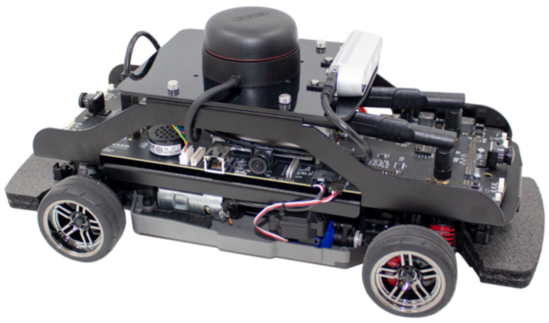
Mobile Robot Control

TP Qcar

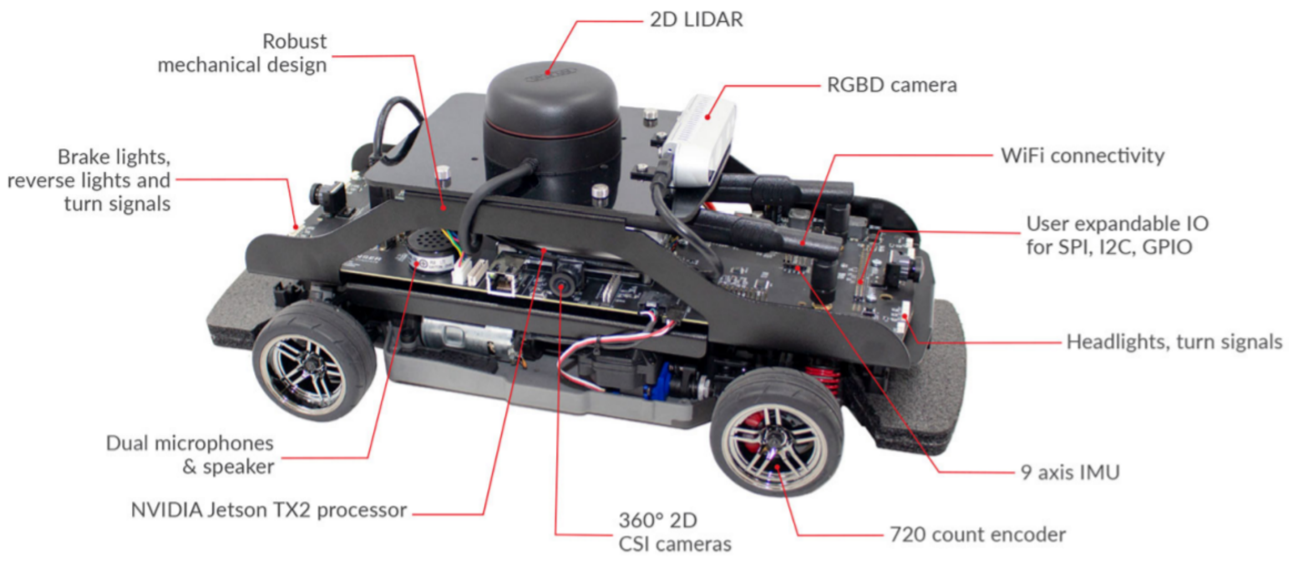


Nathan MARTIN - Alexandre SPRICH - Dmytro SAVCHUK

I - INTRODUCTION

In the context of autonomous vehicles (vehicles that have the capability to have automatic motions and navigate itself depending on its environments and scheduled tasks), we worked on the control of a small model of autonomous vehicle, named Qcar, which is developed by Quanser.

This small vehicle is equipped with several sensors (RGB camera, LIDAR, side cameras, etc.) allowing it to analyze its environment and thus be able to move around in space in a secure manner.


2D LIDAR
Robust mechanical design
RGBD camera
Brake lights, reverse lights and turn signals
WiFi connectivity
User expandable IO for SPI, I2C, GPIO
Headlights, turn signals
Dual microphones & speaker
9 axis IMU
NVIDIA Jetson TX2 processor
360° 2D CSI cameras
720 count encoder

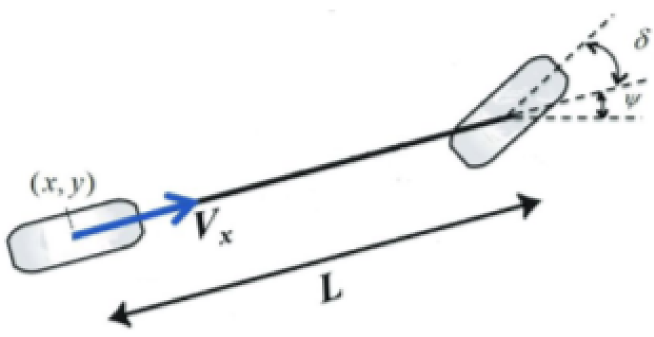
Indeed, thanks to these sensors it will be possible to receive all the data transmitted by the car (angle of the wheels, speed, angle of the car, etc.) in order to be able to process them and carry out actions in function.

It will also be possible to detect obstacles thanks to the Lidar or camera data.

Finally, the purpose of this TP is to program and control the Qcar using Matlab and Simulink to perform position regulation.

II - Control of Qcar

The operation of the Qcar follows the principle of the bicycle model, a kinematic model where the two front wheels and the two rear wheels are replaced by one front and one rear wheel respectively.



Bicycle model

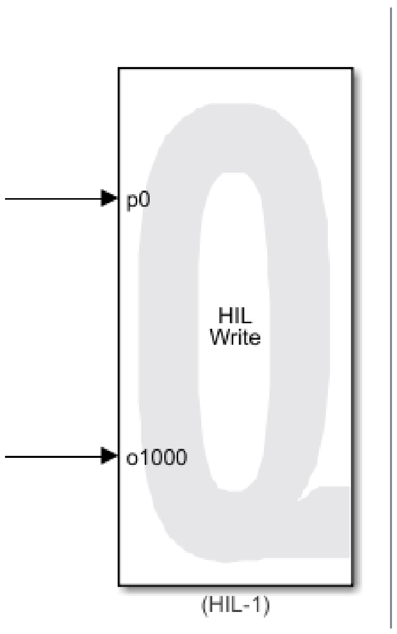
The location of Qcar is defined by its coordinate (x, y, ψ), where x *and* y are the coordinates of the position of the center of gravity and ψ is the orientation (heading) of the vehicle.

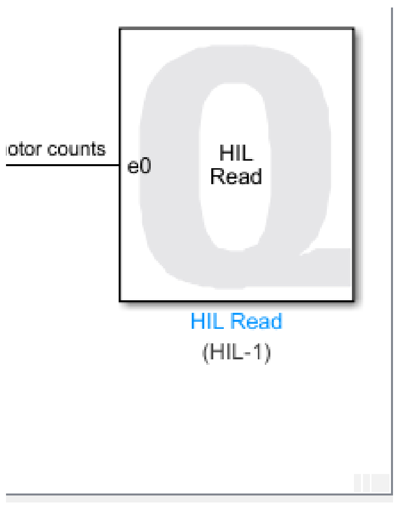
Of course, this model is not perfect but it accurately represents the kinematics of the car while simplifying the analysis.

Then we can control the Qcar using 2 command variables :

- The velocity of the car Vx, to set the speed of the robot.
- The steering angle of the front wheels δ, to set the trajectory of the robot.

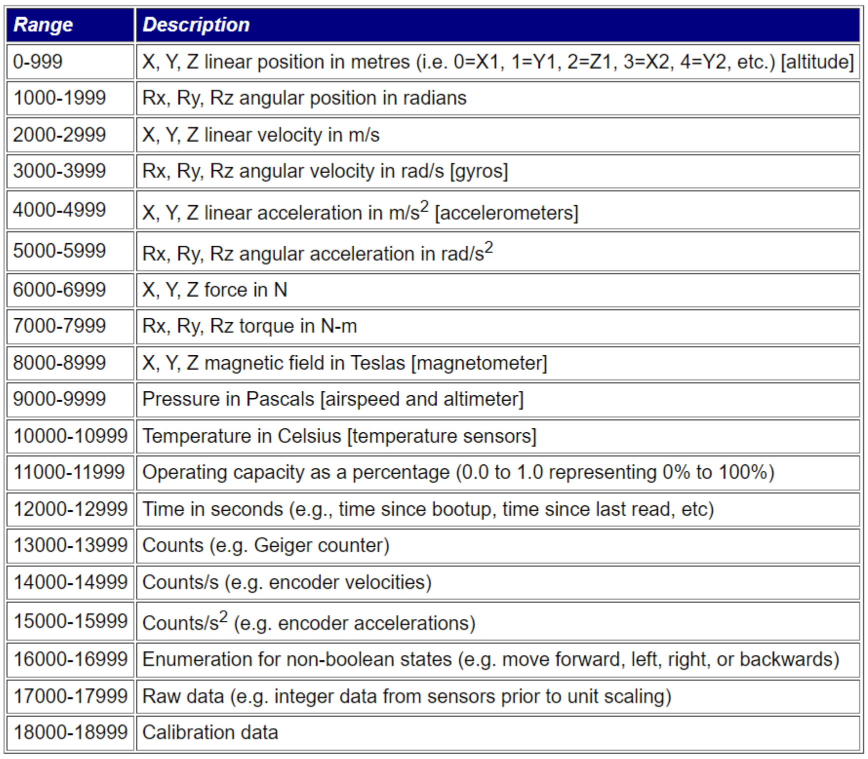
We also used 2 blocks on Simulink to read and write information to the Qcar.





These blocks allowed us to read and write information on the specified channels, which correspond to the types of sensors or actuators on which to act or read.

Just follow this chart :

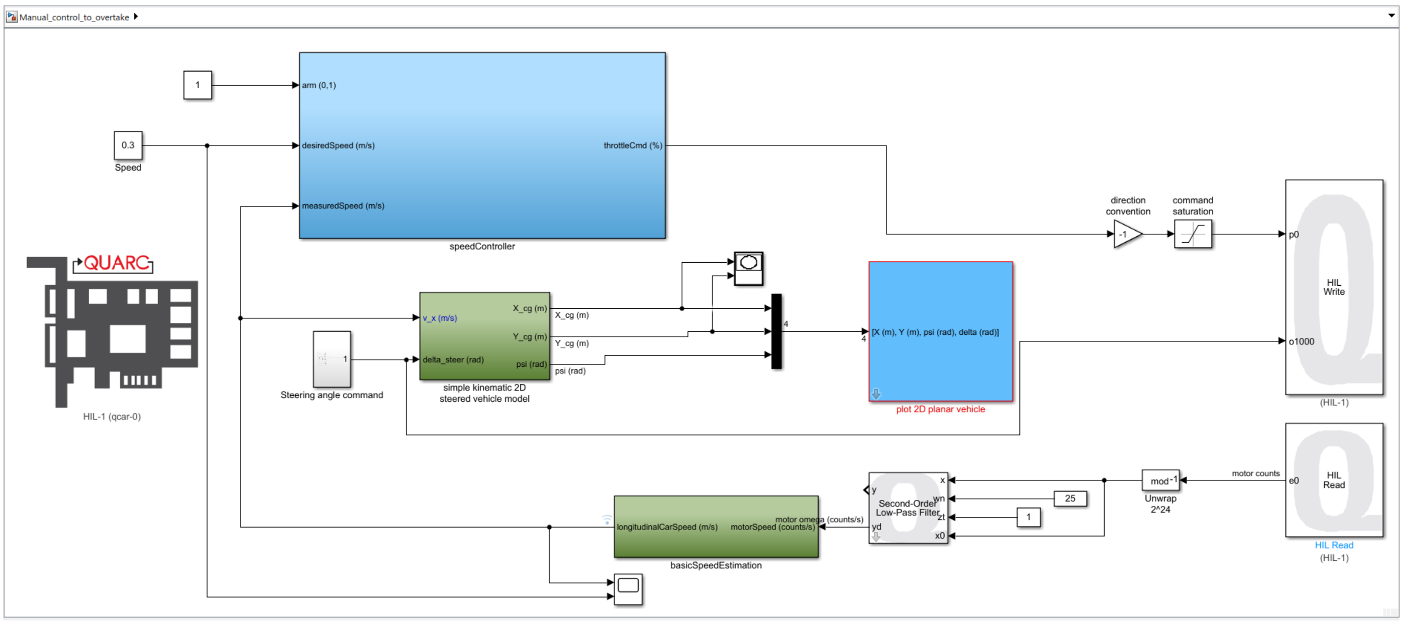


Range	***Description***
0-999	X, Y, Z linear position in metres (i.e. 0=X1, 1=Y1, 2=Z1, 3=X2, 4=Y2, etc.) [altitude]
1000-1999	Rx, Ry, Rz angular position in radians
2000-2999	X, Y, Z linear velocity in m/s
3000-3999	Rx, Ry, Rz angular velocity in rad/s [gyros]
4000-4999	X, Y, Z linear acceleration in m/s^2 [accelerometers]
5000-5999	Rx, Ry, Rz angular acceleration in rad/s^2
6000-6999	X, Y, Z force in N
7000-7999	Rx, Ry, Rz torque in N-m
8000-8999	X, Y, Z magnetic field in Teslas [magnetometer]
9000-9999	Pressure in Pascals [airspeed and altimeter]
10000-10999	Temperature in Celsius [temperature sensors]
11000-11999	Operating capacity as a percentage (0.0 to 1.0 representing 0% to 100%)
12000-12999	Time in seconds (e.g., time since bootup, time since last read, etc)
13000-13999	Counts (e.g. Geiger counter)
14000-14999	Counts/s (e.g. encoder velocities)
15000-15999	$Counts/s^2$ (e.g. encoder accelerations)
16000-16999	Enumeration for non-boolean states (e.g. move forward, left, right, or backwards)
17000-17999	Raw data (e.g. integer data from sensors prior to unit scaling)
18000-18999	Calibration data

III - Lane changing

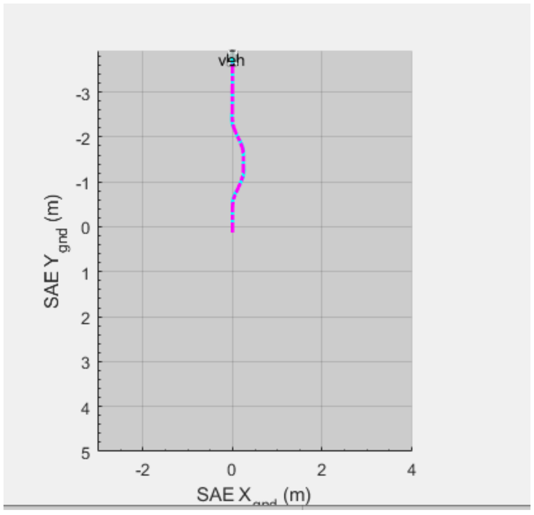
The first exercise we worked on consisted of performing a vehicle overtaking action : the Qcar vehicle moving on one lane of the road must move to the other lane to overtake another vehicle, then return to the initial lane once the overrun has been made.

The Simulink block diagram implemented for this exercise is as follows :



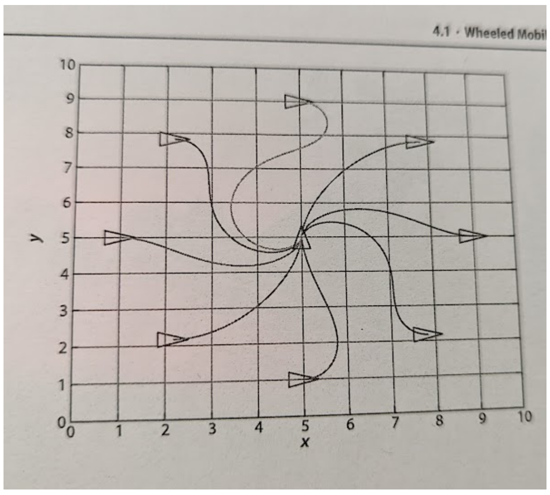
Concretely, we applied a series of 8 steps in Simulink to the steering angle command to modify the orientation of the wheels and therefore the trajectory of the robot, and thus allow it to move to the other lane :

1) Step of -0.37 → steer the wheels to the left
2) Step of +0.37 → put the wheels straight
3) Step of +0.37 → put the robot straight
4) Step of -0.37 → put the wheels straight
5) Step of +0.37 → steer the wheels to the right
6) Step of -0.37 → put the wheels straight
7) Step of -0.37 → put the robot straight
8) Step of +0.37 → put the wheels straight



IV - Moving to a point

The second exercise we worked on consisted of the second exercise we worked on consisted of moving the Qcar robot from an initial point, defined by its coordinates (xo, yo), to an end point defined by its coordinates (xf, yf).

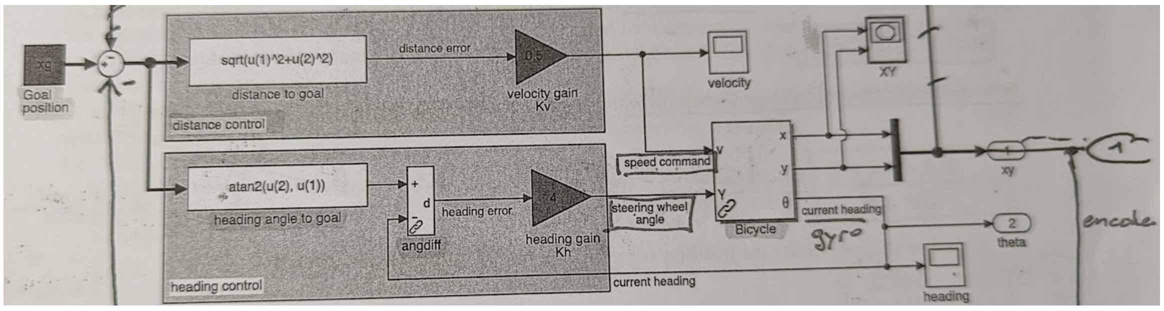


To do that, we used the Robotics Toolbox for Matlab and Simulink, developed by Peter Corke.

Here we needed to use the final position as a control variable, and the system computes itself the steering angle and the speed corresponding to reach this point.

During the TP, an attempt was made to understand how the bicycle model worked to achieve this. Indeed, we had some problems, because basically the model was just spinning the car in circles.

Indeed, we first tried to manipulate the static gains of the angle control and the speed control, because at the start the wheels were always at the maximum angle which caused the infinite circle.



We therefore tried to understand the "*angdiff*" function which makes a difference between the control angle and the desired angle.

We encountered a lot of problems on this subject because we really had to understand each block of the simulink in order to be able to understand where the problem came from.

V - Conclusion

The objective of our study was to understand how the Qcar robot works and understand how to control it using the tools provided by Matlab and Simulink.

- During the first part, we saw how to steer the robot by using the speed and steering angle control variables, to move the Qcar in the room and perform overtaking for example.

- During the second part, we tried to understand how to make the robot move from an initial point to an end point. This part still needs to be explored, so that the robot can move optimally towards the requested point.

We were not necessarily able to accomplish a lot during this TP, because understanding the different systems, detecting and correcting problems and finding solutions took us quite a bit of time.

Thereafter, the control of these 2 concepts will allow it to go further in the use of the functionalities of the Qcar, in particular to carry out line tracking or even the detection of obstacles.