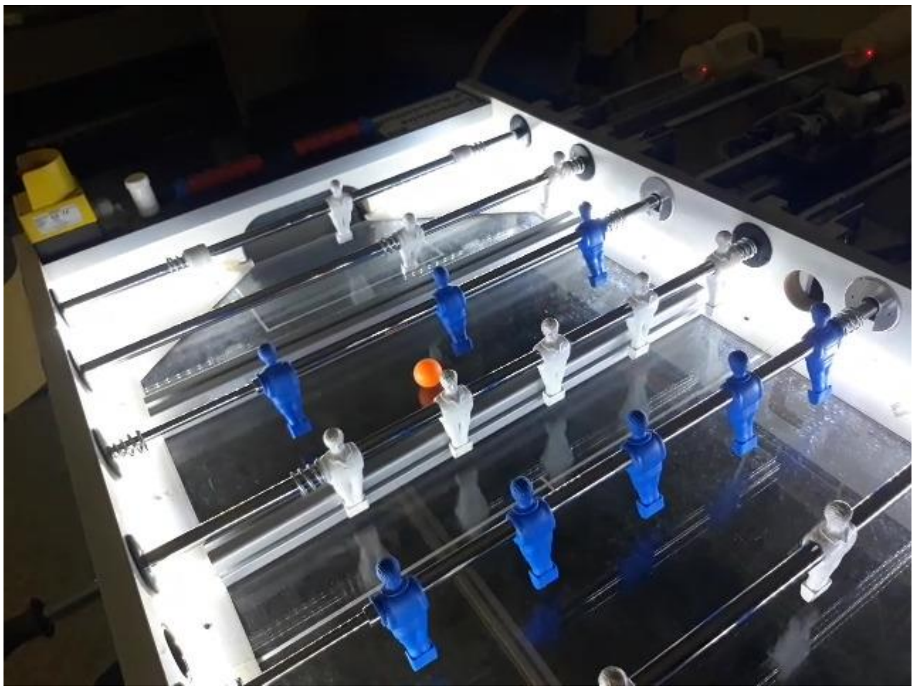
Laboratoire d'automatique

Table football: automated passes between players

Supervised by Christophe SALZMANN



Isaac BERNARDINO DINIS
Autumn semester 2020

1 Table of contents

2 Introduction

The automatic table football has been improved by students doing their semester project along the years. It is now able to compete against human players with simple algorithms and state machines but still has much to learn. Previous work [2] showed the possibility of catching the ball by damping it. This encouraged us developing a new feature for the table football: once the ball is caught next step is to perform some passes to destabilize the opponent. Hence the goal of this semester project is to be able to juggle with two players (on the same rod). As this project has already been conducted but switched to simulation due to the covid-19 restrictions, we will present a simulation to implementation approach. The report is separated in 4 main parts. First the method is discussed globally and the results are presented. Secondly some encountered hardware issues are listed. In the third part the implementation with LabView is detailed to facilitate the next student that may take over the project. In the fourth part the new printed circuits boards for connections are presented.

3 Part I Method and Results

3.1 Prerequisites

First let's define the coordinate system used along the report. The table football being a surface, 2 coordinates are enough to describe the position of the ball: x and y. The origin (0,0) corresponds to table center. For the direction of the coordinates see Figure 1. The players have two degrees of freedom, one linear along the y axis and one rotation around the rod, angle θ (Figure 2). The rotation can also be projected on the table surface, then $x = -\sin(\theta)\, L$ with L the length of a player's length.

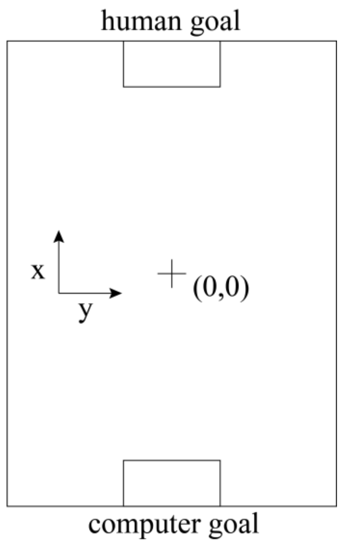


Figure 1 Coordinate system of the table football

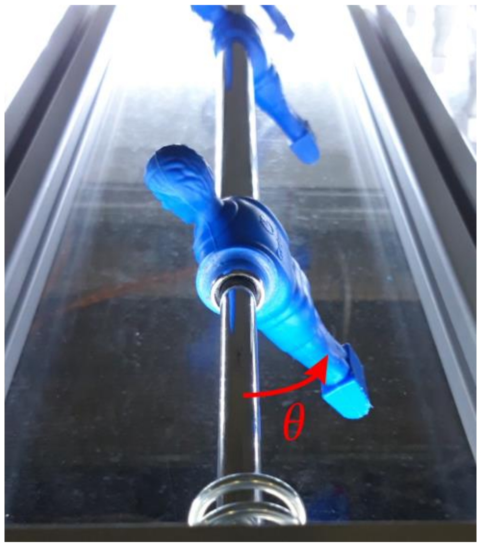


Figure 2 Rotation of a player

3.2 Method

3.2.1 Damped Juggling

This semester project was primarily conducted in spring 2020 [1], but due to covid-19 restrictions had to be switched remotely after a month. Therefore, it was decided that Maxence PERRET, the student in charge of the project, should do a simulation study of the juggling problem. The resultant method (here summarized) is the starting point of the project.

The conclusion of the previous semester project (in a nutshell) is that the ball can only be controlled as long as it is in contact with a player. Therefore, by maximizing the contact duration the desired ball trajectory is more likely to be achieved. This is done by damping the ball between each pass.

The resultant controller for the y axis has for parameters an acceleration and a maximum speed. As the ball hits the player, the latter starts to move with same velocity as the ball. It then accelerates accordingly to the input acceleration until it reaches the maximum speed. The final speed being in the opposite direction of the ball initial speed.

$$v_{y,player}(t) = \begin{cases} v_{0y,ball} + a(t - t_0) & \text{if } t > t_0 \text{ and } v_{y,player} < v_{max} \\ 0 & \text{else} \end{cases}$$

With a the acceleration, v_{max} the maximum speed, $v_{0y,ball}$ the ball speed at collision and t_0 the collision time.

The x axis controller is also inspired from the spring 2020 semester project [1]. It has for parameters two control coefficients K_p and K_v. The resulting x position is defined as:

$$x_{player} = K_p x_{ball} + K_v \dot{x_{ball}}$$

If $K_p > 1$ the ball is sent back in the opposite x direction.

The damped juggling method allows to do a few (often less than 5) passes. Its instability is due to the K_p coefficient which pushes the ball out of the desired trajectory. Next figure explains the effect.

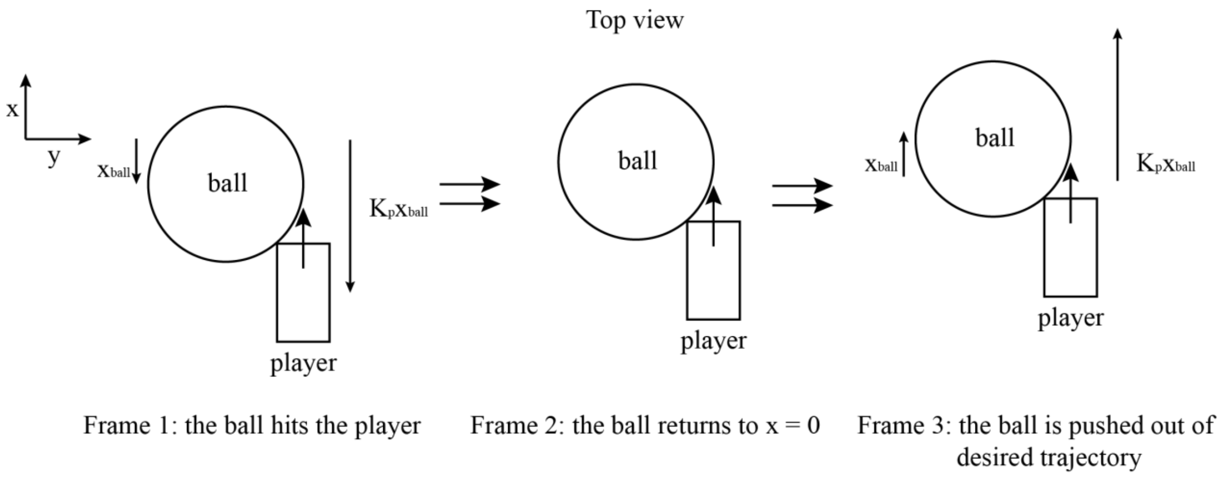

Top view
x
y
Xball
ball
KpXball
player
ball
player
Xball
ball
KpXball
player
Frame 1: the ball hits the player
Frame 2: the ball returns to x = 0
Frame 3: the ball is pushed out of desired trajectory

Figure 3 Damped juggling problematic

As the player is the contact with the ball, x_{ball} will decrease (desired effect) but once it reaches 0 mm and eventually exceeds in the opposite direction, the player (which remains in contact) will push the ball out of desired trajectory.

Furthermore, damping the ball between each pass results in a slower juggling than passing the ball with elastic collisions. For these reasons an elastic juggling method is implemented.

3.2.2 Elastic Juggling

The elastic juggling consists of elastic collisions between the ball and the player for each pass. It requires a model to provide the x_j position of the player as function of the ball offset x_b offset in order to minimize and rectify the latter. The model is constructed with the assumptions that the player's foot is rectangular, and the ball has no friction. All the assumptions are motivated by the fact that there is no absolute need of the ball displacement to be corrected in one hit as long as it is decreasing after each pass (stable condition).

By hitting the steady ball off its axis it will be deflected by an angle α which corresponds to the normal reaction force (the frictions are here neglected) [4].

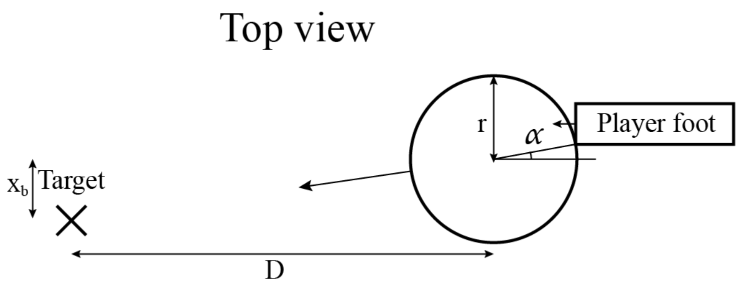


Figure 4 Foot position to hit to the ball toward the target.

From Figure 4 it appears that $\alpha = \operatorname{atan}\frac{x_b}{D}$. In this case D is the distance between the two players and x_b the ball offset.

Next figure shows the geometry needed for the relations.

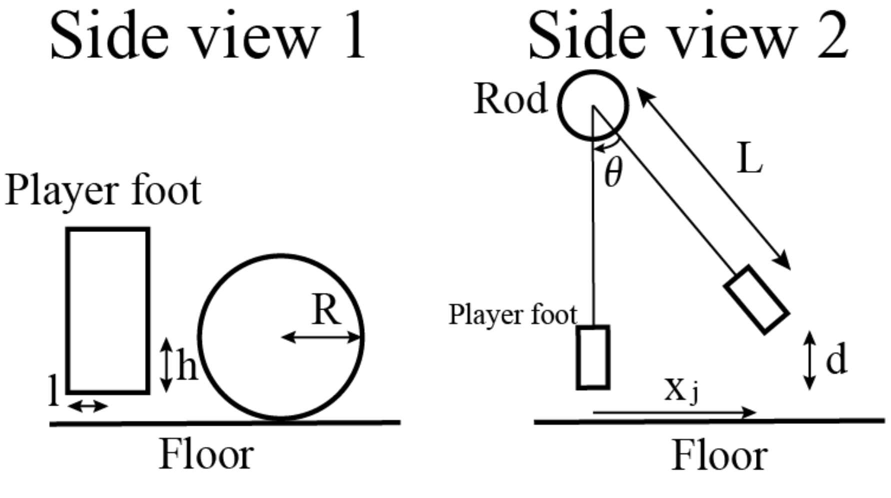


Figure 5 Geometry definitions for the ball model

Note the difference between r and R from Figure 4 and Figure 5: $\begin{cases} r = R & \text{if } d < h \\ r \cong \sqrt{R^2 - (d-h)^2} & \text{if } d \geq h \end{cases}$.
In fact, because of the rectangular shape of the foot the lowest contact point between the ball and the foot is not $d - h$ but would also depend on the position of the ball. It was here simplified.

From Figure 5 the following relation stands $x_j = -\sin(\theta)L$. By taking into account the ball offset (x_b) and half of the foot length (l), the foot displacement becomes $x_j = \sin(\alpha) * r - x_b - l$ in order to correct the ball x offset (x_b) in one hit.

The following result is obtained by partial resolution and interpolation of previous relation.

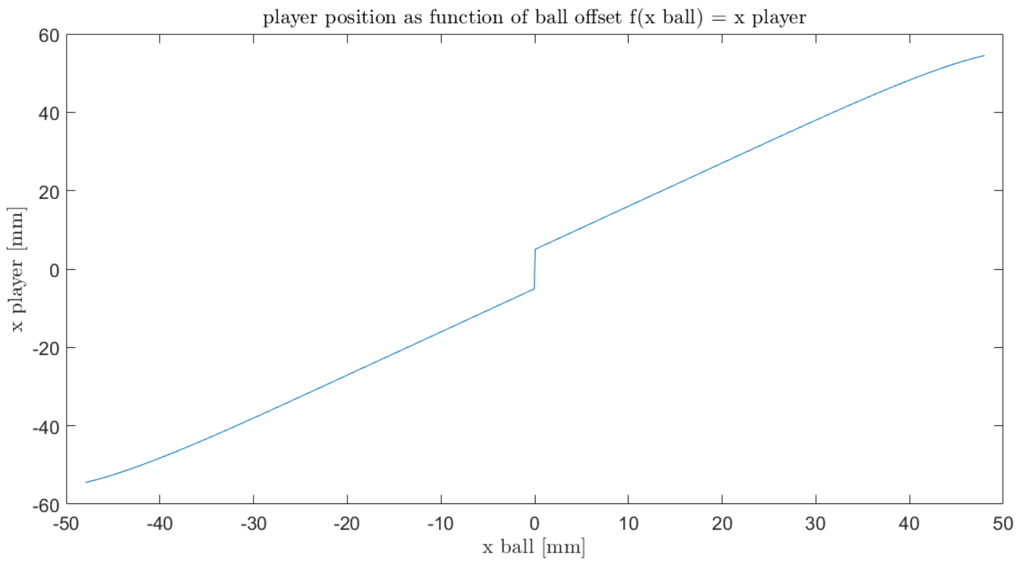


Figure 6 Player position as function of the ball offset

The vertical line at $x_b = 0$ corresponds to foot length and the curvature of the line at both ends shows the impact of the ball radius decreasing as the player foot goes up.

A linear fit is applied to the function with a different vertical intercept for $x_b < 0$ and $x_b \geq 0$. $x_j = a * x_b + b$ with b opposite sign for $x_b < 0$ and $x_b \geq 0$.

With this first relation the players are already able to perform a few passes, but to increase the robustness a correcting factor on the ball x-axis velocity must be considered. A simple proportional factor $x_j = k_v * \dot{x_b}$ is sufficient. The proportional factor is hand-tuned and the coefficient between the player foot position x_j and the ball offset x_b is slightly decreased to avoid overcompensation due to the added velocity factor. The speed is computed by dividing two consecutive position measurements by the time interval between two iterations. An IIR filter is used as moving average $y_{k+1} = \alpha * y_k + (1-\alpha)x_k$.

Finally $x_j = a * x_b + b + k_v * \dot{x_b}$ with b opposite sign for $x_b < 0$ and $x_b \geq 0$.

3.2.3 Path planning

To start juggling the player should be next to the ball. Previous work from past semester project showed the possibility to damp an incoming ball and stop it [2]. Therefore, as initial condition it is assumed that the ball is steady in the range of the player. A path planning algorithm leads the player next to the ball without touching it. Note that the left player must be on the left side of the ball to start juggling and opposite stands for the right player. The path is chosen depending on the position of the ball and the player.

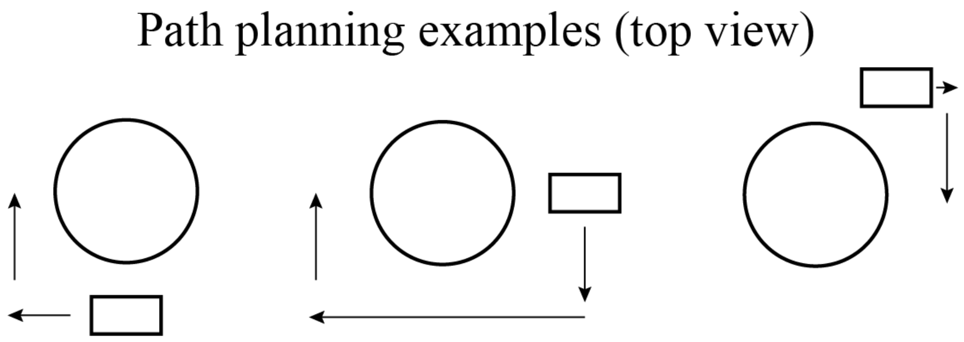


Figure 7 Path planning examples

3.3 Results

3.3.1 Damped juggling

Next the logs of a damped juggling are shown, it fails after a few passes. A video can be found here https://www.epfl.ch/labs/la/wp-content/uploads/2020/11/InitialJuggling.mp4.

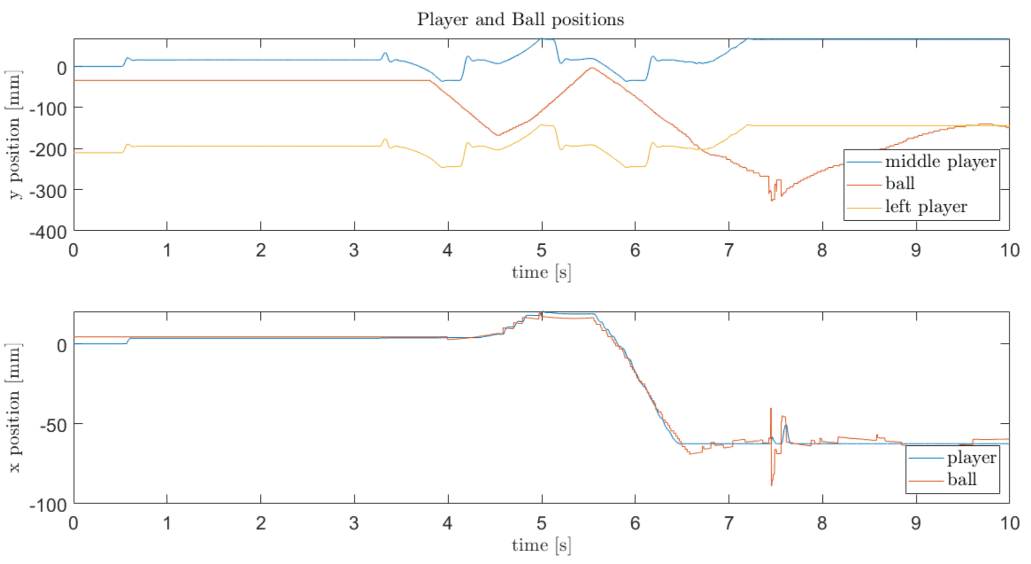


Figure 8 Plot of damped juggling logs

3.3.2 Elastic juggling

The table football can juggle over 3 minutes with the elastic method, a video can be found here https://www.epfl.ch/labs/la/wp-content/uploads/2020/12/baby_16_12_normal.mp4.

Next plots show the logs of 30 sec juggling. The path planning appears at the beginning.

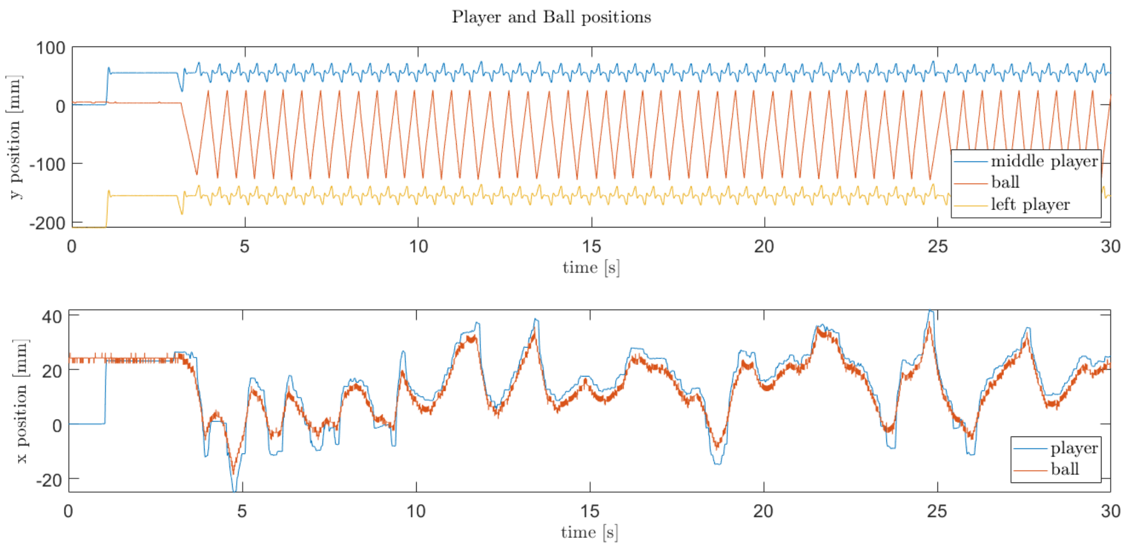


Figure 9 Plot of elastic juggling logs

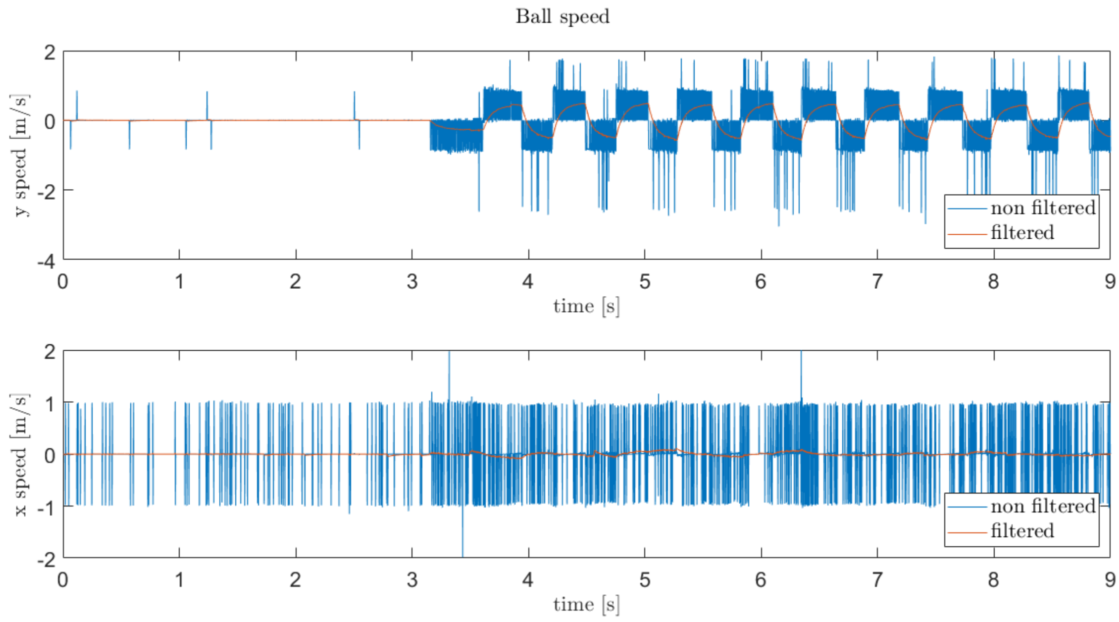


Figure 10 Plot of the ball speed during elastic juggling

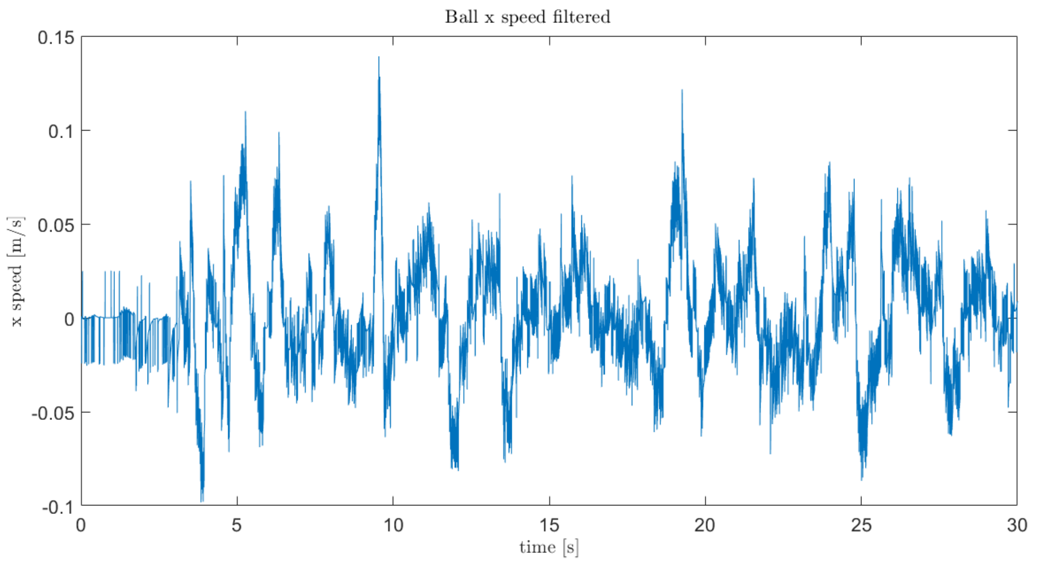


Figure 11 Zoom on the ball x speed filtered during juggling

4 Comments on actual table football issues

4.1 Current limitation

Some observation with ESCON studio showed that with the current PID gains for the angle control, the current limitation (10 A) is reached. Decreasing the PID gains solves it, however the player loses hitting ball strength.

4.2 Vision issues

The vision happens to stop working, freezes. Previous identification of the problem was made without success [2]. It was decided that the camera should be replaced.

4.3 Drift from the rotary control

The rotary control tends to drift. The cause remains unknown but here are the first investigations results. The tests were performed on two rods: the attack and middle ones since the other two were not working. Only the attack rod suffered from drift. The drift is always is in the same direction: θ positive regardless of the rotation direction. It does not depend on the number of rotations done e.g. doing 10*1200 degrees back and forth ends up with a bigger drift than a single 12000 degrees rotation back and forth.

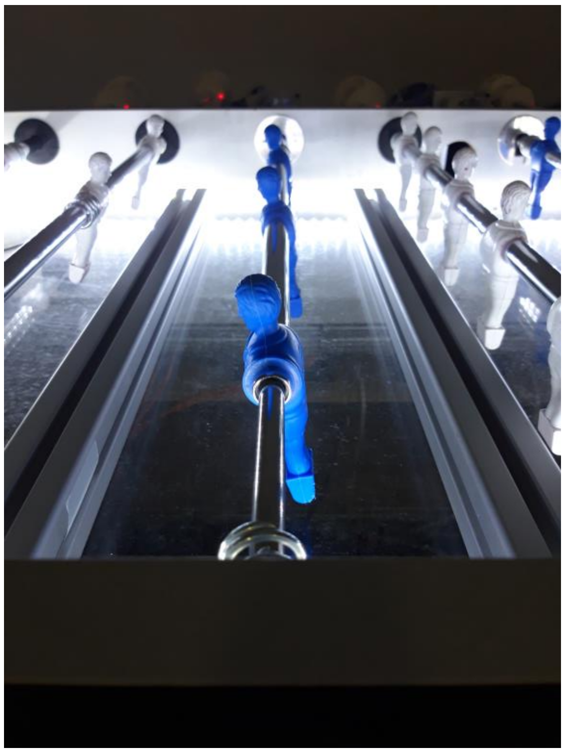


Figure 12 Photo of the drift after a 12000 degrees rotation back and forth

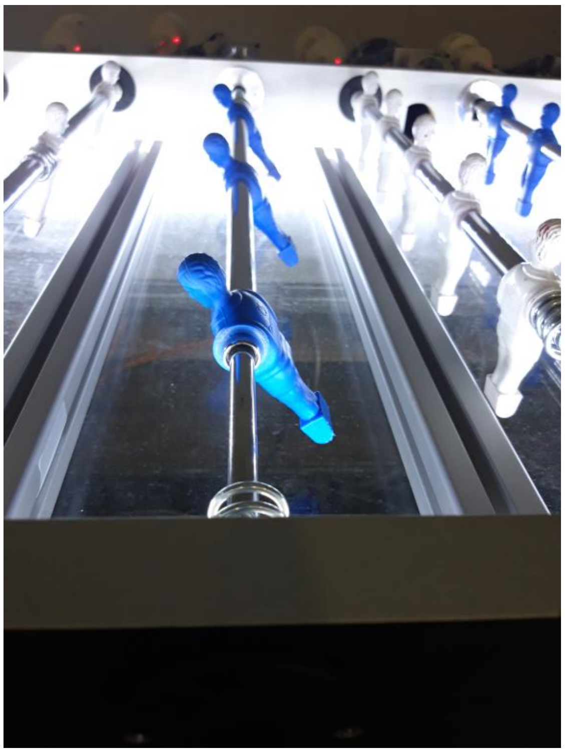


*Figure 13 Photo of the drift after 10*1200 degrees rotations back and forth*

4.4 Vision noise

The vision algorithm identifies the position of the ball discretely. With an accuracy of one pixel which corresponds to approximately 2 mm. This noise, regarding the value of the ball position, has no impact on the performance of the presented juggling algorithm, since other used approximations introduces larger errors. However, if we consider the ball speed obtained by deriving the difference in the position by the time interval between two frames, the vision noise becomes problematic. The vision algorithm typically acquires 500 frames per second (subject to change depending on the exposure). Hence a pixel difference between two frames introduces a speed of $2\,mm * 500\,Hz = 1\,m/s$. Since the speed in the x direction typically varies between ±0.1 m/s (see Figure 11) the signal-to-noise ratio is 1/10. The vision algorithm was taken as it is from previous work. An adaptive moving average filter was implemented [2], which would average over 10 frames at maximum, which could still introduce a speed error of 0.1 m/s while the ball stands still. Therefore, another filter was implemented (§5.1.1). It was also observed that the discrete noise on the vision algorithm was influenced by the ambient light as the behavior changed along the day, the discrete noise tended to increase as there was less light. No experiments were done to quantify it but that is something to keep in mind. Note that for better flexibility the speed averaging is done in the juggling loop but should be done in the vision loop. As the two loops are running approximately at the same frequency (500 Hz) it is a minor problem.

5 Part II LabVIEW explanations

5.1 General usage functions

First part is dedicated to general usage functions developed in LabView meaning that they could be used out of the juggling table football context.

5.1.1 exponential_filter.vi

Inputs: cutoff frequency (fc) [Hz], sampling frequency (fs) [Hz], input sample (input)
Output: output sample (output)

This block is a first order IIR low pass filter and is used to filter the speed of the ball. $y_{k+1} = \alpha * y_k + (1-\alpha)x_k$. With $\alpha = \cos(\omega_c) - 1 + \sqrt{\cos^2(\omega_c) - 4\cos(\omega_c) + 3}$ and $\omega_c = \frac{2\pi f_c}{f_s}$
This filter has a -3dB gain at the cutoff frequency.

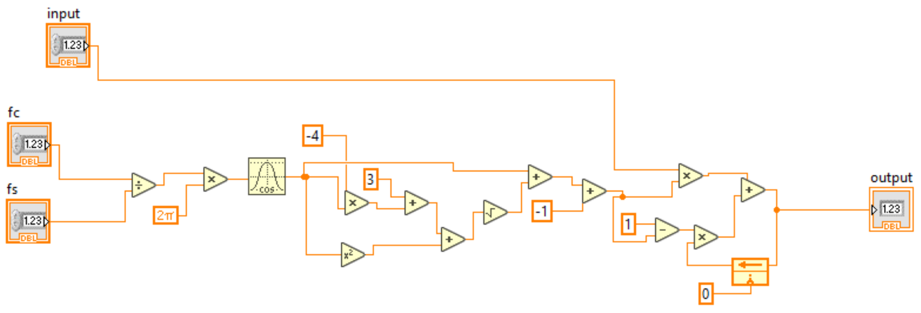


Figure 14 Low pass filter LabView diagram

5.1.2 pseudo_speed.vi

Inputs: enable [bool], speed [mm/s], sampling period (dt) [s], saturation [bool], maximum displacement [mm]
Output: position (output) [mm]

As the linear control (y axis) is in position, this block performs a pseudo speed control by outputting a position at each iteration depending on the *speed* parameter: $y_{k+1} = y_k + v_y * dt$. If *enable* is cleared the output position becomes 0. If *saturation* is set, the absolute *output* position won't exceed the *maximum displacement* set.

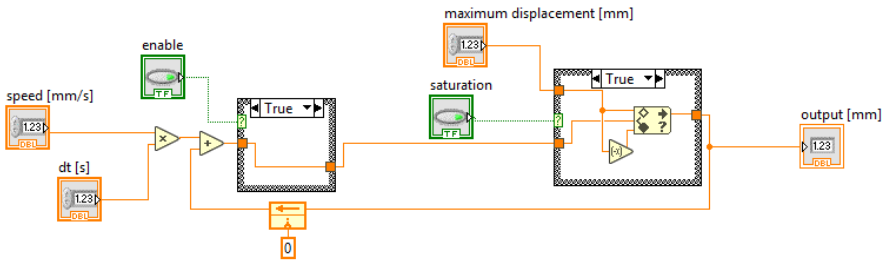


Figure 15 Pseudo speed control LabView diagram

5.1.3 timer.vi

Inputs: enable [bool], sampling period (dt) [s], time [s]

Output: out [bool]

This function is a simple timer, once *enable* is set the counter starts and is increment by *dt* at each iteration. Once the *time* set is reached by the counter the *out* flag is set. This block is used for the transition path planning / juggling. Once the player is next to ball and ready to start juggling it bypasses all the juggling conditions during the *time* set to make the first pass.

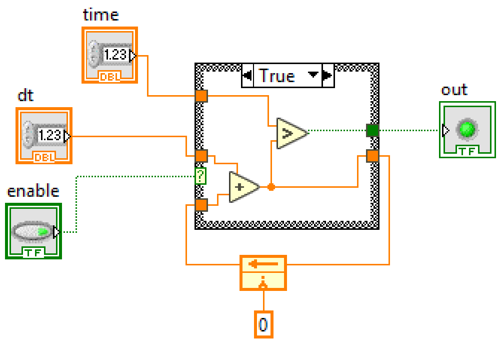


Figure 16 Timer LabView diagram

5.1.4 path_planning.vi

Inputs: start path [bool], current position cluster (x_y) [mm,mm], 2∗N path matrix (path) [mm,mm] ∗N, threshold [mm]

Outputs: reference position (xy_ref) [mm,mm], path_done [bool]

This function is enabled by the *start_path* boolean, the reference output position corresponds to first element of the *path* matrix. If the absolute difference between the current position and the reference position is lower than the *threshold*, the *path* matrix index is incremented and the reference position *xy_ref* is updated. Once the index reaches N-1, the reference position *xy_ref* holds and the *path_done* flag is set.

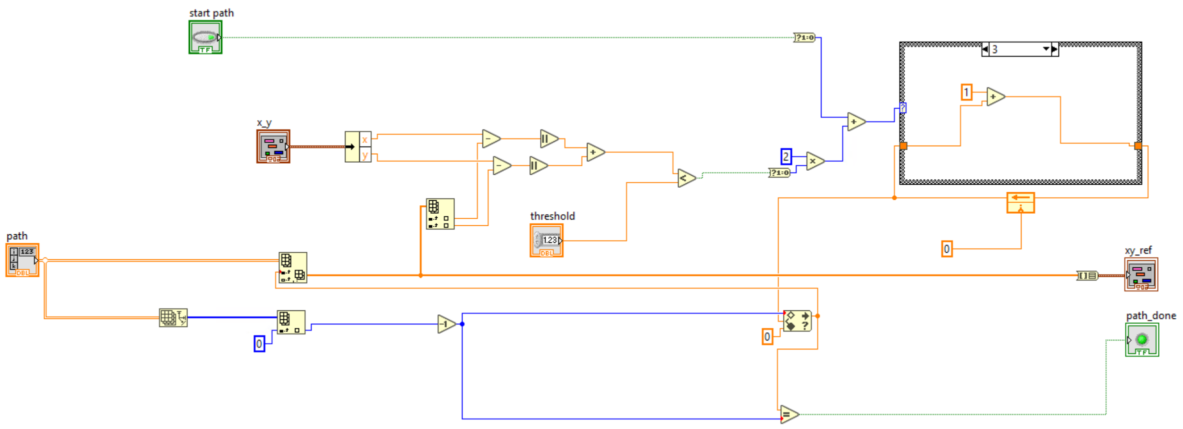
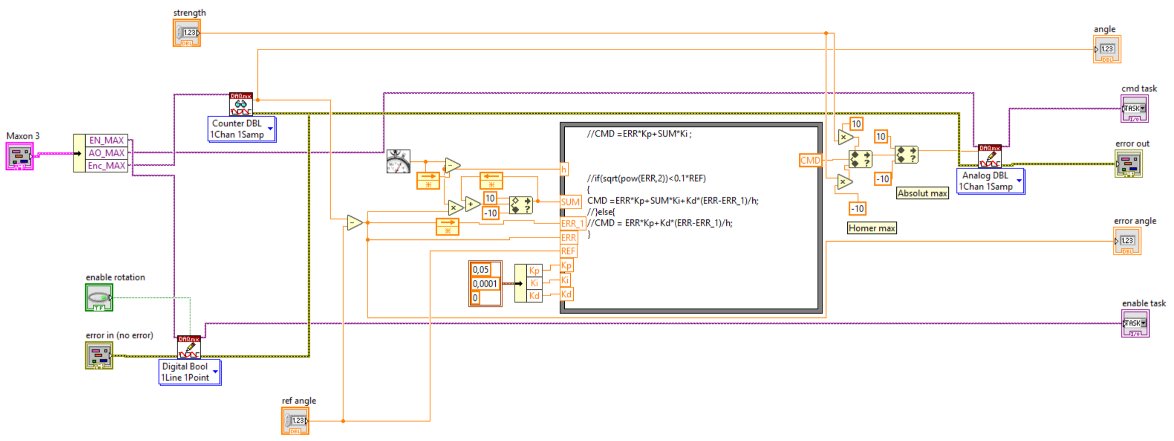


Figure 17 Path planning LabView diagram

5.1.5 ang_control.vi

Inputs: enable rotation [bool], reference angle (ref angle) [deg], Maxon task, error in.
Output: current angle (angle) [deg], error angle [deg], error out, enable task, command task (cmd task).

This function communicates to the Maxon rotational motor, so that the *ref angle* is reached by the player. The command set is computed with a PID and has a gain set by the *strength* input. The input Maxon task corresponds to a rod numerated as follow: 0 to 3 → defense to attack rod. The block outputs the current angle and the angle error. This function was taken from previous work.



```
//CMD =ERR*Kp+SUM*Ki ;


//if(sqrt(pow(ERR,2))<0.1*REF)
{
CMD =ERR*Kp+SUM*Ki+Kd*(ERR-ERR_1)/h;
//}else{
//CMD = ERR*Kp+Kd*(ERR-ERR_1)/h;
}
```

Figure 18 Angle control LabView diagram

5.1.6 lin_control.vi

Inputs: enable linear [bool], reference position (ref lin) [mm], Xenus task, pos control/homing [bool], reset error [bool], error in.

Outputs: current position (y) [mm], position error (error y) [mm], linear task, enable linear, error out.

Like the angle control VI (§5.1.5). The Xenus driver uses a direct position to voltage conversion so there is no PID here. By enabling the *pos control/homing* boolean the Xenus driver performs a homing so that the players are centered. It is needed to activate the *reset error* button after the homing to reset the y position and its error. The input Xenus task corresponds to a rod numerated as follow: 0 to 3 → defense to attack rod. This function was taken from previous work and adapted.

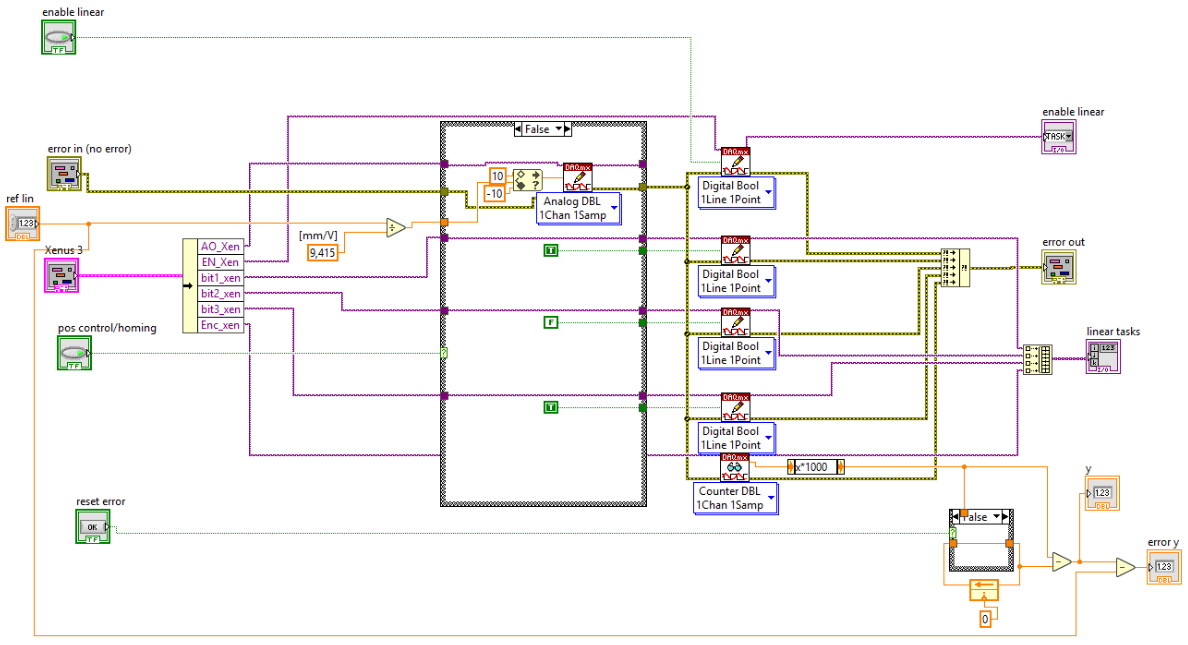


Figure 19 Linear control LabView diagram

5.1.7 xy_control.vi

Inputs: x and y reference cluster (xy_input) [mm,mm], strength, Maxon task, Xenus task, enable linear [bool], enable rotation [bool], pos control/homing [bool], reset error y [bool], error in.

Outputs: current x and position (x_y) [mm,mm], angle and y error (error_angle_y) [deg,mm], angle tasks, linear tasks, enable linear task, enable angle task, error out.

This function calls the previously defined angle and linear functions (§5.1.5 and §5.1.6). A conversion from the x displacement (player foot projection on the surface) to degrees is made for the angle control unit. The angle is then converted back to the x position. $x = -\sin(\theta) * L$ with L the player leg length (70 mm).

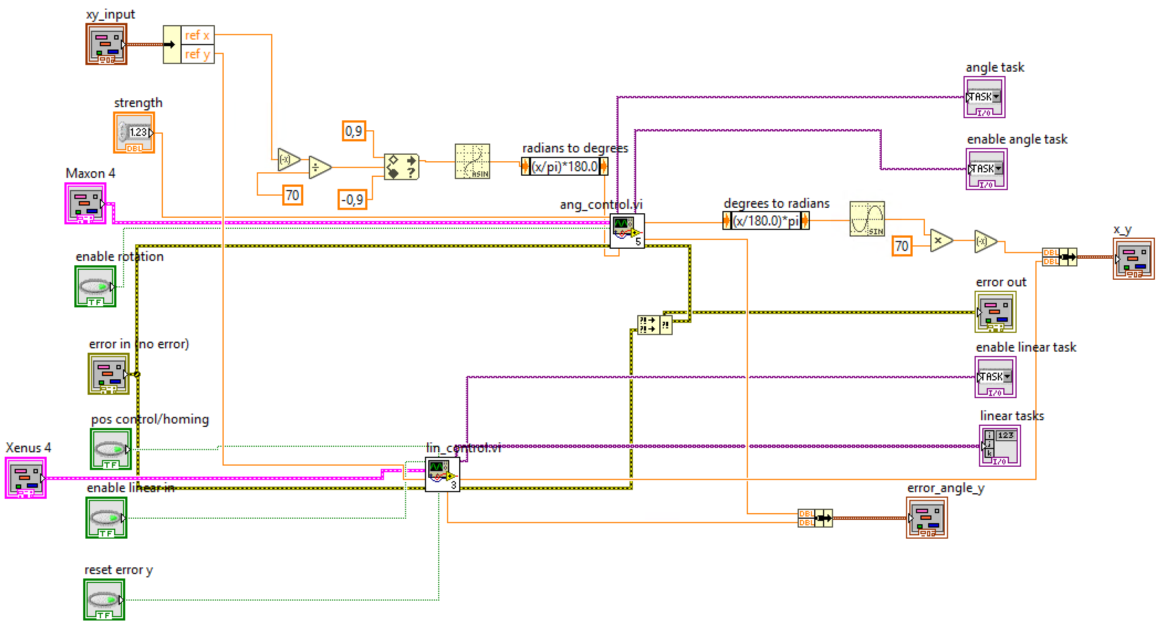


Figure 20 x and y control LabView diagram

5.2 Juggling specific functions

5.2.1 Init_jonglage.vi

Inputs: start path [bool], ball position [mm, mm], middle player position [mm, mm]

Outputs: 2∗N path matrix (path) [mm, mm] ∗N, left/right string.

This function computes the path that the player must follow at initialization in order to get next to the ball without touching it. Depending on the y position of the ball which belongs to one of the 3 defined ranges (left/right/middle), the corresponding player is chosen. If the ball is in the middle range, the middle player is placed right or left to it depending on their position (shortest path is chosen). If the ball is on the left range the left player goes left to the ball, if right range, right player goes right to the ball.

The path is computed once *start path* is enabled, the ball and player position are captured for the path to remain constant. An output string *left/right* tells if the left or right player will juggle with the middle player.

The output path matrix should then be connected to the path planning VI (§3.2.3).

5.2.2 juggling_elastic.vi

Inputs: y position [mm], ball position [mm, mm], left/right string, threshold_y [mm], threshold_x [mm], start jonglage [bool], sampling period (dt) [s], speed [mm/s], minimum speed [mm/s], slope x, offset x [mm], x velocity factor (vx_factor), ball speed [mm/s ,mm/s], cutoff frequency v_x (fc vx) [Hz], cutoff frequency v_y (fc vy) [Hz].

Outputs: player reference position (x_y_out) [mm, mm], filtered ball speed (vx_vy_filtered) [mm/s, mm/s].

This function is responsible for the elastic juggling. The *left/right* input (which comes from the Init_jonglage.vi, §5.2.1), tells if the middle player is juggling with the left or right player. The function is separated in two parts: the y velocity and the x position computation.

The y velocity of player is defined by two parameters: *speed* and *minimum speed*. It was observed that for better robustness the first passes should be stronger as the ball is initially steady. As the ball reaches the targeted speed, the speed of the players should accordingly decrease. Therefore, the *speed* parameter corresponds to the initial y speed of the players and *minimum speed* to the long run speed. The players will start to move toward the ball if the ball is less than *threshold y* away from them and is coming to their direction. Once the direction of the ball has changed the players stop moving in the y direction. There's one exception: as *start jonglage* is set, a timer (§5.1.3) starts, and the player will move toward the ball during the time set (0.3 s) to perform the first pass. The computed y speed is linked to the pseudo speed controller (§5.1.2).

The x position is computed accordingly to the defined model (§3.2.2). $x_j = a * x_b + b + k_v * \dot{x_b}$ with b opposite sign for $x_b < 0$ and $x_b \geq 0$. a is here equal to *slope x* (optimal value found: 1.02) and $b = 5.01$. k_v corresponds to *vx_factor* (optimal value found: 0.05). If the ball x offset (x_b) is lower than *threshold_x* (2 mm) the player stays at $x_j = 0$, this avoids a vibrating effect between $+b$ and $-b$ when the ball offset is close to 0 due to noise. To avoid deviating the ball by moving the player while it is in contact with the ball, the player's x position (x_j) is constant while the player is moving in the y direction.

The input *ball_speed* is filtered with a cutoff frequency *fc vx* (optimal value found: 2 Hz) for the x velocity, and cutoff frequency *fc vy* (optimal value found: 10 Hz) for the y velocity. Note that only the sign of the y velocity matters since it is only used to detect the direction of the ball and thus a change in the direction, therefore, the cutoff frequency is higher than the x velocity one. The latter varies usually between ±0.1 m/s (Figure 11) and has a negative signal-to-noise ratio (Figure 10), therefore the low 2 Hz cutoff frequency.

The *offset x* input corresponds to the distance from the middle of the table (origin) to the attack rod. This value is subtracted to ball position to translate it to the rod referential. The measured distance is 226 mm and is obviously constant. It is a parameter of the VI because of the encoder drift. Thus, it possible to adjust value manually dependently of the drift.

5.2.3 Jonglage_amorti.vi

Inputs: y position (y) [mm], ball position [mm, mm], left/right string, start jonglage [bool], sampling period (dt) [s], max_speed [mm/s], acceleration [mm/s^2], offset x [mm], x velocity factor (K_v), x position factor (K_x), ball speed [mm/s, mm/s], cutoff frequency x velocity (fc) [Hz].

Outputs: player reference position (xy_ref) [mm, mm], filtered ball speed (vx_vy_filtered) [mm/s, mm/s].

This function is responsible for the initial damped juggling. The *left/right* input (which comes from the Init_jonglage.vi, §5.2.1), tells if the middle player is juggling with the left or right player. The function is separated in two parts: the y velocity and the x position computation.

The y velocity of player is defined by two parameters: *max_speed* and *acceleration*. The players will start to move toward the ball if the ball is less than *50 mm* away from them. The initial speed of the player at collision with the ball is equal to ball speed and it accelerates towards the ball until max speed is reached. The speed is incrementally increased by the acceleration $*$ *dt* at each iteration. The resulting speed is linked to the pseudo speed controller (§5.1.2).

The x position is computed accordingly to the defined model (§3.2.1). $x_j = K_x x_b + K_v \dot{x_b}$. With $K_x = 1.1$ and $K_v = 0.1$.

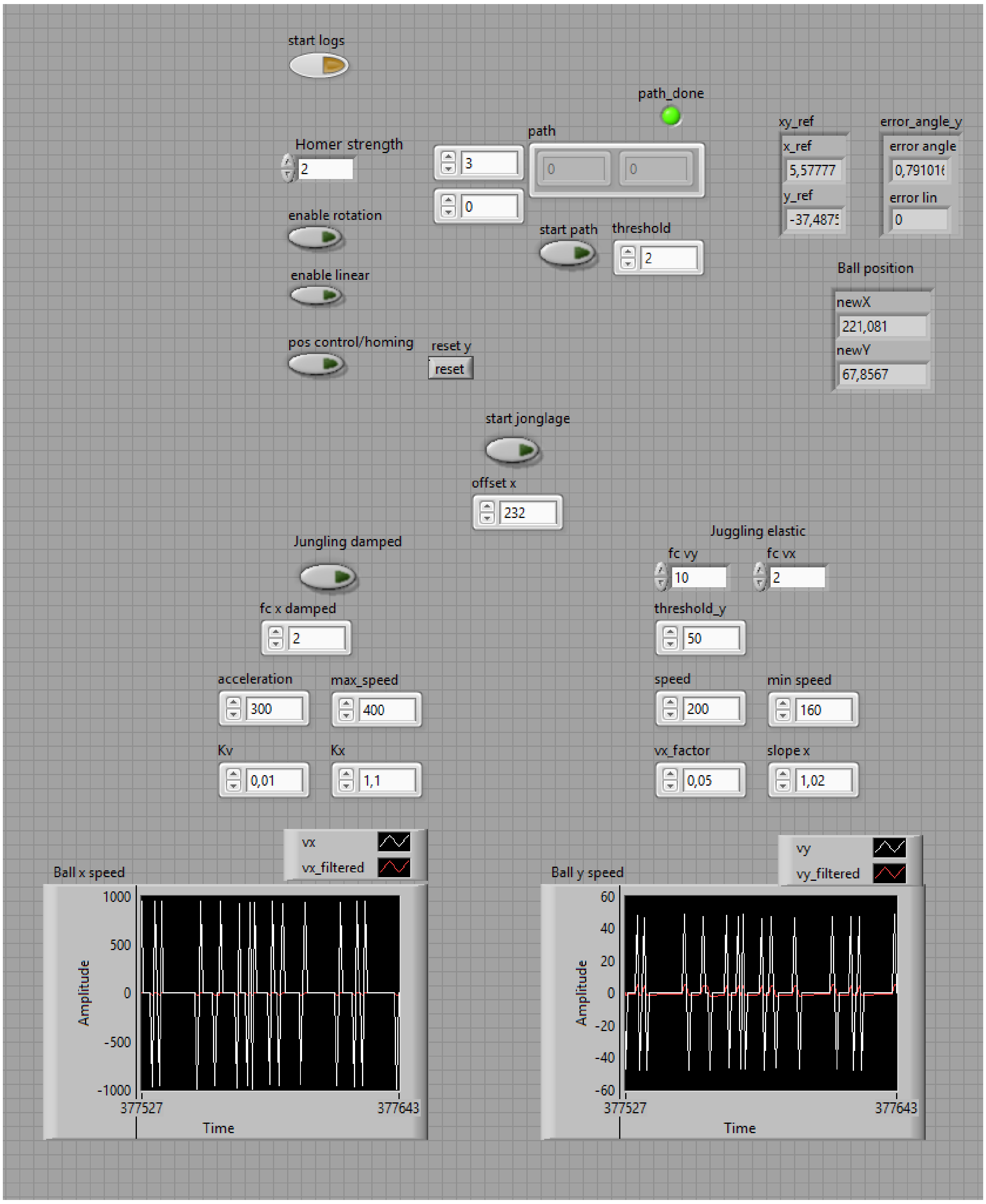
The *offset x* input corresponds to distance from the middle of the table to attack rod. This value is subtracted to ball position to translate it to the rod referential. The measured distance is 226 mm and is obviously constant. It is a parameter of the VI because of the encoder drift. Thus, it possible to adjust value manually dependently of the drift.

5.3 Vision

The vision is imported as it is from previous project, except the speed averaging that is removed. A new calibration has been done, one should be aware that in the calibration step by step guide from a previous report [3], the distance between the circles in the calibration pattern is wrong. The correct values are dx = 58 mm and dy = 68 mm (also written on the pattern sheet). In the last calibration the y axis is inverted but to keep consistent with the axis definition of the previous projects and of the main program, the ball y position and speed are inverted at the beginning of the main juggling loop.

5.4 Front panel

On the front panel are displayed all the parameters previously detailed. They are grouped by module. Here is a screenshot with the parameters used (Figure 21). Note that the *offset x* corresponds to the distance between the rod and the origin (226 mm) but should be changed according to the drift. The *start logs* button will start the writing of the logs to the logs folder.



start logs

path_done

Homer strength 2

3

0

path 0 0

enable rotation

start path

threshold 2

enable linear

pos control/homing

reset y reset

xy_ref

x_ref 5,57777

y_ref -37,487

error_angle_y

error angle 0,79101

error lin 0

Ball position

newX 221,081

newY 67,8567

start jonglage

offset x 232

Jungling damped

fc x damped 2

acceleration 300

max_speed 400

Kv 0,01

Kx 1,1

Juggling elastic

fc vy 10

fc vx 2

threshold_y 50

speed 200

min speed 160

vx_factor 0,05

slope x 1,02

Figure 21 LabView front panel with parameters set

6 Printed Circuit Board

Previous PCBs used to make the connections between the motor/sensors with the drivers were using Molex connectors which are not robust. They eventually broke. These cards are now replaced with new ones using flat cable connectors. It was decided to keep the same amount of connection despite that not all of them are used. Note that the previous card had straight lines between the connectors but the new one have crossed lines (Figure 22). Therefore, while replacing them, one should be careful to connect it accordingly.

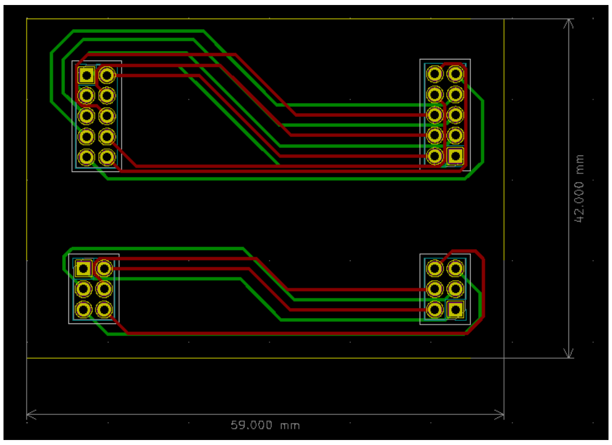


Figure 22 PCB board layout from KiCad

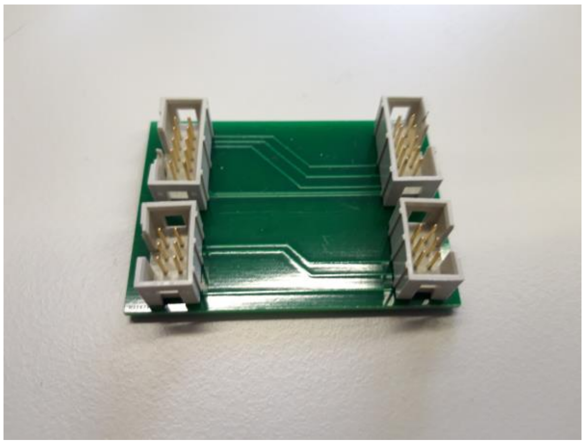


Figure 23 Photo of the PCB

7 Conclusion

In conclusion, while the simulation to implementation approach was not successful, we managed to make the table football juggle. The hardware issues encountered affected the stability, but the players can keep performing passes for more than 3 minutes. We are now looking forward to seeing the juggling algorithm implemented in the main strategy of the table football.

8 Acknowledgments

I would like to thank Christophe SALZMANN for offering and supervising this semester project along the years. It is great to see each student contributing and improving the table football one after each other. The table football provides engineering problematics and remains playful. The downside is that it can be hard to dive in multiple contributor project. But that is an experience I am glad I did.

9 References

[1] Maxence PERRET « Simulation : Passes entre les joueurs », 2020

[2] Yann MORIZE “Vision and strategy: Ball capture”, 2020

[3] L. AFILAL, K. BENABDERRAZIK, D. NIGRI “Babyfoot : Vision et Calibration », 2019

[4] Rod Cross “Cue and ball deflection (or “squirt") in billiards”, American Journal of Physics, 2007

10 Annexes

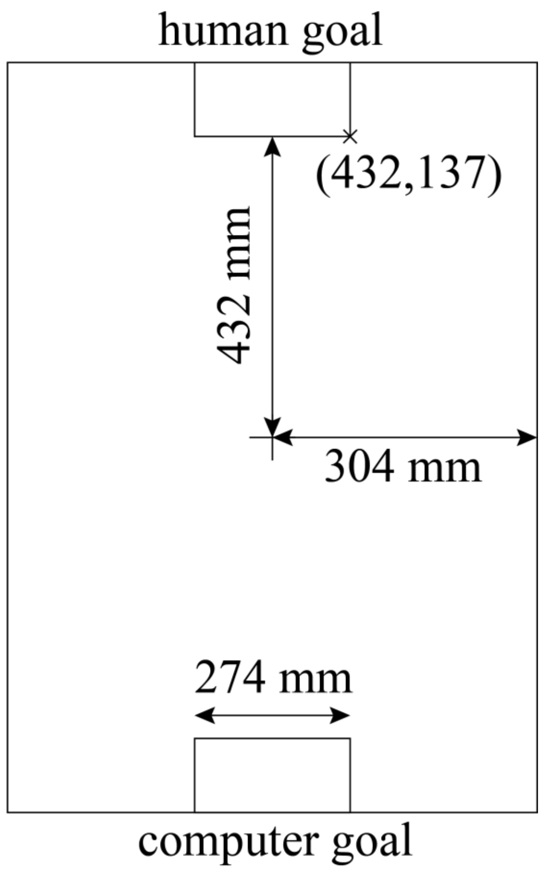


Figure 24 Some measured distances for calibration