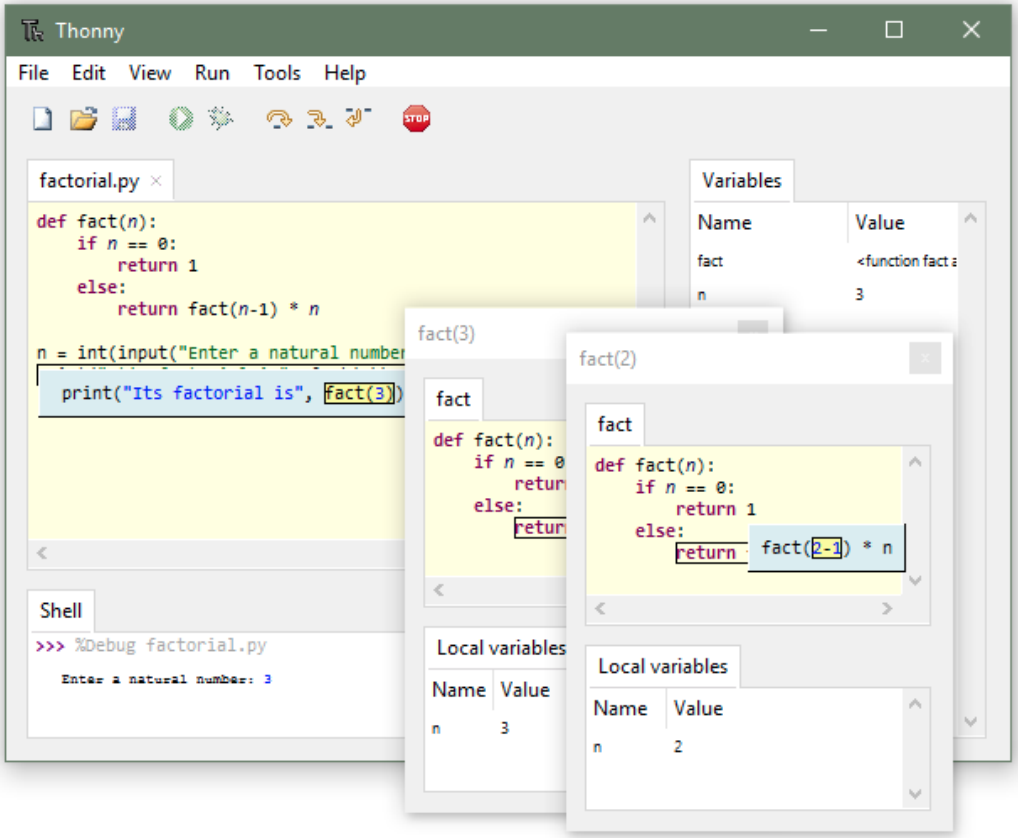
→ **MicroPython/Python – less efficient but easier to write and with 2 phases development cycle (processor independent)**

Thonny IDE – starting with µPython

Thonny

Python IDE for beginners

Download version 3.3.13 for Windows • Mac • Linux



ESP32-C3 (RISC-V) and µPython

In the **first part of this module** we are going to use **µPython** running (with interpreter) on **ES32C3** based board and the IoT DevKiT.

The Kit contains:

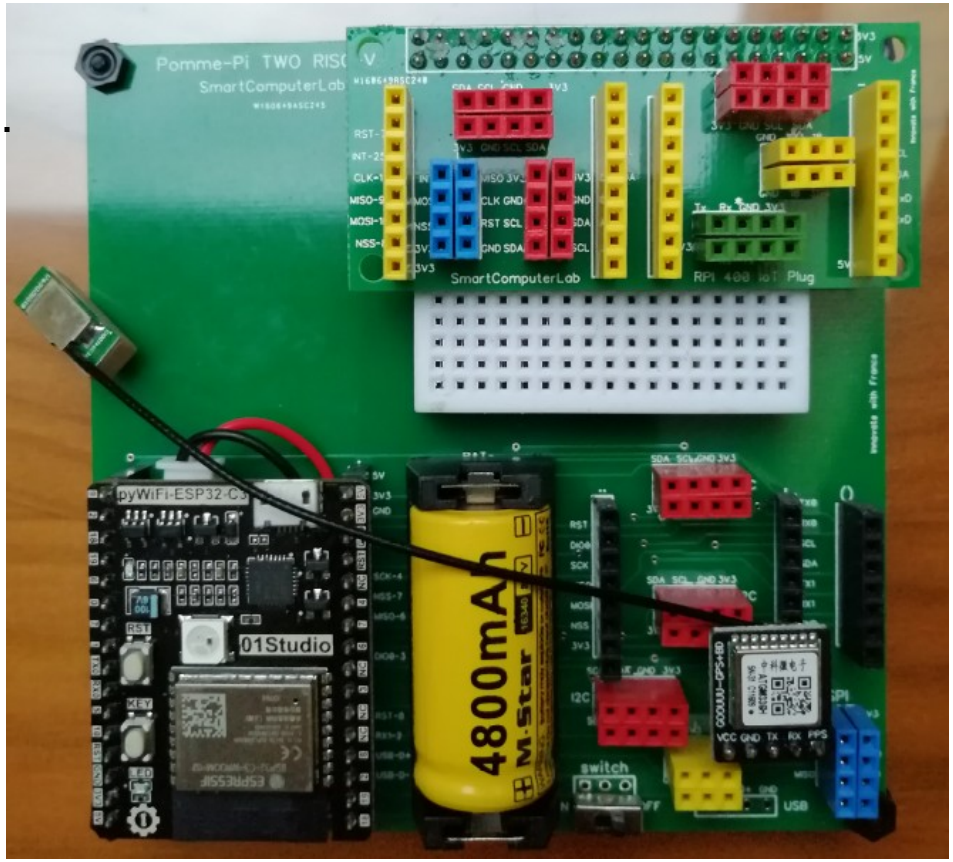
PyWiFi main board with **ESP32C3 SoC**.

Bus interfaces for sensors and actuators

- I2C (**red**)
- SPI (**blue**)
- UART (**green**)
- simple (**yellow**)

+ **MicroBus** and **GPIO**

+ small breadboard



ESP32-S1 (2*LX06) and C/C++

In the **first second of this module** we are going to use **Arduino IDE (C/C++)** **ES32S1** based board with LoRa modem and the IoT DevKiT.

The Kit contains:

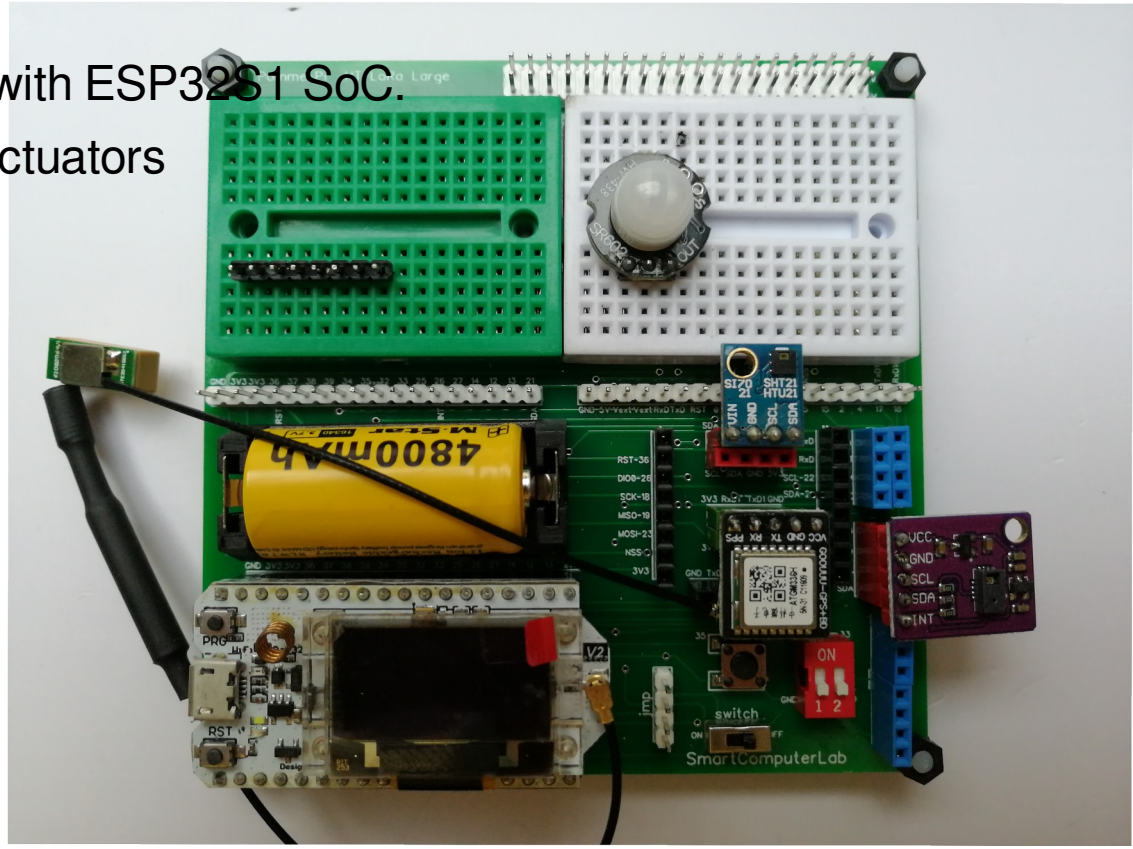
Heltec WiFi-LoRa main board with ESP32S1 SoC.

Bus interfaces for sensors and actuators

- I2C (**red**)
- SPI (**blue**)
- UART (**green**)

+ **MicroBus** and **GPIO**

+ small breadboard



ASR6501/CubeCell and C/C++

In the **third part of this module** we are going to use **Arduino IDE** with **CubeCell (ASR6501)** based board and the IoT DevKiT + PPK2 **power profiler kit**.

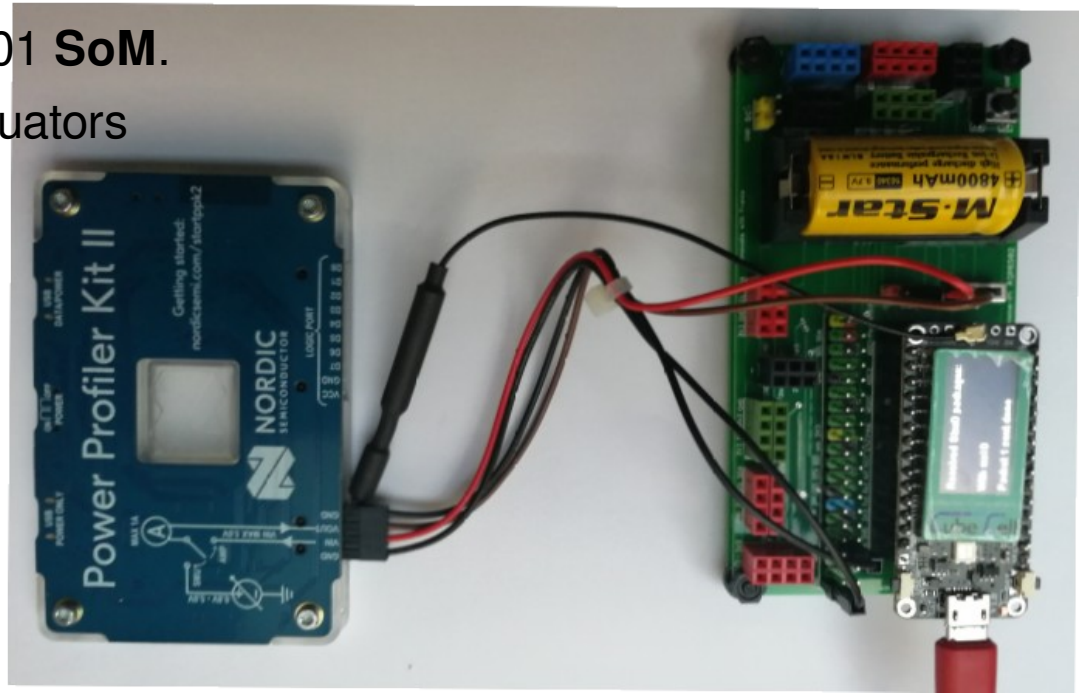
The Kit contains:

CubeCell main board with ASR6501 **SoM**.

Bus interfaces for sensors and actuators

- I2C (red)
- SPI (blue)
- UART (green)
- simple (black)

+ MicroBus and GPIO



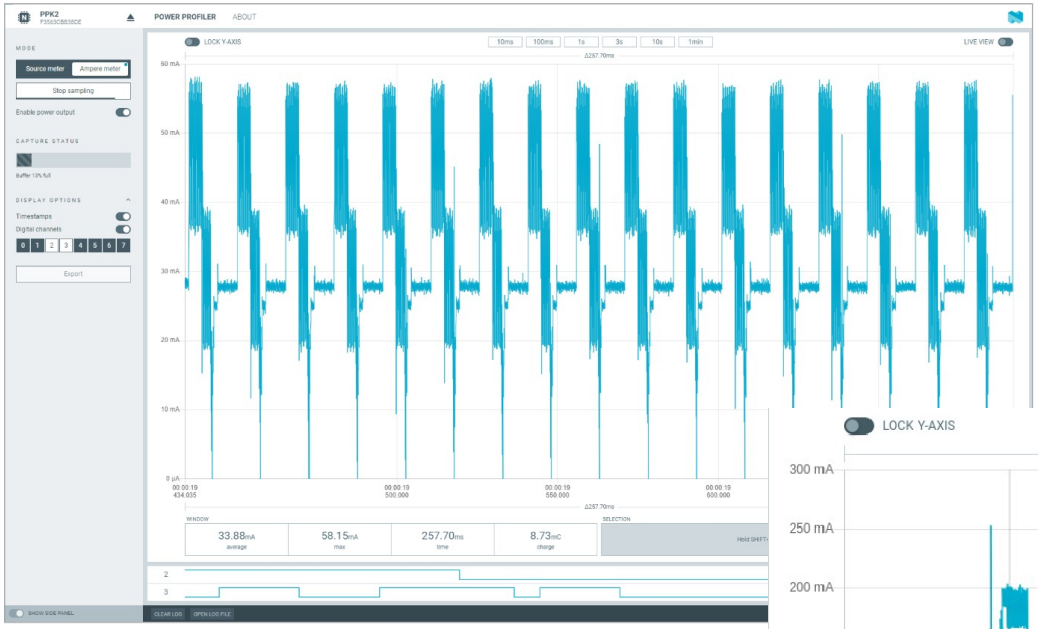
Power Profiler Kit II

The **Power Profiler Kit II** (PPK2) from **NORDIC** Semiconductor is a **standalone unit**, which can measure and optionally supply currents all the way from **sub-uA (2µ)** and as high as **1A** on all SmartComputerLab IoT DevKit boards from Smartcomputerlab.

With PPK2 it is possible to measure
low sleep currents,
the **higher active currents**,
as well as **short current peaks.**

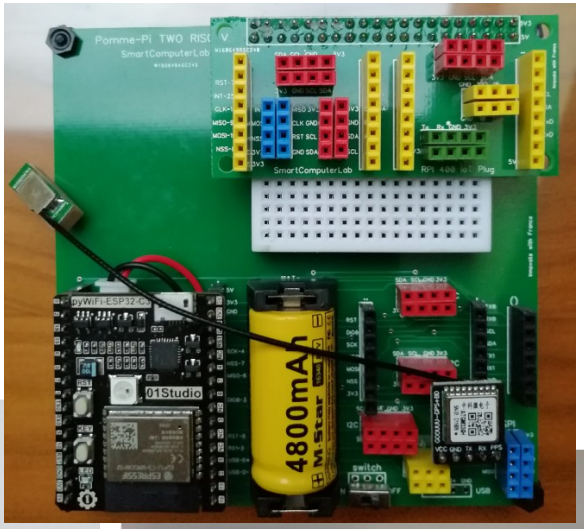
Power Profiler Kit II

PPK2 software and display/analysis functions:



Final mini-project

In the **fourth part of this module** is a kind mini-project where you are going to build your own IoT architectures.

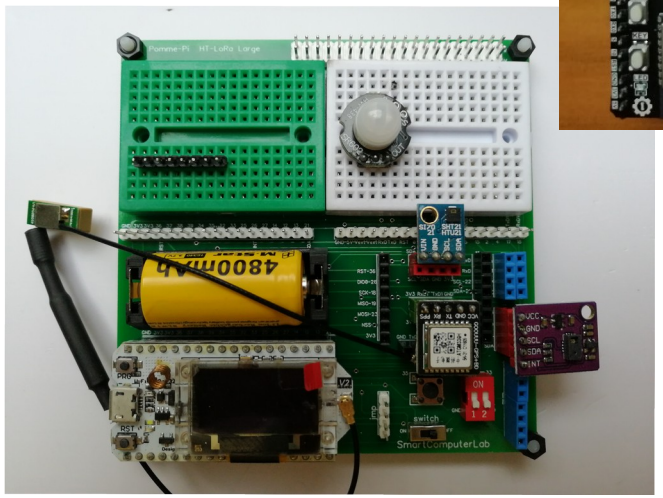


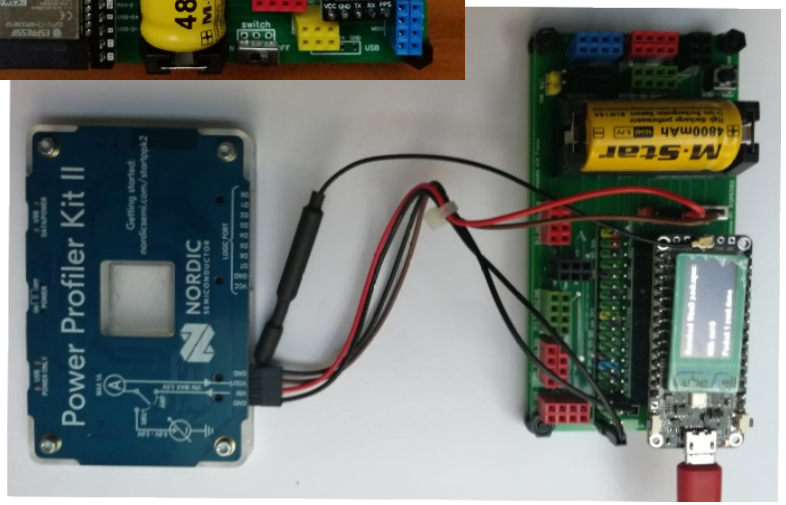
You choose the type of project:

Technical

Application

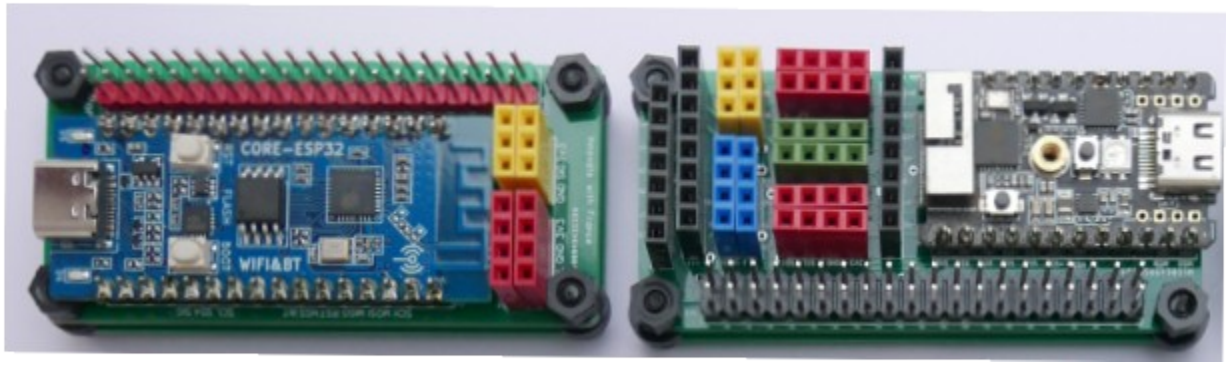
Or ?





To start with RISC-V on Pomme-Pi ZERO

In order to start with the development of **IoT architectures** we may propose the use of **Pomme-Pi ZERO** (CORE or Stamp):



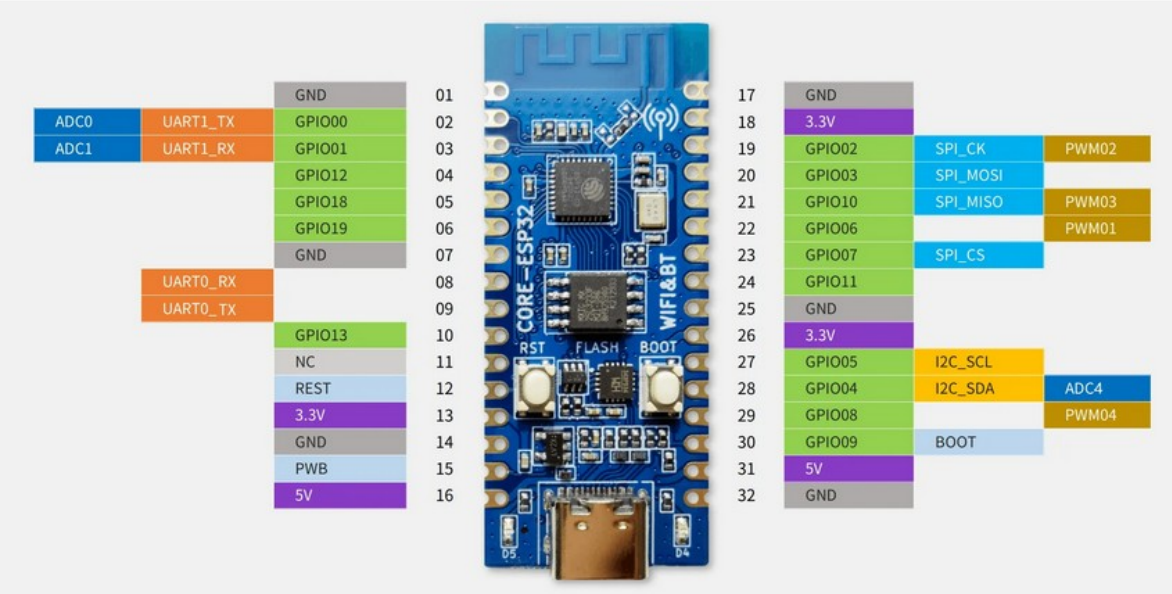
You can now download **Thonny IDE** and start the exploration of these cards (both **ESP32C3 – RISC-V**). The preparations are ready on:

https://github.com/smartcomputerlab/IoT-Labs-with-Pomme-Pi-ZERO-ONE-and-TWO/tree/main/docs

To start with RISC-V on Pomme-Pi ZERO

Attention:

The preparations are ready for Pomme-Pi ZERO (**CORE**), some modifications concerning the use of I2C/SPI/UART buses must be done for the version **Stamp**.



You can now download Thonny IDE and start the exploration of these cards (both ESP32C3 – RISC-V). The preparations are ready on:

https://github.com/smartcomputerlab/IoT-Labs-with-Pomme-Pi-ZERO-ONE-and-TWO/tree/main/docs