# Architecture

Clamp:

The original idea with the clamp was to be able to grab several balls at the same time. Then, since we do not have a lot of motors available, the goal was to design a clamp that do not need any motorisation to size the balls. We ended up with this clamp (picture below). We use the tension of the rope to do the grabbing. The main advantages are that the balls do not need to be in a particular place to be grabbed (remove a lot of indeterminism). Hence the behaviour is very consistent.

The robot:

The robot is mostly symmetric.

We put the sonar sensor on the side. Its position is the key element of our strategy.

Ev3:

We put it here since it was easy to access it both to branch cables (sensors, motors and the alimentation) and to use the buttons.

Arm of the robot:

The design of a scissor-lift-like arm enable us to have a very good precision when lowering or lifting the clamp. A total of about 3500 tacho counts were needed to lower it.

The main drawback with the design of the arm is that some mechanical pieces are stressed a lot when weights are added to the clamp. In addition, when the motors are overloaded, they lose some accuracy (tacho counts).

# Architecture team

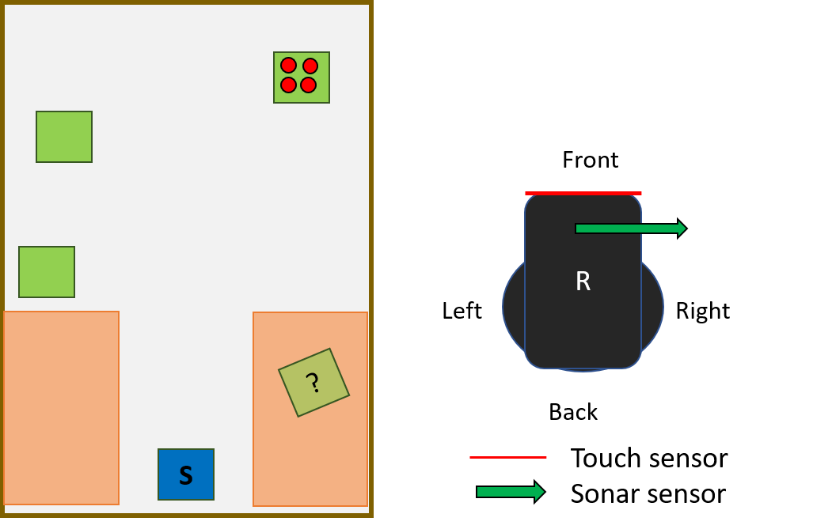
I built the main architecture (the base of the robot and the scissor-lift clamp) of the robot and then Nicolas optimized it (tuned the clamp, improved the stability and the solidity, …).

At first, I thought about using a poly-articulated arm, but the movement of the clamp would have been rather complicated. Using just a simple clamp and using the robot to position the clamp at the right position (a colour sensor to detect a ball) appeared to me like an easier option. I finally ended up with the idea of a clamp that do not need any motor to size a ball, in addition to be able to catch several balls at the same time.

Since getting familiar with the sensors (lot of tests and empirical tunning) may require a lot of time and complexity in this context, I tried to look at an easy way to use them without introducing too much indeterminism. It really pushes me to understand how to use sensors, how to couple them and how to deal with their inaccuracy.

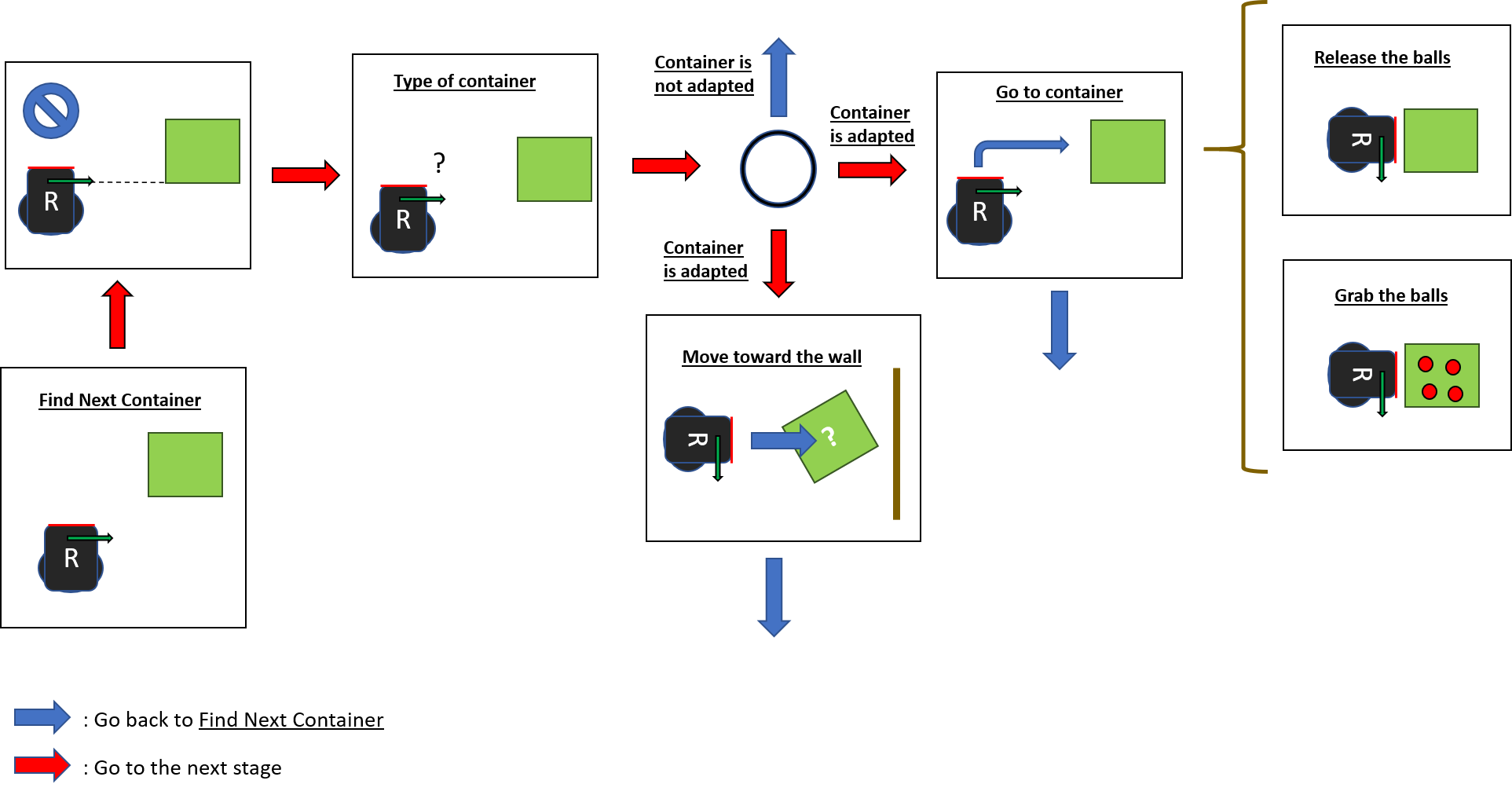
# Algorithm

Warehouse game:



The strategy is based on “replication”.

An illustration of the strategy is provided below:



The unknown cube can have a random angle compare to the others, thereby to apply our “replication” approach, we program the robot to push the cube toward the wall to make it parallel to it. From there, we just need to move the robot backward, make a quarter turn, move backward again and then we can go anew to the first step of the process but this time we know that the unknown cube is in a right position – it will be considered as normal cube.

This approach allows to use the sensors and the robot in a very simple manner. Hence, the behaviour is consistent. In addition, we can update the model quite easily and change the strategy according to our needs.

# Algorithm team

I have designed, coded the strategy, and made the code integration. Nicolas helped me to debug the whole file.

At first the algorithm part was quite challenging. The fact that we cannot meet in person and work physically with the robot make the whole process harder. I tried to find a strategy that would be simple to explain, debug, test and collaborate on. The simple idea that came to my mind was “replication”. I tried to design a way for the robot to always do the same thing when it spots a container.

At the beginning of the project, Brehima did not know how to access the position of the robot so I tried to figure out a way to get around this. I finally came up with the idea of putting the sonar sensor at a side of the robot and the touch sensor at the front. I have learned through this, how to use in an efficient way “flags” to code the strategy. With this approach the robot can adapt to several configurations. It was a plus since we did not at the time how the arena would be.

