Description of Work nº 1	
Course	**Real-Time systems / *Sistemas de Tempo Real***
Year	2018/2019
Aim	Study concepts of Real-Time Systems and develop a project with a real-time kernel
Classes	5 classes x 3 hours + 12 extra hours (outside)
Delivery Date	19/10/2018

Concrete Objectives:

1. Interaction with physical systems (Interface boards, sensors, actuators, …)
    a. Interface board (Data acquisition)
    b. Sensors
    c. Actuators
    d. Interface functions (in C++)

2. Try the base real-time system's concepts, namely:
    a. Task / Process: concurrency,
    b. State, event, actions (triggered by events)
    c. Synchronization and communication between tasks
    d. Accessing shared resources / exclusive access resources

3. Using Real-Time kernels
    a. freeRTOS
    b. C/C++ Language

4. Solve the problem described in Annex 1.

5. Answer the questions in Annex 2 until the deadline date. This annex is provided later, as a word document or online questionnaire.

Anexo 1* – *Description of Work

This Lab-work aims at training the real-time concepts with a real-time kernel. The work comprises the development of a program to perform the management of an **Automated Storage of Perishable Products (ASPP)**, using real-time approaches. Each product stored in the ASPP must be identified and assigned with a "best before"/"expiry date", both using the keyboard. For the sake of simulation model each day as one second (e.g. 60 day → 60 seconds).

The tools necessary to develop the work comprise a real-time kernel (freeRTOS), the Visual Studio C/C++ compiler and the hardware kit that represents the warehouse, illustrated in fig. 1. Other tools available for developing this work are: a storagesimulator; a set of low-level functions for software/hardware interaction, provided in slides.

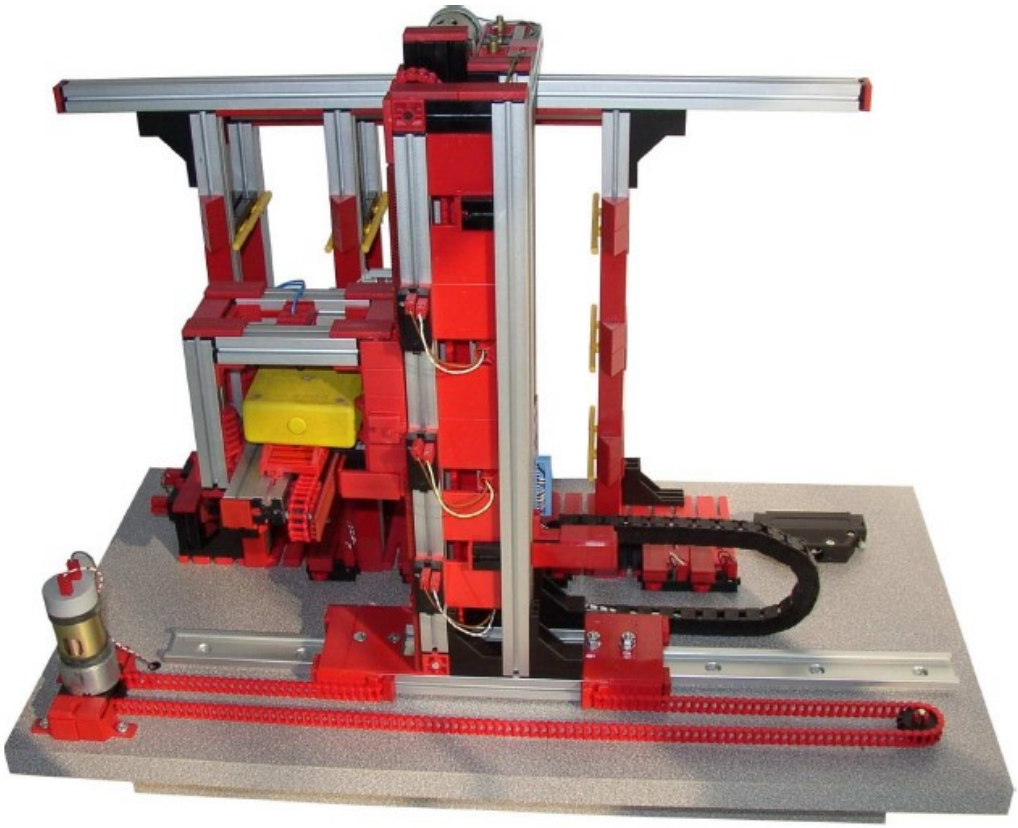

Fig. 1 – Warehouse Kit

The technical characteristics of the hardware kit, representing the storage, can be obtained from its document that can be downloaded from http://www.staudinger-est.de/en/simulation/compact-models/documents/226001.pdf. The interaction between the real-time program and the warehouse is performed through an input / output digital data acquisition board, specifically, the NI USB 6509, as shown in Fig. 2.

Fig. 2 – the NI USB 6509 data acquisition board

The ASPP system must be managed according to the requirements stated in table 1.

Table 1 – Needed requirements

Req.	Description
	Storage utilization requirements
FR1	Using the keyboard, develop a robust (semi-automatic) storage calibration. Using keys, the user defines the direction of the movements. During a movement, the actuator must stop when it reaches a position sensor (sensor state going from 1 to 0).
FR2	Store a product in the storage. Ask for product name (or ID) and expiration time. The system should know which cells is empty.
FR3	If the up_left_button is pressed for 5 seconds, then take all **expired** products from storage. Up_ light should flash with 1 second period until discharge finishes.
RF4	Retrieve a product from the storage, given its name/id with the keyboard.
FR5	While being operated (moving), if both left buttons are pressed, then perform emergency STOP
FR6	During an emergency STOP, if up button is pressed, then RESUME operation
FR7	During an emergency STOP, if down button is pressed, then Reset the system (perform semi-automatic)
FR8	Both lights flash faster (0.5 seconds period) during an emergency STOP (and until resume)
	Information retrieval requirements
FR9	Show list of stored products and expiration time.
FR10	Given a product ID with the keyboard, shows the (x,z) coordinates and expiration time
FR11	Shows the list of all expired products only
	Non-Functional requirements:
NF1	While operating, the system can accumulate Requests
NF2	While operating, the system can provide information (e.g. occupation / free cells / products list)
NF3	The system works well (one product per cell, not taking products from empty cells)
NF4	The system always moves within limits x, y, z
NF5	The system moves along (x,z) only when y is stopped at center position (y==2)

Given that there is a single and very expensive warehouse for all the classes, a simulator for the warehouse is also provided, as illustrated in fig. 3. This allow "using" the kit without annoyingly

waiting for the availability of the real one. Furthermore, the simulator allows the development of the program, as well as testing and error correction, before its use in the real kit. These are best practices in industry (like in manufacture, robotics…), so we suggest you get used and enjoy this design way.

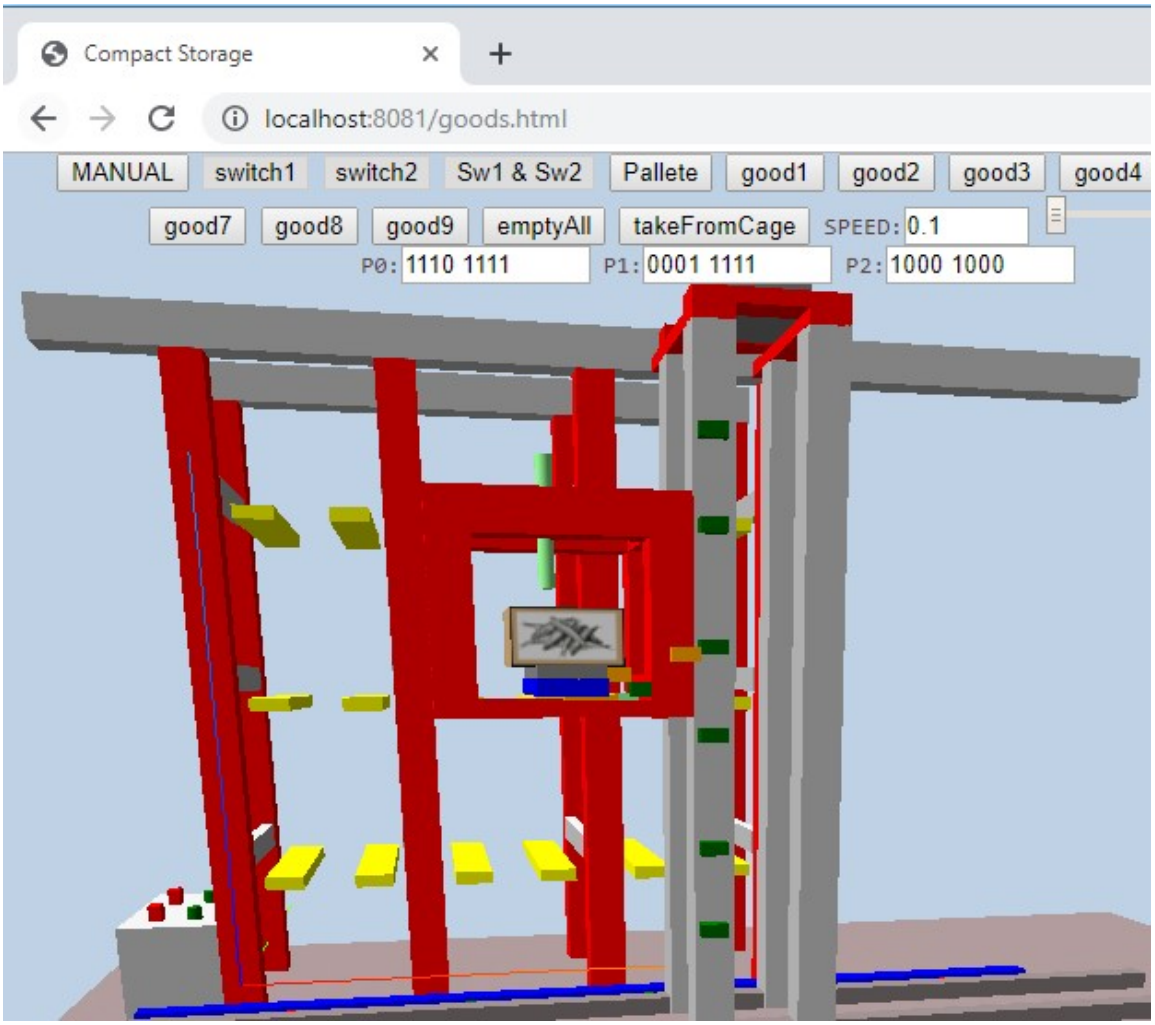

Fig. 3 – Warehouse Simulator

It is recommended that you start creating the low-level code to read sensors and activate actuators. Afterwards you should progressively move to higher-level functions.

Obligatory steps for lab work evaluation

- Each team must fill the **Reports form** that is available in Moodle, which is necessary for the project evaluation. The form can be edited as necessary till the submission deadline.
- Upload a zip/rar archive with your Visual Studio Project (before compacting, run the provided del.bat script to reduce the size of the compacted file).

Classes plan for the following weeks

Class 0:

Installation of software tools and creation of a Visual C++ project.

Class 1:

Interaction between a program and physical systems, through sensors, actuators and a Data acquisition board; Development of the functions for the interaction with the warehouse.

Class 2:

Development of real-time behavior; tasks and semaphores; development of the low-level requirements

Class 3:

Communication mechanisms (mailboxes); development of the high-level requirements to control the warehouse.

Class 4:

Continue develop high-level requirements to control the warehouse.

Class 5:

Conclude the lab-work.