Ch. 4 Embedded System Characteristics and Real-time OS

COMPSCI 147

Internet-of-Things; Software and Systems

WHAT IS EMBEDDED DEVICE?

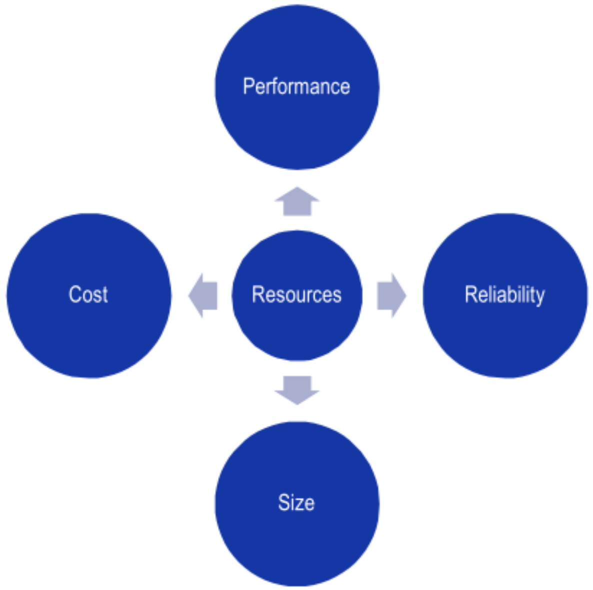




- No clear definition!
  - Characteristics of embedded systems are more **descriptive** than definitive.

CHARACTERISTICS OF EMBEDDED DEVICES

- Embedded devices are intended **to do one (or few) thing**.
- When the user buys a unit, all of the **necessary software is already inside** the **system**, and, updates aside, the user has no further impact on software content.
- Often characterized by: **low power consumption, small size, very limited operating ranges, low per-unit cost, real-time** computing constraints.
- The task is to optimally **manage available resources** => more difficult to **program** and to **interact** with than general purpose devices.



FIVE TIPS FOR A "PC PROGRAMMER" MOVING TO "EMBEDDED PROGRAMMER"

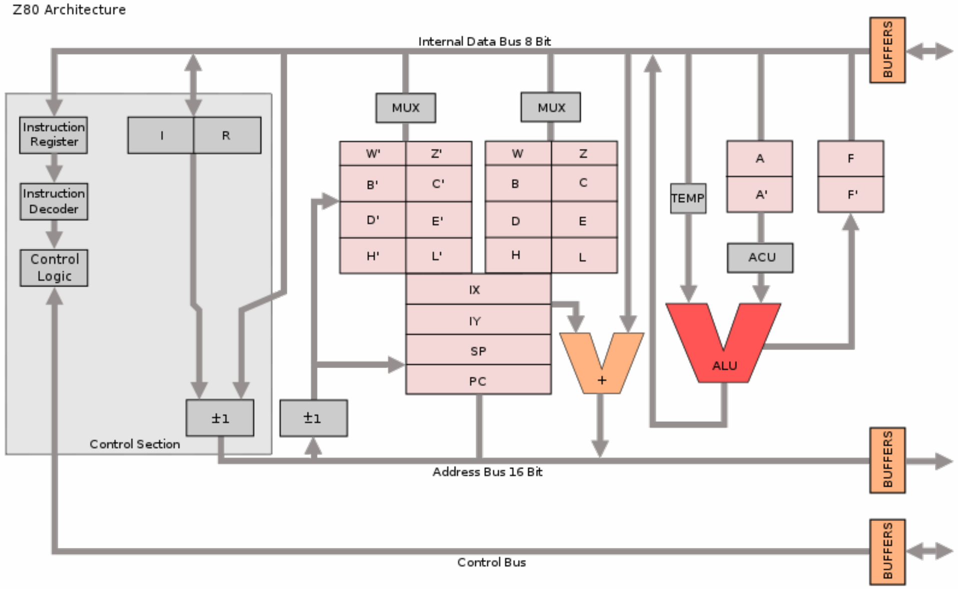
- Remember that embedded system can be **installed** for example in a **satellite**.
- You should assume that system is **inaccessible** afterwards. Albeit, **over the air** firmware upgrades possible on IoT devices.
- Prepare situation where your **embedded real time system** must handle surprisingly **large amount of data**.
- **Reasonable and detailed error messages** are important, because **repeating error conditions can be difficult**, even impossible.
- Importance of **planning the testing**, and **testing during development** phase.

WHAT IS EMBEDDED (IOT) PROGRAMMING?

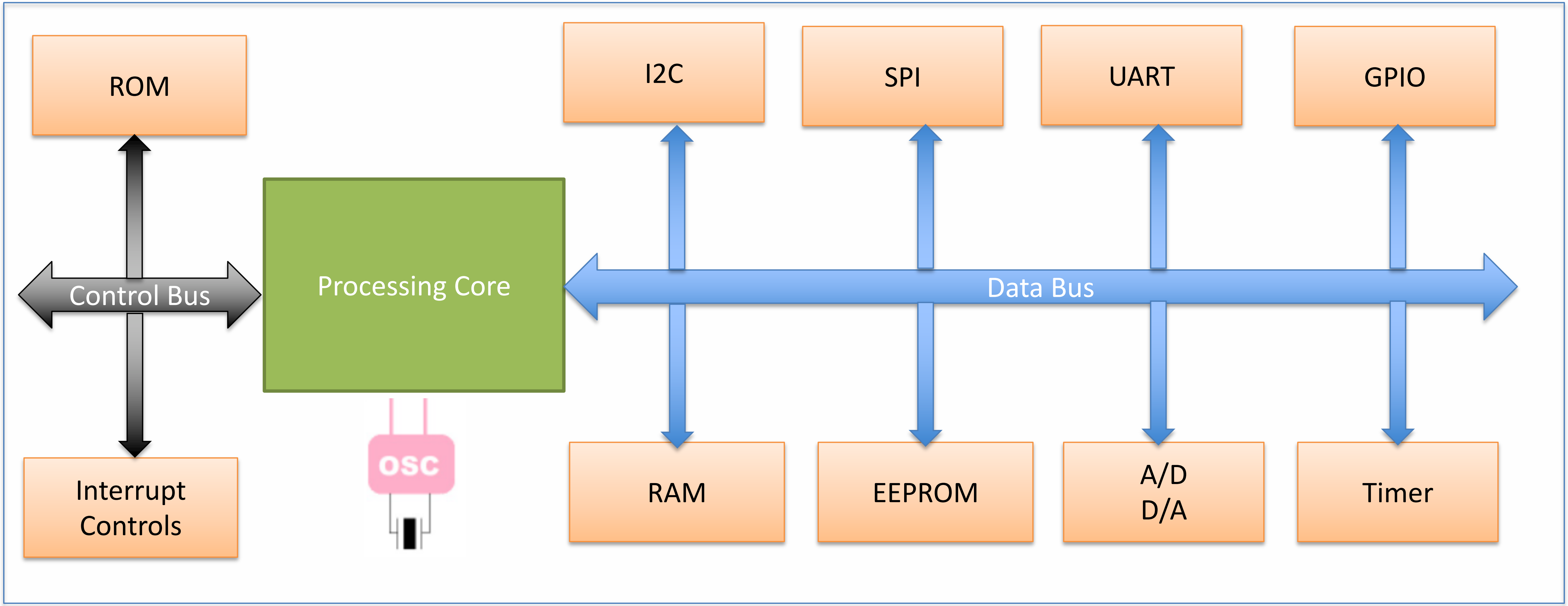
- **Embedded** devices, such as network connected IoT devices, are implemented using both **hardware** and **software** components.
  - **Programmer** needs to understand (at least) how to **interface HW components** in order to write SW for embedded systems => ability to **read component data sheets** required
- Dedicated **hardware components** are used to implement the **interface** with the physical world.
- At the heart of the embedded device is a microcontroller which executes software that interprets inputs and controls the system. => Embedded programming is programming of the controller.

Microprocessor

- A computer processor where the data processing logic and control is included on a single integrated circuit.
- Capable of interpreting and executing program instructions and performing arithmetic operations.
- Contains Arithmetic, logic, and control circuitry required to perform the functions of a computer's central processing unit.
- First introduced by Intel (Intel 4004 introduced in 1971.)
- Prevalent even now (Intel I7): typically intended for general purpose computing



Microcontroller



MPU VS MCU VS SOC

Often confused because share many common features.

– **Microprocessor (MPU)**

CPU, no RAM, ROM, nor peripherals

(e.g. Intel i7)

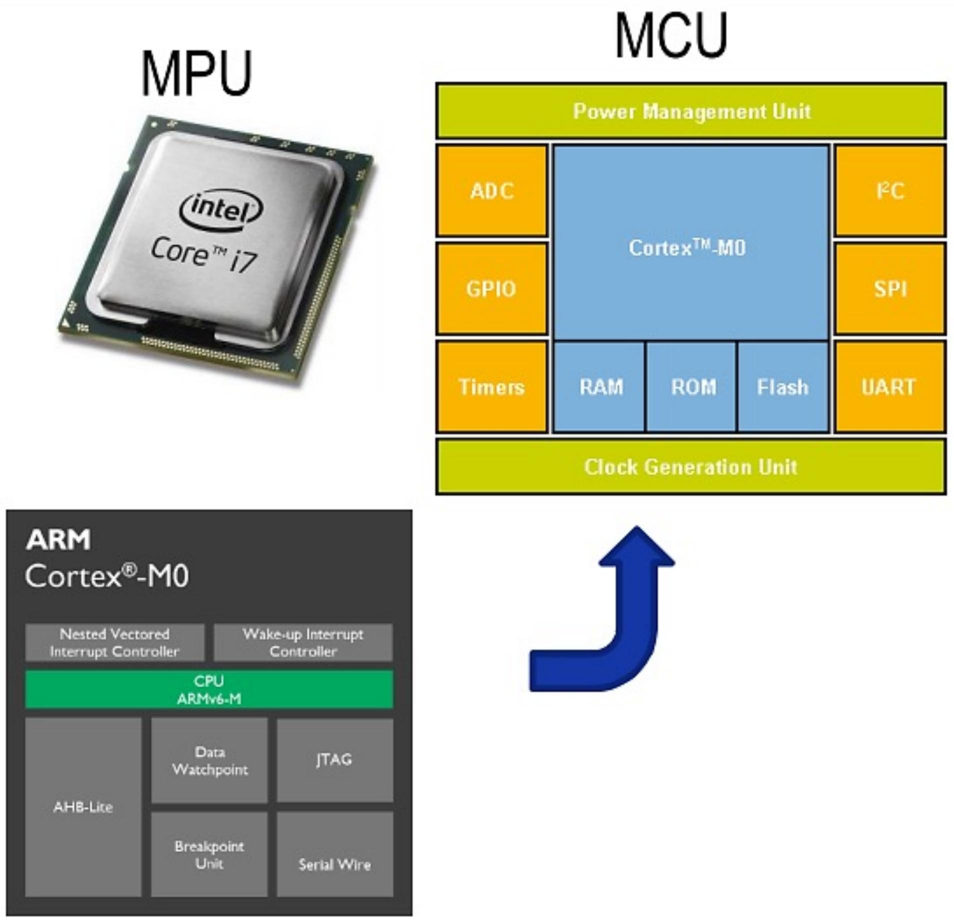
– **Microcontroller (MCU)**

CPU (e.g. ARM Cortex-M series)
+ memory
+ input/output peripherals
on a single chip

- (e.g., ARM Cortex M0)

- **System on a Chip (SoC)**

Microcontroller is a component
+ Integrates advanced peripherals (e.g., GPU/WiFi)
- (e.g., ESP 32)

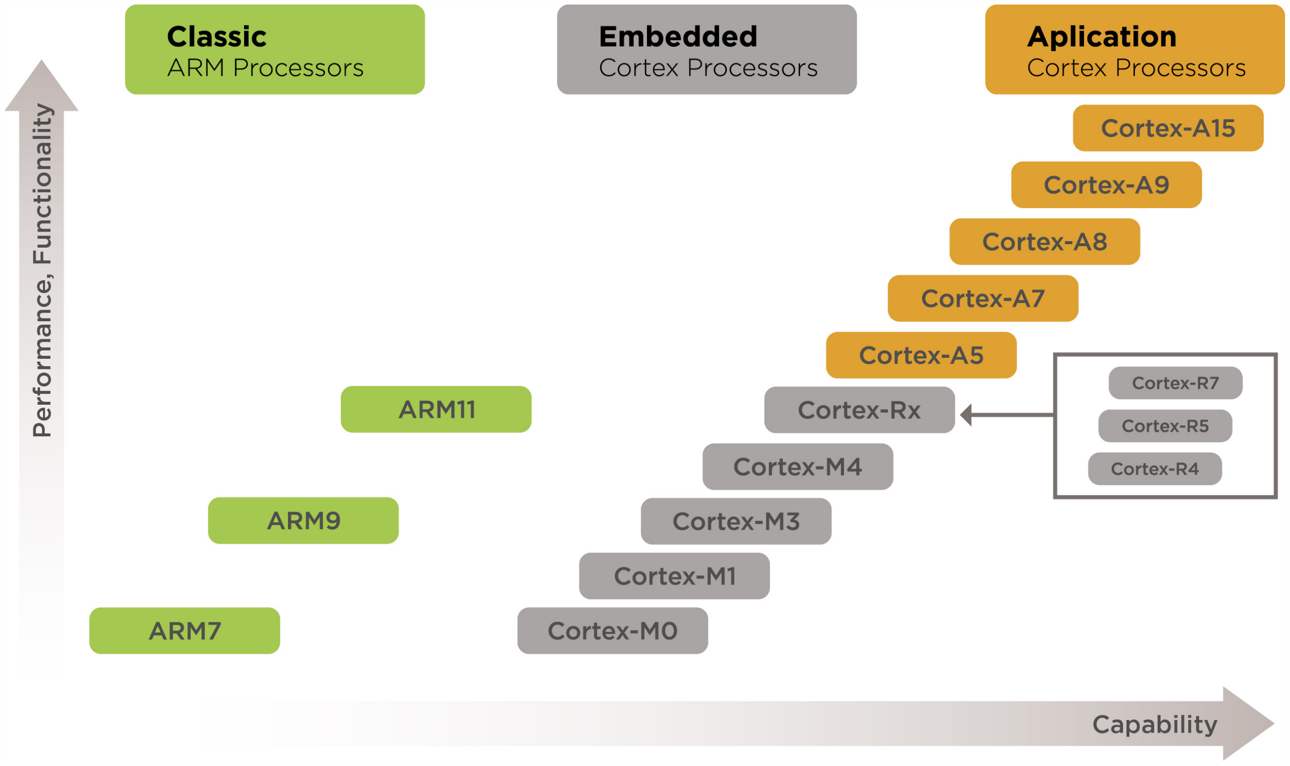


Example nomenclature for ARM

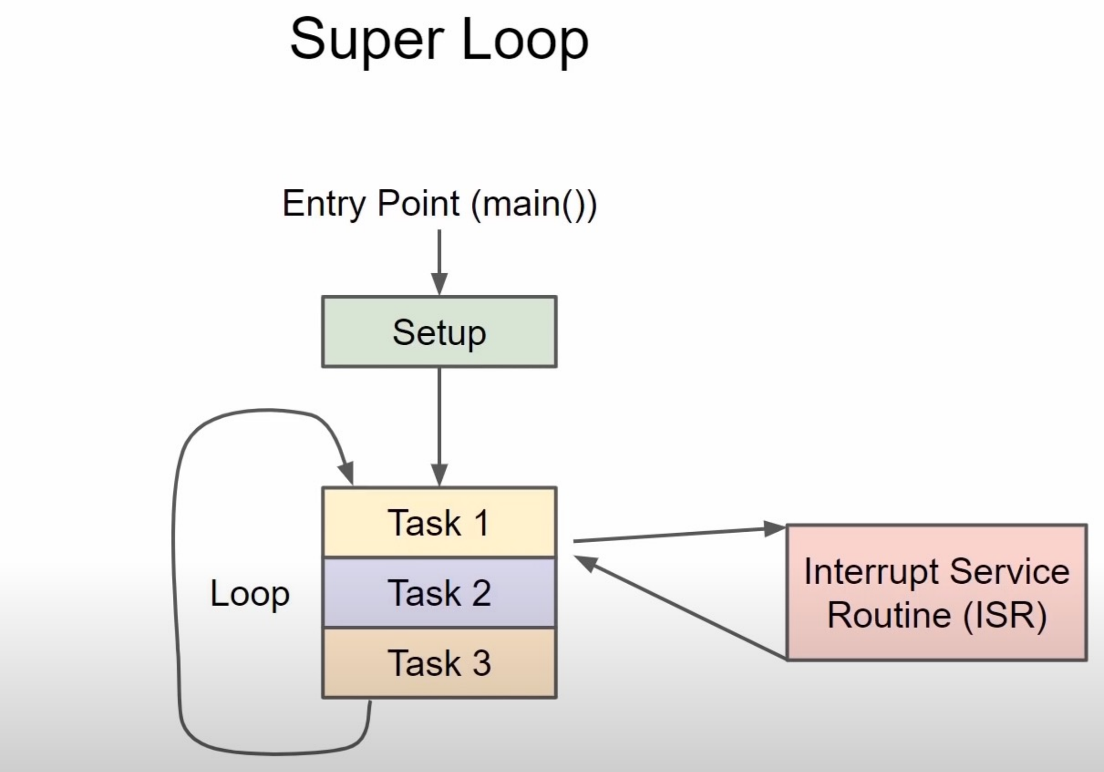
- Cortex M: Microcontroller
  Meant for embedded applications
  E.g., Cortex M0.

- Cortex A: Application
  Meant for general purpose computing
  MMU (Memory Management Unit) is mandatory to support virtual memory.
  E.g., Raspberry Pi uses quad-core Cortex-A based SoC.

- Cortex R: Realtime
  Intended for safety-critical, hard real-time applications.



Microcontroller Super-loop

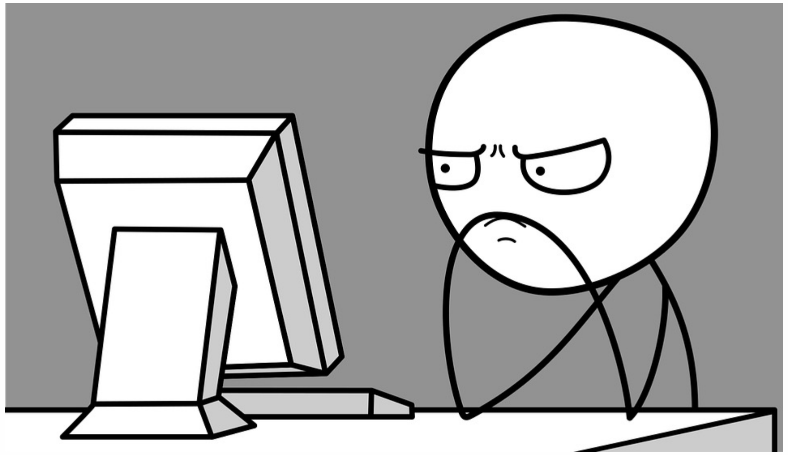


Goal :

Blink LED1 every 1 second + Blink LED2 every 2 seconds + Blink LED3 every 1.5 seconds

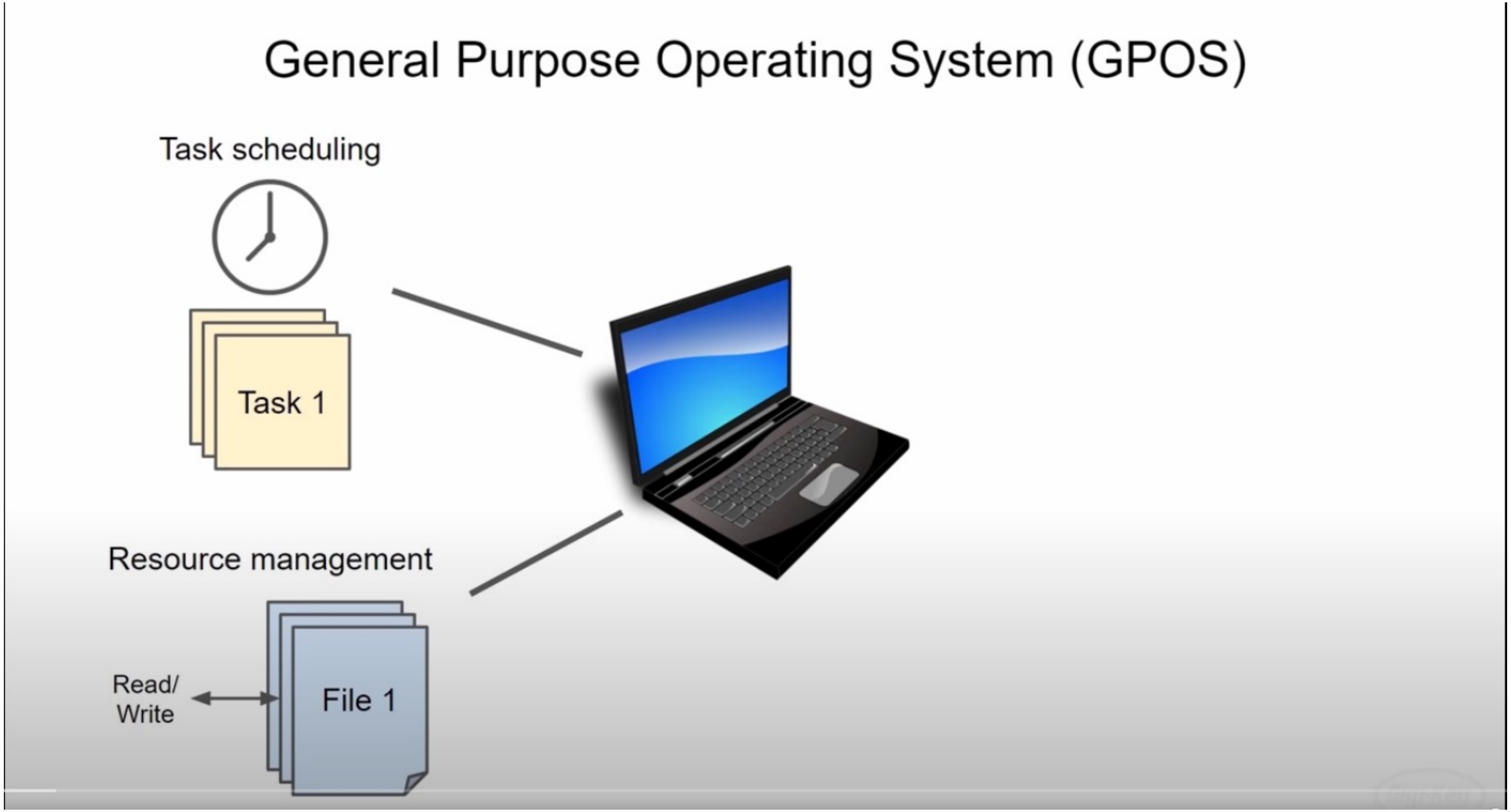
```
void loop(){
    toggle(led_1);
    sleep(1);
}
```

```
int counter=1;
void loop(){
    toggle(led_1);

    if(counter%2 ==0){
        toggle(led_2);
    }

    counter++;
    sleep(1);
}
```

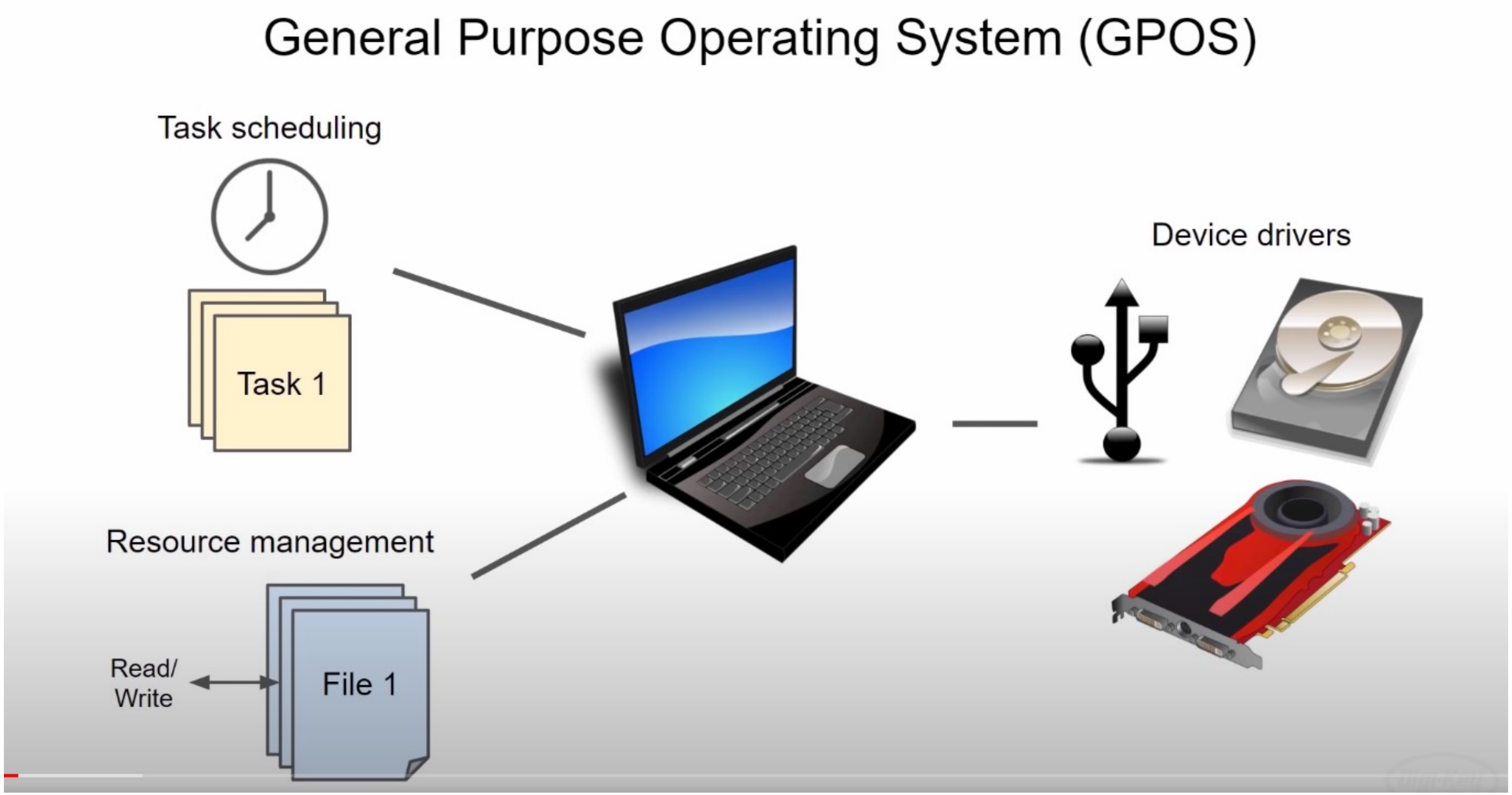


General Purpose Operating System

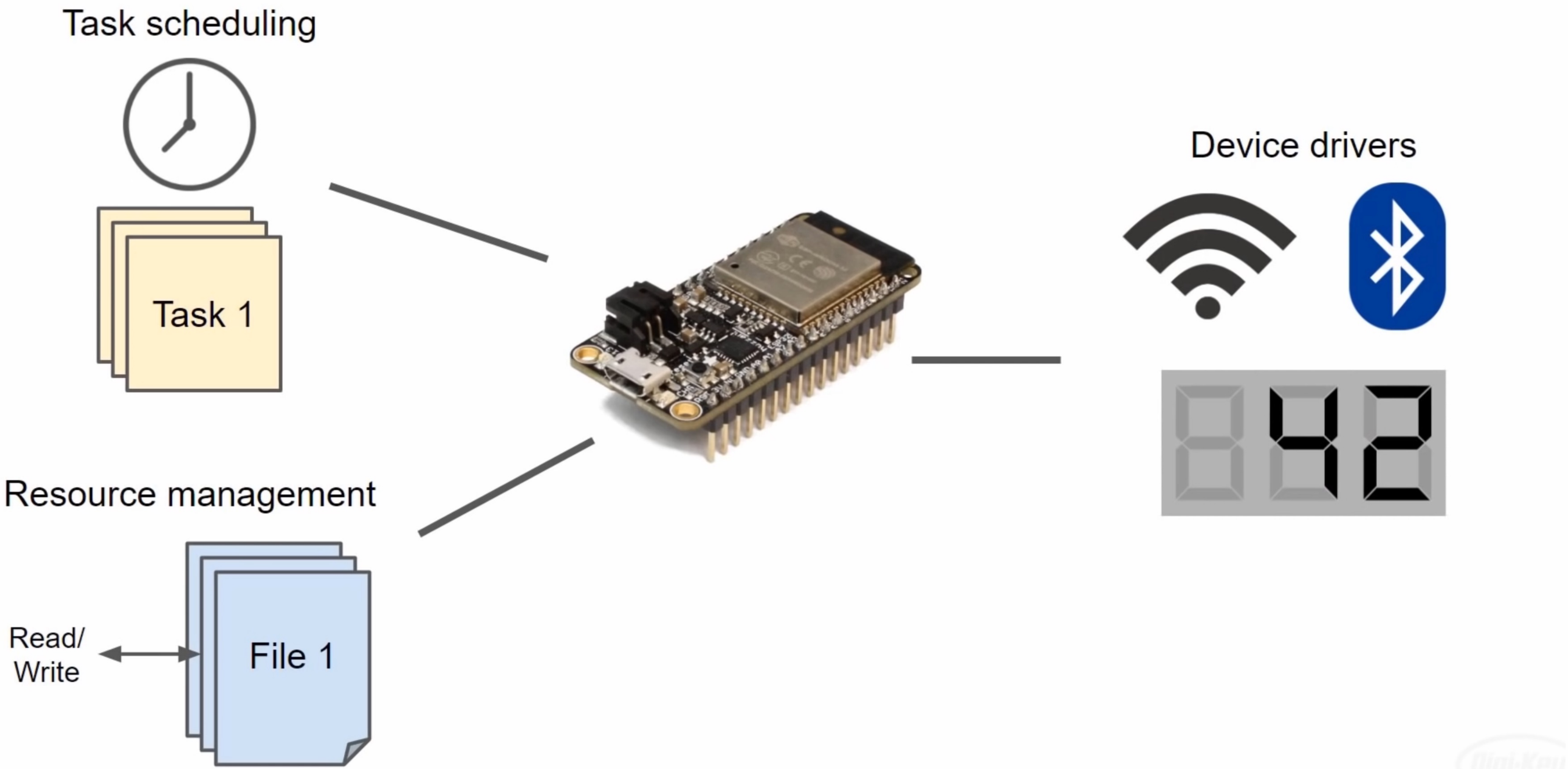
General Purpose Operating System



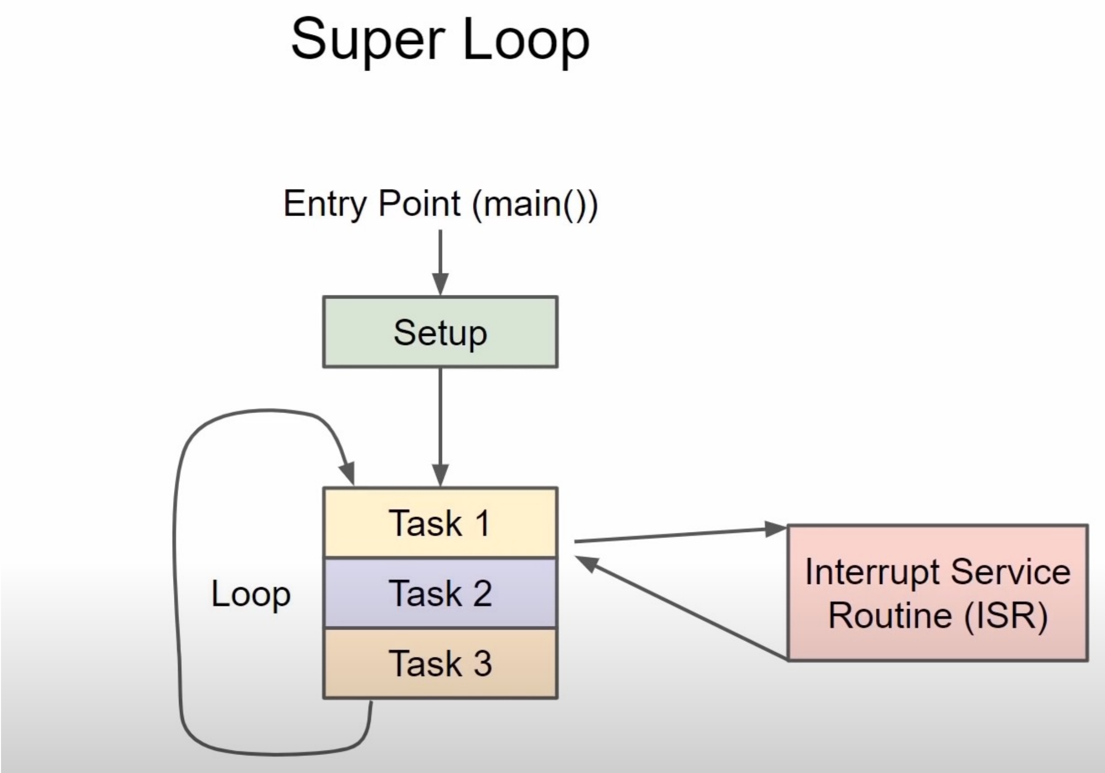
General Purpose Operating System

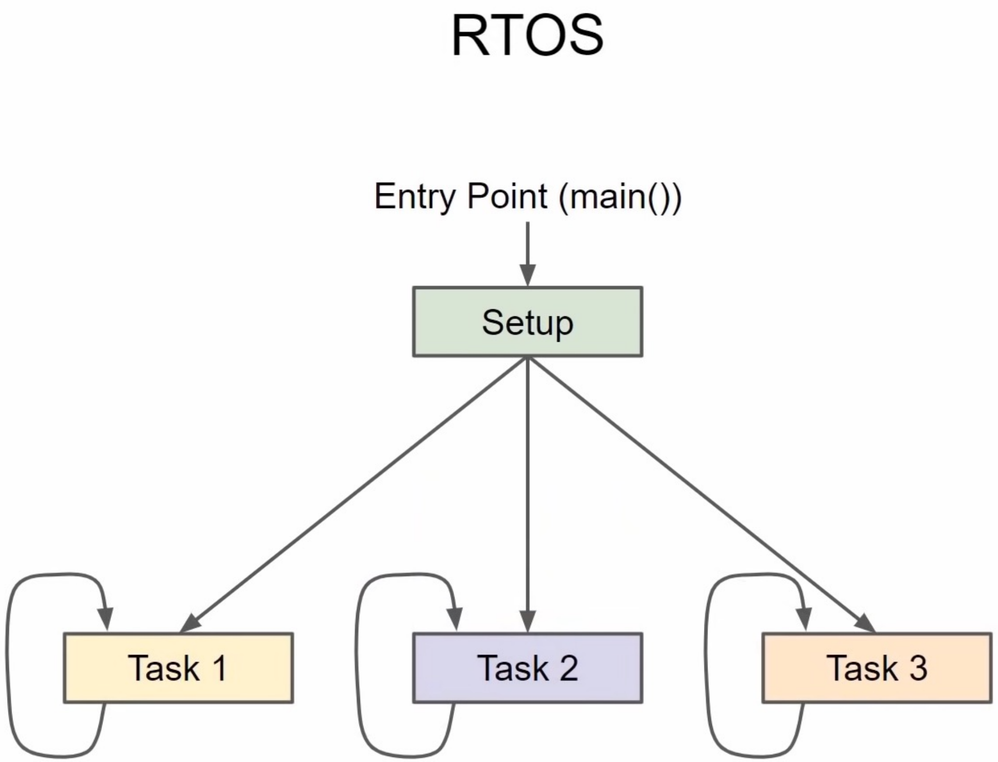


Real-Time Operating System (RTOS)



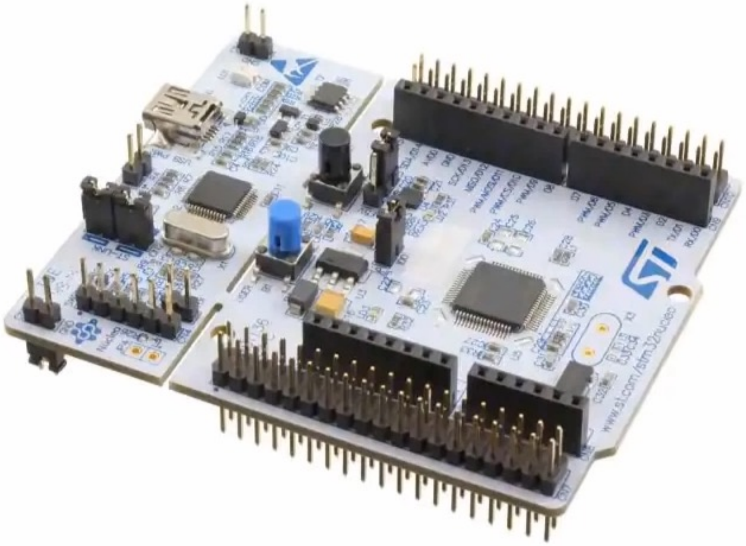
Gives the ability to run concurrent tasks



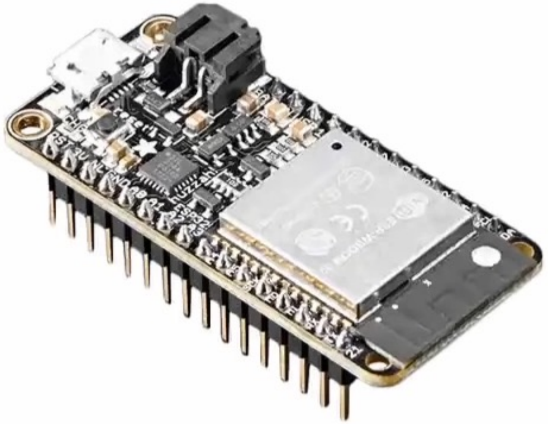


Resources dictate feasibility

ATmega 328p
- 16 MHz
- 32 kB flash
- 2 kB RAM



STM32L476RG
- 80 MHz
- 1 MB flash
- 128 kB RAM



ESP-WROOM-32
- 240 MHz (dual core)
- 4 MB flash
- 520 kB RAM

Super Loop → RTOS

OS OR NOT?

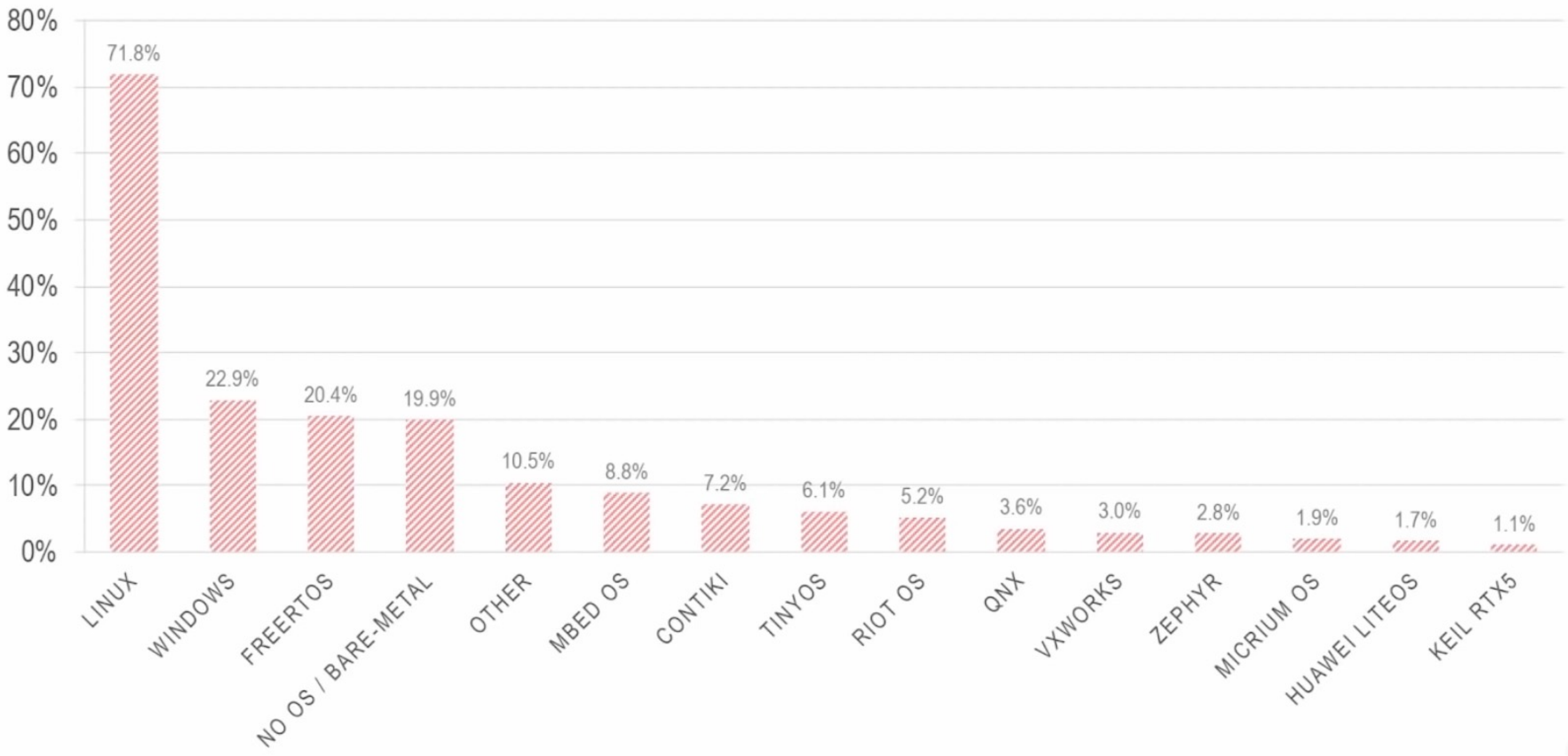
- Embedded systems often have **real-time** requirements.
- IoT devices often use an **operating system** to support the **interaction between the software and the microcontroller**
  - **RTOS** (real time operating system) includes a **scheduler** that is intended to provide **a predictable execution order**. As opposed e.g. **desktop Windows** where the main object of the scheduler is to maintain the **computer responsive**.
  - Considered RTOS must be small enough to fit into embedded system.
- RTOS not necessarily required. System can also be purely **event (interrupt) controlled**.
  - "Sleep until this happens, then do that and go back to sleep".
- **Partial implementations** of full RTOS are also possible e.g. *only scheduler, inter-task communication and syncing*

EXAMPLE OF RTOS

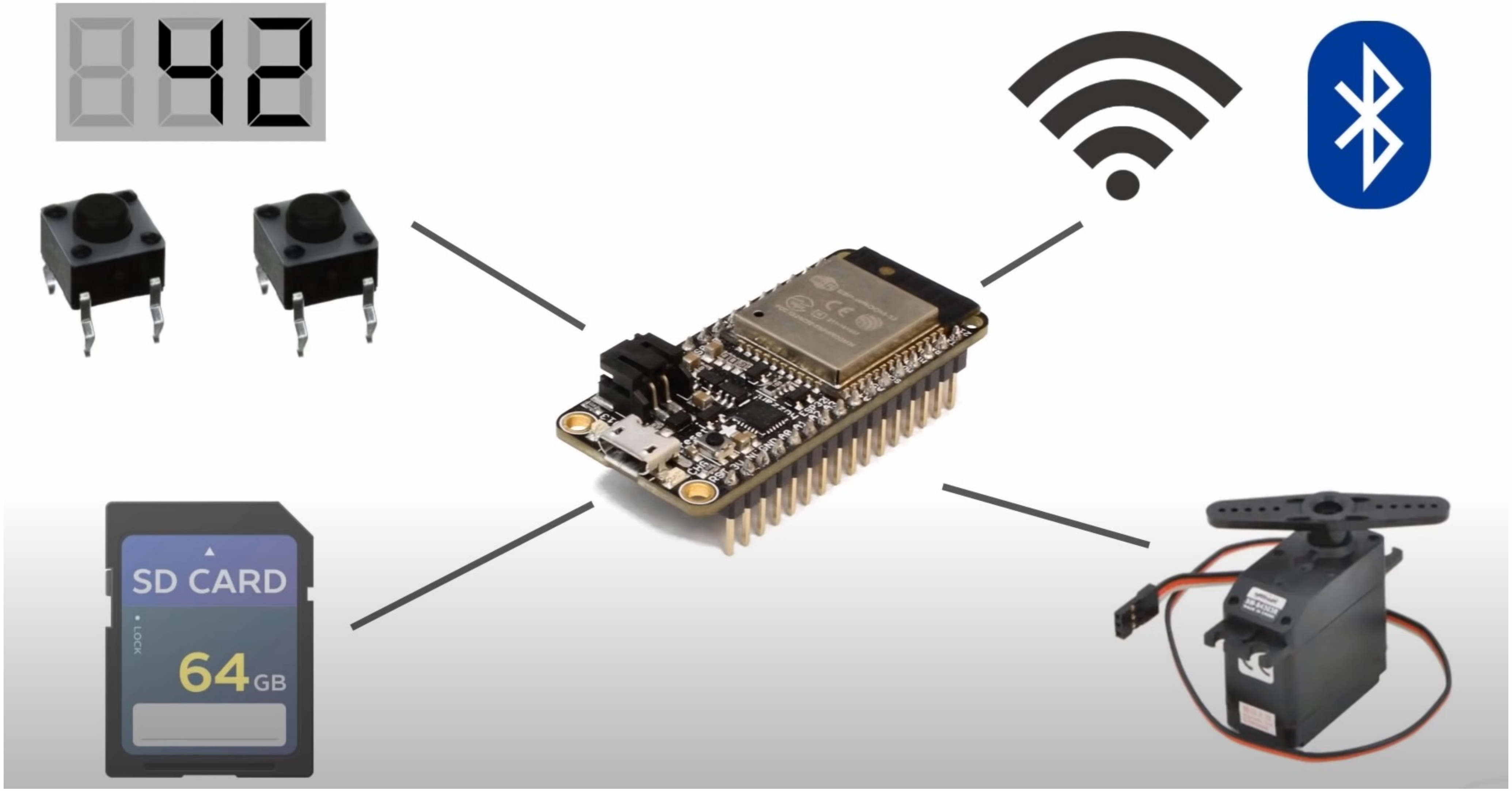
- Currently there is no one OS that rules the MCU market
- Popular IoT OSes
  - ARM mbed, Contiki OS, mynewt, RIOT OS, and Zephyr Project.
- FreeRTOS
  - Distributed freely under the MIT license, ported to many MCUs.
  - Allows the user to set the **priority** of threads (task).
  - Includes an efficient **SW-timer** implementation that **does not use any CPU time unless a timer needs servicing.**
  - Acquired and Maintained by Amazon

IoT Operating Systems

Which operating system(s) do you use for your IoT devices?



ESP32 is extremely capable of running RTOS



Popular IoT Kits

ST MICROELECTRONICS

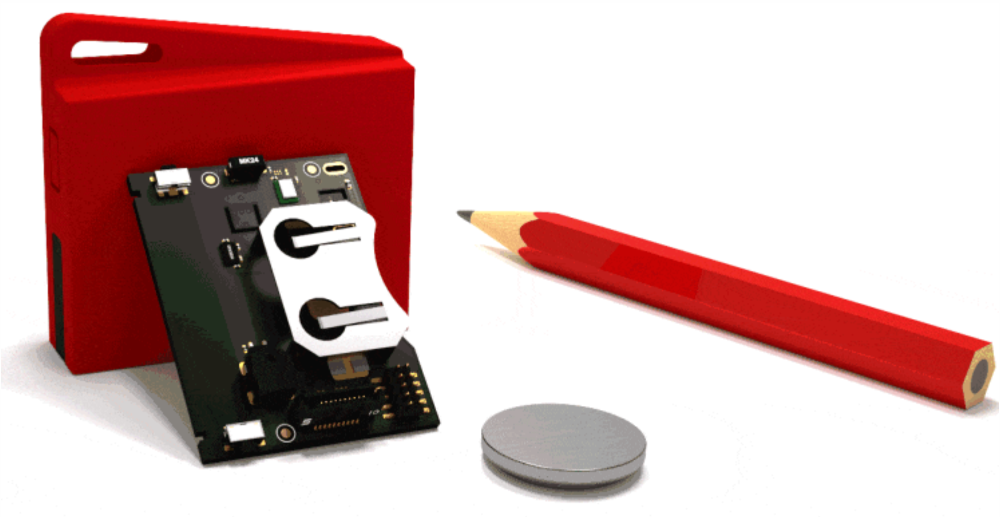
SensorTile kit

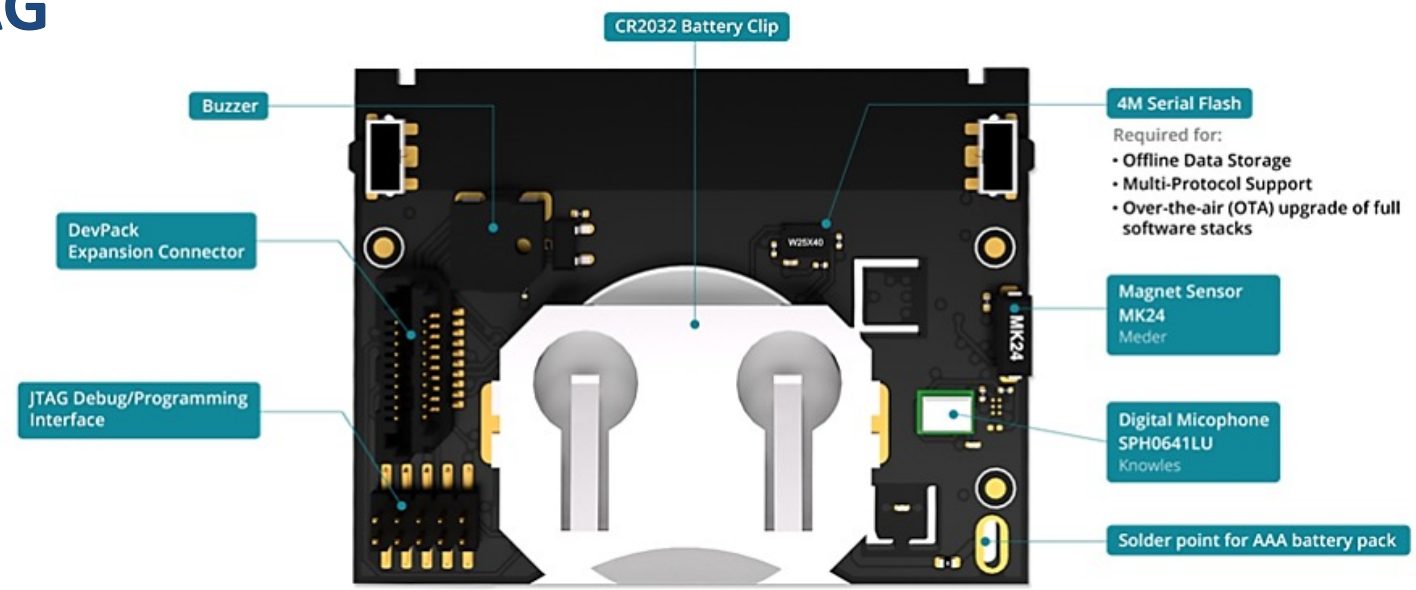
- SensorTile firmware package that supports
  - sensors raw **data streaming** via USB
  - data logging on **SDCard**,
  - **audio acquisition**
  - **audio streaming.**

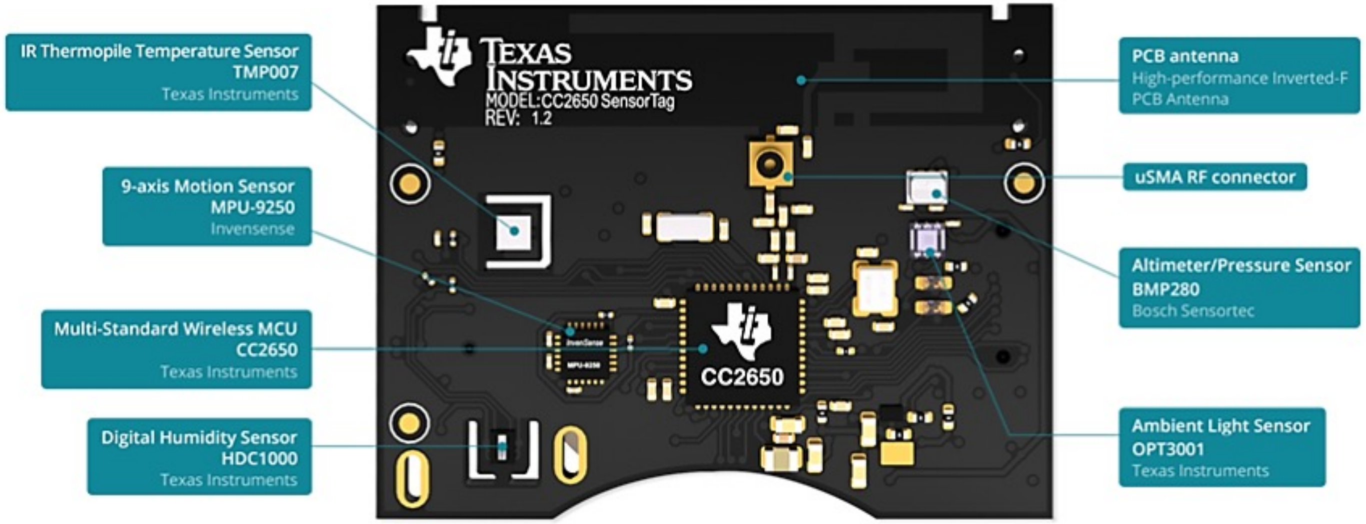
- It includes low level drivers for all the on-board devices.

- iOS and Android demo Apps

TEXAS INSTRUMENT SENSORTAG







More info:

http://www.ti.com/tool/tidc-cc2650stk-sensortag

Demo:

https://youtu.be/zPhjnN0HD2E

NORDIC SEMICONDUCTOR

- Bluetooth 5 and Bluetooth mesh Development Kit
  - nRF52 DK, 2.4 GHz RF, BLE
- nRF module be examined later in a lab exercise
- More Info:

  https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK

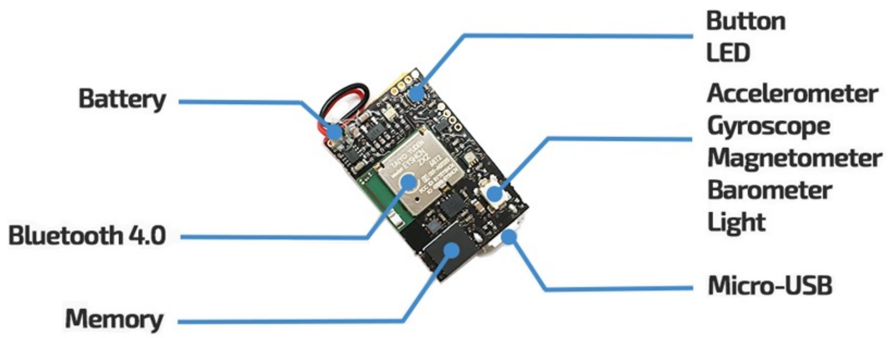
nRF52

- Highly flexible ultra-low power multiprotocol SoC
- 32-bit ARM® Cortex™-M4F CPU with 512kB flash + 64kB RAM
- 2.4GHz transceiver supports
  - Bluetooth Low Energy
  - ANT
  - proprietary 2.4 GHz protocol stack
  - On-chip NFC tag

3 x Master/Slave SPI
2 x Two-wire interface (I^2C)
UART (RTS/CTS)
3 x PWM
AES HW encryption
12-bit ADC
Real Time Counter (RTC)
Digital microphone interface (PDM)

METAMOTIONC, METAWEARC

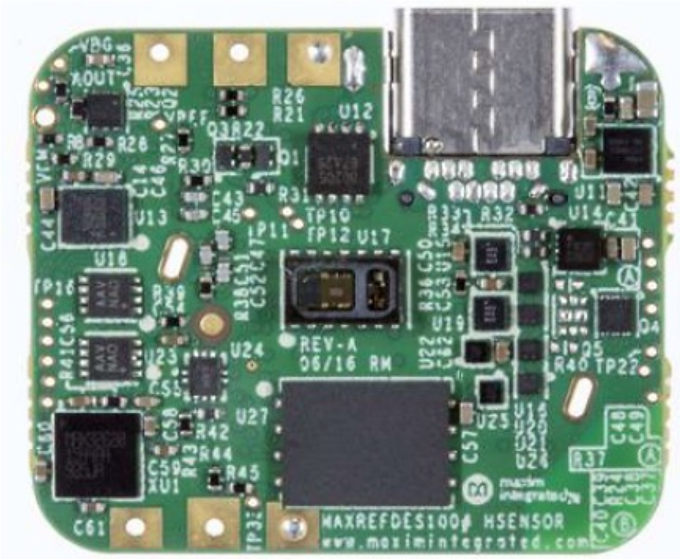
- **MetaMotionC (MMC) and MetaMotionR (MMR)**
  - Real-time and continuous monitoring of motion and environmental sensor data.
  - Logging Sensor w/ 9 Axis inertial measurement unit (IMU) + Sensor Fusion + Pressure + Temp
  - https://mbientlab.com/metamotionc
  - Demo on Kickstarter.
- **MetaWearC**
  - Streaming Sensor w/ 6 Axis IMU + Temp
  - https://mbientlab.com/store/metawearc
- **Lot of stuff available for developers for free**
  - https://mbientlab.com/developers/#documentation



MetaMotionR (MMR)

MAXIM HEALTH SENSOR PLATFORM

- Included sensors:
  - ECG (HR/electrical activity)
  - Photoplethysmogram (PPG: HR/beats per minute)
  - 2x temp. sensor
  - 3-axis accelerometer
  - 3-D gyroscope
  - Barometric pressure sensor



- Both data streaming and logging modes

More Info:
https://www.maximintegrated.com/en/design/reference-design-center/system-board/6312.html

Demo:
https://youtu.be/jiKg-4S4gfs