epsi

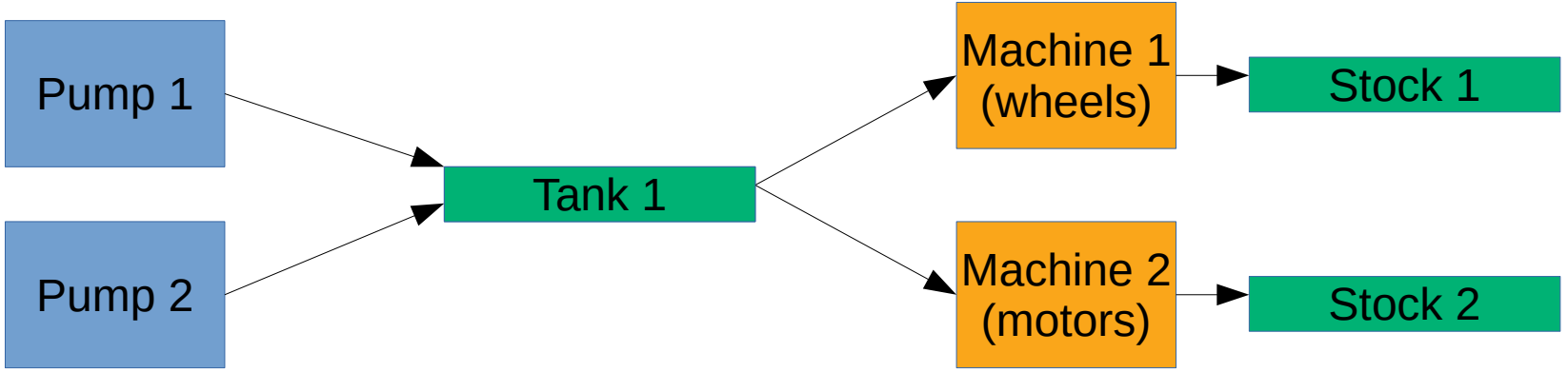
Conception d'une solution embarquée en temps réel

Today organization

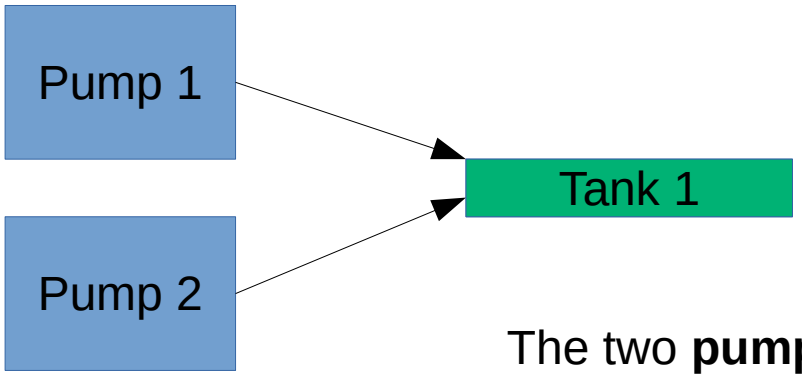
Date	Contenu séance
14h-16h	Examen sous forme de TP
16h-17h	Evaluation écrite

Programming exercice

We are going to program (in python or other language if you want) the tasks that will run a factory. The general idea is that the two pump harvest oil that can be used by the two machines.



Pumps and tanks

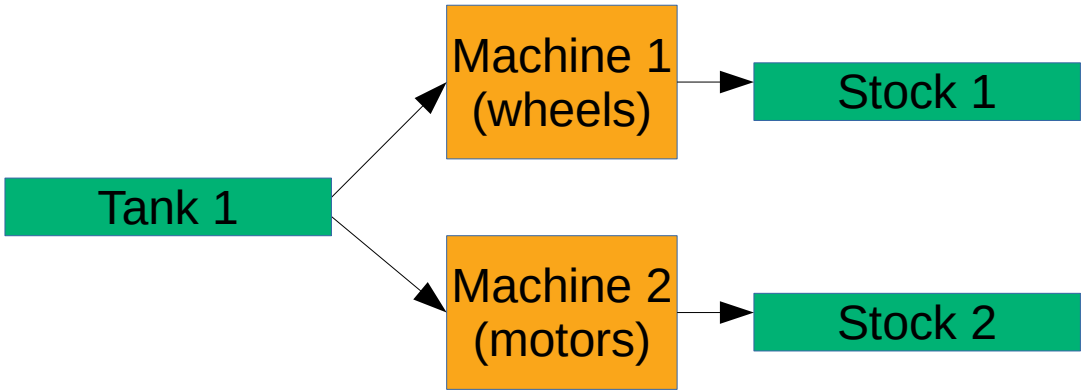


The two **pumps** have different capabilities in term of :

- Period : when the pump can be used.
- Execution time : for how much time it runs
- Production : the **amount of oil** it can extract

The **tank** is used to store oil. It has only one specification : the maximum quantity it can store. If a pump try to add more oil into a filled tank **it will be wasted**.

Machines and stock



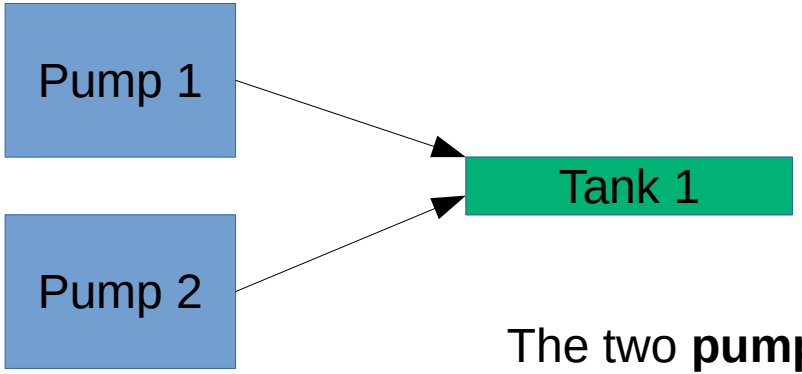
The Stocks are used to store the product of the two machines. Those stocks are **unlimited**.

The machines each need some amount of oil to run and produce **one product** that will then be **added to the stock**. “Machine 1” produce wheels as “Machine 2” produce motors. At the end we want to have the maximum of

[1 engine + 4 wheels] available.

Of course this fabrication will have a maximum period and an execution time.

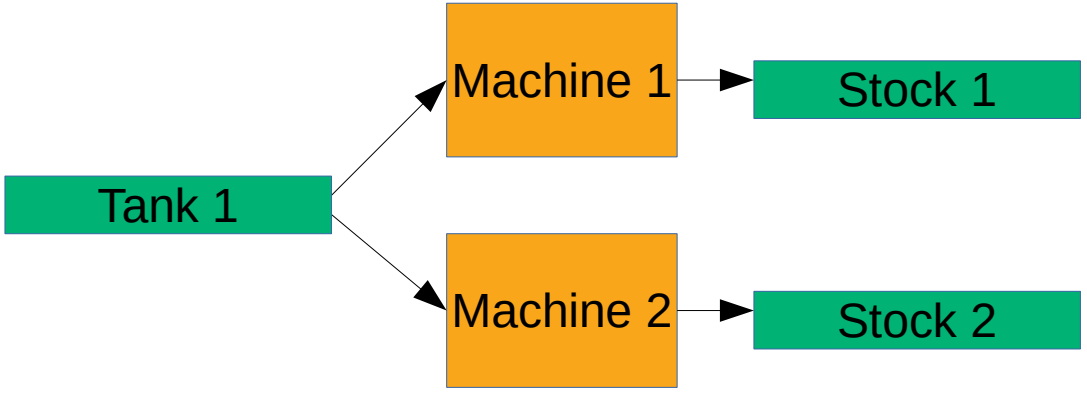
Implementation :Pumps & tanks



The two **pumps** will be two independent tasks described in following specifications.

The **tank** can be represented by a Fifo/Lifo. You can just use a global variable in python.

Implementation: Machines & stock



The Stocks can be implemented by two global variables. You need to print its content to follow production.

The machines are implemented by tasks. These tasks start by checking if there is enough oil available. If yes it removes the necessary amount of oil, execute itself and then put the product in the stock (+1 each time).

epsi

Scheduler

The scheduler you will design and program must build the more [1 motor + 4 wheels] it can.
For that it can implement the following rules :

- If the tank is full, Pumps should have a really low priority.
- If [nb wheels / 4 > nb motors] the Machine 1 will have a higher priority.
- If [nb wheels / 4 < nb motors] the Machine 2 will have a hight priority.

Implementation : Scheduler

As you understood priority will probably change among time. You are free to implement the dynamical scheduler you want (no pre-defined order).

Only one task will be run at the same time : **no multi-threading** !

This scheduler **will not allow pre-emption of tasks** !!!

The system is pure **soft real-time**.

This scheduler will be evaluated by the **amount of engines** it can produce in 2 minute (use datetime.datetime.now().minute). For your test start with one minute.

System specification

	Period	Exec time	Job
Pump 1	5	2	Produce 10 oil
Pump 2	15	3	Produce 20 Oil
Machine 1	5	5	Needs 25 oil Produce 1 motor
Machine 2	5	3	Needs 5 oil Produce 1 wheel
Tank	NA	NA	Can store 50 oil

You can imagine here that the “period” is the time that needs a pump or an engine to cool down, so the time between it runs and the next time it will be able to run (EDF will not work).

epsi

What will the scheduling algorithm look like ?

For each task in my task list :

    If the rules don’t allow my task to be executed : discarded

    Then I decide which task has the more priority regarding other rules

I run my task

Task states : waiting, blocked, running

What will you deliver ?

- A document sent by email to alexandre.harly@mail-formateur.net with title : “Temps réel – NOM Prénom”

- Content of this document :
    - The Github link to your program in python
    - A log of your results (trace in terminal)
    - A pseudo code description of your algorithm
    - The number of [motor + 4 wheels] you produced by minute

How to proceed

1) Read instructions (10 min)

2) Take a piece of paper and draw a fast timeline of your tasks in order to see what is happening (5min)

3) Decide what you are going to use for your tasks (function or thread…) (1 min)

4) Take source code from a previous session and run it on your computer (2 min)

5) Start modifying the tasks to implement period, execution time and behavior (15 min)

6) Run these task with any scheduler to check it works (15 min)

7) Now start modifying the scheduler to improve your results (45 min)

8) Write your results analysis (15 min)

9) When satisfied with your work or when the time is over, put your results on Github and MyLearningBox (5 min)

Keep calm and work well !