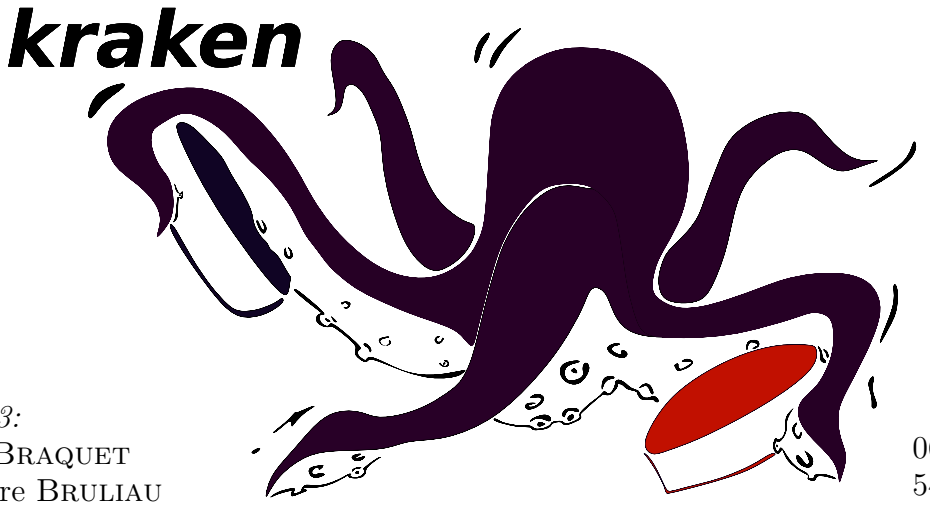
Ecole Polytechnique de Louvain

ELME2002 - Project in mechatronics

Final report – Kraken



Groupe 3:

Martin Braquet	06641500
Alexandre Bruliau	54561500
Miguel Freixo	59671500
Valentin Sokolow	22291300
Arthur Alves Tasca	59891800
Antoine Winant	36311500

June 14, 2019

Academic year 2018-2019

Contents

1 Acknowledgement **3**

2 Introduction **4**

3 General presentation of the robot **4**
3.1 Broad overview . . . 4
3.2 Strategy . . . 8

4 Description of the different functions **9**
4.1 Mobility . . . 9
4.1.1 Low-level control and actuation . . . 9
4.1.2 Path planning and opponent avoidance . . . 10
4.1.3 Localization . . . 13
4.2 Mechanisms . . . 14
4.2.1 Pushing on the ground . . . 15
4.2.2 Grabing atoms on the distributor . . . 15
4.2.3 Pushing atoms in the accelerator . . . 16
4.2.4 Pneumatic circuit . . . 16
4.2.5 Suction pads & vacuum generation . . . 17
4.2.6 Pneumatic command . . . 18
4.2.7 Elevator . . . 19
4.2.8 Accelerator deployment arm . . . 20
4.2.9 Experiment . . . 21

5 Electronics **23**
5.1 Global architecture . . . 23
5.2 Boards design . . . 24
5.2.1 Power board . . . 24
5.2.2 Interface board . . . 24
5.2.3 Emergency board . . . 26
5.2.4 Pneumatic board . . . 26
5.3 Power management . . . 27
5.3.1 Batteries dimensioning . . . 27

6 Programming **28**
6.1 Description of the general structure of the program . . . 28
6.2 Software on the Raspberry PI . . . 28
6.2.1 Software functions for the robot . . . 29
6.2.2 Software functions for the experiment . . . 31
6.3 DE0-nano . . . 32
6.3.1 Hardware functions for the robot . . . 32
6.3.2 Hardware functions for the experiment . . . 32
6.4 LT24 interface and software on a NIOS 2 . . . 32
6.5 Wifi communication . . . 35
6.6 Android application . . . 35

7 Final assessment **36**
7.1 Pros / Cons of Kraken . . . 36
7.1.1 Mechanical and general aspects . . . 36
7.1.2 Hardware and electronics . . . 36
7.1.3 Software aspects . . . 37
7.2 If we had to do again . . . 37
7.3 Evaluation of the group . . . 37

8 Conclusion **38**

A Algorithm of triangulation **40**

B Detail of pneumatic components **41**

1 Acknowledgement

We are really grateful to everyone who has participated, up close or far away, to the realisation of this project. This includes, of course, the technical staff of the project, namely:

- Caroline Bernier: Project coordinator
- Léna Vanthournhout : Mechanics and Design
- Guillaume François: Mechanics, Programing and Minibots
- Christophe De Gréef: Mechanics, Programing and Minibots
- Ludovic Moreau : Digital electronics
- Thierry Daras : Analogic electronics, selection of electronic components and sensors and purchase of electronic components
- Benoît Herman : 3D printing and purchase of Misumi components

We also want to thank the supervising teachers: Renaud Ronsse, Benoit Legat and Bruno Dehez.

Without forgetting our sponsors who allow us to work in the best conditions

Finally, we wish to thank the other three groups of UCLouvain for their mutual help and their good spirit.

2 Introduction

In this section, Antoine drafted the text and Miguel revised it to improve its integration to the whole document.

In September we started a hole project aimed to participate to the belgian robotic cup. From this point we have begin to design a robot that could solve some imposed tasks by the constest. The first semester was oriented on it conception while we build it over the second semester. We finally participate to the robotic cup to test our robot.

This report reflects the final version of the Kraken. It is complementary to reports written previously at earlier stages of the project but do not require them to be understood. It begins with a quick description to remind the main components of the Kraken. It includes also the strategy followed for the robotics cup that influenced our design. After this broad overview, each function will be described and tested. The tests should proof and convince that each solution proposed is appropriate and efficient to complete the task described in the strategy. We will then describe the electronic circuit used to actuate the mechanism and to power the brain and the sensors. So far we explained what was working and what components where needed however this works thanks to a complex program written in the brain of the robot. This is the topic of the next section, programming. We will talk first about the software that runs on the raspberry PI but also on the dedicated hardware implemented on the FPGA to interact with the sensors. To conclude we will explain what were important features of the Kraken while in the mean time we will explain what we would improve and change if all was to be done from the start again. Thus this report briefly summarise one year of work on the same project and highlight the results obtained.

3 General presentation of the robot

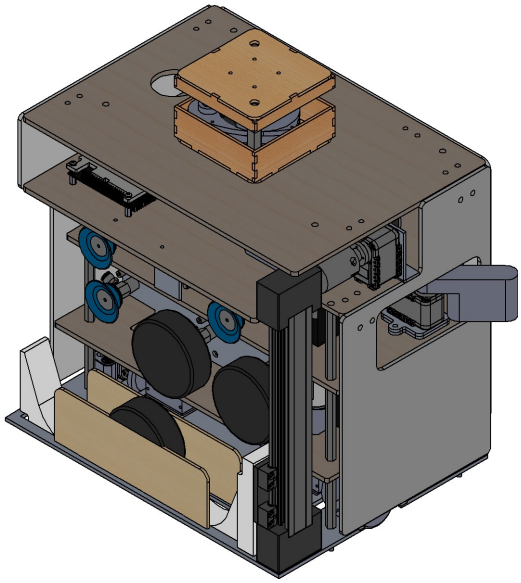
In this section, Miguel and Arthur developed the scientific and technical material; Alexandre drafted the text; and Antoine revised the text to improve its integration to the whole document.

3.1 Broad overview

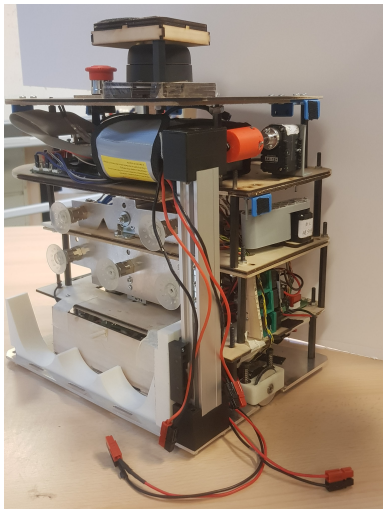
Kraken is decomposed in five layers, connected by rods, in which the actuating, processing and sensing components are fixed. Connected to the first and last layers there is also an elevator system, responsible for rising the atoms between the different levels of the playground. Therefore, the second and the third floor are smaller than the others so that the lift can move from one layer to the other. On its side, a rack is screwed and it holds the atoms. The first floor between the plate of wood is a storage, the second is the level from where the atoms are taken and the third is where the atoms are place in the accelerator.
The mains characteristics of Kraken are detailed in table 1.

Characteristic	**Value**
Total height	0.4 [m]
Nbr of dynamixel	2
Deployed perimeter	1.3 [m]
Undeployed perimeter	1.118 [m]
Nbr of batteries	2
Nbr of pistons	3
Weight	6.3[Kg]
Sensor	Lidar + Odometer/Encoder

Table 1: Kraken's main characteristics



(a) Kraken 3D model



(b) Kraken physical robot

Figure 1: Comparison and changes in strategy

Below is a description of the 5 layers:

1. The first floor is composed of two wheels driven each by one electrical motor. There is also two odometers but they aren't not linked on the same shaft than the wheels. To have a better stability, we added two free balls (one at the front and one at the back of the robot). Respecting to the two batteries, we finally decided to stack them and to move them at the fourth stage for a room and pratical aspect. There is also an important part of the hardware circuit (the power and can board) fixed under this stage (not visible).

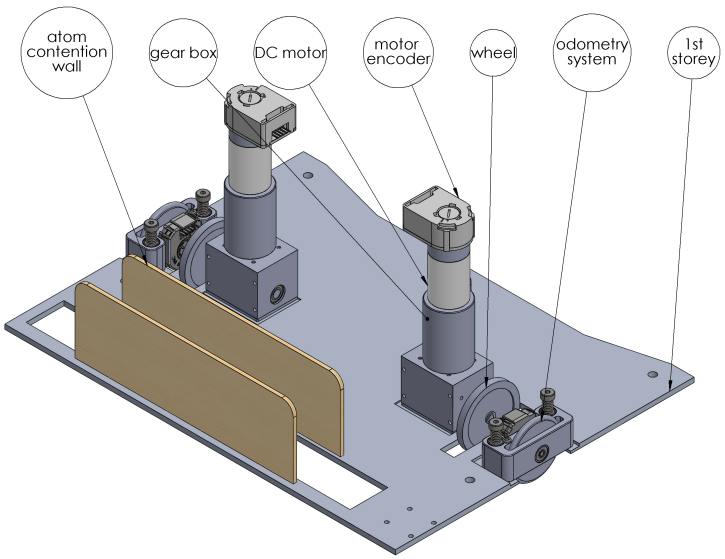


Figure 2: Ground 1

2. The second floor is composed of our first pneumatic actuator (double stroke) linked to two solenoid valves, three vacuum suckers working thanks to three electrical pumps. All of these elements are controlled through the pneumatic card. We also have one pressure regulator to assure that the comprimed air contains in the tank stays always below 4 bars.

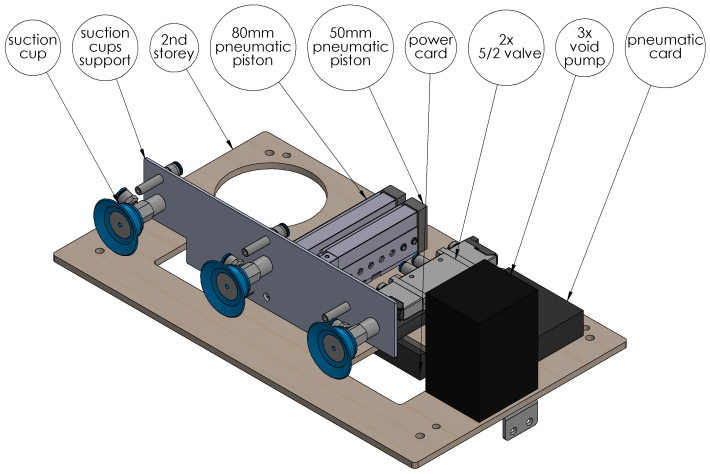


Figure 3: Ground 2

3. The third floor is composed of our second pneumatic actuator (single stroke) actuated by only one solenoid valve. There are also two vacuum suckers working thanks to two electrical pumps. These elements are also controlled through the pneumatic card.Moreover, there is also a rotating arm driven by a dynamixel which is being controlled through the interface card.

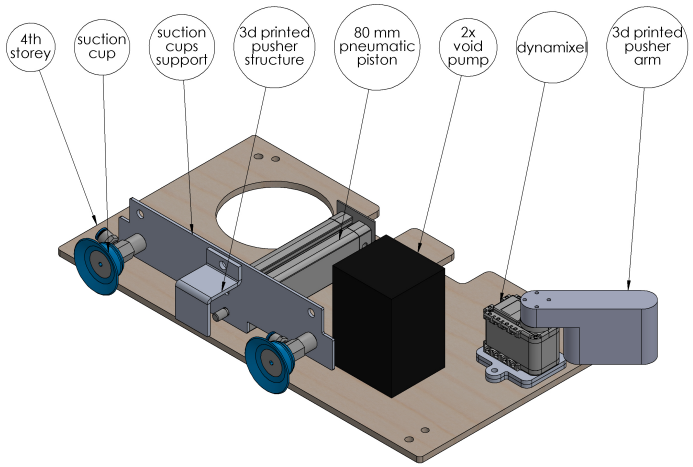


Figure 4: Ground 3

4. The fourth floor is composed of one processor Raspberry Pi, the central brain of the robot, the De0-Nano, where most of the input signals coming from the encoders/odometers are processed and the interface card (described in the section Electronics). We also have a second dynamixel that will drive our elevator by the intermediate of a flexible joint. As explained for the first stage, that is the place for the two batteries.

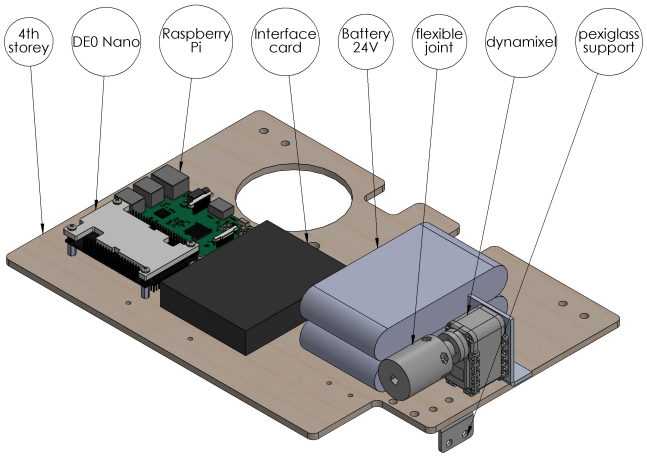


Figure 5: Ground 4

5. The last stage is composed of a Lidar used for the localisation of the robot and of an emergency button to stop the motorisation in case of unexpected failure. Above the Lidar, there is also a support to host the beacon of the opponent.

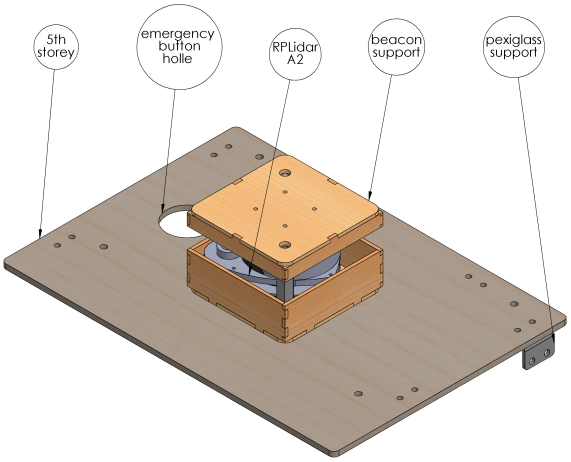


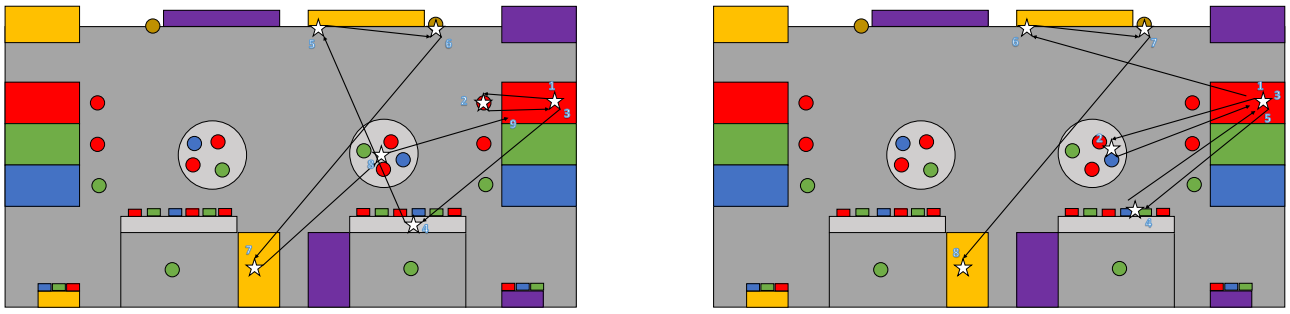
Figure 6: Ground 5

3.2 Strategy

In this section, we compare the initial strategy for which the robot was firstly thought and the strategy we applied at the contest. The remaining strategy is still a predefined path stored in the robot. All the actions are executed one after the other and are not adapted in terms of the opponent strategy.

The motivation for the changes are the following:

1. First straight to the chaos zone (7b point 2) to remove the pucks in order to clear the road to the distributor(7b point 4). The problem occurs when the robot go backwards toward the wall and that a puck is in between.The gap between the wall and the robot is to wide for the suction cup to reach the distributor and the robot is not parallel to the wall. Therefore it is safer to place them in the base first.
2. Then we choose to drop the atoms in the base (7b point 5) instead of the accelerator (7a point 5). In fact placing the atoms in the accelerator is time consuming and requires a great precision. Thus we save time to complete other tasks that give us also points.



(a) Old strategy

(b) Current main strategy

Figure 7: Comparison and changes in strategy

The following table 2 describes the order of the actions performed, gives the time it takes and the points obtained. This is true until point five after a bit of implementation is still required (a similar analysis can be found in the first report for the initial strategy). Thus the time before point 5 are experimental results but after it is an estimation.

In our strategy, the experiment is started by WIFI. The Kraken send a signal and the experiment starts just after the beginning of the match.

Reference	Actions	Time	Score	Status of work
1	Starts from the red base	0 s	+0	functional
2	Goes around the chaos zone	10 s (measured)	+0	functional
3	Brings 6 pucks inside the red base	20 s (measured)	+21	functional
4	Grabs atoms in the distributor	27 s (measured)	+0	functional
5	Drops atoms in the red base	50 s (measured)	+22	functional
6	Unlocks the Goldenium	60 s (estimated)	+20	under test
7	Grabs the Goldenium	80 s (estimated)	+20	not implemented yet
8	Places the Goldenium	90 s (estimated)	+24	not implemented yet
	Experiment		+40	functional
	Total of functional actions		83	
	Total feasible		147	

Table 2: Strategy summarise

4 Description of the different functions

In this section, Arthur, Valentin, Antoine and Miguel, Alexandre developed the scientific and technical material and drafted the text; and all members of the group revised the text to improve its integration to the whole document.

4.1 Mobility

4.1.1 Low-level control and actuation

The low level controller is a PI controller as shown in the blue box in Figure 9. In order to determine the constant K_p and K_i, we start from the model given in the red box. The parameters are given by the datasheet of the motor.

$$J_{eq} = m_{robot}\frac{r_{roue}^2}{2.R^2} + J_{motor}$$

where R is the reduction ratio.

Parameters	Values	Units
U_n	24	$[V]$
I_n	0.82	$[A]$
N_n	4770	$[rpm]$
$K\Phi$	0.0378	$[V/rad/s]$
J_{eq}	6.3939 e-6	$[Kg.m^2]$
K_v	4.3e-5	$[N.m.s]$
R_a	5.84	$[\Omega]$
L_a	560 e-6	$[H]$

Figure 8: Parameters of the motor

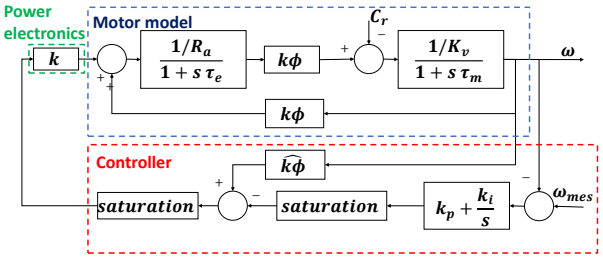


Figure 9: Low level controller schematic

The encoders do not provide a current information therefore we will design a controller without a current loop. Moreover we make the following assumptions in order to simplify our system.

- We consider that the electrical time constant is way smaller than the mechanical time constant, thus we can neglect the electrical transient.

$$\tau_e = \frac{L_a}{R_a} = 9.58e^{-5}s << \tau_m = \frac{J_{robot}}{K_v} = 0.17s$$

- We also consider that the estimation of $K\phi$ is also perfect, thus the feed-froward action compensates exactly the retro-action inside the model.

To determine the values of k_p et k_i, we use cancellation pole-zero. We cancel the slowest pole $\frac{1}{\tau_m s+1}$ and $\frac{k'_p s+k'_i}{s}$. We obtain this equality: $\frac{k'_p}{k'_i} = \tau_m$. Using the first assumption, $\tau_e << \tau_m$, we neglect the pole $\frac{1}{\tau_e s+1}$ and it becomes a first order transfer function, given by : $\frac{K_{gain}}{\tau s+1}$. We can use the electro-mechanical time constant to have a second equality $\tau = \tau_{em}$. Thus we find $K_i = 0.261$, $K_p = 0.0397$

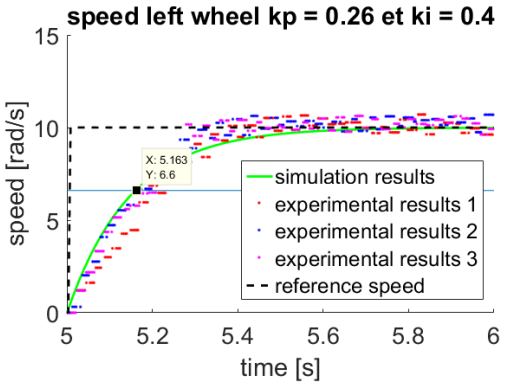


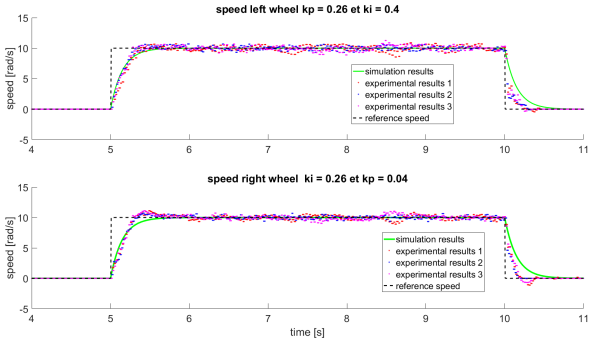
Figure 10: Zoom in the picture

Since we use the electro-mechanical time constant to compute the parameters, we can theoretically have a approximation of the time constant.

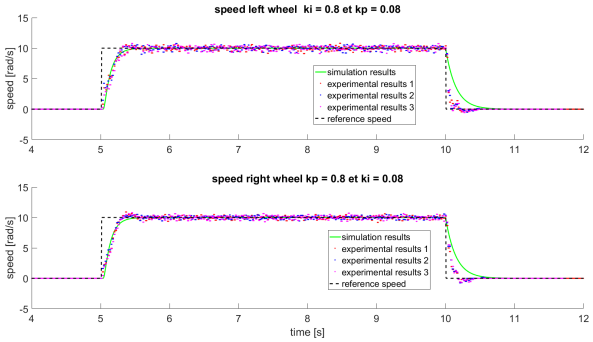
$$\tau_{em} = \frac{J_{eq} R_a}{(k\phi)^2} = 0.0261s$$

In the figure 10 the value at 66% of the steady state value gives $t = 3\tau$. Thus, $\tau_{em} = \frac{5.183-5}{3} = 0.061s$

The estimation is a bit lower than the theoretical and experimental values but the order of magnitude is good. We can notice that the theoretical and experimental results are quite close. After we try several values for the controller to see if there is any improvement but the results are similar. (fig 11)



(a) result to a step $k_p = 0.08$ and $k_i = 0.8$



(b) result to a step $k_p = 0.04$ and $k_i = 0.26$

Figure 11: Comparison of several constants

The graphs 12 shows the left and the right wheel speeds over a complete match of 60 s. The order is given by the path-planning algorithm explained in the following section. At 25 s and 37 s there is a inconsistency, it is explained by the fact that the robot is going towards the distributors and the speeds are not set with the path-planning anymore. Therefore it does not match with the order. For the rest of the time, the curves are following each other very well for positive and negative speeds.

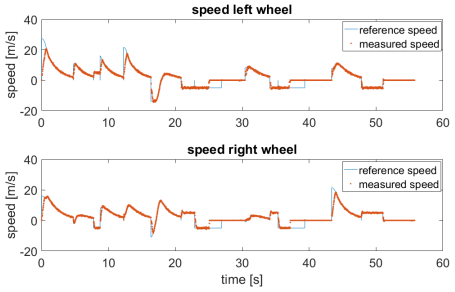


Figure 12: Zoom in the picture

4.1.2 Path planning and opponent avoidance

The robot finds his way with the path planning method called "potential field". Each target is represented as an attractive potential while wall, obstacles and opponents are represented as repulsive potentials.

A representation of the potential map is shown in the fig 13. In this case there are not obstacles, no opponents and no targets. Only the walls are represented to give a felling that the robot will remain in this area during a match period. By adding targets or opponents we can modulate the field and the robot can slide to the smallest potential as water would do.

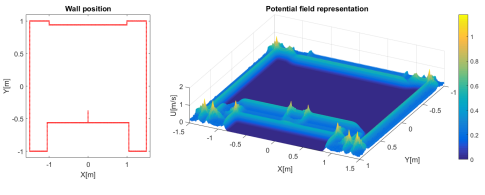


Figure 13: Potential field representation of the map

By setting the target one after the other, we see in the figure 14 that the robot moves to reach them. (The robot starts at the blue dot.) Thus the path planning algorithm works as expected. To test the robustness and the accuracy of the algorithm and sensors. The same strategy is ran 5 times in a row. Each test represent an experimental game play. In black there is the mean of all the tests and we can observe that the standard deviation (red doted line) is less than 1 cm. These results are obtained without the LIDAR (the robot only relies on odometers), thus we can conclude that in the internal map the robot is very precise.

One can observe a circle around each target which corresponds to an area where the robot should enter before moving to the next one. By decreasing the size of it the robot is more precise but it sometimes struggles to find the exact location, thus it is a trade-off between time and precision.

An example of this problem is shown when an exact position is required such as point 5 and 6 in the fig 14. In this case we would like to position the robot in front of the distributor to grab the pucks with the suction pad (described later in section Grabbing atoms on the distributor, it requires great precision 1.5 cm). The position reached by the robot is always the same, but it not the one set by the user.Thus either an offset or a smaller circle can be used to be more precise.

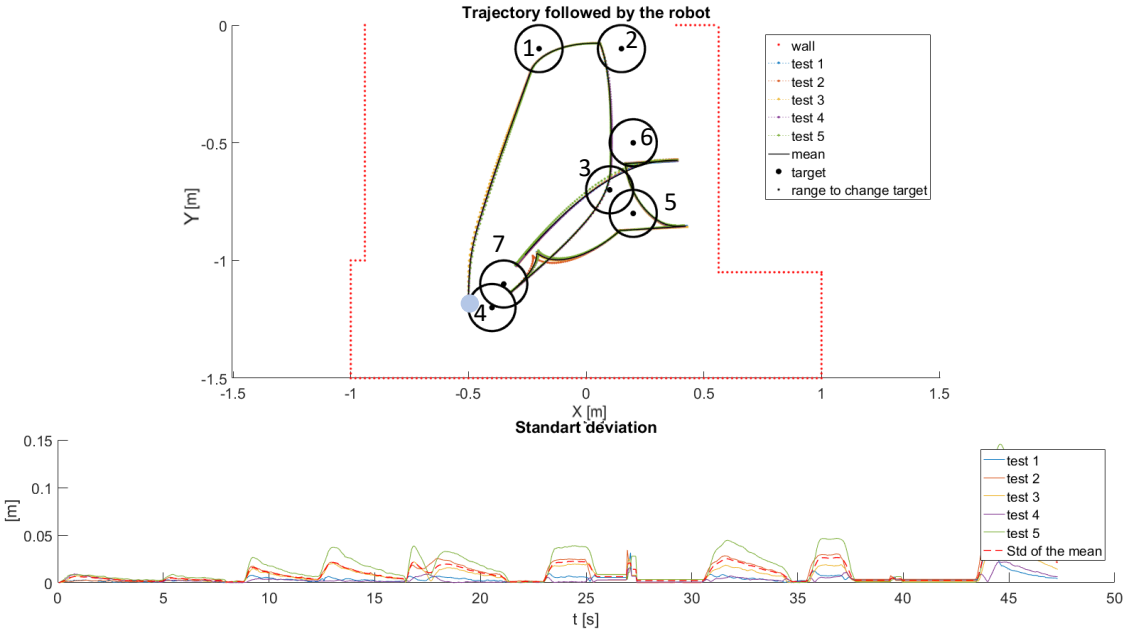
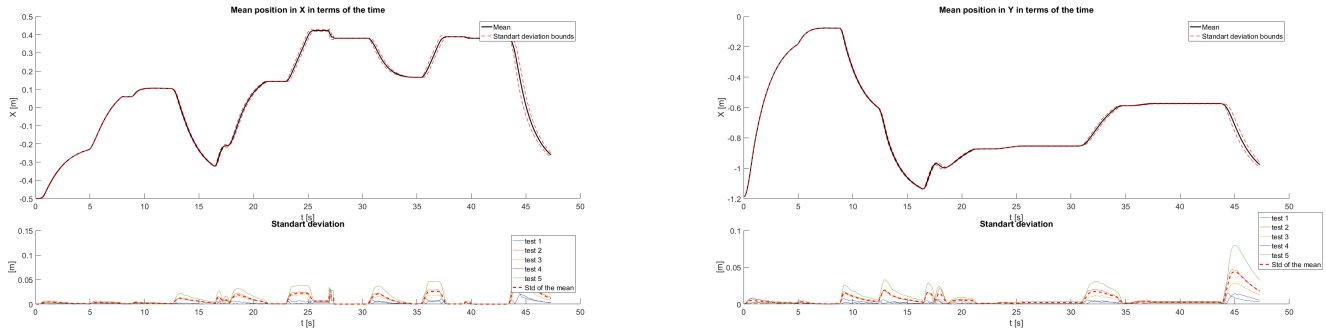


Figure 14: Trajectory followed by the robot

The figure 15 represent the same trajectory than the one in the fig 14 but in terms of the time. This shows that the robot present a stable and reliable behaviour with respect to the time when it evolves in the controlled environment without an opponent.

At time 27 s, the position in x suddenly drops from 10 cm. This is a calibration. It shows that the robot loses little by little track of it real position and that we can use the walls or the LIDAR to correct it. The mathematical aspect of the error is described in the next section.

The error is quite constant in terms of the several trials. The standard deviation is below 5 cm. The mean error is also small compare to the overall travelled distance (about 5 cm before and after 27s). It is quite small compare to the distance done by the robot at this stage. After the calibration the standard deviation is null since the value is known.



(a) Trajectory in X in terms of time

(b) Trajectory in Y in terms of time

Figure 15: Trajectory followed by the robot

We can avoid the opponent thanks to the path planning method. The lidar detects it and a repulsive potential is added when the opponent is closer than 0.7 m.

The figure 16 shows what appends when the robot (in black) is detecting the obstacle in red. The trajectory without the obstacle is represented in blue and the trajectories show to experimental test when the obstacle is placed in the trajectory. The robot can go either on the left or on the right in terms of how it is align with the obstacle.

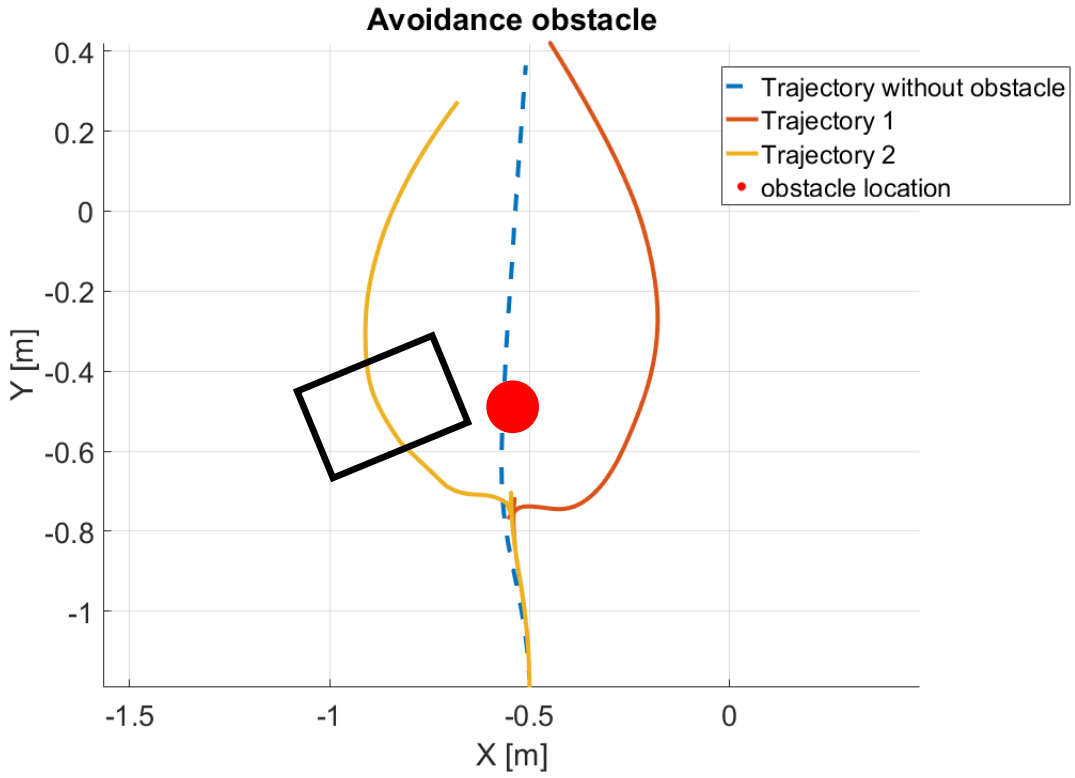


Figure 16: result over a pah

4.1.3 Localization

The robot is localised by two sensors: odometers and a LIDAR. Odometers are relative sensors, they enable a *prediction* of the position by knowing the previous position. The LIDAR gives an absolute position and is thus used to *update* the robot position.

Odometry Odometry consists to compute the robot position based on the rotation of each odometry wheel, with a refresh rate of 20 ms in our current software.

The position is predicted with the relation:

$$\hat{P}_{i+1} = \begin{pmatrix} \hat{x}_{i+1} \\ \hat{y}_{i+1} \\ \hat{\theta}_{i+1} \end{pmatrix} = \hat{P}_i + \begin{pmatrix} \Delta s \cos\left(\hat{\theta}_i + \Delta\theta/2\right) \\ \Delta s \sin\left(\hat{\theta}_i + \Delta\theta/2\right) \\ \Delta\theta \end{pmatrix}$$

and the predicted covariance is

$$\hat{\sigma}_{P,i+1} = \nabla_p f \cdot \hat{\sigma}_{P,i} \cdot \nabla_p f^T + \nabla_\delta f \cdot \sigma_\delta \cdot \nabla_\delta f^T$$

Results of odometry are provided in the figure 15. This figure is describe in the previous section.

LIDAR We use a RPLIDAR A2/M8, this is a really good model and one of the best end-users LIDARs available on the market with its sample frequency of 4000 Hz. The LIDAR can give either an absolute position based on triangulation of three beacons, or an *update* step with only one beacon in a Kalman filter that will modify the position estimated by odometry. The LIDAR detects some beacons on the border of the map, made with hollow pvc pipes. Its speed is 10 Hz, giving it a refresh rate of 100 ms, which is 5 times lower than the odometry update.

Firstly, the *triangulation* of three beacons is achieved by considering the power center of three circles around the beacons (the details of the computations are given in [1]). It has proven really good results in the order of 5 cm but they are not sufficient for our strong requirements. Indeed, this method requires three beacons to localise itself, which means that one of them is necessarily at more than 2 m from the robot. This long distance is the limiting factor since the LIDAR gives only 1 or 2 values per turn due to the low angular resolution of the LIDAR (about 0.9°). Moreover, taking the center mast (used for the experiment) as a beacon is also not precised because it is too large and not circular. As a result, this method is not implemented, while it has been really powerful for the robotics project. The details of the algorithm are given in Appendix A.

Secondly, an *Extended Kalman Filter* (EKF) is implemented to correct the position by odometry. It consists to transfer the data given by the LIDAR (angle and distance) for a beacon in the robot frame into the beacon position in the base frame based on the current estimation of the robot position. The algorithm works as follows.

1. Compute the predicted observation (the angle α_B and the distance d_B from the robot to the beacon):

$$h = \begin{pmatrix} \alpha_B \\ d_B \end{pmatrix} = \begin{pmatrix} \text{atan2}(y_B - \hat{y}, x_B - \hat{x}) - \hat{\theta} \\ \sqrt{(\hat{x} - x_B)^2 + (\hat{y} - y_B)^2} \end{pmatrix}$$

   where $\hat{P} = (\hat{x}, \hat{y}, \hat{\theta})$ is the robot predicted position from the odometry step and (x_B, y_B) is the beacon position.

2. Compute the jacobian of h:

$$\nabla h = \begin{pmatrix} \frac{y_B - \hat{y}}{d_B^2} & \frac{\hat{x} - x_B}{d_B^2} & -1 \\ \frac{\hat{x} - x_B}{d_B} & \frac{\hat{y} - y_B}{d_B} & 0 \end{pmatrix}$$

3. Compute the innovation with the LIDAR data:

$$v = \begin{pmatrix} \alpha_L - \alpha_B \\ d_L - d_B \end{pmatrix}$$

   where α_L and d_L are the angle and the distance determined by the LIDAR.

4. Compute the covariance matrix of the innovation:

$$\Sigma_{IN} = \nabla h \cdot \hat{\sigma}_P \cdot \nabla h^T + \sigma_L$$

where $\hat{\sigma}_p$ is the predicted covariance of the robot after the odometry step. σ_L is the covariance matrix of the LIDAR data and is characterised once and for all by sampling a lot of data in different places of the map. Next the covariance matrix of the angle and distance from the LIDAR is computed as:

$$\sigma_L = \begin{pmatrix} \sigma_{\alpha_L \alpha_L} & \sigma_{\alpha_L d_L} \\ \sigma_{d_L \alpha_L} & \sigma_{d_L d_L} \end{pmatrix}$$

5. Compute the Kalman gain:

$$K = \hat{\sigma}_P \cdot \nabla h^T \cdot \Sigma_{IN}^{-1}$$

6. Update the position:

$$P = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \hat{P} + K \cdot v$$

7. Update the covariance matrix:

$$\sigma_P = (I - K \cdot \nabla h) \cdot \hat{\sigma}_P$$

This algorithm is repeated for all beacons that are sufficiently close to the robot, typically in the 2m range.

This method has proven to be really reliable and with a high precision, in the order of the cm.

4.2 Mechanisms

In order to follow the strategy described in the section 3.2, we have conceived and implemented innovative mechanisms in the robot. The steps to perform are the following:

- Movement 1: move to the linear distributor and bring out the suction pads of the first stage of the robot.
- Movement 2: catch the pucks with the suckers
- Movement 3: move back the suction cups to an intermediate position which is located above the tank and drop the atoms in the container.
- Movement 4: bring out the the suction pads to catch three other pucks.
- Movement 5: move back the suction cups inside the robot.
- Movement 6: move to the accelerator. Thanks to *movement 5*, the container with the three first pucks has space to go up, at the level of the accelerator.
- Movement 7: push the atoms in the accelerator thanks to the piston on the second floor of the robot.
- Movement 8: lower the container and drop the atoms catch during *movement 4 and 5* in the container. This is done by deploying the piston in the intermediate position with the piston of the first floor.
- Movement 9: rise the container and repeat *movement 7*.

The descriptions of these movements will help the reader to understand our choices for the mechanisms. Our guideline was to be **the more efficient**, with simple mechanisms. This section describes their role and characteristics.

4.2.1 Pushing on the ground

Pushing the atoms on the ground is not really a mechanism but it is a quick and easy way to score points. This method brought us the majority of our points scored during the Eurobot competition. At the beginning of the game, we bring the pucks of the chaos zone back into the base by dragging them on the table.

We simply designed the bottom aluminium plate so that we could move as many pucks as possible. As our robot is very wide, we can bring back up to five pucks. We made a rounded cut to avoid losing pucks during robot manoeuvres and to increase the storage area (see fig 2). The radius of the cut was limited by one of the free wheels located at the front of the plate. It was therefore necessary not to make sharp turns and to travel in a straight line as much as possible, otherwise the pucks go out of the robot.

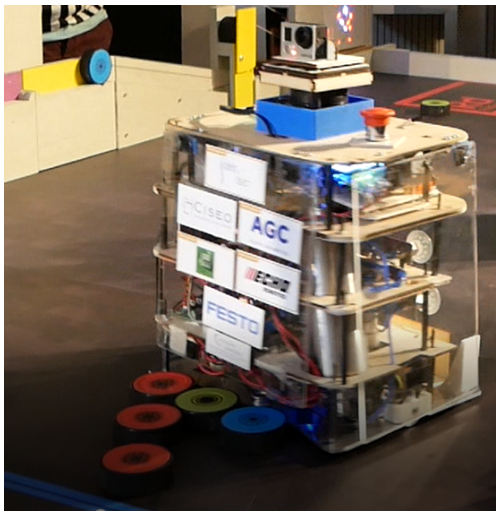


Figure 17: Robot uses the bottom plate to collect atoms

4.2.2 Grabing atoms on the distributor

In order to take the atoms in the distributors, we opted for **pneumatic actuators**. The motivation of this choice is that the movements made to take the pucks are simple linear movements. They are repetitive and must have a precise stroke. The room in the robot is also an important constraint and pneumatic pistons are very compact. Furthermore, they are easy to actuate.

As explained above, two positions are needed on the first floor. Reaching two strokes with pneumatic pistons is not an easy task in a confined space. The solution proposed in the preliminary report, which was to use a double-stroke cylinder, is not possible because the cylinder is too long and does not fit in the robot.

The solution was to use two pneumatic *mini-slides DGSL* from Festo. They are very compact and designed to be fixed on top of each other.

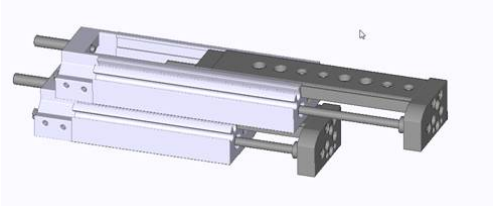


Figure 18: Mini-slides DGSL mounted on top of each other

When the fully-extended configuration is needed (80 mm), both pistons are actuated. For the intermediate position, only the piston having the smallest stroke (30 mm) is actuated. They are each actuated by its 5/2 mono-stable solenoid valve. The reason we chose mono-stable valves is that they can be controlled by a single control signal.

At the extremity of the piston with the small stroke, we screwed an aluminium plate manufactured to hold three suction pads (see fig 3).

Robustness and validation
In order to validate the mechanism and to test its robustness, different tests were made. First of all, we tried to define the parameters that could variate and to give an estimation of their value for which the mechanism is always working. We found the following values:

Parameter	Estimated working value
Pressure range	[3;4] [bar]
Distance from the wall	10 [mm]
Distance from the atom centre	0 [mm]
Time pistons out	2 [s]
Time sucker off	3 [s]

Time pistons out is the time that the the suction pads switched on are in contact with the pucks before bringing them back in the robot, time sucker off is the time the pucks are staying over the elevator with the pumps of in order to make them fall in the atom elevator. Those parameters are estimated to be working each time. Of course, it is way too demanding for our robot, regarding to its localisation and robustness. Starting from those values, we are going to variate those parameters, to see which is the working range of this mechanism. Here are the results we found.

		Pucks taken [%]			Pucks dropped correctly[%]		
Parameter	New value	Left	Middle	Right	Left	Middle	Right
Pressure range	[2;3] [bar]	100	95	65	100	95	65
Distance from the wall	15 [mm]	100	100	95	100	100	95
Distance from the wall	17 [mm]	0	0	0	0	0	0
Distance from the atom centre	16 [mm]	100	100	95	100	100	0
Distance from the atom centre	18 [mm]	0	0	0	0	0	0
Time pistons out	1 [s]	100	95	20	100	95	20
Time sucker off	2 [s]	100	100	100	40	100	20

We first thought that a [2,3] bar range would be sufficient, but after some tests, we found out that a minimum of 3 bar has to be respected for correct results. We also notice that a really good positioning of the robot is needed in order to make this mechanism reliable.

4.2.3 Pushing atoms in the accelerator

The pneumatic piston placed on the second floor allows to push the atoms placed in the elevator into the accelerator. It is also used to catch the *Goldenium*. A single stroke is then sufficient.

We used a double-acting compact piston with 80mm stroke. It is operated by a 5/2 mono-stable solenoid valve. The piston is square to prevent rotation of the shaft. An aluminium plate manufactured to hold two suction pads is also fixed at its extremity (see fig 4).

4.2.4 Pneumatic circuit

The pneumatic circuit of the first and second floor is given in figure 19. Table 20 resumes the pneumatic components we have. A more detailed list with all component references can be found in the appendix B.

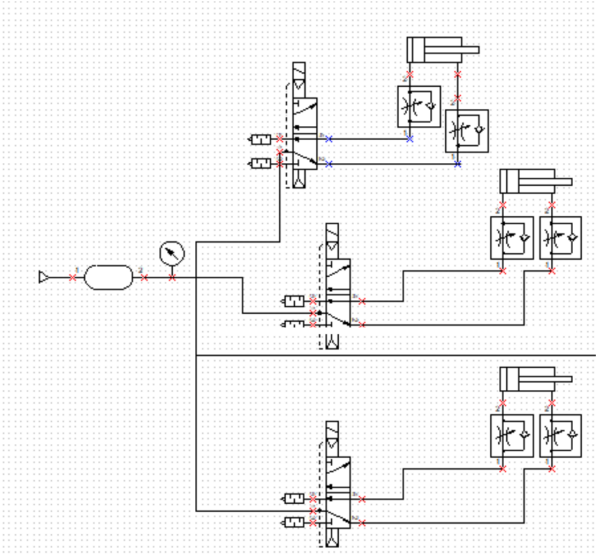


Figure 19: Pneumatic circuit

First floor	DGSL with stroke 50 [mm]
	DGSL with stroke 30 [mm]
	2 Mono-stable electrovalves 5/2
	4 flow limiters
	3 suction pads (⌀40 mm)
	3 diaphram pumps
Second floor	ADN with stroke stroke 80 [mm]
	1 Mono-stable electrovalves 5/2
	2 flow limiters
	2 suction pads (⌀40 mm
	2 diaphram pumps
Tank	Minimal pressure 3 bar
	Normal pressure 4 bar
	Volume 0.75 L

Figure 20: Resume of the components of the pneumatic circuit

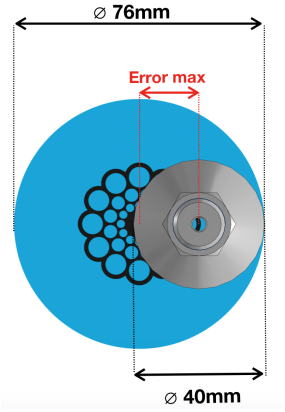
4.2.5 Suction pads & vacuum generation

Vacuum generation We changed the way we grab pucks with suction cups compared to what was presented in the technical report. Our initial idea was to use "passive" suction cups. In this case a simple pressure against the puck (block by the wall on the other side) is sufficient to generate the vacuum and thus "suction" effect wanted. This technique worked well with pucks which have a smooth surface. However a bit later, we received the official pucks from Eurobot and their surface was rougher, making self-sealing impossible.

Using a venturi system was not possible due to the limited tank volume. Indeed, this technique consists in generating a vacuum from air under pressure.

We needed an electrical solution to generate a vacuum. We found **diaphragm pumps** (Fig 21a) that are very compact and perfect for our application. They are powered by 3.5 V via the pneumatic card (see section 5.2.4) and have a vacuum flow rate of 0.9 L/min. There is one pump for each suction cup. Indeed, if several suction cups were fed by a single pump and one of the suckers misses a puck, the others would lose the suction.

(a) Diaphragm pump [ref: 3003VD/0,8/E/LC by Thomas]



(b) Maximum robot positioning error to catch the pucks

Figure 21

Suction pads The suction pads we chose have a diameter of 40 mm. We chose a large diameter because our initial idea of self-sealing suckers worked better with a large contact area. However, a smaller diameter would have been better with the pump suction system. Indeed, with a large suction cup diameter, the positioning of the robot must be very precise. As illustrated in figure 21b, the maximal position error allowed is:

$$R_{puck} - R_{sucker} = 1.8\,\text{cm}$$

. Otherwise, the suction pad is only partially in contact with the puck and suction is not possible. Achieving such a precision is not an easy task and the robot fail sometimes to take all the the pucks. Unfortunately, we had no time to re-order smaller suckers and the final robot keeps these suckers.

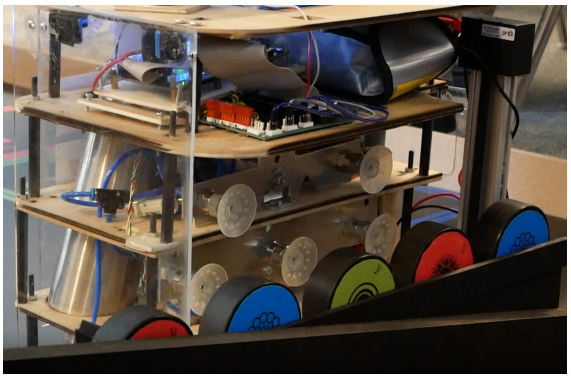


Figure 22: Pneumatic suction pad

4.2.6 Pneumatic command

To activate the pumps and pistons, signals are sent to the pneumatic card (section 5.2.4) by the Raspberry PI, through the SPI interface.
We simply set the PIO corresponding to the element to actuate and the pneumatic card is designed to supply the corresponding piston/pump.

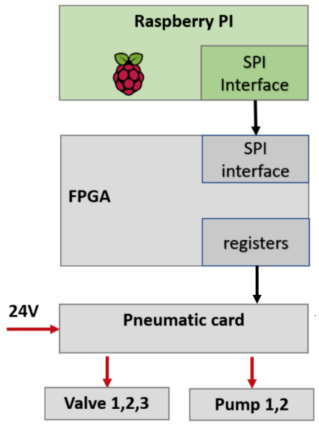


Figure 23: Pneumatic command

4.2.7 Elevator

In order to move the atoms vertically, the robot also relies on one elevator system, based on one dynamixel (AX-12A) and one 160mm stroke linear rail (product code ZLW-0630, from IGUS). The transmission between the dynamixel and the rail is done by one rigid aluminium joint, one circular shaft and one 3D printed joint that has a square mounting hole to be connected to the input of the rail. In the moving part of the rail, there is a 300mm wide container, responsible for storing and lifting up to three pallets at once. The assembly of this set depicted in Figure 24.

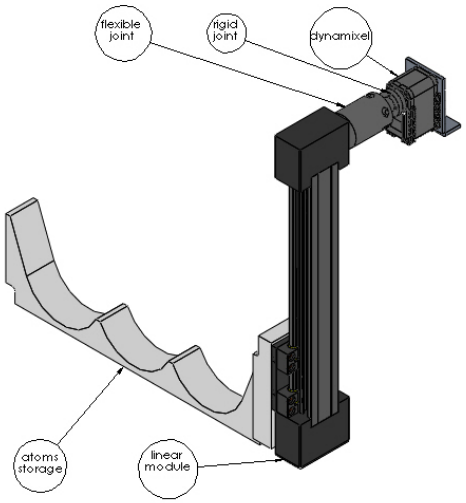


Figure 24: Elevator system

As detailed in the previous rapport, the requirements were that the system should be able to lift a total weight of 20N (including the container) 155mm up in 5 seconds. The previous calculations showed that for such a task an average torque of 0.4 Nm would be required.

Once the system was fully installed in the robot, tests were conducted to verify the torque required from the motor in different situations. Due to limitations of the servo motor, it is controlled in speed, which kept the motion time approximately constant in 3.5s. The torques obtained are characterised according to the number of Greenium's (120g) raised at each experiment, and the results are presented in Table 3.

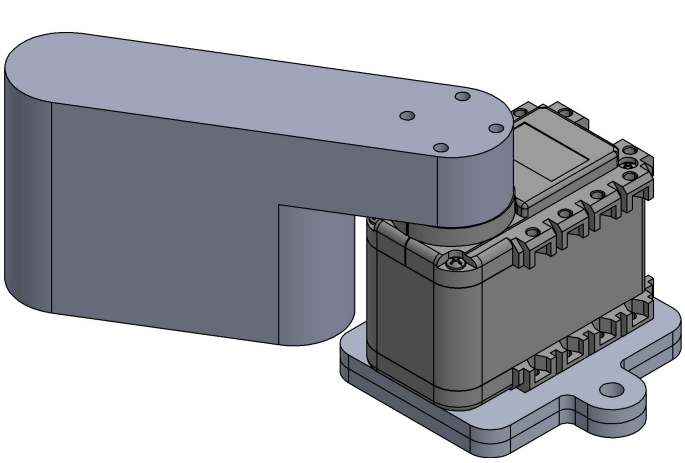
Table 3: Measured torque at elevator for different weights

# pallets	Weigth [N]	τ [Nm]
1	0.12	0.72
2	0.24	0.74
3	0.36	0.77

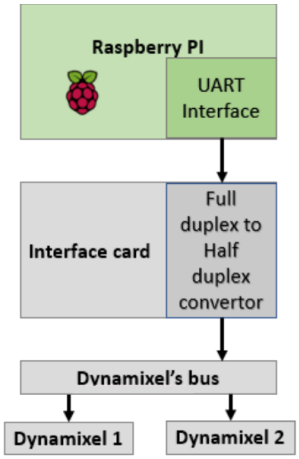
It shows that the specification of lifted weight and time are assessed by the robot implementation, but the torque obtained is considerably higher than expected. We attribute this difference to the internal friction of the motor and the friction of the internal transmission system inside the linear rail, which is based on a timing belt.

4.2.8 Accelerator deployment arm

This subsystem is aimed in pushing one or more atoms into the particle's accelerator. It is composed of one Dynamixel (AX-12A) and one 3D printed arm, connected directly to it, as illustrated in the Figure 25a.



(a) Accelerator deployment arm 3D model



(b) Dynamixel communication scheme

Figure 25: Accelerator deployment arm

Both in the case of this actuator and in the previous detailed system, the data that controls the Dynamixel follow the scheme in Figure 25b. The communication is based in a protocol specific for this class of actuators, which takes place through an UART interface.

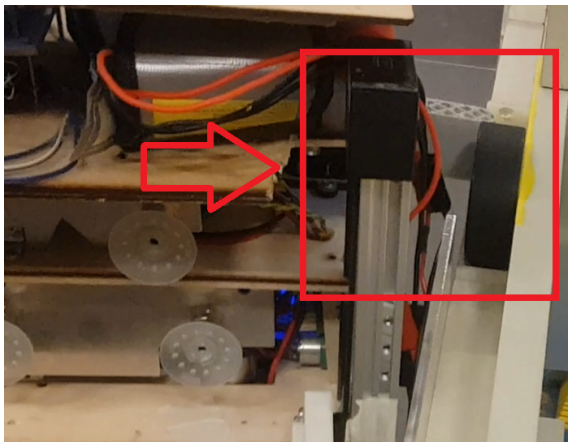
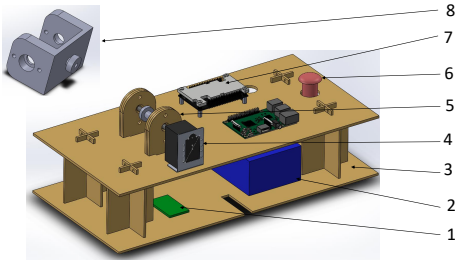


Figure 26: Picture of the arm deployed to unlock the goldenium

4.2.9 Experiment

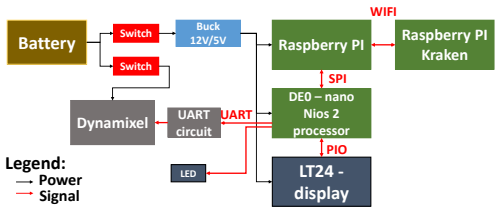
The experiment consists in raising an atom from the base to the mast. The atom is elevated with a belt driven by a pulley (located on the support). An other pulley is attached to the mast. The experiment begins when a wifi signal is sent by the Kraken and starts the Dynamixel for 34 s, which is the time needed by the electron to reach its destination.

In Figure 27a, the base structure is depicted with the main components. The area of the bottom plate is 400mm x 200mm and the global height is 146mm. Figure 27b shows the electrical connections and blocks.



1. Red button card + buck converter
2. Battery
3. Support
4. Dynamixel + support
5. Driven pulley
6. Red button
7. RaspberryPI + FPGA + LT24
8. Mast pulley

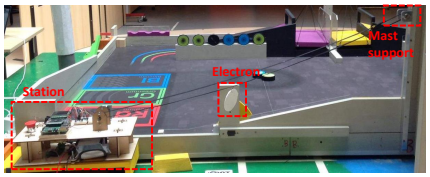
(a) Description of the components



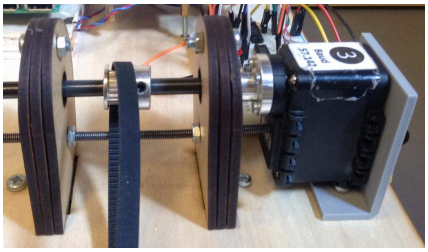
(b) Electrical schematic

Figure 27: Global overview

In comparison with the theoretical plan, Figure 28 describes how the experiment is set up, Figures 28b and 28c highlight the mechanical parts of the final prototype used for the contest.



(a) Overview of the full system



(b) Driven pulley

(c) Mast pulley

Figure 28: Experiment photography

The experiment is performed several times to test the robustness and reproducibility. Table 4 summarises the results obtained for each match of the contest. In fact the contest is an appropriate place to guarantee the wifi connection, since it is the environment with perturbations. The timing is given by an internal clock and the height reached by the electron is always sufficient to consider the experiment as finished.

Match	WIFI message received	Electron reaches the mast	Match	WIFI message received	Electron reaches the mast
1	yes	yes	5	yes	yes
2	yes	yes	6	yes	yes
3	yes	yes	7	yes	yes
4	yes	yes	8	yes	yes

Table 4: Experiment test results

Looking at the table, this proves that the experiment has been a reliable feature of the Kraken.

In the technical report we designed the experiment with a smaller motor "28BYJ-48". We changed for a Dynamixel because it is easier to implement something already functional in the robot. Also the mass of the electron was overestimated

0.003 kg, as the electron is now lighter, 0.005 kg.

Since we increased the torque capability of the motor and we decreased the load, we did not expect a problem however we tried to understand how much weight we could lift.

We are using a Dynamixel A-12 and it stall torque is given in the data sheet 1.54 Nm for (12 V,1.5 A). In the figure 29, it is shown how the current increase in terms of the load. In one hand we can see that the working point of the experiment is safe because we could lift a heavier electron without problem from the motor side. In the other hand we could not test for a heavier load because the wooden support is not strong enough. Thus we can conclude that the the Dynamixel works well for lifting the atom but it is an overestimated design since it could lift a much heavier load.

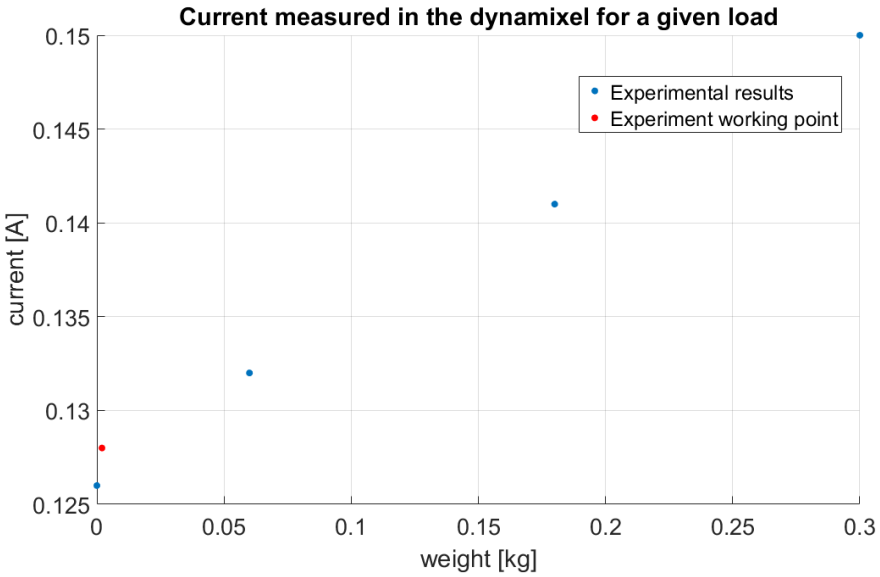


Figure 29: Current in the Dynamixel in terms of the load

5 Electronics

In this section, Alexandre and Martin developed the scientific and technical material and Alexandre drafted the text; and Miguel revised the text to improve its integration to the whole document.

5.1 Global architecture

This section reminds the functional blocks we use in the technical report with certain modification that we decided to bring. The architecture of the electronic circuits of the robot is shown in Figure 30.

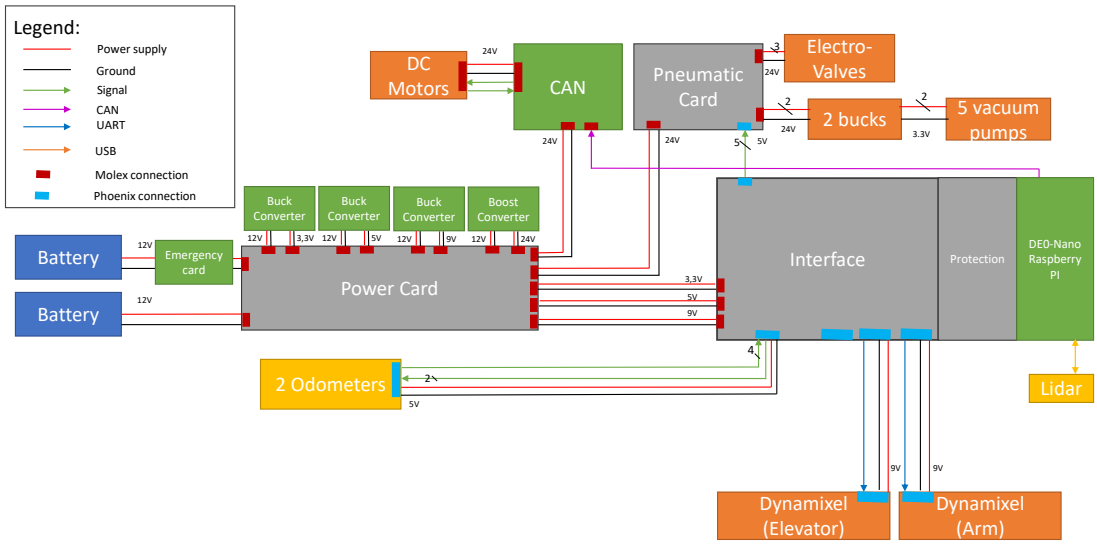


Figure 30: Electrical architecture of the robot

The mains blocs of this architecture are composed of

- Power card : it distributes the voltage to the different components we need to power through converters DC–DC (buck/boost converters).
- Interface card : it makes the link between the De0-Nano and the rest of the robot by recover all of the inputs signals that will be processed into the FPGA and distributes all the output signals coming from the brain of the robot, the Raspberry Pi.
- Can card : it makes the link between the raspberry Pi and the motors trough the bus Can. It allows to modulate the tension send to the motors.
- Pneumatic card : it allows to activate the piston as well as the vacuum suckers depending on the signal sent from the interface card.
- Emergency card : it allows us to cut off the motors and the pneumatic part in case of failure without powering off the Raspberry PI.

As mentioned above, we had to make some adjustments for the pneumatic part and the interface card. Firstly, we decided to remove the 4 ultrasonic sensor because the LIDAR was enough accurate to detect the opponent. Secondly, we chose to dedicate only 3 signals to activate the 3 electro-valves/pistons and the 2 remaining signals for the 5 vacuum pumps (1 signal will set on the three pumps on the second stage and the other signal is used for the third floor.

5.2 Boards design

5.2.1 Power board

This board contains several capacitances in the input to assure a stable voltage. Moreover, it is mainly composed of 4 DC–DC converters (1 boost and 3 bucks). The boost converter is used to power on the CAN board for the motors that works under 24 [V] as well as to power up the pneumatic card because of the nominal voltage required by the Electro-valves. The first buck is used to power on the Dynamixel that works under 9 [V]. The second is to power the FPGA-Raspberry PI that works under 5 [V]. It is essential because all the signals we need (for the odometers, the LIDAR, the encoders, the pneumatic signals) works at this nominal voltage. The last buck converter is used to transform all of these 5 [V] signals in 3.3 [V] required on the pins of the FPGA.

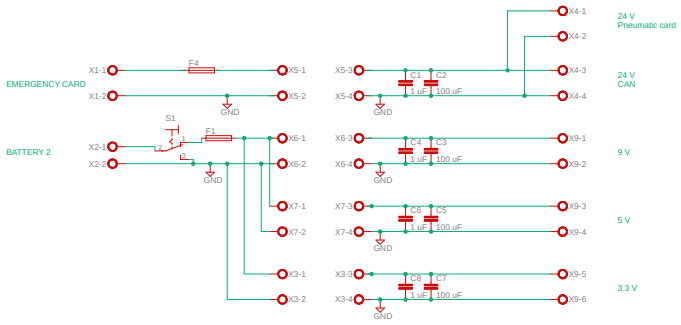


Figure 31: EAGLE schematic of the power card

5.2.2 Interface board

This board is the link between the processing unities of the robot (FPGA and Raspberry Pi) all the actuators and sensors that are connected through GPIO. For reducing the number of electrical connections, this card also distributes the energy to all low power components (FPGA, Raspberry Pi, Lidar, etc), fed by just one battery in different voltages .

To achieve it, the board receive signal from the 2 odometers/encoders and root them to the FPGA, where they are pre-processed and made available to the Raspberry Pi. It also roots the command signals from the electro-pneumatic system and from the Dynamixel communication bus to the FPGA.

In order to make all those connections with the FPGA, the signal lines had to be converted from 3.3V, in the FPGA side, to 5 V in the side of the actuators and sensors. It was achieved by using a TXB0108 voltage translator, requiring an special power source to the signal at 3.3 V.

Another part of the interface board that deserves an extra attention is a circuit for making the UART protocol conversion from half duplex, in the Dynamixel side, to full duplex in the Raspberry Pi side. Take note that it had to be implemented on hardware because the FPGA used did not supported three state internal connections, but if a new model

Figure 32 shows an update version of the connections of the pins from the interface card on the FPGA.

Figure 33 shows the Eagle scheme of the interface board

Pins	Signals destination	Pins	Signals destination
1	Robot start	2	Robot Start
3	Plus 1	4	/
5	Dynamixel	6	/
7	Dynamixel	8	/
9	Dynamixel	10	/
11	FPGA power supply [5V]	12	GND
13	Dynamixel	14	Not connected
15	Dynamixel	16	/
17	Dynamixel	18	/
19	Pneumatic	20	/
21	Pneumatic	22	/
23	Pneumatic	24	Encoder Right
25	Pneumatic	26	Encoder Right
27	Pneumatic	28	Encoder Right
29	Not connected	30	GND
31	Odometer Right	32	Encoder Left
33	Odometer Right	34	Encoder Left
35	Odometer Right	36	Encoder Left
37	Odometer Left	38	Odometer Left
39	Voltage shifter	40	Odometer Left

Figure 32: DE0 Connections

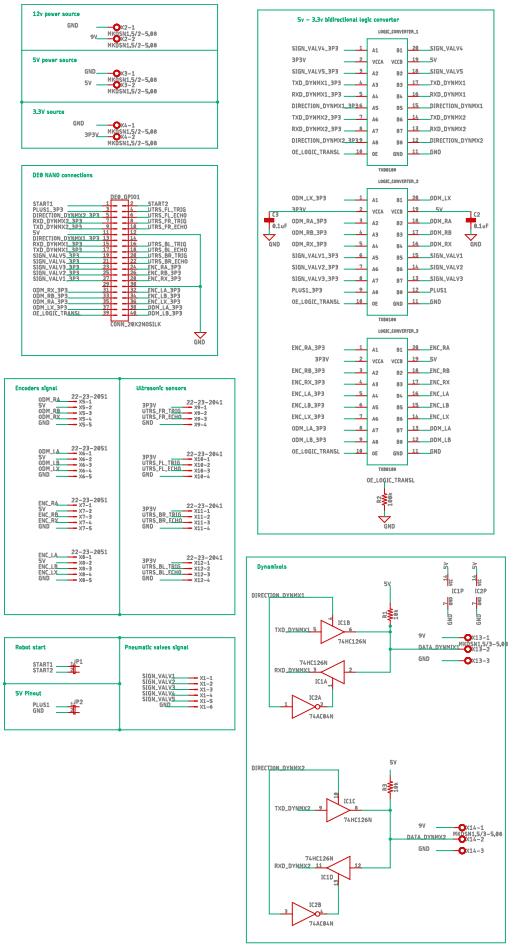


Figure 33: EAGLE schematic of the interface board

5.2.3 Emergency board

This board allows us to have a certain security in case of polarity inversion or short circuits. It is based on a PMOS technology. If we apply a negative voltage on the gate of the transistor, it will prevent the current to go through the rest of the circuit and might damage some components. Otherwise, if the voltage applied on the gate is positive (this voltage is fixed by the battery), it will let the current go through it and we will obtain the same voltage on the output of this board. The computation of the different resistors/capacitors/diodes was done in the technical report.

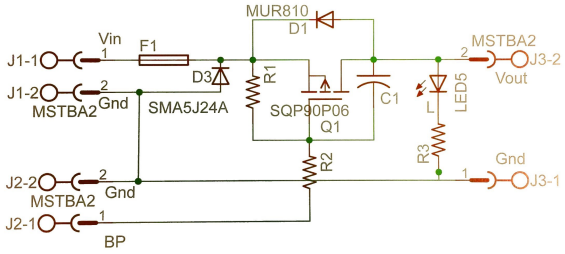


Figure 34: EAGLE schematic of the emergency card

5.2.4 Pneumatic board

The function of this card is to activate the 5/2 monostable electro-valves and the vacuum pumps depending on the value of the signals received by the interface card. When the command coming from the DE0-NANO is high, the NMOS powers the valves with 24 V. A flyback diode is added in order to protect the transistor. On the gate of the NMOS, a small resistor of 20 Ω is added in order to polarise the transistor and limit the current on the gate. A pull-down resistor of 100 kΩ is also connected to the gate in order to not have a floating voltage when the command is at low state. One of the main modifications we did regarding the technical report is that we won't use two rows of 2/2 electrovalves for the suction pads anymore. Instead we decided to use 5 electrical pumps that will be controlled by only 2 signals. These pumps work under 3.3 [V]. Thus, we had to add 2 buck converters at the output of this board.

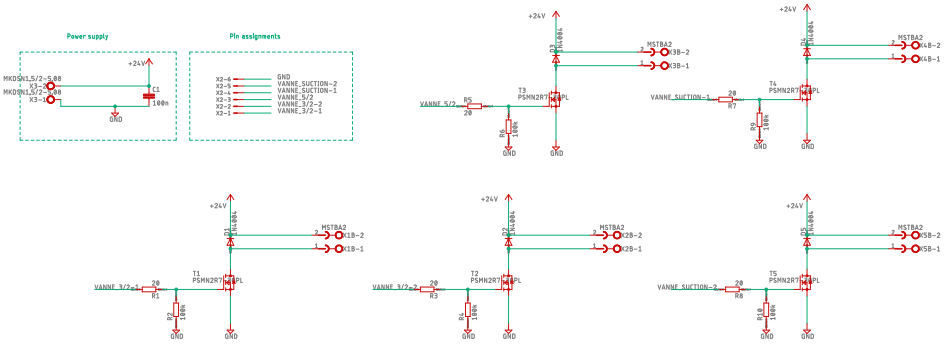


Figure 35: EAGLE schematic of the pneumatic card

5.3 Power management

5.3.1 Batteries dimensioning

The robot uses two batteries of 12 [V].

One of the batteries is mainly used for the FPGA and the Raspberry Pi such that they are always supplied, thus all the data present in the micro-controller are not erased if we press the emergency button. The second battery supplies only the power for the motors and the pneumatic part. The different actuators/sensors with their maximal power consumption and supply voltage are shown in Table 5.

Component	Components	Voltage (V)	Max power (W)	Active time/match (s)	Energy (Wh)
RasPi + FPGA	1	5	6	100	0.166
Dynamixel (AX-12A)	2	9	8.1	20	0.09
DC Motor (Faulhaber 2642)	2	24	28	100	1.5
Electro valve MH3	3	24	5	40	0.166
Diaphragm pumps 3003	5	3.5	0.743	40	0.04
Odometer AMT103	2	5	1	100	0.055
LIDAR A2M8	1	5	2.5	100	0.0694
Total			101.4		2

Table 5: Electronics power consumption

It was considered that the robot has to be powered during three matches. According to these assumption, the two batteries together pack must be able to supply at least 6 Wh with a maximum power at least of 101.4 W. We choose two batteries in the components provided by the university (Ref: B12003). The main characteristics are :

- Voltage: 12 V
- min Capacity : 3200 mAh

The minimal energy contains in this battery is $3200\,\text{mAh} * 12\,\text{V} = 38.4\,\text{Wh}$. We could theoretically have an autonomy of 18 consecutive matches.

6 Programming

In this section, Antoine, Martin, Alexandre and Arthur developed the scientific and technical material; Antoine drafted the text; and Athur revised the text to improve its integration to the whole document.

6.1 Description of the general structure of the program

This section focuses on the micro controllers represented in Figure 36 and the implementation of the code. It gives a global overview of all the components that bring the robot and the experiment alive. It includes two raspberry PI, two De0-nano board and one LT24 screen. Raspberry PI contain the main code in C++, FPGA enable dedicated hardware electronics and LT24 is the interface human-robot, specially for official matches, being the display of the scores and the way we would select the colour of the robot and the strategy to follow. We decided to connect the LT-24 display on the experiment because the FPGA on the robot had no more free header.

The two raspberry PI communicate through WiFi, where one creates a hot-spot (server) while the other one connects to it. Once connected to the same network, information between the two can be exchanged through LAN. While the Raspberry in the Kraken is really a processing unity for the matches, the one in the experience is much more a communication interface with the other. In both cases, those board computers also communicate with their own slave De0-Nano through SPI. The PIO are also added into the FPGA to exchange data with the screen or with the interface card, that later routes data for all the actuators of the robot.

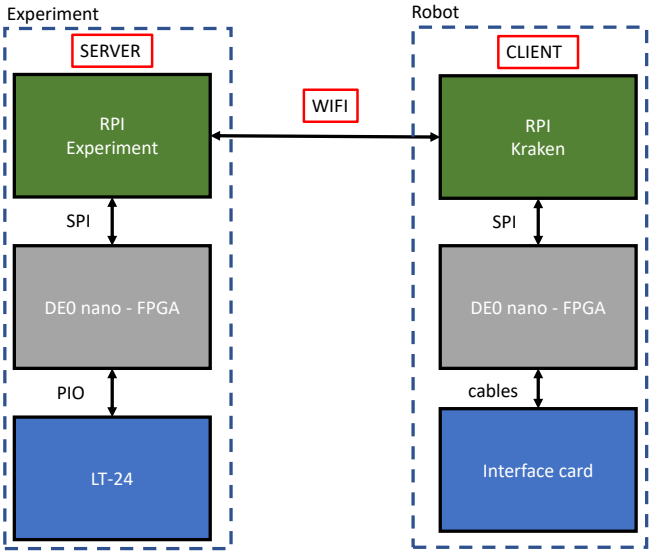


Figure 36: Schematic overview of the micro controller used

6.2 Software on the Raspberry PI

We use two Rapsberry Pi, the being one for the experiment and the other as the brain of our robot. It communicates to the robot with several busses such as a CAN bus to set the motors speed. UART bus enables to control the Dynamixel and SPI to communicate with the FPGA. The Kraken also creates a hot-spot to interact with the experiment station. We use two raspberry that are programmed with the threads to allows several actions to take place. The code is written in C++. The choice is mainly motivate because we had a base all ready written in C++. Furthermore the library for the LIDAR is also written in C++ thus for convenience it was easier to stick with it. It is also a language faster the python.

6.2.1 Software functions for the robot

We will start to describe briefly the organisation of the code before deeping into its implementation. In order to keep the code as clear as possible we started with the base code provided the robotics class, keeping the robot class hierarchy proposed there. The main object structures used was called CVS, which holds all main attributes (also defined as classes) of the robot: inputs and outputs, path, strategy, speed regulation, etc. It provided the code with a good modularity, which enhanced the development process.

We are now focusing on the implementation itself, by describing all the threads and their functionalities developed. Figure 39 gives a good overview off the number of tasks and their usage within the program. We use the library *Pthread* to do multitasking. The good point is the easy implementation because it only requires a few line to be implemented. However the priority can not be set and the type of scheduling can not be changed, making it not possible to implement real time constrains in the code. In order to improve the performance, we manually set tasks to sleep for some time after execution, and we would often turn on or off tasks according to the conditions in which we were using the robot (eg. turn on the keyboard task for testing, but turn it off during the matches). By those means, we managed to achieve satisfactory result as far as we could see in our tests, but without the real time constrains there is no objective implementation in the system guarantees robustness in this sense.

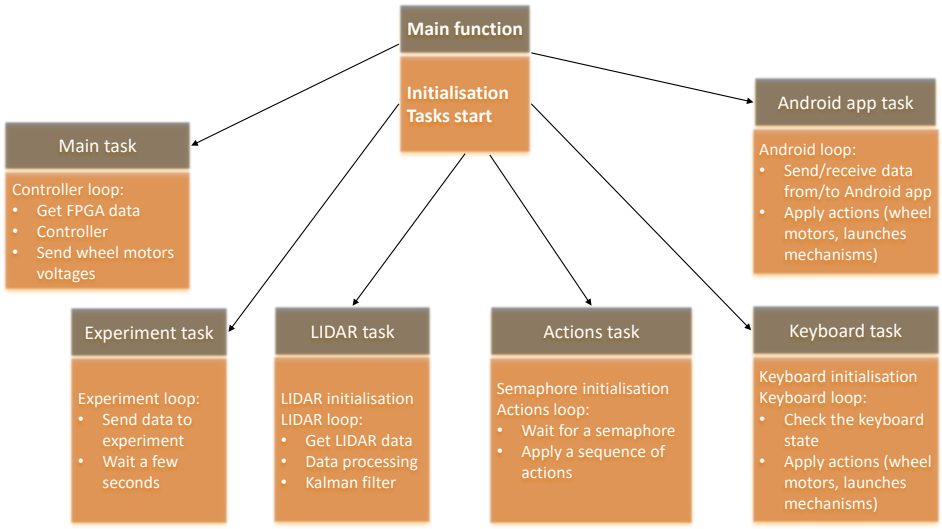


Figure 37: Tasks division

Once the match code is ran, the Kraken structure and the tasks configured to run are initialised.

- **Main task** is mostly dedicated to the mobility data and controller. First it will request from the FPGA, by SPI, the encoder pre-processed data, obtaining the angular speed and position from each odometer and motor encoder. Given this data it can update the position and thanks to a path planning algorithm it can compute new speeds for the wheels and send the current command through the CAN bus to the CAN card.

- **Experiment task** is written for the communication between the experiment and the robot. The raspberry of the experiment is the server of this communication, while the raspberry of the Kraken is the client. In this task, after the launching jumper is pulled off, the robot waits for a few seconds for receiving settings chosen by the user in the LCD of the experiment. When the communication is successfully established, it receives such data and sends the command to launch the experiment to the server. If the data is not transmit, it will wait until next time to try again.

- **Lidar task** is the task that will collect the data from the lidar and signalise that the triangulation can be calculated. Once it is done, its result is merged with position obtained from the odometers, by means of a Kalman filter. Besides detecting the land-markers, the lidar also can detect the opponent and introduce one new component in the path

planning algorithm, which is treated as an obstacle..

- **Action task** is a collection of predefined sequences for giving signals for the actuators, following sequential logic. Most of the sequences have already been describe earlier in the pneumatic section, but others include some commands given to the dynamixel in series with the pneumatic actions. We point out that, since no sensor was used for the pneumatic actuators, there is not feedback signal about whether they have or not been properly actuated, and this part of the system is considered to be totally in open loop controlled with timers. It was a design option due to the big number of extra GPIOs pins required that the boards available could not support. Another important aspect of this task is that it requires communication with other tasks, which is done with semaphore objects. This signalisation is import to decide when a certain action should be launched according to the position in the path planning, or whether the robot can start moving again if the action does not require the robot to blocked in the same position.

- **Android app task** is not aimed for the contest but it is a nice application for demonstration of the robot to the public, and also quite useful for driving the robots during the tests for the strategy. It send and received data from a smart phone using the same protocol as the experiment.

- **Keyboard task** gives the possibility to control the robot with the Keyboard, which again was useful especially for placing the robot in the initial position during the strategy tests, without need of a person to move manually. We use the *SDL library* to have access to it.

We have described so far how the program is ran and how its different tasks interact with each other, but we will now detail a little more how the program is structured. In the figure 38, a schematic description of two structures is given. The one on the right summarises each geometrical parameters of the robot and the map so that we can modify the value rapidly. Each function that needs it has also a easy access thanks to the CVS class, that includes all the robots definitions. The other structure is more important and contains substructure that contains parameters for specialised function.

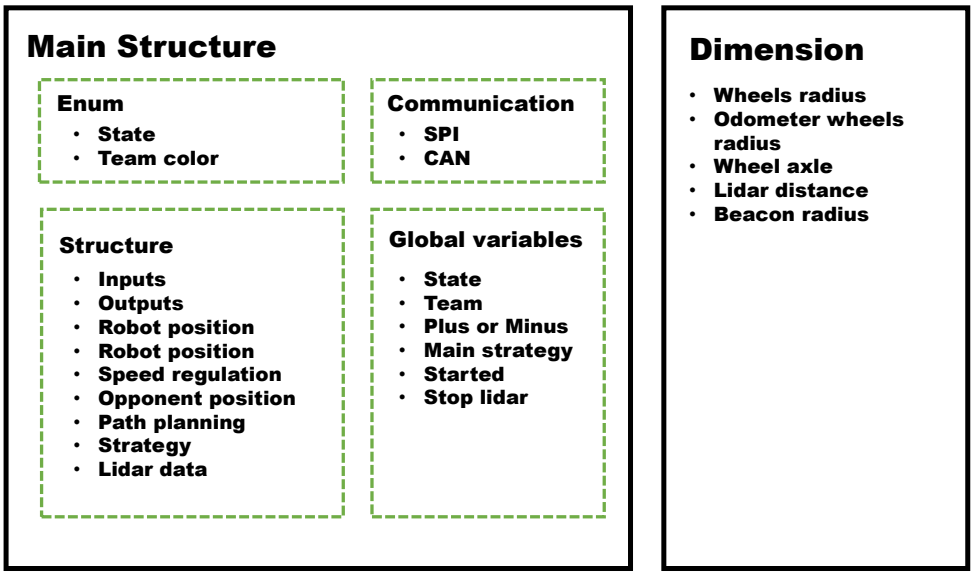


Figure 38: Main structures

Enum simplifies the reading in the code, global variables gives a control on the all program. For example state switch sub-function in term of the game flow. Plus or minus is keeps the path planning target symmetric depending on were the robot starts, etc. We will explain a bit deeper each subs structure.

- **Inputs** is a structure that collects the data usefull for the controllers such as the odometers angular position.The internal time is consider as an input.

- **Outputs** are results or directives given by the robot. For example the command to set the speed for the wheel or the score.
- **Robot position** hold every data concerning the position estimated, either by the odometers or the lidar.
- **Speed regulation** refers to the parameters of the motor details in the section about the low level of control.
- **Opponent position** are the data that will describe the position of the opponent.
- **Path planning** hold every target and walls position. Also repulsive and attractive gain and some useful parameter to the algorithm.
- **Strategy** tell what state has been achieved over a match in order to know what is the next step.
- **LIDAR** memorises every distance to each beacons and the number of beacons to use. Also if the opponent is detected and it distance.

6.2.2 Software functions for the experiment

On the experiment side the code is much simpler. The raspberry PI is mostly used for it integrated WIFI circuit. Also the UART bus is interesting to launch the experiment. The code is also written in C++ to be coherent with the Kraken code. We had already a python code to write on the LCD display, thus we kept it and integrate it inside by executing the file with the *system* command. It is composed of only three tasks. Two of them are created by the main function and this two are dedicated two the communication. The last one is executed once when the experiment needs to start.

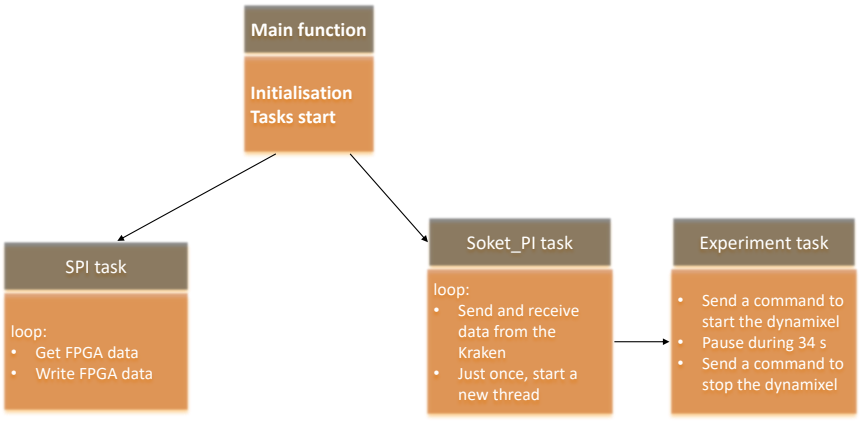


Figure 39: Tasks division

- **SPI task** collects and sends the data related to the LT24 screen. The data receive are the settings and the data send is the score.
- **socket PI task** is a loop that keeps sending and receiving informations from the Kraken. We used also the *socket library* for this purpose.
- **Experiment task** is implemented as a timer that starts, waits and stops the dynamixel.

6.3 DE0-nano

This board includes an FPGA which is use to create dedicated hardware such as timer or counter for the odometer. It uses the concept of parallelism because several circuit can proceed data from different sources in the same time. For the experiment a NIOS 2 is also added to the FPGA to run the LT24. To program the devices we used Quartus and the code is written in system verilog.

6.3.1 Hardware functions for the robot

We mainly implemented 3 modules :

- SPI module : it allows us to share informations with the Raspberry Pi (considered as the master) such as the incremental speed / angle of the encoders of the wheels / motors. The signal is a 5 byte buffer initialised by the master. The data of 32 bits are overwritten in the first four packet and the fifth one allows us to choose in witch register we would like to process. Indeed, the last bit of this packet tells the FPGA that he should write or read the informations.
- Encoders module : this module allows us to get the speed / incremental rotation by processing the 2 quadrature signals A and B send back by the encoders. We implemented a FSM to know in which direction the wheel is turning depending the channel that has the first positive edge. To compute the speed, we had to increment a counter for each positive edge synchronised on the frequency of a clock. The incremental rotation in degree is defined as : $R = \frac{Count*360^{\circ}}{N}$ with N is the number of tick for one turn of wheel.

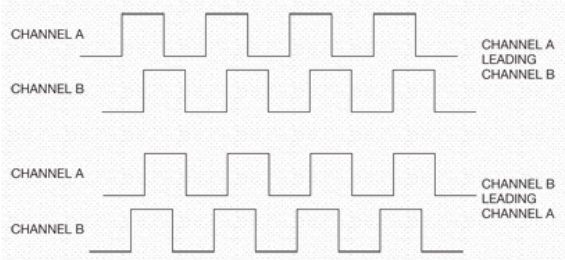


Figure 40: Signals from the encoders

6.3.2 Hardware functions for the experiment

We implemented 3 modules :

- SPI module: Same characteristics as develop in the previous section.
- NIOS 2 processor: Using Quartus we add a NIOS in the FPGA to be able to control the LT24 display. PIO are also use to communicate and receive informations from the screen.
- Connection: The UART-bus is controlled by the Raspberry PI but the phisical connection to use it is on the DE0-nano board.

6.4 LT24 interface and software on a NIOS 2

The code for the LT24 runs on a NIOS 2 processor. The main goal of the screen in the overall project is to be able to send basic settings to the Kraken and to received it score. The scope of the software is limited to the communication to the FPGA, other sections explain the wifi communication. The basic template comes from a template given by teaching staff a few years ago.

The software handles several actions such as display informations and get interactions with the touch screen. Also it enables the user to receive and send data to the FPGA using PIO.

The architecture of the software is based on a single task 42. This task can be divided in 2 parts, initialisation code and loop code.

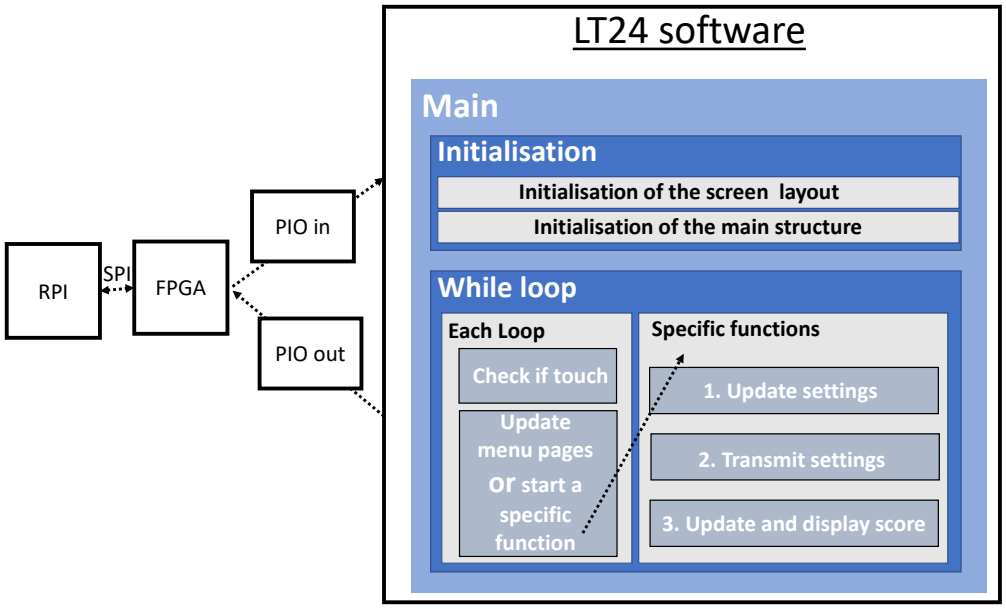


Figure 41: Software LT24 general architecture

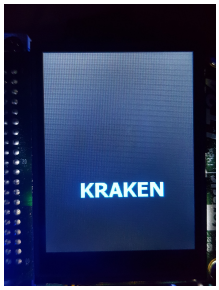
First the screen is set and the main structure is initialised. A schematic overview of the structure is given here below. Variables are related to the interactions and displays of the screen. Data to send the setting and to receive the score are also in this structure to facilitate the access between functions. The last part is composed of several structures containing all the in formations concerning menus (title, name of the button, numbers of buttons, etc).

LT-24 main structure

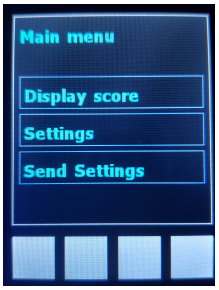
- **Variables for the screen set up**
- **Variables for the data to communicate**
  - **Number of opponent**
  - **Number of beacons**
  - **Team color**
  - **Strategy**
  - **Summarise of the settings on for digit**
  - **Score**
- **Structure to save all the menu**
  - **Name of the button**
  - **Number of buttons**
  - **...**
- **Variable to tell if the structure is initialised**

Figure 42: Software LT24 general architecture

The second part of the software is a loop. It keeps checking if a touch is produced by the user. Either it updates the menu or it can launch a dedicated action.



(a) Welcome Screen



(b) Main menu screen

Figure 43: LT24 Welcome and Menu screens

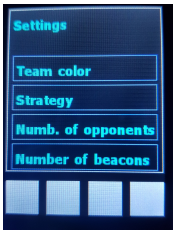
We will now focus on the three actions :

1. **Update settings** (fig 44a) It is possible to modify four settings, each are saved in the global structure :
   - Team color (Purple – Yellow)
   - Strategy (0 to 7)
   - Number of opponents (0 to 3)
   - Number of beacons (0 to 7)

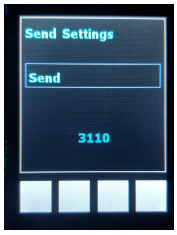
   The settings are transmit with one number computed with the following rule :

$$settings = teamColor + 10 * strategy + 100 * nbre_{opponents} + 1000 * nbre_{beacons}$$

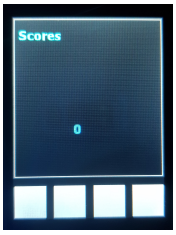
   Thus we have a number composed of 4 digits, where each digit has a meaningfull value. When a value is changed a message « OK » is displayed to tell the user that a change has occured.
2. **Transmit setting** (fig 44b) This function enables the user to modify the value written in the PIO_{out}. It could be done automatically every time the configuration is changed but it gives more control to the user to be able to decided when to send the configuration. Also the setting code is display thus the used can check it is the right one before applying the change.
3. **Update and display the score** (fig **??**) When the display shows the score, then the PIO_{in} is read The display changes and shows the score. If the PIO value is modify by the FPGA then the score display is automatically updated.



(a) Settings



(b) Send settings



(c) Score

Figure 44: Experiment photography

6.5 Wifi communication

As explained previously, we have to send informations such the score and the strategies to the RasPi on the robot from the one dedicated for the experience. This communication is realised between a client (RasPi of the robot) and a server through a WIFI network .
On one hand, the server is waiting for a connection on a port in the local network created by itself. Once a client is connected to the server, this one create a socket to send informations such as the strategies we choosed on the LT24 to the client.
On the other hand, the client establish a connection to the socket and decides when the transfer of the score performs by the robot is executing and sending to the server.
In other to communicate with the right server, an static IP address is assigned to the raspberry of Kraken (the one that is creating a WIFI hotspot).

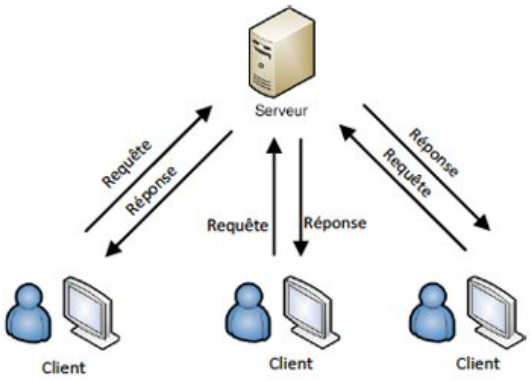


Figure 45: Communication socket

6.6 Android application

We created an application on smartphone to be able to manually control different tasks of the robot such as sending the robot wherever we want on the map, taking the 6 tokens on the dispenser by activating the double stroke actuator / 3 electrical pumps and releasing 3 tokens them our elevator. We could also retrieve the Goldenium token by setting on the single piston if we would have implemented the task to make the rotating arm turn on the side of the robot.
According to what has been told concerning the WIFI communication, the client, here, is the user and his smartphone and the server is the RasPi present on the robot. Indeed the client is defined as the device that chooses the time of the data transfer, whenever it is for sending or receiving data. Here the Android app chooses when it sends the the commands, and it then receives the score on its screen.
Here is the graphical interface of the android application One can notice we can control the displacement of Kraken with a *Joystick* mode but we implemented another mode where the displacement is controlled by the tilt of the smartphone (*Tilt* mode).
To make the application working, the user has to specify on which port and IP address he would like to connect and send the data (see Figure **??**). It can then connect to the robot and immediately stop it.

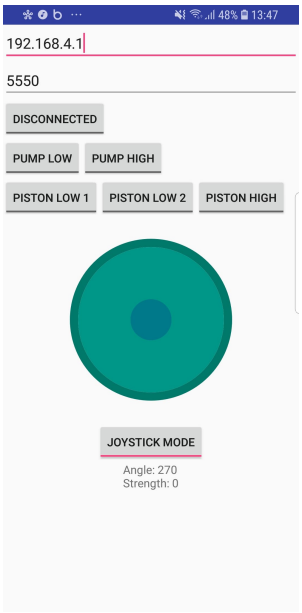


Figure 46: Android application

7 Final assessment

7.1 Pros / Cons of Kraken

In spite of the good results achieved by the Kraken, while assembling and testing it a number of points showed up to need improvements. In this section, we make an assessment of general and specific characteristics of the robot, pointing out which were its strong and weak aspects. We also propose some hypothetical modifications, so that future works can benefit from our reasoning and experience.

7.1.1 Mechanical and general aspects

The mechanism proposed for grabbing the atoms proved really fast and reliable, specially after the introduction of the pumps, which were not predicted in the beginning of the project. The size and power consumed by such pumps have fit well in the requirements, but the fact that they were not purchased at the appropriate time have brought them to have a considerable unpredicted impact in the budget (approximately 9%). The double stroke pneumatic pistons were a good choice made at the design phase, and the solution provided by our sponsor FESTO suited well those requirements in a compact manner.

The elevator system also attended the specifications designed, but the options made brought some difficulties to the implementation phase. The Dynamixel is a robust servo, but it does not suit for cases were the set-points in position require wide angular movements (higher than 300°, in the case of the AX-12 series). We believe that a step motor with a controller board would be more appropriate for this task, since they can provide high torques and do not require much resources from the processing system. This would be a first modification if the robot could be rebuilt.

Concerning the structure of the robot, it was visible that it would not stay straight, which interfered considerably in the results obtained from the lidar, placed at the top of the robot. We attribute such behaviour to the association in series of the coupling rods with wood sheet and uncontrolled hole diameters. We suppose that better results could have been achieved by the use of thin aluminium plates and extruded 'L' profiles for connecting vertically the different levels of the robot.

When it comes to the mobility, we realised that during an intense testing phase the rubber of the driving wheels was considerably damaged. Due to the lack of modularity in the assembling of the robot and to the position of the wheels, they could not be changed easily, which lead to a significant impact in the system performance and speed. For this reason, we suggest that other teams take such aspect into consideration in the design phase.

7.1.2 Hardware and electronics

The processing unit (Raspberry Pi and DE0-Nano) proved to meet our needs, enabling us to properly control all the actuators and most of the needed sensors. All the high level structure was computed by the board computer, whereas low level control and signal processing was split among the former (e.g. control of wheels, lidar data) and the later (e.g. speed and position calculation from encoders, command signals for pneumatic).

The only limitation seen in this aspect was brought by the processing of the data from the lidar. As this sensor fed the Raspberry Pi with a considerable amount of data, about 0.2s were needed for extracting the information needed by the mobility system. For this reason, some software optimisation or dedicated hardware is would have to be implemented in a rebuilt of the model, and they are also suggested for works using such category of sensor.

The most significant issue that we had with the robot was the electronic connectors, though. Concerning the data transmission, we have opted for board-to-board connector that must be individually soldered in the wires, which was impractical. After that, we moved to another category (rather more expensive) that enabled an easy mounting on the wires in the connector. In spite of the practicality, we had several problems of false connections involving it, which consumed a considerable working time. Therefore, a selection of new connectors and the redesign of the PCB is among the top list of changes needed for the version two of the Kraken. For teams following similar work in the future, we strongly advice that before starting the PCB design, teams should make a good benchmark of which type of connectors they will use between the developed boards and the electrical components used. One other aspect that could save some working hours is giving preference for buying connectors board connector whose cables can be also bought ready to use.

Also with respect to the boards we have made, another redesign change to be made is their position inside the robot for facilitating the access to the connectors for either probing and changing wires. The path of the wires inside the robot is also an important aspect for making assembling and maintenance easier, and it has also to be reviewed..

7.1.3 Software aspects

In spite of not running in a RTOS nor in an optimised operational system, the application developed was sufficiently reliable, specially when tested in the conditions of matches. It means that when the mechanical and hardware components were properly set, the general software would perform correctly and predictably, meaning that no significant real-time issues were observed. The main difficulty faced was with regard to the lidar, but by limiting its use acceptable results were achieved.

We attribute such behaviour to the task structure implemented, which could be easily changed according to the scenario in which the robot was used. By simply changing a few defines, we were able to switch on and off tasks used for testings. It would both optimise the testing time and reduce the time needed to adapt the code for the match conditions.

Nevertheless, trying an RTOS would be something to be studied in further details. If it enables the robot to use the lidar more often, it would signify an important improvement in the localisation and mobility system, which was one of the big issues we have faced.

Another aspect that proved to be useful were the tools for command remotely the robot, from a notebook keyboard or from a smart-phone. The main advantages of such implementations were that we were able to bring back the robot from its final position during the test to the initial one without the need of a person to carry it, and also the possibility of testing if a problem was the result of some software implementation in the autonomous program or if it was caused by something else (e.g. low battery, false contact, etc.). We could notice in the other groups a huge lack of time when they so often walk around the table to bring back their robot.

The LIDAR used only with 3 beacons seemed to be a big challenge at the start of the year (and even impossible from certain people), but we managed to deal with these few marks to obtain a good precision. This shows that it is not always useful to stay straight in the line, it is important to broaden his horizons. So it is a good satisfaction from this point.

7.2 If we had to do again

- Taking a longer time at the first quadrimester to review the design of the structure of the robot / defining in cleaver way the right place where should still the right component.
- Taking a longer time on the orders for mechanical/electrical components to be sure to avoid unpleasant surprise.
- Review the design of the most critical PCB : the interface card
- Use better electrical connections to save a precious amount of time.

7.3 Evaluation of the group

To better organise the work, we favoured to use Slack instead of a group on Facebook in order to communicate each other in case of someone misunderstood something.

To dispatching the work between us, we decided to use Trello, a software specialised in the project management.

Moreover, each Wednesday,before the consultancy, we used to discuss about our achievements during the week to have a better overview of the remaining work to respect the deadline.
Hereunder is a table 6 with an personal evaluation of each student for the whole project. The result is noted on 5 points.

	Communication	Organisation	Ambiance	Work repartition	Quality of work
Alexandre	3	4	3	3	4
Antoine	3	4	3	3	4
Arthur	4	3	3	4	4
Martin	3	4	3	3	4
Miguel	4	4	3	3	4
Valentin	3	4	4	3	4

Table 6: Evaluation of the group

8 Conclusion

In this section, Valentin drafted the text and Martin reviewed it.

This whole year was mostly dedicated to this mechatronics project. We managed to improve and to apply our skills and knowledge in different domains. We first spent time to design a whole robot starting from Eurobot contests rules, covering the mechanical aspects, with our first contact with complete engineers machine design and 3D assembling. We then included electrical aspects, both in power management and signal electronics. After half a year spent designing our robot, we started building it up, facing and solving different issues. We improved it, changed some parts of our preliminary project. Finally, we defined a strategy for the contest, and started coding the robot, adding features that were thought to us during our courses during this year. We managed to implement a path planning, in addition of a localisation and avoidance system, thanks to the sensors we had implemented.

After this school-year working on this project, we finally arrived to the Belgian Eurobot contest, Robotix's; We earned a wonderful second place. This achievement is the result of a year of hard work and motivation, which got us closer than any time to the true aspect of team work, project management, in a word, engineer job.

References

[1] Pierlot, V., and Droogenbroeck, M. V. A new three object triangulation algorithm for mobile robot positioning. Available in: www.telecom.ulg.ac.be/publi/publications/pierlot/Pierlot2014ANewThree/, 2019. Accessed: 2019-05-02.

A Algorithm of triangulation

Supposing that the three beacons coordinates are well known:

$$B1 : (X1, Y1, \alpha_1)$$
$$B2 : (X2, Y2, \alpha_2)$$
$$B3 : (X3, Y3, \alpha_3)$$

These angles are defined with respect to the position of the robot on the map θ.
We are looking for the position of the robot $R = (X_R, Y_R)$ and its orientation θ.
We first translate the world frame into the local frame of the second beacon B_2 and we compute the new coordinates of each of the three beacons:

$$X1' = X1 - X2; \qquad Y1' = Y1 - Y2;$$
$$X3' = X3 - X2; \qquad Y3' = Y3 - Y2.$$

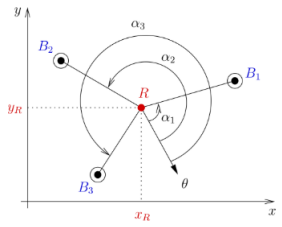


Figure 47: Algorithm for triangulation [?]

Then we look for the center of the locus (it is an arc of circle) where the robot should stay in order to see 2 beacons with a constant angle ϕ. In Figure 48, we define $\phi = \alpha_2 - \alpha_1$.
With trigonometry, we compute the coordinates of the center of the 3 circles in the new frame.

$$X'_{12} = X'_1 + T_{12}Y'_1; \qquad Y'_{12} = Y'_1 - T_{12}X'_1$$
$$X'_{23} = X'_3 - T_{23}Y'_3; \qquad Y'_{23} = Y'_3 + T_{23}X'_3$$
$$X'_{31} = (X'_3 + X'_1) + T_{31}(Y'_3 - Y'_1); \qquad Y'_{31} = (Y'_3 + Y'_1) - T_{31}(X'_3 - X'_1)$$

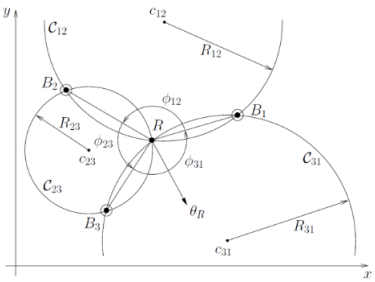


Figure 48: Algorithm for triangulation [?]

where $T_{ij} = \cot(\alpha_j - \alpha_i)$.
We introduce 2 parameters to obtain the final result of the position of the robot based on the power lines and the power center of these three circles.

$$K'_{31} = X'_1X'_3 + T_{31}(X'_1Y'_3 - X'_3Y'_1)$$
$$D = (X'_{12} - X'_{23})(Y'_{23} - Y'_{31}) - (Y'_{12} - Y'_{23})(X'_{23} - X'_{31})$$

Finally we compute the final robot position:

$$X_R = X_2 + \frac{K'_{31}(Y'_{12} - Y'_{23})}{D}; \qquad Y_R = Y_2 + \frac{K'_{31}(X'_{23} - X'_{12})}{D}.$$

B Detail of pneumatic components

Here is a detail of our pneumatic components. We received them from Festo. They kindly accepted to be a material sponsor. Therefore, the part number of the components is from Festo.

Commande FESTO		
	Description	**Référence**
Partie ventouses	Ventouses	VAS-40-1/4-SI-B
	Clapets anti-retour	H-QS-4
	Silencieux	UC-QS-4H
	Distributeurs	MHE2-M1H-3/2G-QS-4
	Fixations ventouses	LJK-1/4-I/I
	Raccords	QS-G1/8-4
	Connections T	QST-4
Partie verrins:	Verrin	ADN-16-80-A-P-A-Q
	et les raccords:	QSML-M5-4
	limiteur de débit	GRLZ-M5-QS-4-D
	Accouplement	KSZ-M6
	Fixation	HNA-16
	DGSL	DGSL-6-30-PA
	DGSL	DGSL-6-50-PA
	et les raccords:	QSML-M3-4
	Distributeurs (avec silencieux)	VUVG-LK10-M52-AH-Q4-U-1H2L-C2-S
	limiteur de débit	GRLA-M3
	raccords	QSM-M3-4
	cables	PUN-4X0,75-BL

Figure 49: Detail pneumatic components