Università degli Studi dell’Insubria
Dipartimento di Scienze Teoriche e Applicate

Real-time software development
Exam project 2020-21

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
luigi.lavazza@uninsubria.it

Note

- This description concerns a simplified system
- It is possible to modify this description if
  - You have a valid reason (e.g., the given description is incomplete in some respect)
  - You have discussed the modification with the teacher, who approved it

Context: metro train control

- The metro train controller has to
  - Respond to emergency brake requests
  - Enforce stop commands by stop signals
  - Activate/deactivate regular brake commands issued by the driver
  - Increase/decrease speed according to the driver’s commands
  - Manage communications between from the train traffic managers to the driver

Inputs: braking and accelerating commands issued by the driver.

- The driver operates a lever that has 7 positions:
  - +3: maximum acceleration
  - +2 : medium acceleration
  - +1 : minimum acceleration
  - 0: no acceleration and no braking
  - -1 : minimum braking
  - -2 : medium braking
  - -3 : strong braking

- Positive positions indicate acceleration requests (i.e., they control the amount of power provided by the motor)
- Negative positions indicate breaking (with different braking strength)
- Position 0 indicates that neither acceleration nor braking are exercised.

Inputs: braking and accelerating commands issued by the driver.

- The positions of the control lever are communicated to the controller as inputs via pins 2 to 8 of GPIOB.
  - Pin 2 is position -3, pin 8 is position +3
  - Note that at any moment, one and only one pin is 1 (since the lever is in one and only one position at any time).

- For regular operations, these commands must be read every 10 ms.

Inputs: stop signal

- The signal is asynchronous and unpredictable
- When this request is received, the train must stop “gently”
  - Motor acceleration is set to zero,
  - Brakes are activated at medium force
    - corresponding to lever in position -2.
  - This request has greater priority than the commands issued by the driver. Therefore, the position of the lever is ignored.
- After stopping because of this signal, the train must not restart until the stop signal is active. When the stop signal is cleared, the train can resume normal operation. However, to this end it is first necessary that the control lever is set in position zero.

- The stop signal is an input received on pin 1 of GPIOB

Inputs: emergency brake request

- This request is asynchronous and unpredictable
- When this request is received, the train must stop as soon as possible.
  - Motor acceleration is set to zero,
  - Brakes are activated. The maximum braking force is applied.
- This request has the maximum priority. It must be satisfied immediately, whatever the current state of the train and the controller.
- Maximum braking force is maintained indefinitely after the signal is received. The system needs to be manually reset after such event.

- The emergency brake request is an input signal, received on pin 0 of GPIOB

Inputs: messages from traffic management center

- The traffic management center can send communications to drivers via a serial line.
    - USART1 (GPIO A) is used to receive messages, which are shown on a display
- Managing these communications has the lowest priority.

Outputs: Braking

- There are 4 braking force levels
- These are communicated to the braking systems via pins 8-12 of GPIOC.
    - Pin 8 on means minimum braking force
    - Pin 12 on means maximum braking force
- Pins 8-11 are used to report the position of the braking lever.
- Pin 12 is used only for emergency braking.

- Outputs must be configured as push-pull.

Outputs: Acceleration

- There are 3 power levels, corresponding to the three positive position of the lever.
- These are communicated to the motor via pins 0-2 of GPIOC.
  - Pin 0 on means minimum power
  - Pin 2 on means maximum power
- Outputs must be configured as push-pull.

Controller's responsibility

- Implement commands and signals, show the incoming messages.
  - In normal conditions, the position of the lever (input) is simply converted in the corresponding acceleration and braking outputs
  - However, for security reasons, maximum acceleration must never be maintained for more than 4 consecutive seconds. When this condition happens, acceleration output must be set to 2, even though the lever is on position 3.
- Ensure that the train is in a consistent state
  - with respect to the given requirements
  - With respect to obvious considerations: e.g., the controller should never ask the motor to accelerate and the brakes to brake at the same time.

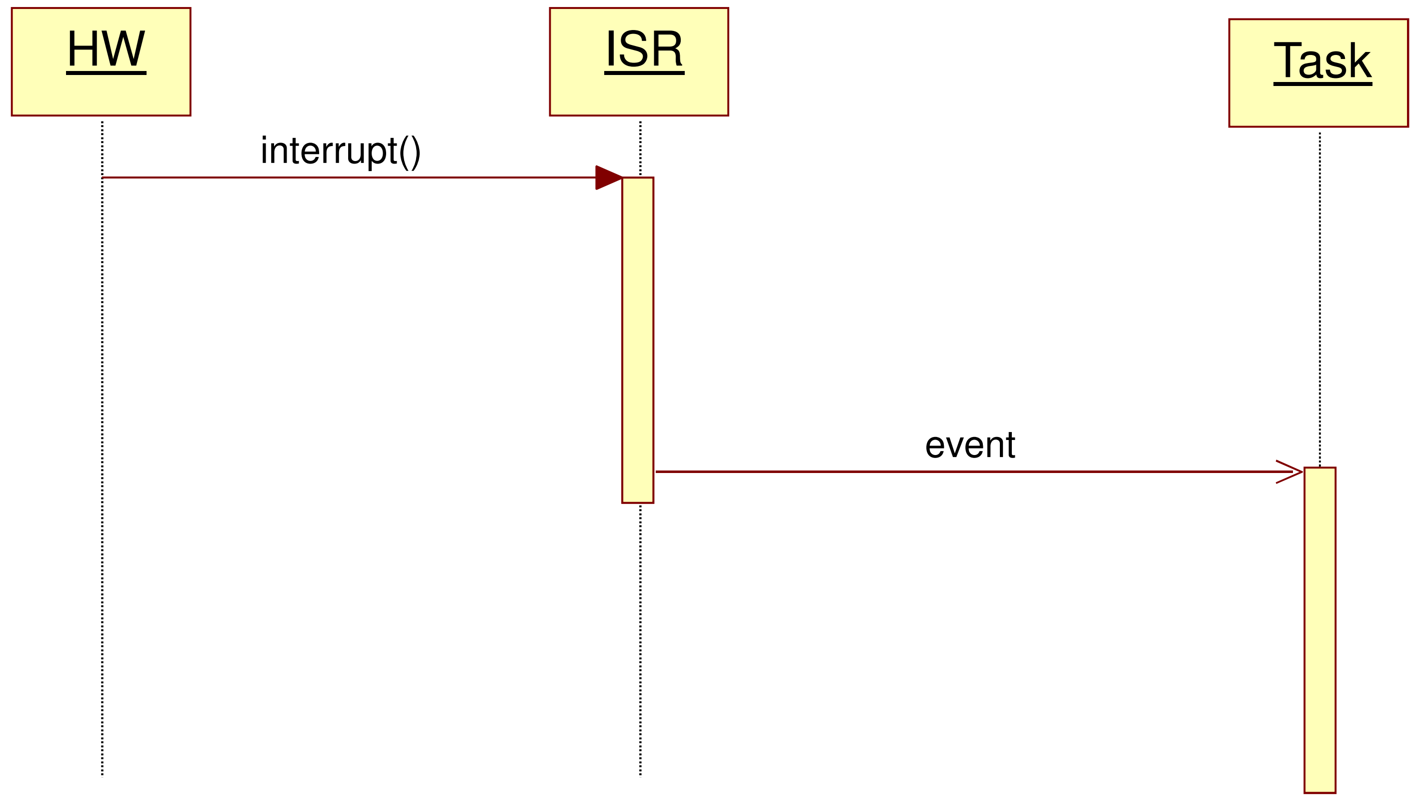
Suggestions

- Implement the system in incremental steps.
- E.g.,
  - First implement the execution of driver’s commands
  - Then add the emergency braking
  - Etc.

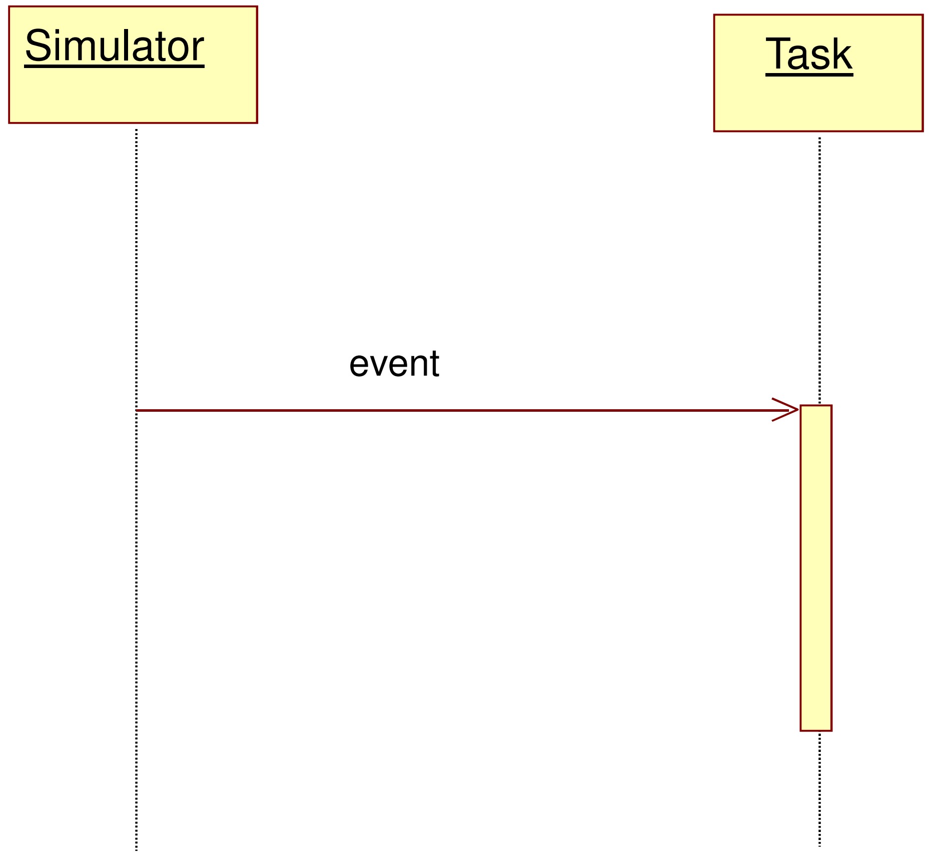
Suggestions

- To test the system, add a task that
  - Has maximum priority
  - Generates events that have the same meaning as the expected inputs
  - The task reads form an array of ⟨event, taskIds ,delay⟩
  - In a loop (executed until the array is finished)
    - Reads a triple ⟨events, taskIds ,delay⟩
    - Sends the event to task taskId
    - Waits for the specified delay
- In this way, you do not have to provide inputs manually. Instead, you can prepare a sequence of events that are expected to exercise a given behaviour of the system, and see if the expected behaviour is actually shown by the system
- This techniques works if the system is made of tasks that are waked up by events. No input is provided to the system, hence ISRs are not executed. If they perform some other operations, in addition to generating events, this technique does not work.

Real system



Simulated situation



What you must do

- Specify the system via UML
- Design the system and describe the design via UML
- Implement the system using µVision (microprocessor STM32F10x, as usual)
- Test the system and document tests and results.

Deliverables

- A pdf file containing the documentation
  - Specifications
  - Design
  - Test cases and results
- A zip file containing uVision working directory (source files, etc.)
- A readme.txt with installation and execution indications, and any other useful or necessary information.
  - After placing the project directory in the directory where other uVision code is usually placed, the system must work.
  - If some <u>minor</u> adjustments are needed, they must be clearly indicated

- Send everything to the teacher via email
- Indicate clearly who are the authors of the project
  - Remember that the project can be carried out by pairs of students