Torque Vectoring for Race Cars

Luís Carlos Dias Duarte
luiscdduarte@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

November 2021

Abstract

Abstract

This article presents the development, construction, testing and results analysis of a remote controlled car, with an engine on each wheel so that Torque Vectoring (TV) controllers capable of distributing torque independently between each wheel may be tested, equipped with sensors to measure the position, speed, direction, and other telemetric values. To help with the task of converting computer-sent information, a digital analogic converter connected to the car's controller was used. Using the state machine concept as basis, the car has three defined states, activated through a selected three-way switch, installed on the car's controller. Said controller was the target of several internal modifications so it could issue commands both manually and via computer. Every installed component was validated with the help of the cameras of the Qualisys system, present in the lab, to obtain viable, precise and accurate results. The dynamic study of the vehicle, a theoretical study and a practical study for two controllers responsible for the trajectory control were shown. The first was a Linear Quadratic Regulator (LQR) with, in spite of being capable of good results in theory, the required physical limitations ensured the wanted trajectory couldn't be followed in practice. To balance the observed difficulties, a Proportional-Integral-Derivative (PID) controller was implemented, with verified improvements in the theoretical and experimental results, which made the car follow the expected trajectory.

Keywords: Torque Vectoring, Sensors, Control, State machine, Qualisys, LQR, PID.

1. Introduction

The work developed in [1] relies on the development of a torque vectoring controller for a Formula Student prototype. The knowledge from this combined with [2] allowed to assemble the car presented in this article.
The inexistence of a platform capable of supporting this type of test was the main motivation. The car built was the result of many different components and types of communications implemented. The car was equipped with four motors, one in each wheel, encoders, IMU, two Atmega328 and one Arduino Due. The car works like a state machine, having three states. One is the fail-safe that cuts the power to the motors, another is responsible for the direct drive mode which allows to control the car manually and the last one is the torque vectoring mode that applies the power to each wheel based on the steering angle input. A strong investment on the communication system was made in order to have an acceptable response and accurate telemetry data. The majority of the previous works developed in the lab discarded the Traxxas remote controller. This article presents a customization of this remote using a digital to analog converter in order to take advantage of the RF communication of the remote to send information from the computer to the car. The dynamic of the car is presented using [1] and [3]. In [4] torque vectoring was implemented in a Formula Student car using PI and LQR controllers for yaw tracking. The Ackerman geometry was studied and implemented in the car presented here too.

2. Vehicle Model

2.1. Vehicle coordinate frame

The dynamics and a model of the vehicle are the most important information that needs to be defined. The equations of motion, steering kinematics, velocity vectors and slip angles were considered. Being important to refer that the rolling resistance, wind resistance and vertical force were neglected because the car is small enough and not too fast for these parameters to influence the results. The vehicle body coordinate frame to be used is shown in Fig.1, having B(C_{xyz}), attached to the center of mass C.

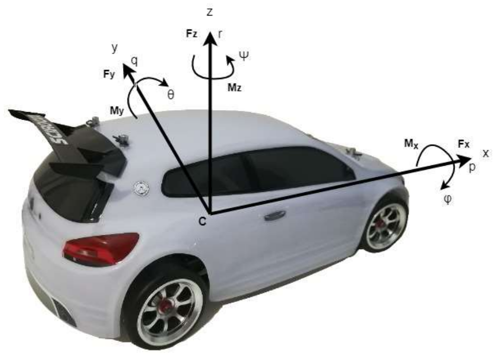

Figure 1: Body coordinate system

To compute the body orientation, the rotations are defined as:

$$\varphi : Roll \rightarrow \ \dot{\varphi} = p : \ Roll\ rate$$

$$\theta : Pitch \rightarrow \ \dot{\theta} = q : Pitch\ rate$$

$$\Psi : Yaw \rightarrow \ \dot{\Psi} = r : Yaw\ rate$$

The forces are described in the body frame as:

$$F^B = F_x\vec{\imath} + F_y\vec{\jmath} + F_z\vec{k} \tag{1}$$

$$M^B = M_x\vec{\imath} + M_y\vec{\jmath} + M_z\vec{k} \tag{2}$$

where F_x is the longitudinal force, F_y is the lateral force, F_z is the vertical force, M_x is the roll moment, M_y is the pitch moment and M_z is the Yaw moment.

2.2. Newton-Euler Dynamics

The car is considered to be a flat box moving horizontally. Three degrees of freedom are needed, translation in x and y and rotation around z axis. The Newton-Euler equations used on B coordinate frame system are:

$$F_x = m\dot{v}_x - m\omega_z v_y \tag{3}$$

$$F_y = m\dot{v}_y - m\omega_z v_x \tag{4}$$

where v_x and v_y are the velocity components. The inertia matrix of the body is given by equation 5, but since it is considered that the body only rotates around z axis, it is taken into account only the value of I_z

$$I^B = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{5}$$

2.3. Force system acting on rigid body

Using the bicycle model without the roll component, the tire lateral force (F_y) is given by:

$$F_{yi} = -C_\alpha \alpha_i \tag{6}$$

where C_α is the cornering stiffness of the tire, and α is the tire sideslip angle and can be physically interpreted as the angle between x axis and the velocity of the tire. Mathematically, α can be written as:

$$\alpha = \beta - \delta \tag{7}$$

$$\beta = \arctan\left(\frac{v_y}{v_z}\right) \tag{8}$$

For a small β, equation (7) can be written as:

$$\beta = \frac{v_y}{v_x} \tag{8.1}$$

Neglecting the aligning moments, M_{zi}, the forces are:

$$F_x = F_{xf}\cos(\delta) + F_{xr} - F_{yf}\sin(\delta) \tag{9}$$

$$F_y = F_{yf}\cos(\delta) + F_{yr} - F_{xf}\sin(\delta) \tag{10}$$

$$M_z = a_1 F_{yf} - a_2 F_{yr} \tag{11}$$

where the indices r and f means "rear" and "front" wheel respectively and a_1 and a_2 are the distance between front and rear wheel in relation to center of gravity. In order to linearize the equations, small rotations should be considered ($\delta = 0$), hence, the the equations 9, 10 and 11 can be written as:

$$F_x \approx F_{xf} + F_{xr} \tag{12}$$

$$F_y \approx F_{yf} + F_{yr} \tag{13}$$

$$M_z \approx a_1 F_{yf} - a_2 F_{yr} \tag{14}$$

It should be noticed that when using bicycle model, the car becomes a one-track model, meaning that only one front steer angle can be controlled. It is worth referring that the slip angle α_i is calculated using the tire side slip angle β_i and an expression as a function of β is required. For that, the lateral wheel velocity v_{yfr} has an additional component because there is a yaw rate in the mass centre of the car with distance a_1 and a_2 has can be seen in Fig.2.

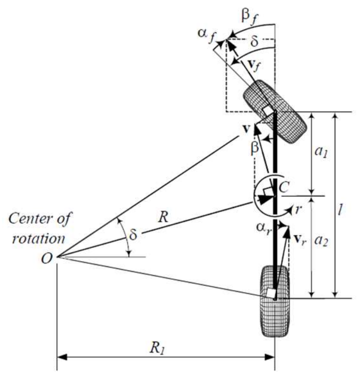

Figure 2: Two-wheel model for a vehicle moving with no roll

Taking equation 7 and applying for the front and rear wheel, it is written as:

$$\alpha_i = \beta_i - \delta \tag{15}$$

$$\alpha_f = \beta + \frac{a_1 r}{v_x} - \delta \tag{16}$$

$$\alpha_r = \beta - \frac{a_2 r}{v_x} \tag{17}$$

where F_y and M_z only depend on the forces in y axis that are functions of the wheel sideslip (α_f, α_r). Hence, these equations can be approximated as:

$$F_y = \left(-\frac{a_1}{v_x}C_{\alpha f} + \frac{a_2}{v_x}C_{\alpha r}\right)r - (C_{\alpha f} + C_{\alpha r})\beta + C_{\alpha f}\delta \tag{18}$$

$$M_z = \left(-\frac{a_1^2}{v_x}C_{\alpha f} - \frac{a_2^2}{v_x}C_{\alpha r}\right) - (a_1 C_{\alpha f} - a_2 C_{\alpha r})\beta + a_1 C_{\alpha f}\beta \tag{19}$$

The implemented state space is given by [3]:

$$\begin{bmatrix}\dot{v}_y \\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{m v_x} & \frac{-a_1 C_{\alpha f} + a_2 C_{\alpha r}}{m v_x} - v_x \\ -\frac{a_1 C_{\alpha f} - a_2 C_{\alpha r}}{I_z v_x} & -\frac{a_1^2 C_{\alpha f} + a_2^2 C_{\alpha r}}{I_z v_x}\end{bmatrix}\begin{bmatrix}v_y \\ \dot{\psi}\end{bmatrix} + \begin{bmatrix}\frac{C_{\alpha f}}{m} \\ \frac{a_1 C_{\alpha f}}{I_z}\end{bmatrix}\delta \tag{20}$$

2.4. Torque differential in rear wheels

The goal with the torque vectoring is to generate yaw moment based on controlling the torque (longitudinal force) in each wheel. For this it will be necessary to introduce a new term M_z that will represent the additional yaw moment generated by the torque distribution, so the new state space can be described as:

$$\begin{bmatrix}\dot{v}_y \\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{m v_x} & \frac{-a_1 C_{\alpha f} + a_2 C_{\alpha r}}{m v_x} - v_x \\ -\frac{a_1 C_{\alpha f} - a_2 C_{\alpha r}}{I_z v_x} & -\frac{a_1^2 C_{\alpha f} + a_2^2 C_{\alpha r}}{I_z v_x}\end{bmatrix}\begin{bmatrix}v_y \\ \dot{\psi}\end{bmatrix} + \begin{bmatrix}\frac{C_{\alpha f}}{m} \\ \frac{a_1 C_{\alpha f}}{I_z}\end{bmatrix}\delta + \begin{bmatrix}0 \\ \frac{1}{I_z}\end{bmatrix}M_z \tag{21}$$

And the car linear model is shown in Fig.3,

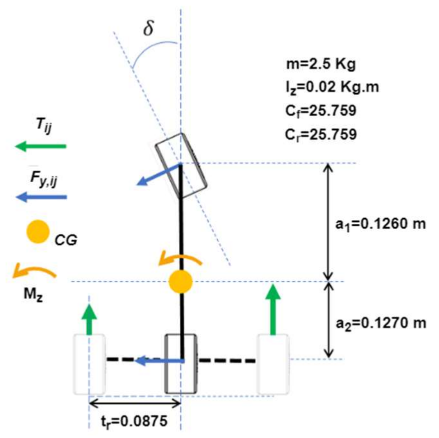

Figure 3: Vehicle linear model with additional moment

where CG is the center of gravity, t_r corresponding to $w/2$, T_{ij} is the torque, F_y is the tire lateral force, I_z is the inertia moment around z axis, m is the mass of the car, C_f and C_r are the cornering stiffness constant, front and rear respectively, This added moment resulted from the difference between the left and the right wheel torque, T_{rl}, T_{rr}.

$$M_z = \Delta T * t_r = (T_{rr} - T_{rl}) * t_r \tag{22}$$

The torque at the wheel is the same as the torque at the motor because it is direct drive. To obtain the force on the ground the torque is divided by the wheel radius R_w.

$$\Delta T = \frac{R_w}{2t_r}M_z \tag{23}$$

Thus, the M_z can be replaced by ΔT in equation 21. In case of 4-wheel torque vectoring there is one more moment to be added due to the torque difference of the front wheels. In this work due to the loss of the front motors only the Ackerman Geometry will be considered.

2.5. Front wheel differential

To simulate a mechanical differential on the front wheels, Ackermann Geometry was used. The steering angle of each wheel should be considered. Knowing the desired steering angle, the angle of inner and outer wheel can be calculated separately resulting in equation (24) and (25). The Ackerman geometry is shown in Fig.4.

$$R_{in} = \frac{a_1 + a_2}{\sin^{-1}(\delta_{in})} \tag{24}$$

$$R_{out} = \frac{a_1 + a_2}{\sin^{-1}(\delta_{out})} \tag{25}$$

After describing a curve, the inner and outer wheels must travel different distances due to the different arcs of their trajectories, meaning that the outer wheel must rotate faster than the inner one. As the wheels are attached to the same axle, in order to comply with that, they must rotate at different speeds. The following equations demonstrate the relation between the inner and outer wheels. Considering a turning of $360°$ in a certain amount of time:

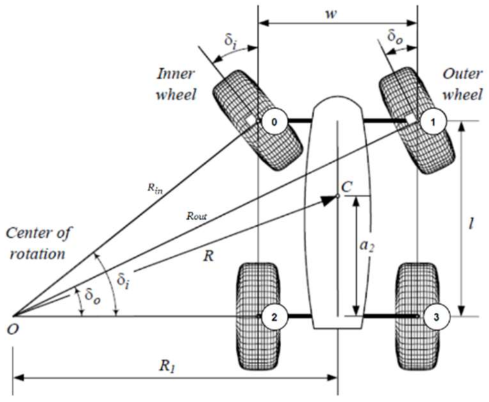

Figure 4: Ackerman geometry

$$\frac{d_{in}}{t} = \frac{2\pi R_{in}}{t} \tag{26}$$

$$\frac{d_{out}}{t} = \frac{2\pi R_{out}}{t} \tag{27}$$

where d_{in} is the distance travelled by the inner wheel in relation to the centre of rotation, d_{out} is the distance travelled by the outer wheel in relation to the centre of rotation and t is the time. Therefore, the relation between the outer and inner wheel velocity is described as:

$$\frac{v_{out}}{v_{in}} = \frac{2\pi R_{out}}{2\pi R_{in}} = \frac{R_{out}}{R_{in}} \tag{28}$$

3. Controllers

3.1. Arduino differential control

To implement control, a reference signal must be created. The chosen signal was the yaw rate and it should be a function of the steering δ. This signal is adapted to the characteristics of the car's behaviour. It can be defined by the ratio between front and rear masses and between the front and rear tire cornering stiffness

$$K_u = \frac{a_2 m}{C_{\alpha f}(a_1 + a_2)} - \frac{a_1 m}{C_{\alpha r}(a_1 + a_2)} \tag{29}$$

If K_u is positive (K_u > 0), the car is said to have an under-steer behavior. In case of K_u< 0, the car has a oversteer behavior. When K_u= 0, it means the car has a neutral steer (ideal yaw rate). To avoid instability, the under-steered vehicle is chosen. The desired yaw rate can be defined by the velocity and the radius of curve:

$$\dot{\psi}_{desired} = \frac{v_{CG}}{R} \tag{30}$$

Giving the velocity and steering angle of the car, with known steer gradient and wheelbase, the radius is described as:

$$\frac{1}{R} = \frac{\delta}{(a_1 + a_2) + K_u v_{CG}^2} \tag{31}$$

With equation (30) and (31), a function of δ to find yaw rate desired is computed as:

$$\dot{\psi}_{desired} = \frac{\delta}{(a_1 + a_2) + K_u v_{CG}} \tag{32}$$

The under-steer K_u can be tuned for driver preference. The bigger the K_u, the bigger the difference between the desired and actual yaw rate, the car will have near under steer characteristics and it will be harder to drive.

3.2. Linear Quadratic Regulator (LQR)

LQR is an optimal control solution for linear systems. The performance index to design the LQR controller is written in equation (33). It is a quadratic cost function and the main objective is to find a state-feedback law $u = -Kx$ that minimizes this function.

$$J(u) = \frac{1}{2}\int_{t_0}^{t_f} [(X_d - X)^T Q (X_d - X) + u^T R u] dt \tag{33}$$

The state space will have as state variables v_y, $\dot{\psi}$ and ψ. For two motorized wheels, δ will be the only control input, since the δ will be imposed by the driver. To estimate the lateral velocity, an integration of the acceleration acquired from the IMU should be done. It was necessary to add a state variable to the system to have an output which is the yaw ψ (equation 34). Since the velocity was imposed to be constant, in this state the longitudinal dynamics ended up being neglected too.

$$\begin{bmatrix} \dot{v}_y \\ \ddot{\psi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{m v_x} & \frac{-a_1 C_{\alpha f} + a_2 C_{\alpha r}}{m v_x} - v_x & 0 \\ -\frac{a_1 C_{\alpha f} - a_2 C_{\alpha r}}{I_z v_x} & -\frac{a_1^2 C_{\alpha f} + a_2^2 C_{\alpha r}}{I_z v_x} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_y \\ \dot{\psi} \\ \psi \end{bmatrix} + \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{a_1 C_{\alpha f}}{I_z} \\ 0 \end{bmatrix} \delta \tag{34}$$

MATLAB was used to design the controller gains. Using the command [K,S,e]=LQR(A,B,Q,R), it returns not only the gains K, but also the solution for the Riccati equation (35) and the closed-loop eigen values.

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{35}$$

K is then derived by *P* using:

$$K = R^{-1} B^T P \tag{36}$$

The first Q matrix used can be described as:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

And $R = 1 \times 10^{-1}$.

The best results obtained with this controller were influenced by the physical limitations of the steering and the real response of the system. The best simulation obtained to control the car had a new Q matrix that is described as:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

with $R = 1 \times 10^{-1}$ and the resulting gains were:

$$K = [0.1294 \quad 0.0361 \quad 0.3162]$$

The results obtained in these conditions, provide the step response shown in Fig.5.

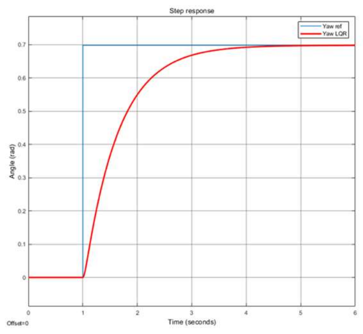

Figure 5: LQR Step response

In comparison with the projected controller shown before, it is possible to see that this one is slower. In practice, it shows low repeatability and does not achieve the desired direction. The trajectory made by the car (blue plot) is shown in Fig.6.

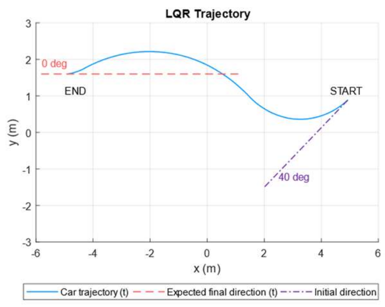

Figure 6: LQR Trajectory

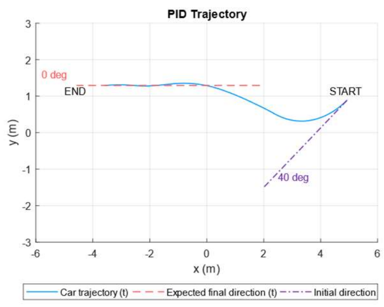

Figure 7: PID Trajectory

Despite the car's attempts to follow the desired direction it was never able to finish the trajectory pointing at 0 degrees direction. Perhaps with more time and distance, the results would be better.

3.3. Proportional Integral Derivative (PID)

The state space from equation (34) was used for this part too. To compare one with the other similar response were wanted. The obtained results presented below, were obtained using Simulink PID block knowing that it has the following structure:

$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

The values of each part of the PID used were strategically changed until had: P=14, I=0, D=6. Note that for similar response, identical rise time, settling time and no overshoot was the goal for the step response in the same conditions. The step response response is presented in Fig.8.

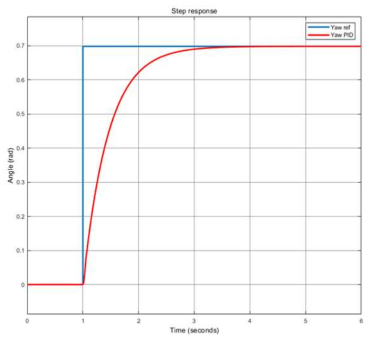

Figure 8: PID Step response

It is possible to see that the response of this controller is better from a theoretical perspective. This last sentence could be confirmed considering the result presented in Fig.7. This result allows to prove that the desired direction was successfully obtained because the car was able to follow the same direction since halfway until the end.

4. Tests results

4.1. Hardware developed

Communication was very important to this work, since all the telemetry needed to be sent to the computer through Wi-Fi. On the other hand the computer communication from the computer to the car is established using the digital-to-analogic converter NI USB 6008 because only commands of steering and power need to be sent. Most communication protocols appear in the car telemetry due to the use of several different microcontrollers connected between each other requiring different protocols. The communication diagram between Qualisys, computer, remote control and the car is shown in Fig.9, final configuration is shown in Fig.10 and Remote with upgrades is shown in Fig.11.

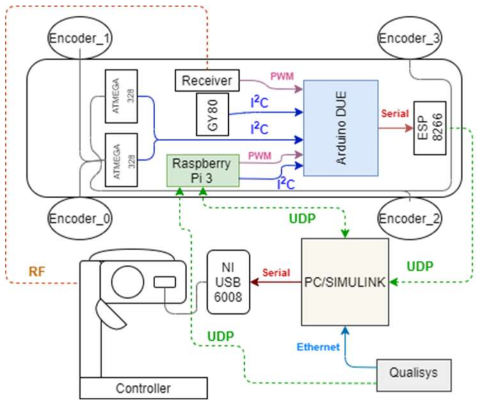

Figure 9: Communication diagram

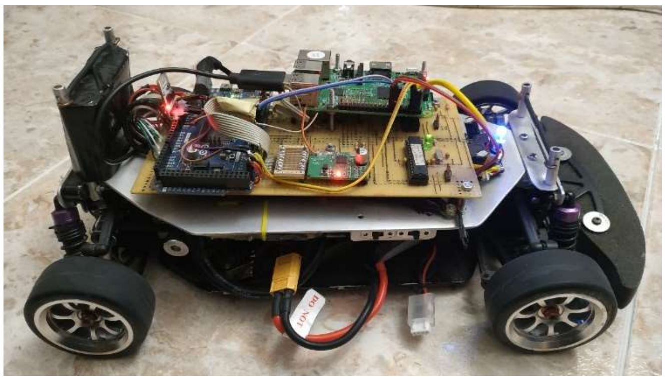

Figure10: Final configuration

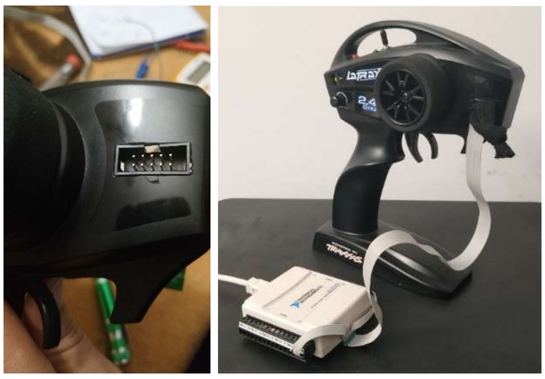

Figure 11: Remote and NI USB 6008

The data from the IMU and encoders were obtained using Arduino libraries such as “GY80” [5], and "Wire" library to get data using I2C communications [6]. To read the encoders were used two ATMEGA328 as slaves to get data from two encoders each and send it, by I2C, to the Arduino Due working as master to implement the TV controller. It is important to explain that due to the assembly of the two new encoders the Arduino responsible for the front wheels was reconfigured to read the new ones. With that upgrade, better results were achieved. To connect and receive the IMU data, the library code was adapted to fulfill some requirements and be able to receive and measured accurately all pretended values. The UDP connection was used to send data from Arduino DUE to Computer. The UDP connection was already used in [2], but now was used with upgraded Simulink toolboxes to receive data. In this case the data sent by ESP8266 was transformed in one string and then decoded on Simulink. The ESP8266 send UDP messages taking advantage of the AT Commands [7]. These AT Commands are part of the AT Firmware of the ESP8266 and they allow a large variety of customization and uses. Qualisys connection to the laptop was the same one used in [8], the only difference were the configurations of the IP address of each machine inside of the “Config.txt” file, and then running the “Qualisys UDP Receiver.exe” file to receive the data. One of the validation tests made was the speed comparison, the results are shown in Fig.12. Other one was the IMU validation shown in Fig.13

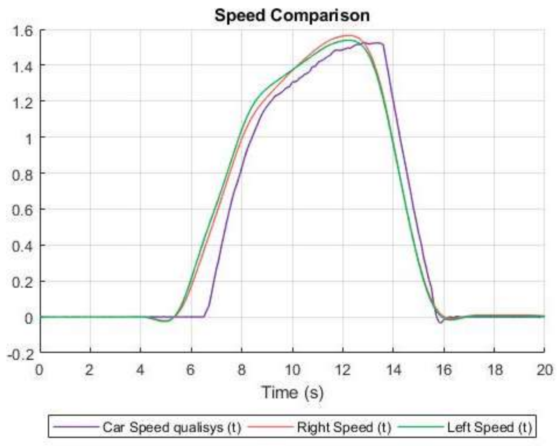

Figure 12: Speed comparison

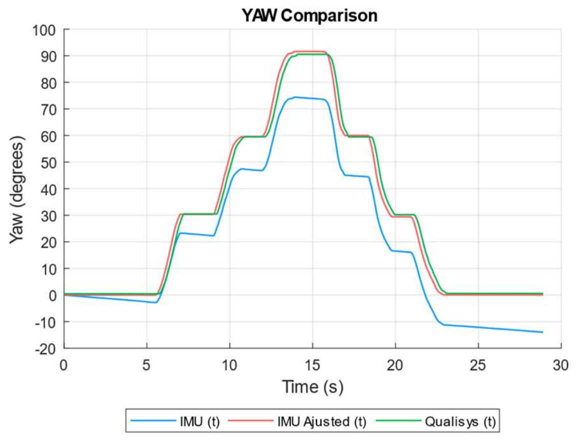

Figure 13: Yaw comparison

4.2. Experimental results

Now that the controller has been chosen and the car is assembled and had sensors validated, the torque vectoring implemented on the state machine needed to be tested too. To evaluate if torque vectoring makes any difference in trajectory, some comparison tests were performed. The car was programmed to start the trajectory in straight line and then perform a right or left J-shaped curves. It is important to explain that in each test (right and left) all the conditions were maintained, the only difference between each test was the toggle switch in the remote controller turned on activating the torque vectoring. The results for each pair of tests are shown in Fig.14 and Fig.15.

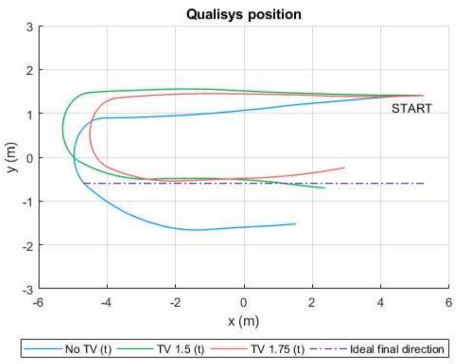

Figure 14: Left turn comparison

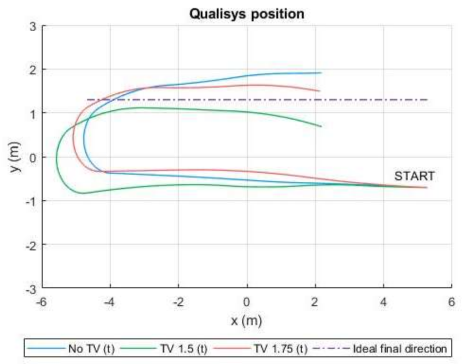

Figure 15: Right turn comparison

In the environment of the laboratorial tests, sending a signal of 0.8V for the steering corresponding to $\delta = 14.29° = 0.249\ rad$ it was expected to have a left turn with the expected radius $R \approx 1m$. Evaluating Fig.16 and Fig.17 it was possible to determine that the radius of the trajectory described was bigger than the expected, probably caused by the understeer of the car on the arena. The activation of the torque vectoring forces the car to describe the turn in a minor radius. In a limited space like the Qualisys arena, the starting point had to be moved in order to have similar trajectories. The major struggle was the inversion of the signal to the ESC responsible for speeding up the right wheel, sometimes it was not able to deliver the signal with efficiency and the car tended to slow down. However, the car ended up with good results, reducing the radius of the curve and having the best control of the PID in the straight. Despite all the issues associated with communications and hardware, good results were successfully achieved with this controller, ending up being only a PD, since integral part was zero.

5. Conclusion

Between all the setbacks during this time, the purpose of this work was successfully reached, providing a new platform able to be used in different control projects. The car was tested with the Raspberry Pi 3 but some issues related to the delay between Qualisys and the car were not solved in due time, so this configuration was set aside, but remained able to use as future implementation. The results obtained with the electric differential were good and prove that the torque vectoring allows the car to turn in a small radius of curvature, proving that the developed platform has a good capability to realize different tests describing the expected trajectories. The Qualisys system proved to be a very important tool for this work after the calibration was done. Since most of the work developed relies on the assembly and connections between different components, the knowledge achieved in terms of communications and programming languages was very satisfactory. With Traxxas remote, good results were achieved too, all the setup proved to be reliable, working well and avoiding the use of batteries. In the end the reliability of the state machine proved to be good because the transition from one state to other occurs immediately without delay. One of the setbacks was the upgrade of the encoders, but the proto-board developed proved to be a good choice as well because the pull-up resistor was easily implemented. In other hand, the most intricate part was the encoders' mounts' construction and the resizing of the belts.

References

[1] J. Antunes, A. Antunes, P. Outeiro, C. Cardeira, and P. Oliveira, "Testing of a torque vectoring controller for a Formula Student prototype," *Rob. Auton. Syst.*, 2019, doi: 10.1016/j.robot.2018.12.010.

[2] N. Gonçalo and P. Martins, "Integration of RC Vehicles in a Robotic Arena," no. November, 2016.

[3] R. N. Jazar, *Vehicle Dynamics - Theory and Application*. Springer.

[4] J. Pedro and M. Antunes, "Torque Vectoring for a Formula Student Prototype," no. June, 2017.

[5] Muggn, "GY80 arduino libray. url: https://github.com/muggn/GY80." .

[6] Arduino, "Wire library. url: https://www.arduino.cc/en/Reference/Wire." .

[7] Espressif, "ESP8266 AT Instruction Set," p. 70, 2016.

[8] A. Antunes, P. Outeiro, C. Cardeira, and P. Oliveira, "Implementation and testing of a sideslip estimation for a formula student prototype," *Rob. Auton. Syst.*, 2019, doi: 10.1016/j.robot.2019.01.018.