# Architectures for Embedded Systems

# Real-time Operating Systems
# The FreeRTOS case
# Laboratory assignments

Arnaldo S. R. Oliveira

Academic year 2024/25

Universidade de Aveiro – Dep. de Eletrónica, Telecomunicações e Informática

# Outline

## Real-time operating systems

- General concepts
- FreeRTOS aspects

## Lab assignments

- Introducing tasks and timers
- Adding shared variables and mutual exclusion access
- Scaling the system with more tasks

# Questions for Autonomous Study

**General aspects**

- What are real-time systems? Are “real-time” and “fast” synonymous?
- What is a deadline, and a deadline miss, in real-time systems?
- What is the difference between hard and soft real-time systems?
- How severe is missing a deadline in a hard real-time system? And in a soft real-time system?
- What is a Real-time Operating System (RTOS)?
- What distinguishes a RTOS from a general-purpose operating system?
- Which services and abstractions are typically provided by a RTOS?
- What are the pros and cons of an RTOS-based system (compared to a standalone/bare-metal system)?
- Name (list) a few examples of RTOSs (free, academic, commercial).

# Questions for Autonomous Study (cont.)

- What are RTOS tasks?
- What is the role of RTOS scheduler?
- What are the typical task states in a RTOS?
- What is the purpose of task priorities?
- What is the RTOS tick?
- What are RTOS timers?
- What is a shared variable?
- Why is mutual exclusion of access to shared variables important?
- What are critical sections (in the code)?
- What are queues?
- What are semaphores and mutexes?

# FreeRTOS – Introduction to selected functions

**Search for the purpose and parameters of the following FreeRTOS functions**

- xTaskCreate(...)
- vTaskDelay(...)
- xTaskDelayUntil(...)
- xTimerCreate(...)
- xTimerStart(...)
- xQueueCreate(...)
- xQueueSend(...)
- xQueueReceive(...)
- vSemaphoreCreateBinary(...)
- xSemaphoreTake(...)
- xSemaphoreGive(...)

# Sources of Information

- FreeRTOS (generic)

https://www.freertos.org/

- FreeRTOS Overview (ESP32-C3)

https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/api-reference/system/freertos.html

- FreeRTOS (IDF) (ESP32-C3)

https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/api-reference/system/freertos_idf.html

# Laboratory Assignment 1 – Introducing tasks and timers

- Develop a FreeRTOS-based system that
  - Collects a temperature from the TC74 sensor, every second (in a timer-based function) and computes the average of the last three values
  - Prints the average temperature in the terminal whenever the user presses a key (this functionality must be implemented in a separate RTOS task)

# Laboratory Assignment 2 – Adding shared variables and mutual exclusion access

- Create a copy of the previous program and add the following capabilities:
  - Timestamp every temperature read from the sensor (suggestion: define a data structure with a timestamp and a temperature)
  - Keep the maximum and minimum temperatures read from the sensor and corresponding timestamps
- Depending on the key pressed, print on the terminal
  - The average temperature and corresponding timestamp
  - The maximum temperature and corresponding timestamp
  - The minimum temperature and corresponding timestamp
- Important note: ensure safe access to the shared variables (structures) containing the timestamps and the temperatures

# Laboratory Assignment 3 – Scaling the system with more tasks

- Create a copy of the previous program and add tasks to
  - Read the temperature from the TC74
  - Read the temperature from the BME280
  - Read the temperature from the LM335
- The temperatures displayed on the terminal must be the average of the temperatures read from the three sensors

# Final Remarks

- At the end of this week, you should be familiar with the:
  - The basics of real-time systems
  - RTOS main features, advantages, and limitations
  - FreeRTOS capabilities and programming
  - Utilization of FreeRTOS in systems with sensors and interaction with the user

- Additional topic for discussion:
  - Analyse the approach and the effort required to implement the previous assignments in the case of a standalone/bare-metal system (no RTOS)