

Tutorial 3: software architectures (part 1)

Thomas Braipson

October 16, 2026

Theoretical reminder

Architecture	latency of non-urgent operations (in exec time)	latency of urgent operations (in exec time)
RR	entire loop	entire loop
RRI	entire loop + interrupts	interrupts of higher priority + maximum time during which interrupts are disabled
WQ	max of all tasks + interrupts	interrupts of higher priority + maximum time during which interrupts are disabled
RTOS	tasks of higher priority + interrupts	interrupts of higher priority + maximum time during which interrupts are disabled

How to choose the best architecture?

The simplest one that fulfills timing requirements.

Tips:

- ▶ Identify and list the different tasks of the problem.
- ▶ Identify the period and execution time of each task.
- ▶ Identify which tasks can/have to be performed by interrupt routines. (exec time $\sim \mu s$ is ok, $\sim ms$ is not).
- ▶ Estimate the delay induced by each architecture, and choose the simplest one whose delay is acceptable (i.e., smaller than the period+exec time of each task)
- ▶ The following dichotomous key can be useful:
 1. Is preemption needed? Yes $\rightarrow$ RTOS.
 2. Are interrupts needed? No $\rightarrow$ RR.
 3. Is a static CPU assignment acceptable? Yes $\rightarrow$ RRI, No $\rightarrow$ WQ.

Ex. 1 (January 2015)

3. [6 points] The onboard microcontroller of a homemade quadcopter has to receive and execute instructions sent by the pilot via a wireless link (such as taking off, landing, flying in a specified direction, ...). In addition to processing those instructions, it constantly stabilizes the vehicle. In order to monitor and control stability, the quadcopter is equipped with two 3-axis accelerometers and one gyroscope. Moreover, it includes a camera whose signal, along with others monitoring data, has to be sent back to the pilot. The microcontroller has to perform the following tasks:

- receiving and processing instructions: 10 times per second, execution time = 5 ms;
- controlling the motors: 100 times per second, execution time = 2 ms;
- acquiring data from sensors: period of 4 ms, execution time = 1 ms;
- sending the video signal and other data: period of 40 ms, execution time = 20 ms.

(a) What is the best software architecture for this system? Justify.

Ex. 2 (September 2017)

3. A nanosatellite in charge of taking pictures of the earth is controlled by an embedded system. This system is equipped with a radio transceiver that receives telecommand data packets from a ground station. Upon receiving a data packet, the radio transceiver sends an interrupt request to the onboard microcontroller. Such requests are always separated by a delay of at least one second.

The microcontroller has to perform the following tasks:

- τ_1 fetches and processes the data packets communicated by the radio transceiver;
- τ_2 performs computations for estimating the position of the satellite, every 10 ms;
- τ_3 regulates a DC/DC power converter, every 2 ms;
- τ_4 acquires an image from a camera, after telecommand data requesting to perform this operation has been received from the ground station. It then performs image processing operations.

Tasks τ_1 and τ_3 execute in less than 0.1 ms; task τ_2 needs 1 ms. The execution of task τ_4 may require up to 500 ms of CPU time, depending on the image processing operations that must be carried out.

(a) What is the best software architecture for this system? Justify.

[3/20]

Ex. 3 (January 2022)

2. A marine research buoy contains sensors for measuring water temperature and salinity. It is equipped with an embedded system whose processor performs the following tasks:

- Starting a new round of measurements every hour. This operation takes negligible time. The measurements are carried out by a separate data acquisition peripheral, and take a few seconds. An interrupt request is send by this peripheral when the results of the measurements are available.
- Processing the measurements results, which is done each time that they are obtained, and takes up to 500 ms of CPU time. The results are written into flash memory. A write operation takes up to 10 ms, and is performed by a peripheral. This peripheral sends an interrupt request when the write is complete.
- Sending by radio the last processed measurements. This operation should be performed once every 24 hours, and takes up to 30 s.

Since the amount of onboard power is limited, it is important to put the processor in a low-power sleep mode whenever possible.

- (a) What is the best software architecture for this system? Justify carefully your answer. [3/20]

Ex. 4 (September 2025)

2. A monitoring system for an aircraft engine controller is built around a microcontroller that receives data from a large number of sensors, processes this data, and sends periodic reports as well as alarm signals through a communication channel.

The sensors operate asynchronously, meaning that they generate data at unpredictable times, independently from each other. When a sensor has new data available, it sends an interrupt request to the microcontroller, which has to fetch this data as soon as possible. This operation can be assumed to take negligible time. Then, the microcontroller processes the sensor data, which takes between 1 and 20 μs . It is known that in the long run, the microcontroller has enough computing power to process all incoming data and perform all communication duties. However, it may happen that rapid bursts of sensor data force the data processing to be sometimes delayed for a short while. The result of data processing can take two forms: nominally, it consists in updating statistics that are (globally for all sensors) sent every 10 ms through the communication channel. Besides that, if the processing detects a problematic situation, then an alarm message must immediately be sent through the communication channel. Transmitting on the communication channel simply amounts to writing appropriate data to the registers of a dedicated device, which takes at most 1 μs .

Finally, a key feature of the system is that the sensors are classified in decreasing order of importance as being *life-threatening*, *critical*, *essential*, *auxiliary*, *indicative*, ..., the idea being that data issued by more important sensors must as much as possible be processed before that of less important ones.

- (a) What is the best software architecture for this system? Justify carefully your answer.

Homework: try to implement in pseudocode Ex. 3

Implementation issues will be the topic of next week's tutorial.