TÉCNICO LISBOA

Torque Vectoring for Race Cars

Luís Carlos Dias Duarte

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Carlos Baptista Cardeira
Prof. Paulo Jorge Coelho Ramalho Oliveira

Examination Committee

Chairperson: Prof. Duarte Pedro Mata de Oliveira Valério
Supervisor: Prof. Carlos Baptista Cardeira
Member of the Committee: Prof. Mário António da Silva Neves Ramalho

December 2021

Acknowledgements

I would like to thank my supervisors, Professor Carlos Cardeira and Professor Paulo Oliveira for all the support and availability during all this work, helping every time a setback appeared.

I would like to thank Eng. Camilo Christo and Luís Raposeiro for their support in the lab, helping in the development of new parts and teaching me how to calibrate the Qualisys system.

Special thanks to my parents, girlfriend and all my family for all their priceless and unconditional support, not only during this dissertation, but during all the years and all the setbacks in our life.

Special thank goes to my friend Eng. João Cunha, his parents António and Antónia Cunha and his brother Duarte Cunha for welcoming me into their home during the pandemic while working on the final stretch of my dissertation.

Special thanks to Maximilian Waibel, Marco Dorsch and Salvador Llácer Gómez with their parallel work and contribution for this project.

Finally, I would like to thank my friends for their support, friendship and all the relaxing moments out from work.

Resumo

Esta dissertação apresenta o desenvolvimento, construção, testes e análise de resultados obtidos de um carro telecomandado, com um motor em cada roda para que possam ser testados controladores de Torque Vectoring (TV) capazes de distribuir a força de modo independente por cada roda, equipado com sensores para avaliar a posição, velocidade, direção, entre outros dados telemétricos.

Para auxiliar na tarefa de converter informação enviada por computador, foi utilizado um conversor digital analógico ligado ao controlo remoto do carro.

Tendo como base o conceito de máquina de estados, o carro tem três estados definidos, sendo ativados através da seleção de um interruptor de três posições, instalado no controlo remoto. Este controlo remoto foi alvo de grandes alterações internas para poder enviar as ordens de controlo quer pelo computador, quer manualmente.

Todos os componentes instalados foram validados recorrendo também ao conjunto das câmaras do sistema Qualisys, existente no laboratório, de forma a obter resultados fiáveis, precisos e fidedignos.

Foi apresentado o estudo dinâmico do veículo, um estudo teórico e um estudo prático para dois controladores responsáveis pelo controlo de trajetória. O primeiro escolhido foi um Regulador Linear Quadrático (LQR) que apesar de em teoria ser capaz de originar bons resultados, na prática não conseguiu seguir a trajetória desejada devido a limitações físicas impostas. Para colmatar as dificuldades verificadas, foi implementado um controlador Proporcional-Integrativo-Derivativo (PID), tendo-se verificado melhorias nos resultados teóricos e experimentais, o que fez com que o carro seguisse a trajetória esperada.

Palavras-chave: Torque Vectoring, Sensores, Controlo, Máquina de estados, Qualisys, LQR, PID

Abstract

This dissertation 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) which, 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-Integrative-Derivative (PID) controller was implemented, with verified improvements in the theoretical and experimental results, which made the car follow the expected trajectory.

Key words: Torque Vectoring, Sensors, Control, State machine, Qualisys, LQR, PID

Contents

List of Tables

List of Figures

Nomenclature

Greek symbols

α_i	Sideslip angle of wheel i
β	Sideslip angle
δ	Steer angle
ρ	Air density
φ	Roll
θ	Pitch
Ψ	Yaw
ω	Generic angular velocity

Roman symbols

A_f	Front area of the vehicle
C_D	Drag coefficient
$C_\alpha, C_{\alpha f}, C_{\alpha r}$	Cornering stiffness, generic, front and rear
e_{ss}	Steady state error
F_x	Longitudinal force, forward force, traction force
F_y	Lateral force
F_x, F_y, F_z	Force components
I	Identity matrix
I_x, I_y, I_z	Principal moment of inertia
K_u	Stability factor
M_x, M_y, M_z	Roll, Pitch, Yaw Moment
M_z	Yaw moment, aligning moment
P	Electric motor power
R_{in}	Inner radius of the curvature
R_{out}	Outer radius of the curvature
R_w	Tire radius (wheel)
a_i	Distance of the axle number from the mass centre
d_{in}, d_{out}	Inner and outer distance travelled
M	Vehicle mass
M_p	Overshoot
r	Position vector

t	time
t_s	Settling time
t_f, t_r	Front and rear axle size (track)
v_x, v_y	Velocity components
x,y,z	Displacement

Subscripts

0	Initial value
i	Quarter suspension index
x, y, z	Cartesian components

Superscripts

{-1}	Inverse
{-}	Indication of variable computed in previous instant
B	Body reference frame
F,R	Front and rear
FL, FR, RL, RR	Front left, front right, rear left and rear right
T	Transpose

Others

$\dot{\psi}$	First derivative of ψ to time (yaw rate)

Acronyms

AC Alternating Current
ASCII American Standard Code for Information Interchange
CG Centre of Gravity
D2A Digital-to-Analogic
ESC Electronic Speed Controller
Esc Electronic stability control
GPS Global Positioning System
I2C Inter-Integrated Circuit
IDE Integrated Development Environment
IMU Inertial Measurement Unit
IP Internet Protocol
LED Light Emitting Diode
LLC Logic Level Converter
LPV Linear Parameter-Varying
LQR Linear Quadratic Regulator
MAC Media Access Control
PID Proportional-Integral-Derivative
PWM Pulse Width Modulation
RC Radio Controlled
RF Radio Frequency
TSL Torque Slip Limiter
TV Torque Vectoring
UDP User Datagram Protocol
Wi-Fi Wireless Fidelity

Chapter 1

1. Introduction

1.1 Motivation

In the future, most cars will use only electric motors. This way, the implementation of electric motors in each wheel will allow for the study of Torque Vectoring control strategies individually, per wheel. Torque vectoring control is a common case study and all started in the “Mechatronic Systems” course, where the base platform of the car was developed and assembled. For that, new parts were mounted on the customized old frame and electronically it works like a state machine.

In this work, the RC car built allows to implement control techniques on the prototype in order to test it in the real world. A strong investment on the communication will increase the quality of the response. This was so important because the response depends directly on the data received from the sensors installed on the car.

Usually in laboratorial projects, when a RC vehicle is used or transformed, the RF remote is neglected and discarded. The RF remote is a powerful tool in the communication system of the RC vehicles because it has almost no delay.

However, the development of an interface using equipment available on the laboratory, with quite a few changes on the TRAXXAS RF remote and using new approaches, is needed. One of the main goals of this dissertation is to use TRAXXAS RF controllers to send the control action to the car.

The developed work relies on the communications between the RC vehicle and the computer. The idea is that the car shares all the telemetry with the computer and, in turn, the computer is responsible for all the data treatment, allowing the car to describe the desired trajectory as precisely as possible.

Communications using UDP protocol are very useful to send and receive packages. This protocol seems to be the best choice for the platform. A deep study of this implementation in the car is needed as well.

1.2 Objectives and contributions

The inexistence of a platform capable of supporting torque vectoring tests inside the lab led to this dissertation, expecting that the developed platform would be capable of being used in the future, not only to control the RC vehicle using the computer, but to control manually, recording the telemetry when driven by human and be capable of describing trajectories in order to improve the control techniques in different study areas.

The main objective is to evaluate the response of the car with torque vectoring, having access to all the telemetry provided by the car and making use of the laboratorial vision tool called Qualisys.

The communication speed and the computational power installed on the RC vehicle is very important due to physical area limitations imposed by the vision system.

1.3 Thesis Outline

This dissertation is organized as follows: In Chapter 2 the state of the art related to torque vectoring and platforms already developed are presented. The details of the model, physical representation and the stability are explained and are presented in Chapter 3. The state space used, the control approaches and simulations done are presented in Chapter 4. In Chapter 5 the hardware development and the assembly of the car are presented. In Chapter 6 all the communication used in the dissertation are presented. In Chapter 7 the experimental results and validations are presented, analysed and discussed. Finally, in Chapter 8 some concluding remarks are presented, and possible future work is suggested.

2. **State of the Art**

In [1] Torque vectoring techniques were studied and tested in a Formula Student prototype. It was a complete work including controllers and estimators using real data and applying all the dynamics studied in plenty of courses of the master's degree. When the torque vectoring is implemented it allows to substitute the mechanical differential improving stability and handling of the car. The electrical implementation is a complex task, since it has to consider all the vehicle dynamics. The afore-mentioned article developed a simulation using a dynamic model of the vehicle, allowing for the fine tuning of the presented controllers. This approach proved to be a good one, avoiding damage to the real equipment and having the knowledge about how the changes in the controller affect the system, since good results were achieved in the real Formula Student car.

The study in [2] discarded the Traxxas RF remote because of the high cost of the interface responsible to communicate between computer and remote. The UDP communication method was used.

In [3] are shown the benefits of using UDP Protocol, proving its high efficiency and low CPU occupancy in communications.

In [4], the prototype used in this dissertation was built only to the point of being capable of moving mimicking the commands of the Traxxas RF. The sensors were mounted but not wired or programmed, showing no information at all.

In [5], theoretical concepts are discussed and applications about vehicle dynamics are shown. A complete description and example models are available, contributing in a large scale to the work developed in the torque vectoring techniques presented in this dissertation.

In [6], a torque vectoring control was used and implemented in a Formula Student car using PI and LQR controllers for yaw rate tracking. Linear and non-linear tests were performed, and his performance evaluated when implemented in the vehicle. The Ackerman Geometry was studied and applied to the front differential. This type of geometry is applied in different types of cars, transforming the input steering angle in two different angles, one in each wheel.

In [7] the front differential with two independent motors was studied. A Linear Parameter-Varying (LPV) controller was used to control the longitudinal and lateral behaviour, while a Torque Slip Limiter (TSL) was tuned to work as a trade-off between tracking the longitudinal velocity and the yaw rate.

In [8] and [9], trajectory tracking and control of an RC car on a circuit were studied. The platforms used were simpler but similar. The control techniques studied proved to be a good help for this dissertation, since the car need to be controlled in a circuit too.

In [10] Linear-Quadratic Regulator (LQR) information regarding application and use of this controller was found, among more helpful information about the search engine of MATLAB, allowing to improve the computation and the development of the work.

In [11] information and Arduino libraries for IMU (GY-80) are available to use and modify according to the necessary use. This community allows for code contributions and helps each member with code development.

In [12] all the information regarding to Wire library could be found. The I2C communication is described and examples of implementation are shown, helping in further development of the code needed in each different situation.

In [13], [14] and [15], developed code and information regarding its use, about memory flash steps and wiring diagrams was presented. The tools provided are very useful when using the ESP8266 module, allowing Wi-Fi communications.

In [16], implementation of the torque vectoring in one RC car and the communication system that use Qualisys camera system to evaluate the car's positioning inside the arena was presented. This communication systems uses a Raspberry Pi 3 model B to control the car and communicate with Qualisys. The steps to configure the Raspberry were provided by the authors of the work.

3. **Vehicle Model**

3.1 Vehicle coordinate frame

The dynamics and a model of the vehicle are the most important information that needs to be defined. Equations of motion must be computed, steering kinematics must be taken into consideration and wheel velocity vectors and slip angles must be implemented. The article [4] will be followed, noticing that rolling resistance, wind resistance and vertical force will be neglected because the car is small enough and it will not be too fast for these parameters to be important for this dissertation.

The vehicle body coordinate frame to be used is shown in Figure 1, having B(C_{xyz}), attached to the center of mass C.

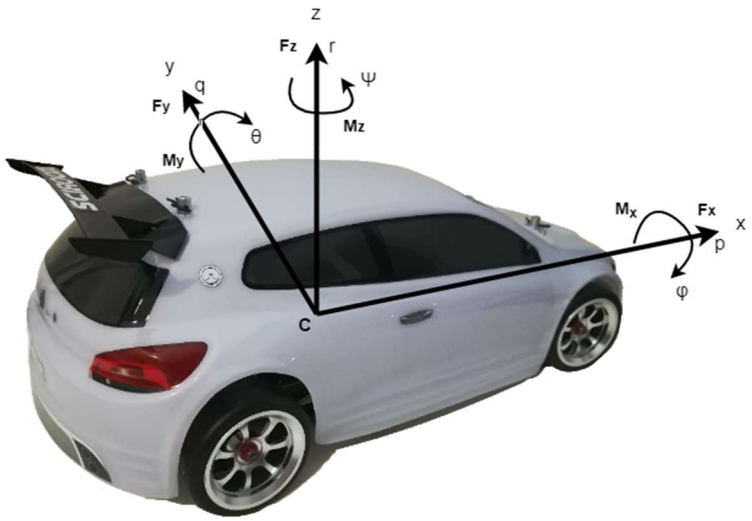


Figure 1 - Body coordinate system

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:

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

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

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.

3.2 Newton-Euler Dynamics

The model used is small and flat, and some simplifications were made as the car was considered to be a flat box moving horizontally. Three degrees of freedom are needed, being translation in x and y and rotation around the z axis. The Newton-Euler equations used on the body 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 the z axis, it is taken into account only the value of I_z

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

At some point, it will be important to compute the car trajectory. The trajectory can be defined as:

$$\psi = \psi_0 + \int r dt \tag{6}$$

$$x = \int \left(v_x \cos\psi - v_y \sin\psi\right) dt \tag{7}$$

$$y = \int \left(v_x \sin\psi - v_y \cos\psi\right) dt \tag{8}$$

where r is the position vector and ψ_0 is the initial yaw value.

3.3 Force system acting on the rigid body

The forces applied in the wheel i are given as:

$$F_{xi} = F_{xwi} \cos\delta_i - F_{ywi} \sin\delta_i \tag{9}$$

$$F_{yi} = F_{ywi} \cos\delta_i - F_{xwi} \sin\delta_i \tag{10}$$

$$M_{zi} = M_{zwi} + x_i F_{yi} - y_i F_{xi} \tag{11}$$

where x_i, y_i are the cartesian coordinates of each wheel in relation to the centre of gravity, δ_i is the angle between the wheel i with the x axis of the body

represented in the force vector system. This representation is shown in Figure 2.

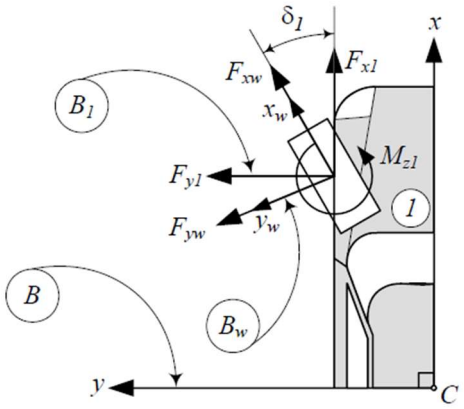


Figure 2 - The force system at the tire [5]

The tire lateral force (F_y) is given by:

$$F_{yi} = -C_\alpha \alpha_i \tag{12}$$

where C_α is the cornering stiffness of the tire, α is the tire sideslip angle and can be physically interpreted as the angle between the x axis and the velocity of the tire as represented. This representation is shown in Figure 3.

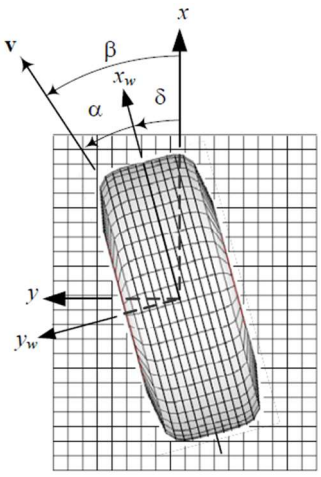


Figure 3 - Angular orientation of a moving tire [5]

Mathematically, α can be written as:

$$\alpha = \beta - \delta \tag{13}$$

$$\beta = \arctan\left(\frac{v_y}{v_z}\right) \tag{14}$$

For a small β, equation (13) can be written as:

$$\beta = \frac{v_y}{v_x} \tag{15}$$

As a simplification of the model in use, the bicycle model is used without the roll component. Neglecting the aligning moments, M_{zi}, forces applied are given by:

$$F_x = F_{xf}\cos(\delta) + F_{xr} - F_{yf}\sin(\delta) \tag{16}$$

$$F_y = F_{yf}\cos(\delta) + F_{yr} - F_{xf}\sin(\delta) \tag{17}$$

$$M_z = a_1 F_{yf} - a_2 F_{yr} \tag{18}$$

where the indexes r and f mean "rear" and "front" wheel respectively and a_1 and a_2 are the distance between front and rear wheel in relation to centre of gravity. In order to linearize the equations, small rotations should be considered having $\delta = 0$, and as such, the equations 16,17 and 18 can be written as:

$$F_x \approx F_{xf} + F_{xr} \tag{19}$$

$$F_y \approx F_{yf} + F_{yr} \tag{20}$$

$$M_z \approx a_1 F_{yf} - a_2 F_{yr} \tag{21}$$

It should be noticed that when using the 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 desired. 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 Figure 4.

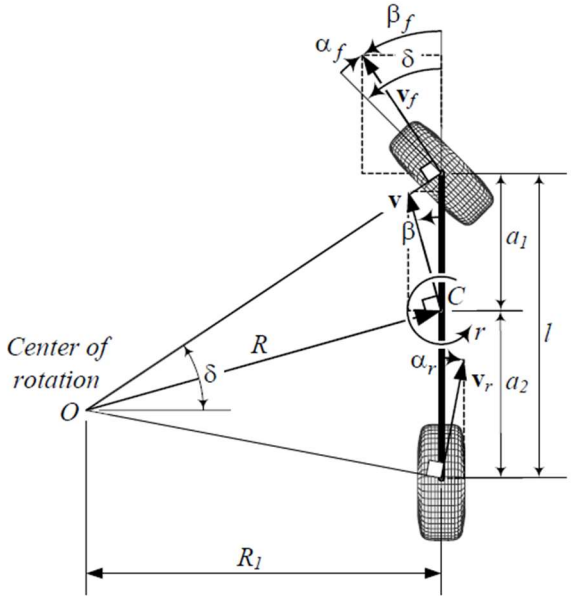


Figure 4 - Two-wheel model for a vehicle moving with no roll [5]

Taking equation 13 and applying for the front and rear wheel, it is written as:

$$\alpha_i = \beta_i - \delta \tag{22}$$

$$\alpha_f = \beta + \frac{a_1 r}{v_x} - \delta \tag{23}$$

$$\alpha_r = \beta - \frac{a_2 r}{v_x} \tag{24}$$

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

$$F_y = \left(-\frac{a_1}{v_x} C_{\alpha f} + \frac{a_2}{v_x} C_{\alpha r}\right) r - \left(C_{\alpha f} + C_{\alpha r}\right)\beta + C_{\alpha f}\delta \tag{25}$$

$$M_z = \left(-\frac{a_1^2}{v_x} C_{\alpha f} - \frac{a_2^2}{v_x} C_{\alpha r}\right) - \left(a_1 C_{\alpha f} - a_2 C_{\alpha r}\right)\beta + a_1 C_{\alpha f}\beta \tag{26}$$

The implemented state space is given by [5]:

$$\begin{bmatrix} \dot{v}_y \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{m v_x} & \frac{-a_1 C_{\alpha f} + a_2 C_{\alpha r}}{m v_x} - v_x \\ -\frac{a_1 C_{\alpha f} - a_2 C_{\alpha r}}{I_z v_x} & -\frac{a_1^2 C_{\alpha f} + a_2^2 C_{\alpha r}}{I_z v_x} \end{bmatrix} \begin{bmatrix} v_y \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{a_1 C_{\alpha f}}{I_z} \end{bmatrix} \delta \tag{27}$$

The longitudinal dynamic used is also presented by [5]:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{k v_0}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{P_{max}}{m R_w w_{max}} \end{bmatrix} pp \tag{28}$$

where P is the electric motor power, R_w is the wheel radius, w the wheel angular velocity, $k = \rho C_D A_f$ and pp is the pedal position.

3.4 Implementation of 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 [6], so the new state space is:

$$\begin{bmatrix} \dot{v}_y \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{m v_x} & \frac{-a_1 C_{\alpha f} + a_2 C_{\alpha r}}{m v_x} - v_x \\ -\frac{a_1 C_{\alpha f} - a_2 C_{\alpha r}}{I_z v_x} & -\frac{a_1^2 C_{\alpha f} + a_2^2 C_{\alpha r}}{I_z v_x} \end{bmatrix} \begin{bmatrix} v_y \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{a_1 C_{\alpha f}}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ \frac{1}{I_z} \end{bmatrix} M_z \tag{29}$$

and the car linear model is shown in Figure 5,

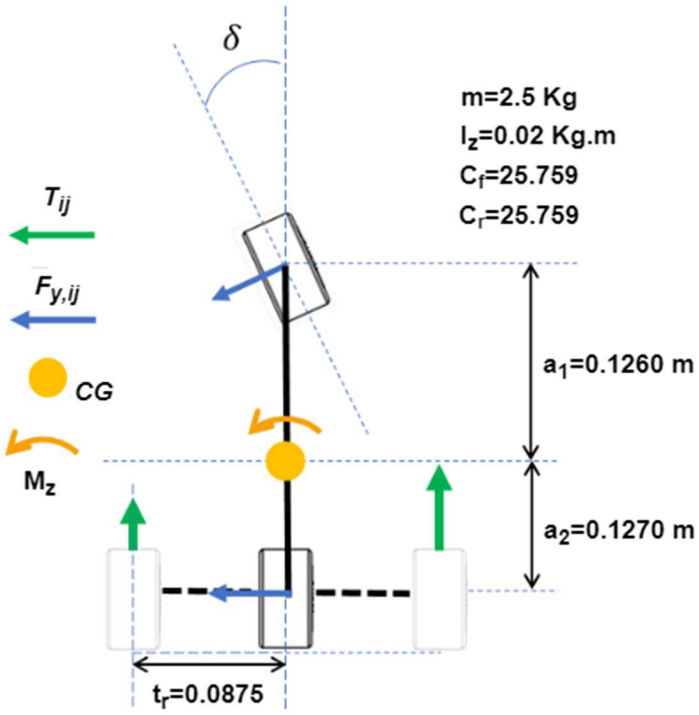


Figure 5 - Vehicle linear model with additional moment [6]

where *CG* is the centre of gravity, t_r corresponds to $w/2$, T_{ij} is the torque, F_y is the tire lateral force, I_z is the inertia moment around the z axis,*m* is the mass of the car, C_f and C_r are the cornering stiffness constants, front and rear respectively.

This added moment resulted from the difference between the left and the right wheel torque, T_{rl}, $T_{rr}$is given by [6]

$$M_z = \Delta T * t_r = (T_{rr} - T_{rl}) * t_r \tag{30}$$

In this dissertation, 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.

$$\Delta T = \frac{R_w}{2t_r} M_z \tag{31}$$

Thus, the M_z can be replaced by ΔT in equation 29. 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 [7], an approach is presented using a Linear Parameter-Varying controller (LPV) but in this dissertation, due to the loss of the front motors, only The Ackerman Geometry described in [6] will be considered.

3.5 Front wheel differential

To simulate mechanical differential in front wheels the steering angle should be known. Knowing the desired steering angle, the angles of the inner and outer wheel can be calculated as equations (32) and (33), using the Ackermann Geometry [6] theory. The Ackerman condition says that to have all wheels turning freely on a curved road, the normal line to the center of each tire-plane must intersect at a common point. This condition is needed when the speed of the vehicle is small and slip angles are zero because there is no lateral and centrifugal force to balance each other. The Ackerman Geometry is shown in Figure 6.

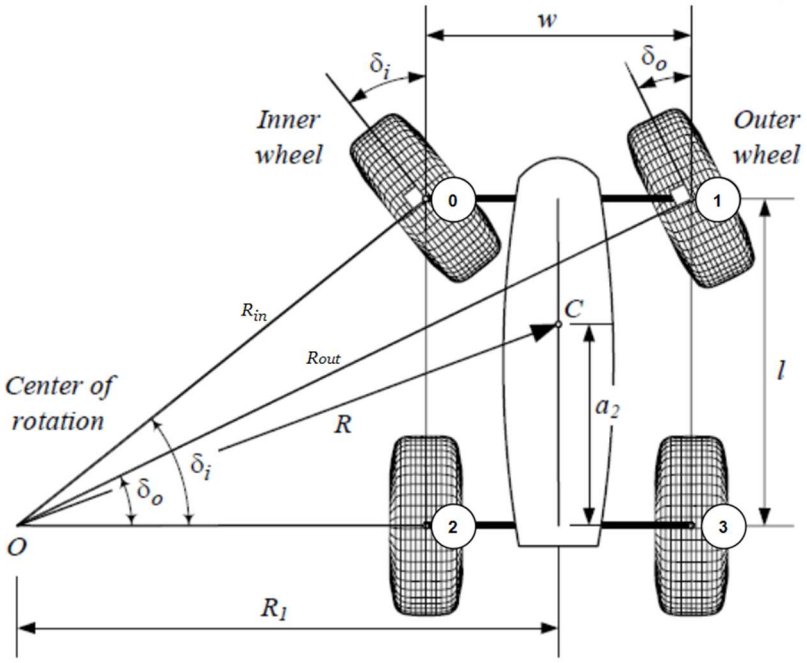


Figure 6 - Ackermann geometry [5]

$$\delta_{in} = \tan^{-1}\left(\frac{l}{l\cot(\delta) - \frac{t_f}{2}}\right) \tag{32}$$

$$\delta_{out} = \tan^{-1}\left(\frac{l}{l\cot(\delta) + \frac{t_f}{2}}\right) \tag{33}$$

To compute inner and outer radius:

$$\delta_{in} = \sin\left(\frac{a_1 + a_2}{R_{in}}\right) \tag{34}$$

$$\delta_{out} = \sin\left(\frac{a_1 + a_2}{R_{out}}\right) \tag{35}$$

Resulting in equations (36) and (37)

$$R_{in} = \frac{a_1 + a_2}{\sin^{-1}(\delta_{in})} \tag{36}$$

$$R_{out} = \frac{a_1 + a_2}{\sin^{-1}(\delta_{out})} \tag{37}$$

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:

$$\frac{d_{in}}{t} = \frac{2\pi R_{in}}{t} \tag{38}$$

$$\frac{d_{out}}{t} = \frac{2\pi R_{out}}{t} \tag{39}$$

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 given by:

$$\frac{v_{out}}{v_{in}} = \frac{2\pi R_{out}}{2\pi R_{in}} = \frac{R_{out}}{R_{in}} \tag{40}$$

and the radius described by the CG is gyven by:

$$R = \sqrt{a_2^2 + l^2\,(\cot\delta)^2} \tag{41}$$

3.6 Controllability

The controllability matrix allows to evaluate if the system is controllable.

Controllability is defined as the capability to transfer the system from any initial state, $x(t_0)$, to any other state in a finite interval of time.

The results that are shown in this section are obtained in a similar way of the study developed in [8] and [9], but with the characteristics of the new car.

Controllability matrix is defined as:

$$[B \quad AB \quad A^{n-1}B] \tag{42}$$

where A and B being the state and input matrix, respectively. Using the car characteristics presented above, it can be seen that the rank of the controllability matrix is 2, thus it is full rank and this MATLAB result is shown in Figure 7.

```
>> Co=ctrb(A,B)

Co =

   1.0e+03 *

         0   -0.2906
    0.1946   -5.3485
```

Figure 7 - Controllability matrix

3.7 Stability

To study the stability of the system, the root locus method was used, allowing to evaluate the system and the stability zone.

For better understanding of the system, the transformation from State Space to Transfer Function was performed using MATLAB commands. This way we can better evaluate the location of the poles and the zeros of the system. The transfer function obtained is given by:

$$tf = \frac{162.3s + 2238}{s^2 + 41.22s + 378.8}$$

where the zeros were (-13.79) and poles were (-27.39, -13.83). This way, the root locus of the system is shown in Figure 8.

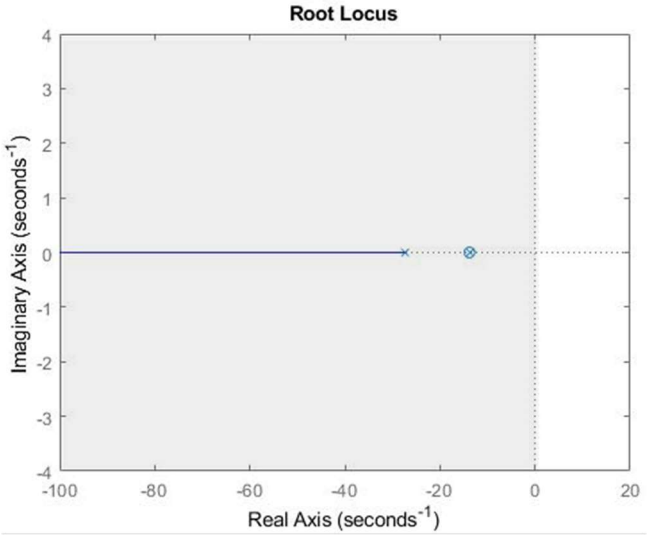


Figure 8 - Root Locus

As could be seen, the pole and the zero close to each other were on the shaded area and the other pole goes to minus infinity proving that the stability exist because all the poles and zero are in the negative side of the real axis.

4. **Proposed Controller**

To control the torque in each wheel, a Linear Quadratic Regulator (LQR) and a Proportional-Integral-Derivative (PID), using the linear model, will be used assuming it is enough. The system requirements are presented in Table 1.

Table 1 - System requirements

	Specs required
$M_p\ (\%)$	0
$t_s\ (s)$	<1
$e_{ss}\ (\%)$	0

The method to be used is to distribute the left and right torque, proportional to the amount of steering input $\Delta T = f(\delta)$.

4.1 Data acquisition hardware

To control the car, at least 2 sensors are needed: Inertial Measurement Unit (IMU) and encoders. Using an IMU, while a lot of telemetric data could be obtained, for this dissertation only acceleration and gyroscope data were used. Having the acceleration of all the axes, the speed of the centre of gravity of the car can be estimated. The most important parameters would be the yaw since it will be the reference to the control loop. With encoders, the speed of each wheel can be estimated by differentiating its values in time.

To close the control loop, Qualisys system was used to have the real yaw angle. This architecture joined with the code implemented in Arduino Due was the best configuration found. The memory and processing capacity were the main reasons in choosing an Arduino Due as the main microcontroller. The differential controller will be all coded and implemented inside the Arduino to try avoiding major communications with Simulink, while Simulink was only responsible for processing the received car telemetry and the Qualisys data, applying the trajectory controllers.

A failsafe system to stop the motors in case of emergency was implemented in the RC transmitter. This failsafe was implemented in a switch, which will allow the user to change from direct drive mode, where the Arduino will be a bypass from the receiver for the motors just doing data conditioning, to torque vectoring control, lighting up a small LED implemented in the remote controller or activating the failsafe.

4.2 Arduino differential control

To implement control, a reference signal must be created. The chosen signal is 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

$$K_u = \frac{a_2 m}{C_{\alpha f}(a_1 + a_2)} - \frac{a_1 m}{C_{\alpha r}(a_1 + a_2)} \tag{43}$$

If K_u is positive (K_u> 0), the car is said to have an under-steer behaviour. In case of K_u< 0, the car has a oversteer behaviour. When K_u= 0, it means the car has a neutral steer (ideal yaw rate). Being the latter the ideal, it is chosen as the reference. However, this can take to over-steer instability [6] and the under-steered vehicle is chosen.

The desired yaw rate can be defined by the velocity and the radius of curve:

$$\dot{\psi}_{desired} = \frac{v_{CG}}{R} \tag{44}$$

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

$$\frac{1}{R} = \frac{\delta}{(a_1 + a_2) + K_u v_{CG}^2} \tag{45}$$

With equation (44) and (45), a function of δ to find yaw rate desired is computed as:

$$\dot{\psi}_{desired} = \frac{\delta}{(a_1 + a_2) + K_u v_{CG}} \tag{46}$$

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.

4.3 Linear Quadratic Regulator (LQR)

LQR is an optimal control solution for linear systems. To design this type of controller an optimal gain K is calculated to minimize the energy function J.

The state space will have v_y, $\dot{\psi}$ and ψ as state variables. For 2 motorized wheels, δ will be the only control input, since the δ will be imposed by the driver [6]. To estimate the lateral velocity, an integration of the acceleration acquired from the IMU should be done.

The performance index to design the LQR controller is written in equation (46). It is a quadratic cost function and the main objective is to find a state-feedback law $u = -Kx$ that minimizes this function.

$$J(u) = \frac{1}{2}\int_{t_0}^{t_f} [(X_d - X)^T Q (X_d - X) + u^T R u] dt \tag{47}$$

Since the velocity was imposed to be constant, the longitudinal dynamics ended up being neglected too. It is necessary to add a state variable to the system to have an output which is the yaw ψ and the new state space is given by:

$$\begin{bmatrix} \dot{v}_y \\ \ddot{\psi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{m v_x} & \frac{-a_1 C_{\alpha f} + a_2 C_{\alpha r}}{m v_x} - v_x & 0 \\ -\frac{a_1 C_{\alpha f} - a_2 C_{\alpha r}}{I_z v_x} & -\frac{a_1^2 C_{\alpha f} + a_2^2 C_{\alpha r}}{I_z v_x} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_y \\ \dot{\psi} \\ \psi \end{bmatrix} + \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{a_1 C_{\alpha f}}{I_z} \\ 0 \end{bmatrix} \delta \tag{48}$$

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 (49) and the closed-loop eigen values e [10].

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{49}$$

K is then derived by P using:

$$K = R^{-1}B^T P \tag{50}$$

The Q matrix used is:

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

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

4.4 LQR Simulation

To test all the assembly of the car and the communications system the simple state space from equation (48) was used. With this state only the yaw reference must be followed.

To simulate the system and design the controller, all the car parameters must be taken into account. These characteristics were shown previously in Figure 5.

The step response of the system for a step of $40°$ (approximately 0.7 rad) is shown in Figure 9.

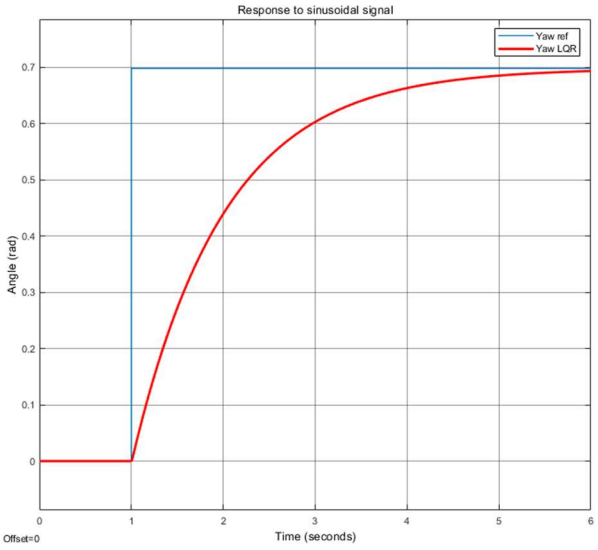


Figure 9 - Step response

As can be seen, the step response is very slow with these first parameters. By manipulating the values of Q matrix and evaluate the response again, it is possible to improve it. This was made iteratively until a reasonable response was obtained. The new Q matrix was:

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 100 \end{bmatrix}$$

The R value was the same, $R = 1 \times 10^{-1}$. The gains of the controller were:

$$K = [6.796 \quad 3.194 \quad 31.623]$$

And the results are shown in Figure 10.

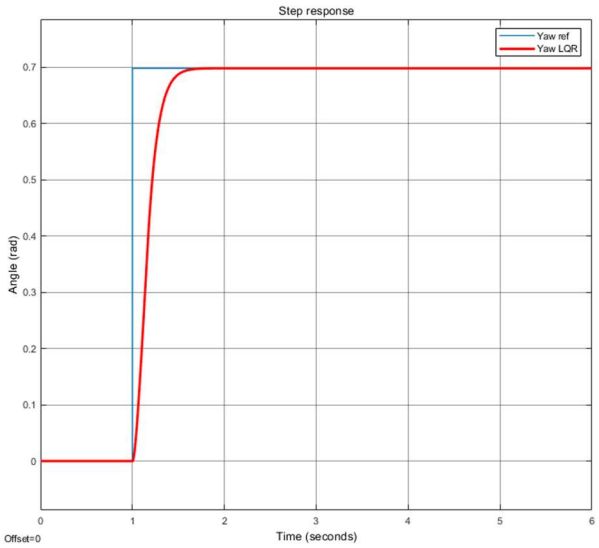


Figure 10 - LQR Step response

It is important to refer that, for the previous results, only the system response was considered, no physical limitations were considered. Further ahead, on the laboratorial implementation section, such limitations will be considered because the maximum steering angle for each side is approximately $30°$.

4.5 PID Simulation

The state space from equation (48) was used in the following simulation as well, allowing for the comparison between the two control methods. The 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 manually adjusted until had:

$$P = 50$$

$$I = 0.5$$

$$D = 5$$

Note that similar response, identical rise time, settling time and no overshoot were the goals for the step response in the same conditions. The step response is presented in Figure 11.

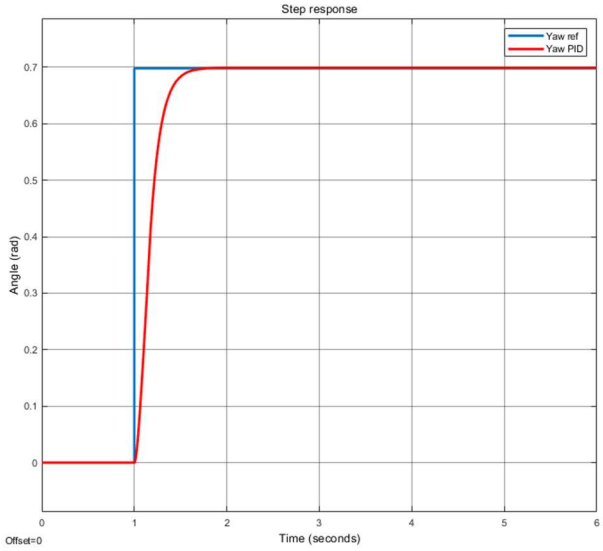


Figure 11 - PID Step response

5. **Hardware implementation**

To implement Torque vectoring in a RC car, if the car only had one motor it should have a braking system that allows to brake each wheel independently. The solution presented is to build, from sketch, a car with a motor in each wheel, independently controllable.

The first thing is to implement a switch in the remote controller to choose car modes as will be explained in section 5.2, in which a simple finite state machine can be controlled.

After implementing this switch, the signal sent from the controller to the receiver, must be configured using Arduino Due to make the motors rotate in the same direction. The differential of the car must be implemented electronically, this is presented in section 5.3.

5.1 Car Assembly

From an original base of a RC car, it was built new supports for rear and front axles in order to make the new motors fit, the result is shown in Figure 12.

Figure 12 - Supports for motor fitting

The important characteristics of the motors, the model being Turnigy 4206 530kv Brushless Multi-Rotor Motor, are presented in Table 2.

Table 2 - Motor characteristics

Power	130W
Weight	68g
Max Currents	20A
Max Voltage	16V
Kv	530rpm/v

For each motor it is used an electronic speed controller (ESC). In each wheel is installed an encoder to control wheel position and estimate the wheel velocity. Two slave ATMEGA328 were needed to be used as counters, and the Arduino DUE was used as Master to control the RC car, sending information to the laptop wirelessly using an ESP 8266. The communication between each component will be explained on chapter 6.

To maintain the suspension geometry performance the axles were made shorter as shown in Figure 13.

Figure 13 - Axle shortened

To reinforce the top part of the suspensions, an aluminium sheet was cut and modelled. This reinforcement allows to separate all the sensitive electronics from the motors that work with AC current, avoiding noise capturing.

All the wiring went through a hole made on the sheet and a connector was assembled on the bottom part, allowing future disconnection for maintenance or replacement of parts without resoldering components. On the bottom part, the steering servo was attached too, as well as two switches that allow turning the motors on and off and another switch for the electronics. These characteristics are shown in Figure 14.

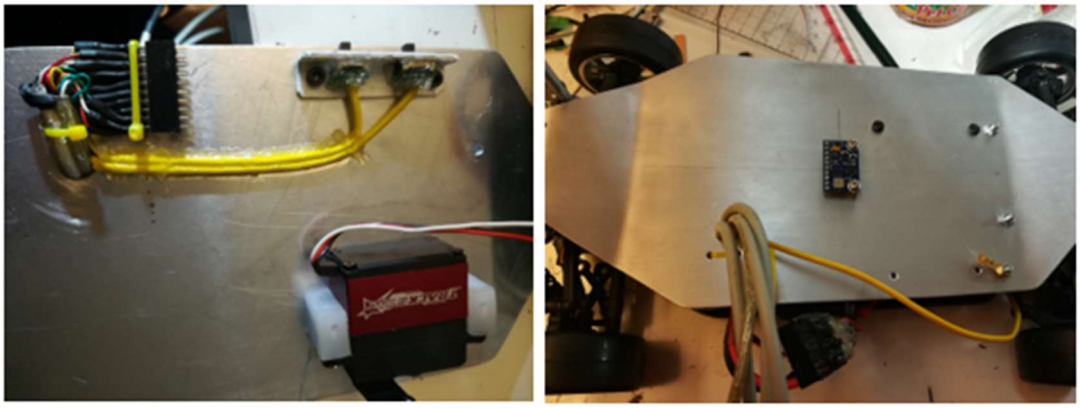


Figure 14 - Bottom part (left) Top part (right)

It is possible to see that the IMU was attached approximately in the centre of the car to avoid more data treatment.

One of the most critical and important parts of this dissertation was the continuous development of the proto-board responsible for connecting all components between each-other. The proto-board allows some versatility to modify a few things and create new ones. The first prototype of this board is shown in Figure 15.

Figure 15 - Developed proto-board

During the development of the work, the board suffered some changes in order to improve the quality and stability of the whole platform. With this configuration it is possible to enumerate some advantages/changes:

- ➔ Eliminate one of the Logic-Level-Converters (LLC);
- ➔ Create a speed limiter using a jumper;
- ➔ Create pull-up resistor for the new encoders;
- ➔ Easy troubleshooting;

➔ Easy access to program the different Arduinos;

➔ Reduce hanging wires;

Due to malfunction, probably caused by trying to brake the car, one ESC burnt out, leaving the car with only 3 powered motors. At that point, the best choice was using only rear wheel drive configuration, letting more space available to future modifications.

One of the biggest changes performed in the car was the assembly of two new encoders with more resolution. The explanation for this change is explained in more detail showing results in Section 7.1.2.

The assembly of the new encoders led to a reorganization of the components location. New mounts for the encoders were built and attached to the car, as well as new customized belts and pulleys. These changes are shown in Figure 16.

Figure 16 - New encoders assembly

The final overall configuration of the car without the protective body is shown in Figure 17.

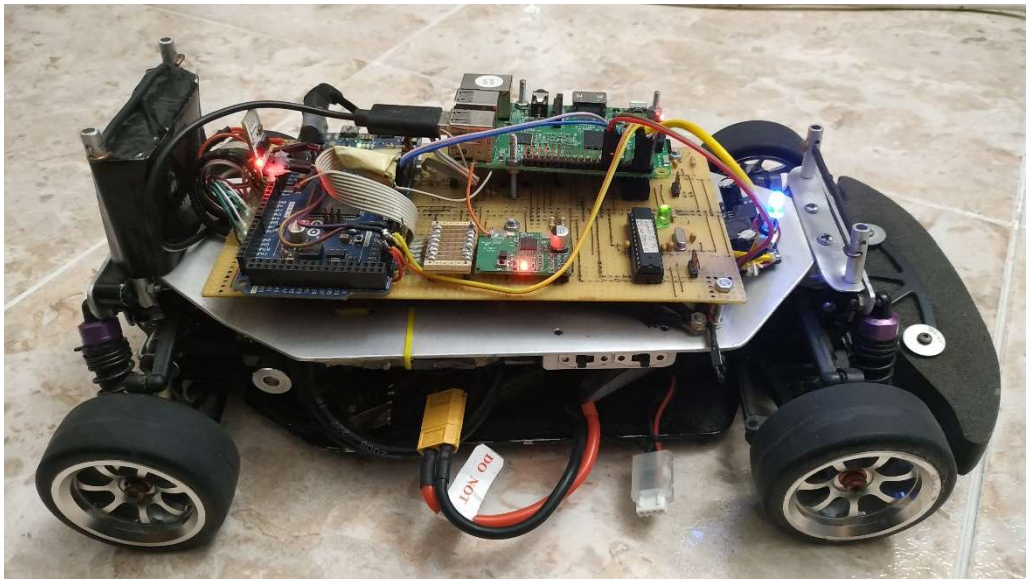


Figure 17 - Final configuration

5.2 Remote Controller Modification

In the remote controller, a switch for three different modes was implemented. The left position activates the mode where the car has the torque vectoring control on. In the middle is the mode where the Arduino is just bypassing the signal from the controller to the motors. Finally, the right position stands for the failsafe position, where the power from the motors was cut. This switch sent a PWM signal for the Arduino. The implemented circuit inside the remote is shown in Figure 18.

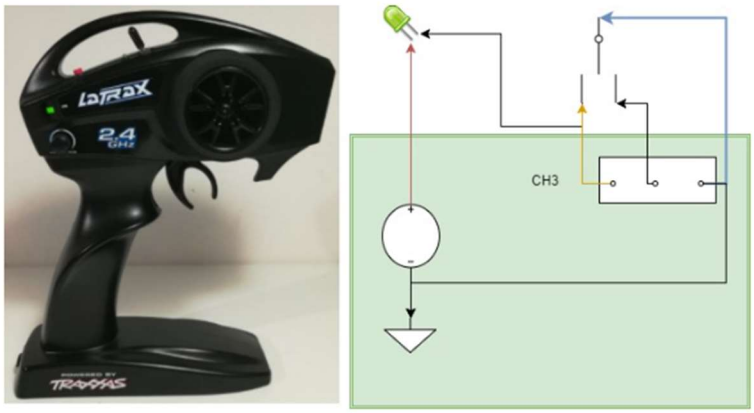


Figure 18 - Implemented circuit in the controller

5.3 Arduino algorithm

The following flowcharts illustrate how the Arduino code works. The two first ones can be considered as the main, Figure 19 and Figure 20. The following ones are the blocks for the differential in Figure 21 and to calculate the desired angular velocity in Figure 22. The Arduino Uno flowchart responsible for the interrupt count is shown in Figure 23.

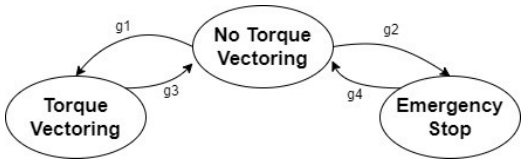


Figure 19 - Finite State Machine for the car modes

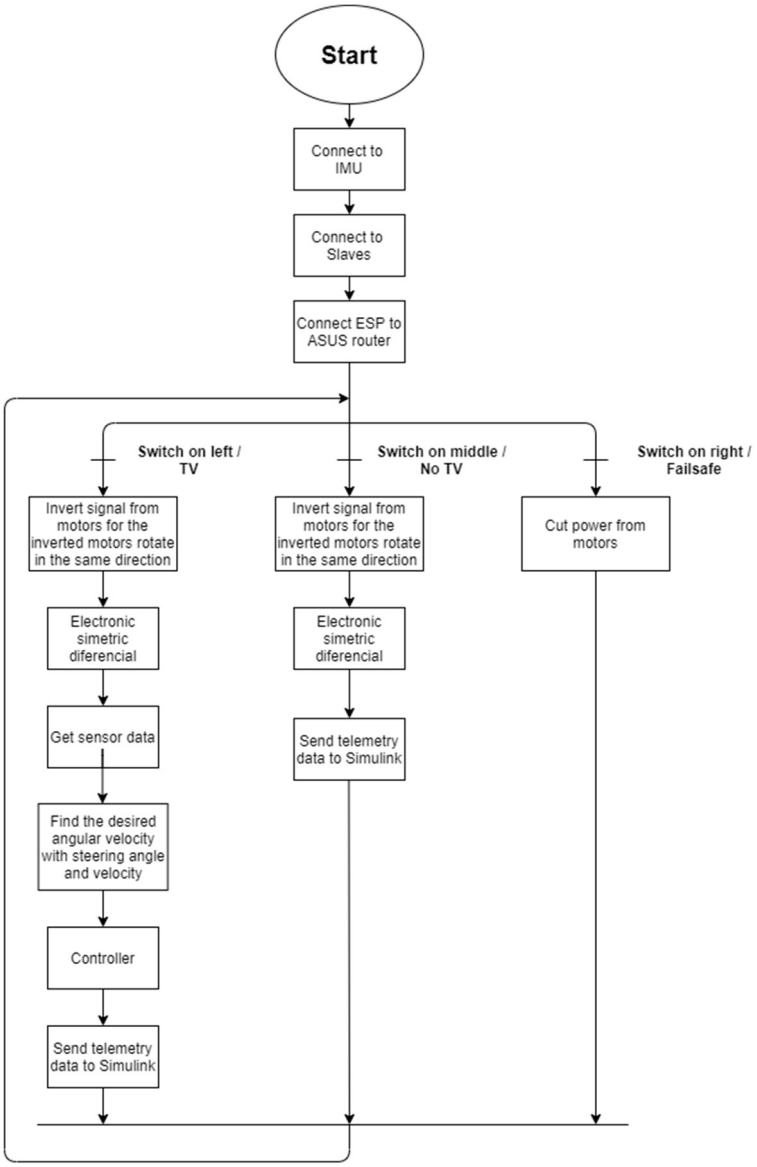


Figure 20 - Arduino Due (Master) flow chart

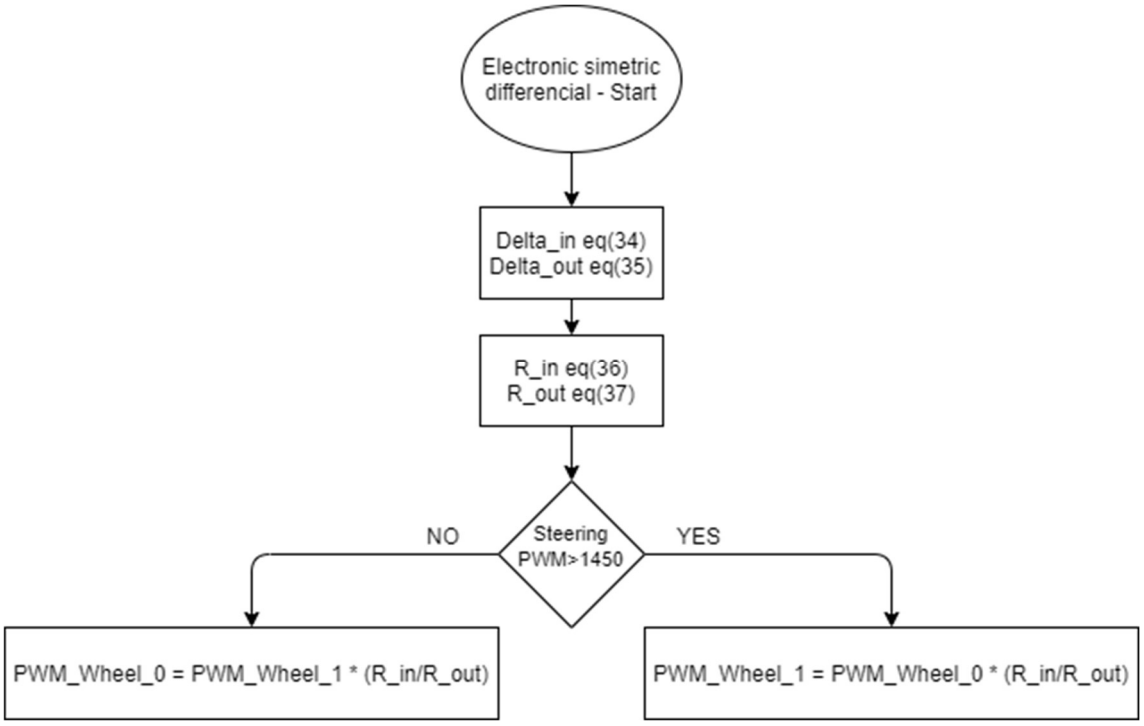


Figure 21 - Electronic differential flow chart

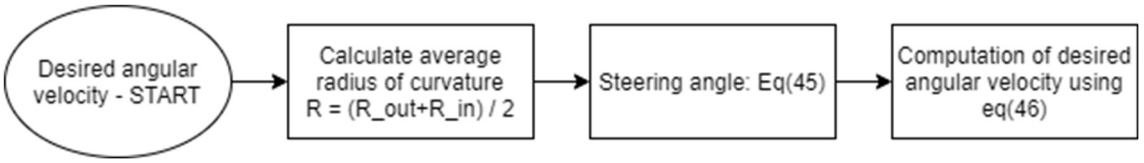


Figure 22 - Desired angular velocity flow chart

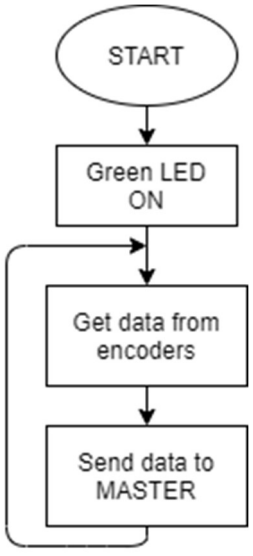


Figure 23 - Arduino Uno (Slave) flow chart

5.4 Electric diagram

The importance of the wiring diagram in this type of platform is so high because it allows to spend less time in troubleshooting in case of crash, maintenance or changing parts. Once only rear motors were used, it is possible to see the red cross over the two front motors. The wiring diagram is shown in Figure 24.

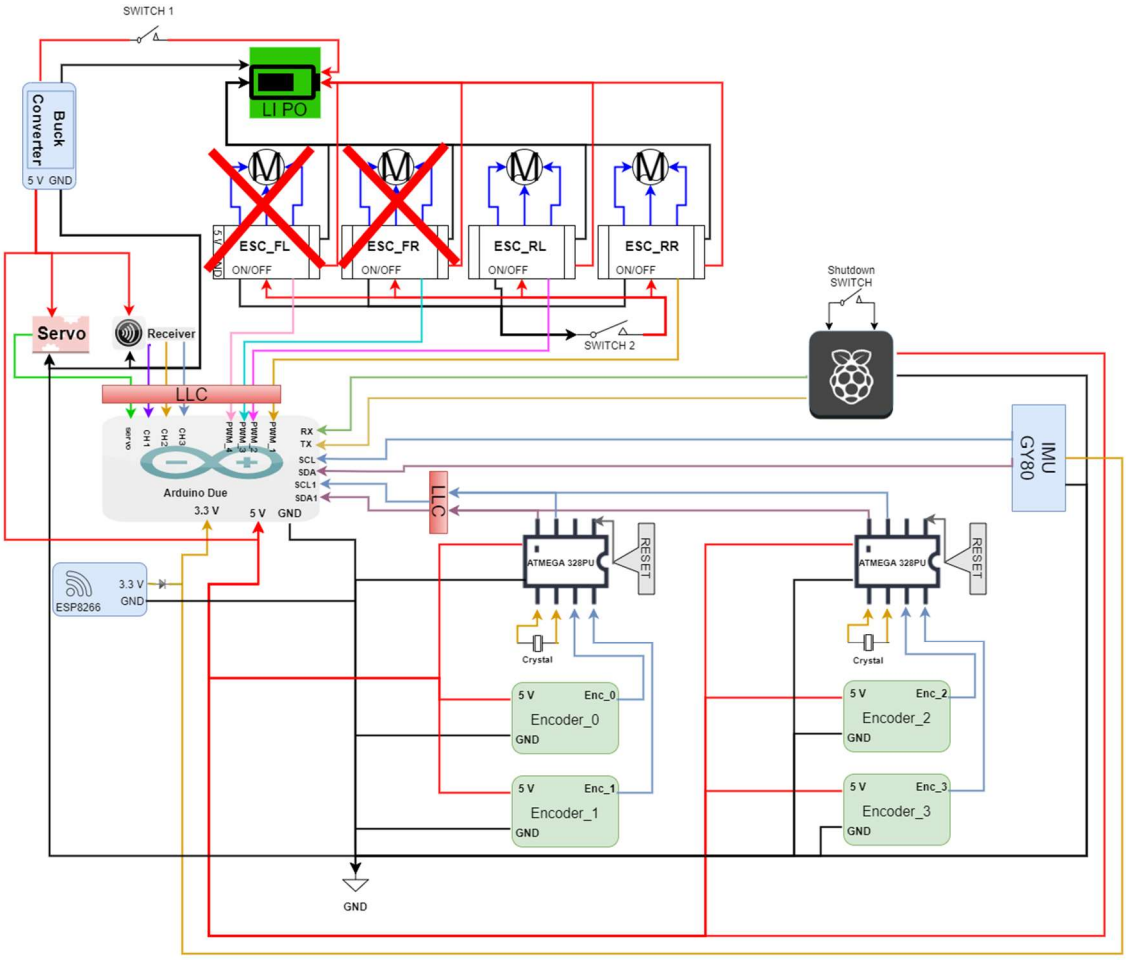


Figure 24 - Wiring diagram

6. Communications

6.1 Communication cycle

It's very important to correctly establish communications technologies and protocols, since all the telemetry data should be sent to computer with near zero delay. Using Wi-Fi connection, UDP protocol was established to send telemetric data from the car to the computer. The converter board NI USB 6008 was used to send commands of steering and power to the car. This communication from the computer to the car was established using the D2A converter.

In the next sub-sections, the different used protocols and type of technologies used will be explained in further detail.

Most of the invested time in communications was in the car side due to the use of several different microcontrollers connected between each other's requiring different protocols. The communication diagram between Qualisys, computer, remote control and the car is shown in Figure 25.

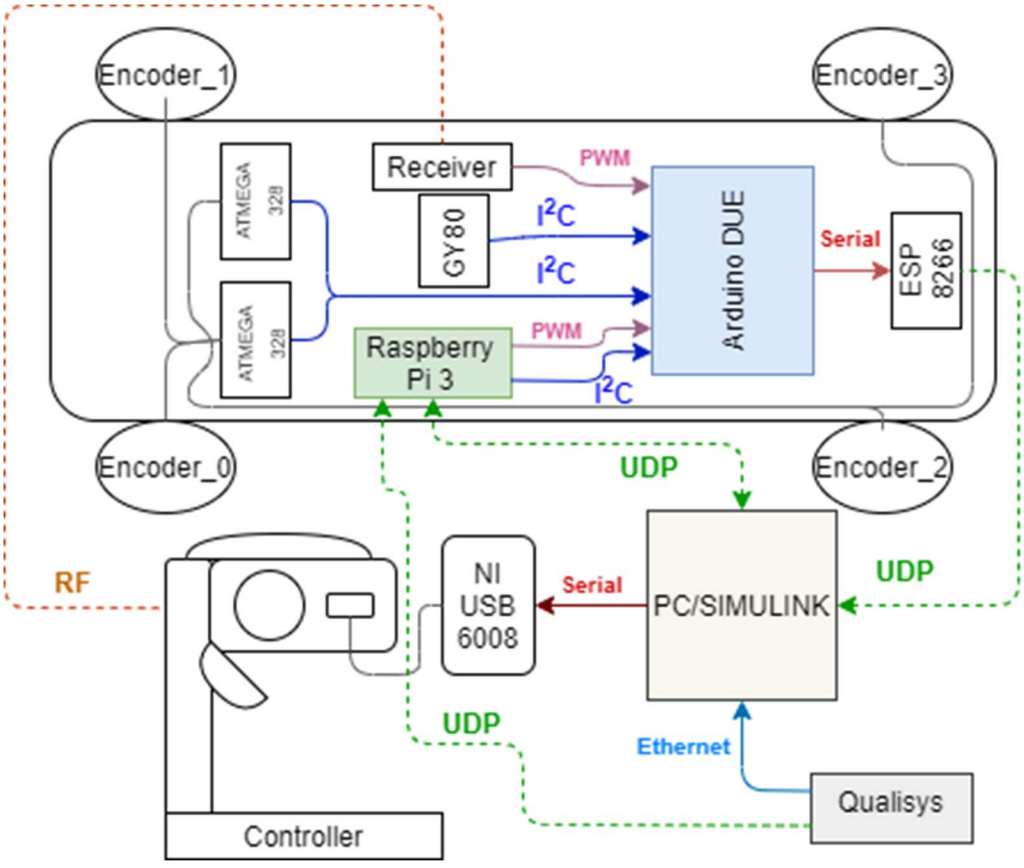


Figure 25 - Communication diagram

The data from the IMU and encoders were obtained using Arduino libraries such as “GY80” [11] and "Wire" library to get data using I2C communications [12]. 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 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, it was possible to compare the results between old encoders and the new ones, this will be explained in section 7.1.2.

New code was developed to connect and receive the IMU data allowing versatility in data treatment. The available library code was adapted to fulfil some requirements and be able to receive and measure accurately all pretended values. A validation of the values was made and is presented in section 7.1.1.

An issue was detected originating a persistent freezing of the Due and causing a big amount of crashes. The responsible for this issue was the ESP power supply and it was solved using one IN4007 diode on the ESP power wire and connecting the IMU in a dedicated I2C channel. This diode is visible in the wiring diagram shown in Figure 24.

The UDP protocol was already used in [2] but now is used with upgraded Simulink toolboxes to receive data. In this case the data sent by ESP8266 is transformed in one string and then decoded on Simulink. The ESP8266 send UDP messages taking advantage of the AT Commands [13]. This AT Commands are part of the AT Firmware of the ESP8266 and they allow a large variety of customization and uses. To flash the memory, “*flash_download_tools v3.6.8*” program was used.

Qualisys connection to the laptop was the same one used in [16]. First the config.txt file must be configured saving the changes. The configuration used is presented in Table 3.

Table 3 - Qualisys configurations

Config.txt
Qualisys Computer information: QualisysIP=169.254.54.27 QualisysPort=22223 NumberOfObjects=1 Local Computer information: Port to use=9091 Communication Type (0-TCP; 1-UDP): Comms=1 Host Computer information (only for UDP): HostIP=192.168.1.40 HostPort=9089 Debug Information (Flags): Messages in queue=1 Print message bytes=1

After save the previous file, the “Qualisys UDP Receiver.exe” file must be open. If everything is good, a black screen appears changing values quickly. To properly exit the program, press simultaneously the keys:

A+S+ENTER.

It is important to mention that randomly the Qualisys required special attention when gathering data because often system reboots were needed due to delays in the communication. To improve the results and accurate validations, a new calibration of the system was made.

6.2 Raspberry Pi

A Raspberry Pi 3 model B was implemented on the car trying to receive and control directly the data from the Qualisys system, however and after a

large amount of attempts this configuration revels a higher delay in the communication compared with previous configuration.

The use of this solution was abandoned. However, all the connections and devices were left in the car being available for future works. The adjustments needed are to change the Arduino code, replacing it by the appropriate code to work with the Raspberry. The code is already developed allowing to receive PWM commands from the Raspberry and is attached with all the technical information that is commented inside the Arduino Due. To connect the IMU to the Raspberry is necessary to disconnect it from Due to avoid crash of the state machine when Raspberry is running.

It is important to remind that a new shutdown switch was implemented on the Raspberry Pi to avoid corruption of the system when the car switch is turned off. When the button is pressed once, it makes the shutdown program run. Then, when the blue light turns off, the green light of the Raspberry shows a steady green and then goes off, it is secure to shut down the car. Note that sometimes when Simulink is running and the button is pressed for the first time, the Raspberry restart and the process must be repeated in order to properly shutdown.

6.3 ESP8266 AT Commands

The Wi-Fi module ESP8266 was used to send the telemetry data from the car to the computer. In order to be able to use the AT Commands a flash of the memory with the original firmware was performed.

AT Commands allows to configure the working environment of the ESP8266 according to the main objectives of the desired work. To configure the connection between the ESP and the Router, the Monitor Series on Arduino IDE, was used following the next steps:

- AT
- AT+CWMODE = 1
- AT+CWJAP = ”ASUS”,”latraxcar”

- AT+CIPSTART = ”UDP”,”192.168.1.40”,25001
- AT+CIPSEND = (STRING)

After the test was concluded with success, the steps of this communication were made implementing the code in Arduino DUE. The main structure of this communication code was from [15] with some specific modifications accordingly to the network connection. The string was responsible for transport the telemetry from ESP to Simulink.

More AT Commands exist, and some were used to test the hardware reliability.

To be able to understand how connections were made and how the diode was implemented in the circuit, the wiring diagram is shown in Figure 26.

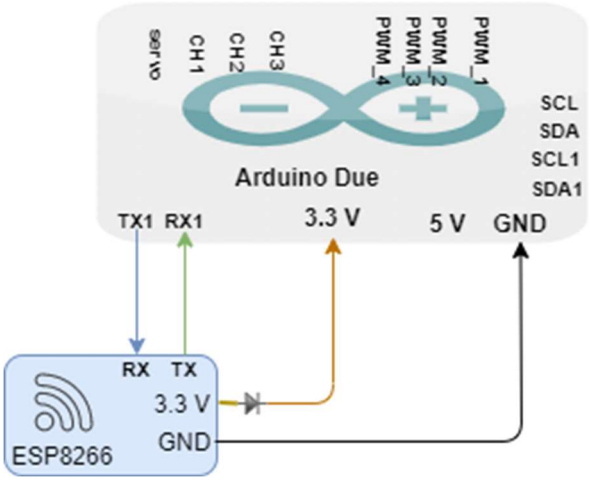


Figure 26 - ESP8266 Wiring diagram

6.4 UDP (Data Sent and Received)

The UDP is one of the most powerful protocol in this communication environment because it allows wireless data transmission between the car, computer and Qualisys. This protocol relies on two steps: send data to computer and receive data in the computer. To send the data the Due needs to create a string with the values in a specific layout for the computer to be able to read it on Simulink. The AT Commands were used to create and constantly send the data string. The format of the string sent was:

rpm0, rpm1, rpm2, rpm3, pitch, roll, yaw, ch1, ch2, packet, accx, accy

where 8bit string need to be specified and only after that give order to send, accordingly AT Commands specifications.

To receive the data on Simulink the Data Acquisition toolbox was used, having the UDP receive block configured, the message should be received with almost no delays. It was noticed that for the Simulink to read the configured UDP port it is imperative that the Windows firewall stay disconnected, or the permissions of the MATLAB to see over the firewall must be reviewed.

Once the message was received, decoding the message was the priority. With the help of the toolbox mentioned this job was done with success, the main blocks used were to convert from ASCII to string and to read the string converting to double values.

To prevent any more delays or external influence in the control, a dedicated router was used and configured. The configuration of the router was made associating the same IP address to each Mac address of the ESP8266 and of the computer. This was necessary because if the IP of any of them changes, the message will not be successfully transferred between the devices, resulting in extra time to configure the connection every time when connecting to the Simulink. The stablished connection diagram is shown in Figure 27.

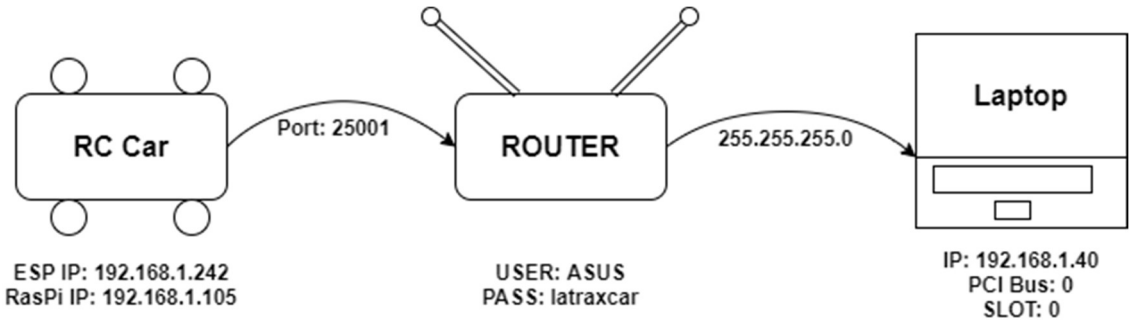


Figure 27 - Network Diagram

6.5 Send Control Action

The RF transmission used by the original Traxxas controller was considered because of the almost inexistent delay in its transmission. To use this controller, many hardware modifications were necessary and performed.

The signal emitted and the voltage values associated were studied in order to replicate them with the NI USB 6008. The first step was a deep study of the limits of the voltage values and which wires were responsible for send the PWM signal to the receiver in the car. One IN4007 diode was used to protect the batteries against the input voltage of the NI USB 6008. The inside connections are shown in Figure 28.

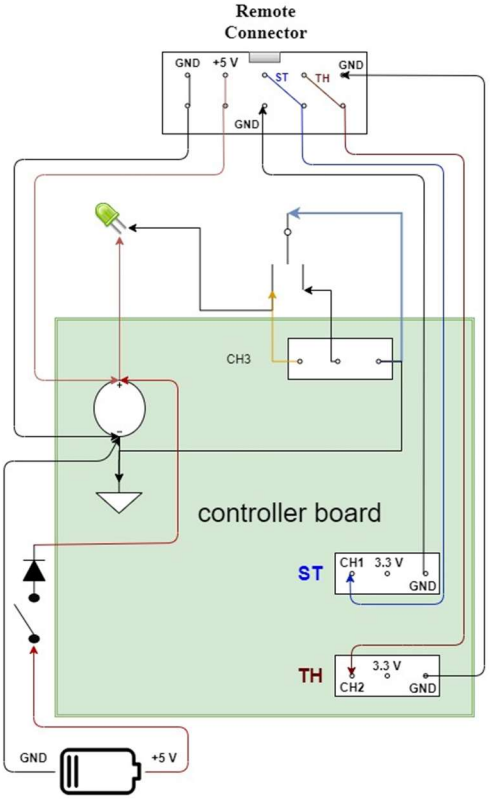


Figure 28 - Remote inside connections

It was noticed that the signal oscillates between approximately 0V and 3.3V (left or right/forward or back), and that the neutral position was around the mean of the values 1.65V. In the case of the steering, the physical limitation was the maximum steering angle of 30º for each side. Associating the voltage with the PWM and steering angle it was possible to achieve the relation of 0.056V/deg. Having a steering signal of 0.8V on the remote means that the steering angle will be -14.29 degrees.

To read the values, Simulink was used with NIDAQmx AddOn installed. In other hand to send the control action some details had to be taken into account, such as: saturation of the signal in order to prevent the physical failure of the remote, the first start of the program needs to be adjusted in order to

allow the controller to be paired with the receiver, and for that the neutral position of the channels must be settled.

Finally, the independency of the system was achieved with the supply of the working voltage (5.0V) to the controller, allowing to automatically be turned on when connected to laptop, not depending on his own batteries. Special care was taken in a way to make it look clean and useful and the controller is shown in Figure 29.

Figure 29 - Remote and NI USB 6008

. The diagram of this layout is shown in Figure 30.

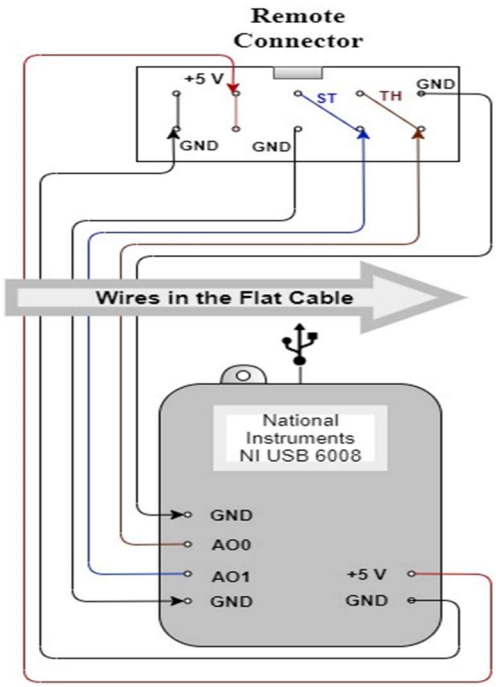


Figure 30 - Remote outside connections

6.6 Simulink Project

A Simulink project was built to communicate with the car and computing the data received from it and send commands back again. This was achieved working in small parcels with different objectives and then joining all the blocks together in one single project.

The NI USB 6008 must be connected before running the MATLAB, as have been said before the firewall must be disabled because of the UDP connection with the ESP8266.

To prevent equipment failure, the project needs to run at least one time without connecting the remote to let the NI interface assume the reference values.

The project was able to record trajectories created manually or program time functions. A fail-safe mechanism is available to prevent accidents, noting that the state machine fail safe is always the best way to cut power immediately because it is the master of the system. A sketch of the program developed in Simulink is shown in Figure 31.

The trajectories were described by time-based functions and show small errors, affirming the good communication between car and Simulink.

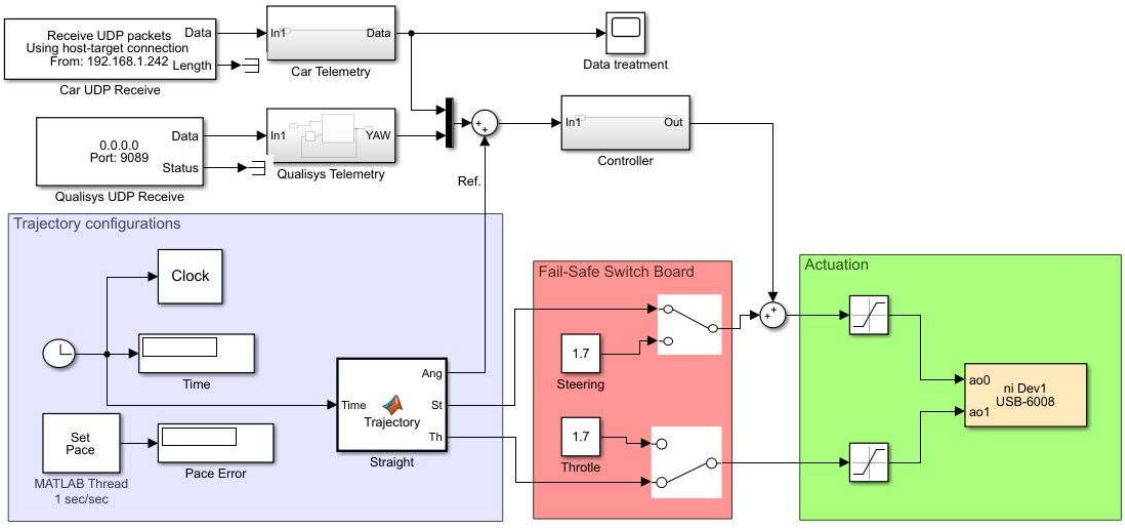


Figure 31 - Simulink project

7. **Tests and Validation**

Along this chapter all the hardware implemented was tested and validated. Some upgrades were made to improve the resolution of the existing encoders. The wire connections of the IMU were improved to eliminate a consistent crash of the hardware leading to incoherent data. These improvements were documented in section 5.1.

7.1 Hardware validation

7.1.1 IMU validation

The code implemented on the Arduino DUE allows the IMU to reset its values to zero once the car is switched on. To test the sensor, one platform of test was developed so that the car could rotate and stop all times in each defined angle in the most accurate and possible way. For the following experiment the sequence of angles defined was 0º→ 30º→ 60º→ 90º→ 60º→ 30º→ 0º and the schematic of the platform used is shown in Figure 32.

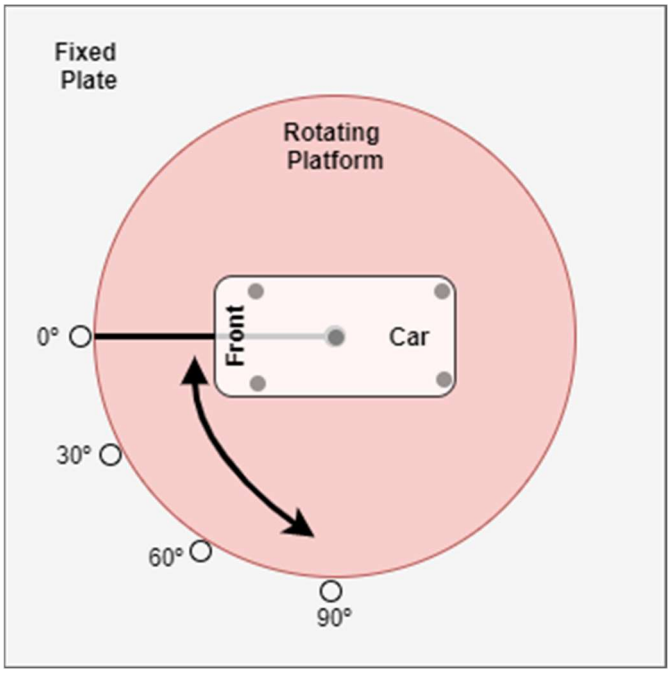


Figure 32 - Angle platform

The results acquired allow to understand the yaw variations during the sequence mentioned before. For a better evaluation a graph was created, and it is shown in Figure 33.

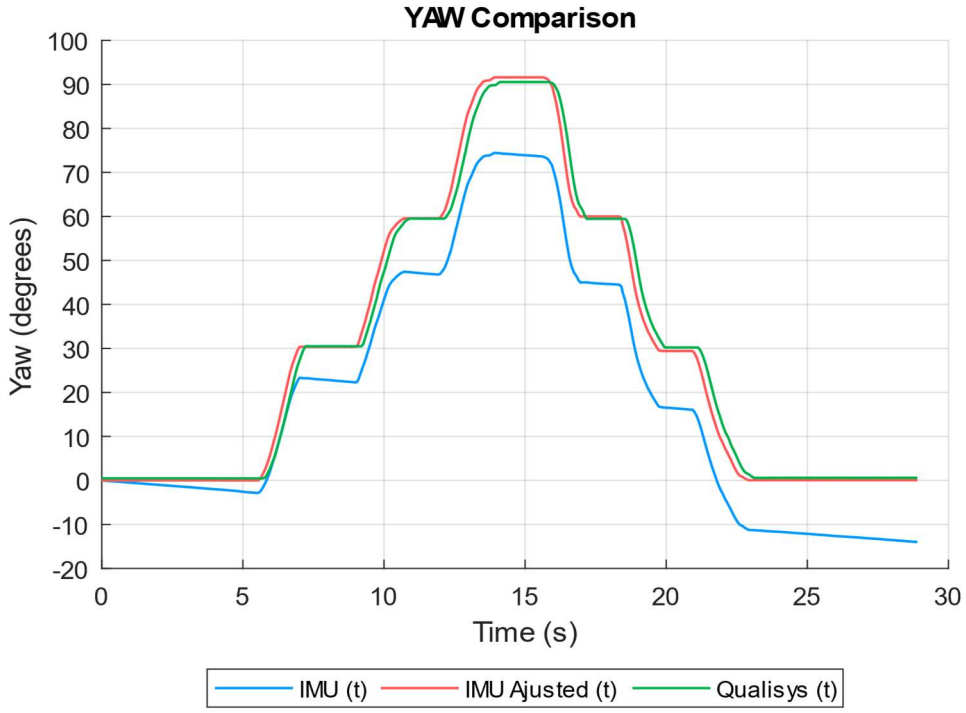


Figure 33 - Yaw comparison

From the graph interpretation, with the car stopped and in a rest state, the values of the IMU had oscillations making the reference values diverge from zero (blue plot), the obtained values are presented on Table 4 - IMU.

Table 4 - IMU

	IMU							
Angle	Test 1	Test 2	Test 3	Test 4	Test 5	Mean	Diff.	Error
0º	-3,40	-1,26	-1,28	-1,41	-1,16	-1,70	1,70	0,47
30º	19,00	22,65	24,74	22,82	22,57	22,36	7,64	25,48
60º	41,95	51,54	48,37	47,09	46,80	47,15	12,85	21,42
90º	66,74	73,38	74,95	74,02	72,53	72,32	17,68	19,64
60º	36,68	43,65	46,30	44,74	43,71	43,02	16,98	28,31
30º	7,08	15,63	19,43	16,02	16,15	14,86	15,14	50,46
0º	-22,87	-13,91	-10,83	-12,65	-13,60	-14,77	14,77	4,10

A filter composed by a saturation and a gain was successfully implemented to clean the signal and acquire more reasonable data. The filtered

telemetry (red plot) is clearly more accurate and it is also presented on Table 5 - IMU Adjusted.

Table 5 - IMU Adjusted

	IMU Adjusted							
Angle	Test 1	Test 2	Test 3	Test 4	Test 5	Mean	Diff.	Error
0º	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
30º	31,02	29,74	31,92	30,37	29,46	30,50	-0,50	1,67
60º	59,62	59,44	60,12	59,55	58,78	59,50	0,50	0,83
90º	90,26	90,19	91,95	91,61	89,89	90,78	-0,78	0,87
60º	59,05	58,66	61,42	60,00	59,56	59,74	0,26	0,44
30º	28,82	28,95	32,87	28,96	30,36	29,99	0,01	0,03
0º	-0,73	-1,32	2,24	0,05	0,74	0,20	-0,20	0,05

Besides the platform of rotation, the Qualisys system was used to confirm and validate the data received from the IMU (green plot). This data is presented on Table 6 - IMU Qualisys.

Table 6 - IMU Qualisys

	IMU Qualisys							
Angle	Test 1	Test 2	Test 3	Test 4	Test 5	Mean	Diff.	Error
0º	-0,49	0,18	0,05	0,46	0,59	0,16	-0,16	0,04
30º	30,22	29,88	30,73	29,45	30,72	30,20	-0,20	0,67
60º	59,61	59,43	59,21	59,37	59,62	59,45	0,55	0,92
90º	90,00	90,43	89,44	90,32	89,05	89,85	0,15	0,17
60º	60,14	60,28	59,55	59,72	60,02	59,94	0,06	0,10
30º	29,76	30,73	31,02	30,23	30,30	30,41	-0,41	1,36
0º	0,20	0,05	0,47	0,58	1,35	0,53	-0,53	0,15

Once only the yaw data is needed to the study, the next tables show the results of the tests only for the yaw angle.

Complementary information showing that the IMU could have accurate results with the applied filter are presented on Table 7 - IMU Comparison.

With the acquired values read from the telemetry in each test, it was possible to evaluate data, first the mean values are given by:

$$mean\ value = \frac{Test\ 1 + Test\ 2 + Test\ 3 + Test\ 4 + Test\ 5}{5} \tag{50}$$

where *Test 1, Test 2, Test 3, Test 4, Test 5* are the angle values obtained for each different test.

The relative error is given by:

$$relative\ error = \frac{desired\ angle - measured\ angle}{desired\ Angle} \quad (51)$$

where *desired angle* corresponds to the angle pre-established for each test and the *measured angle* is the value given by the IMU and Qualisys.

Table 7 - IMU Comparison

	IMU			IMU Adjusted			IMU Qualisys		
Angle	Mean	Diff.	Error	Mean	Diff.	Error	Mean	Diff.	Error
0º	1,70	1,70	0,47	0,00	0,00	0,00	0,16	-0,16	0,04
30º	-22,36	7,64	25,48	30,50	-0,50	1,67	30,20	-0,20	0,67
60º	-47,15	12,85	21,42	59,50	0,50	0,83	59,45	0,55	0,92
90º	-72,32	17,68	19,64	90,78	-0,78	0,87	89,85	0,15	0,17
60º	-43,02	16,98	28,31	59,74	0,26	0,44	59,94	0,06	0,10
30º	-14,86	15,14	50,46	29,99	0,01	0,03	30,41	-0,41	1,36
0º	14,77	14,77	4,10	0,20	-0,20	0,05	0,53	-0,53	0,15

In spite of not being useful for control purposes the roll and pitch angles were tested to verify if they were consistent with the data already obtained for yaw. Similar results to the yaw were obtained proving that the IMU is producing accurate data.

The data comparison for roll and pitch is shown in Figure 34.

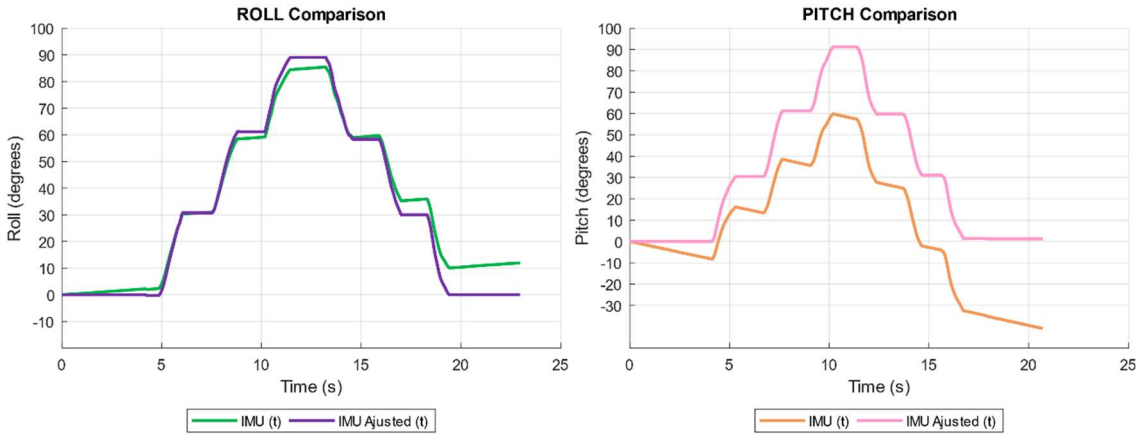


Figure 34 - Roll and Pitch comparison

Interpretation of the previous figure allow to conclude that without the filter, the roll (green plot) is acceptable but not perfect, but the pitch (orange plot) is way too far from acceptable. The filter is well designed, and results could be seen in the purple and pink plots that shown the roll and pitch respectively.

7.1.2 Encoders validation

The original encoders assembled in the car were chose originally due to the easier implementation factor. In an early stage were used hall effect sensors that directly read the magnets from the rotors. This choice proved to be an insufficient one because of their resolution. Taking into account that the velocity values are important to the controller, the encoders' resolution of only 7 pulses per turn of the wheel showed to be not sufficient. Some improvements were made to increment this resolution, using Arduino function to caption change of value instead rising, allowing to have 14 pulses per revolution. With this environment and knowing that one revolution of the wheel corresponds to 200mm, means that per each pulse the car moves approximately 14.3 mm forward.

A new pair of encoders improved the resolution to 400 pulses per revolution. This means that between each pulse the car moves approximately 0.5 mm forward, allowing to have more detail in the car velocity which is

extremely important in the context of this dissertation. It is important to mention that the wheel to encoder ratio is one to one.

The accuracy of the distance is good enough with both encoders and that is shown in Figure 35 and Figure 36.

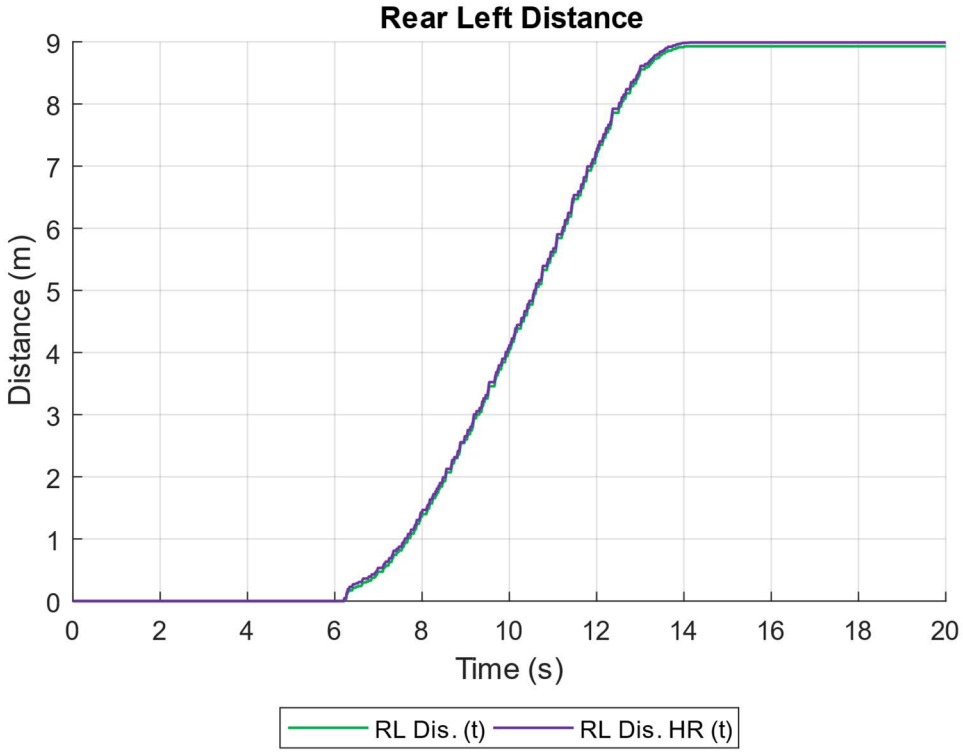


Figure 35 - Rear left distance comparison

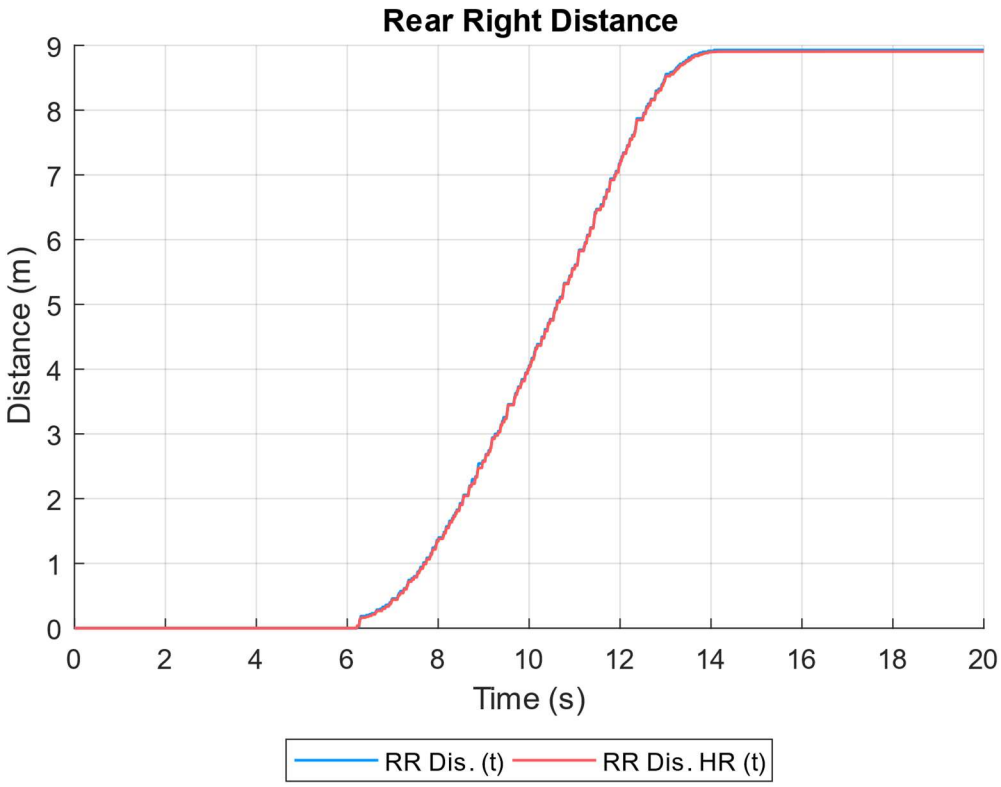


Figure 36 - Rear right distance comparison

The plots in green and blue represent the distance measured by the low-resolution encoders. In purple and red it is possible to see the distance measured by the encoders with high resolution.

The low resolution causes a big impact on speed values because of the lack of readings, as shown in Figure 37.

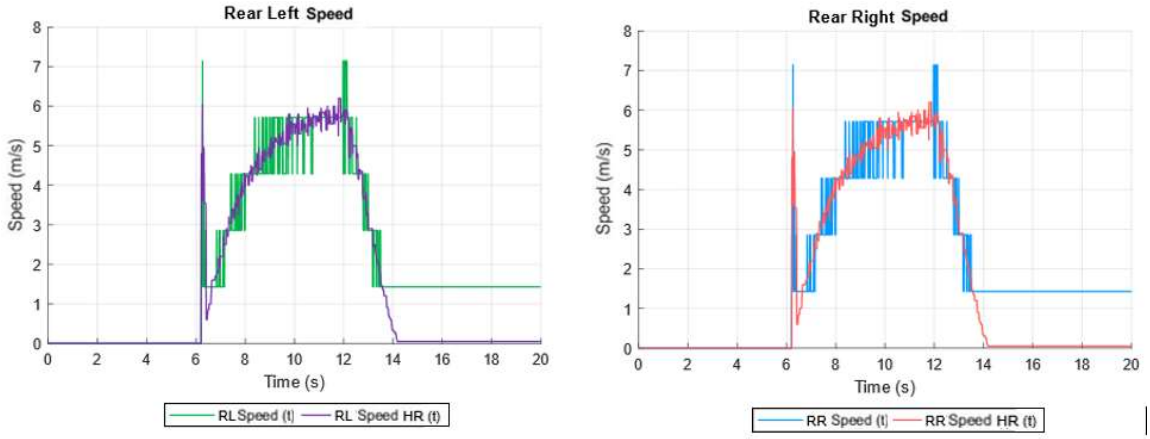


Figure 37 - Speed comparison

A visible peak described by the two encoders was caused by the car start, mechanical clearances, and the choice of motors with low torque. Besides

that, the accuracy of the speed was improved when the new encoders were used, it is possible to see that by evaluating the purple and red plots.

It was possible to conclude that the speed graph had a strange shape, to reshape the speed graph a lowpass filter was implemented. The MATLAB function *lowpass(x,wpass)* allowed to obtain good results shown in Figure 38

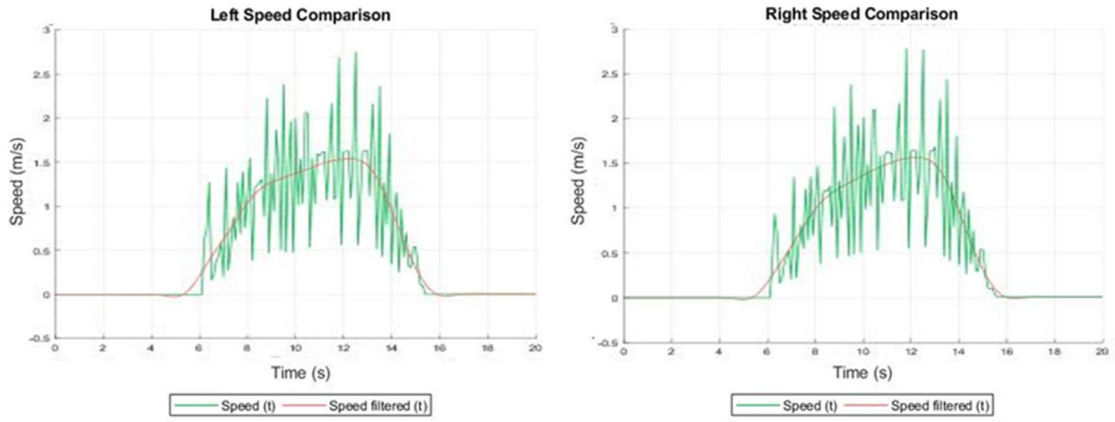


Figure 38 - Speed low-pas filter

To validate the encoders, *Qualisys* system was used again. After a new calibration of the system good results were obtained, showing that the speed in both systems were identically the same. The results for straight line are shown in Figure 39.

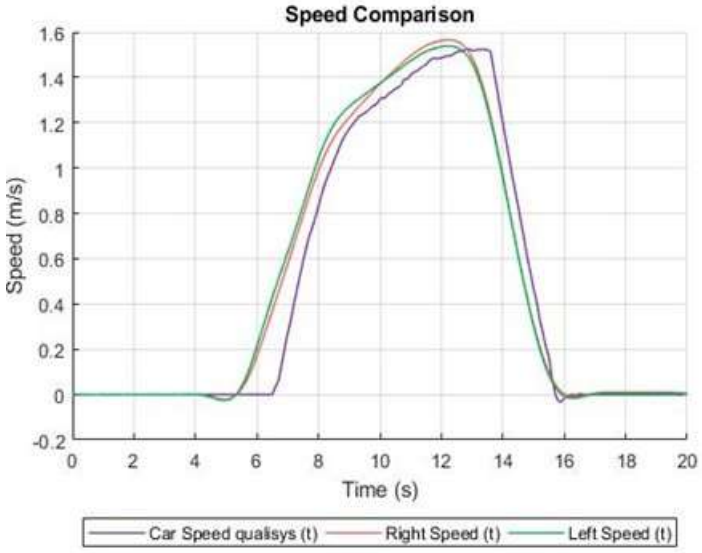


Figure 39 - Speed comparison

To be more visible, two J-shaped curves were performed to evaluate the difference between the speed of each wheel. The first curve consists in a straight line ending in a left turn. The other ended in a right turn. The results are shown in Figure 40.

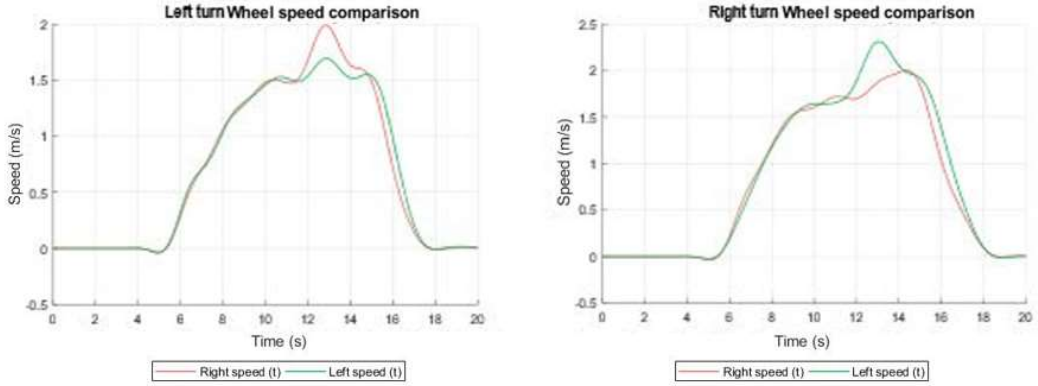


Figure 40 - Left and right turn wheel speed comparison

7.2 Controller comparison

On this section, the results obtained for the control methods are shown. It was chosen a test were the car start in a $40°$ angle at it was expected to drive until follow the $0°$ angle as reference, moving in straight line until the end.

In preliminary tests, the controller values obtained before do not result as expected, causing crashes.

Two control methods were used and modified trying to achieve stability and a more accurate controller. The LQR and PID gains were changed until the stability was achieved. The new gains and simulation are shown in the next two sub-sections.

7.2.1 LQR implementation

The results obtained with this controller were influenced by the physic 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 was:

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

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 that is shown in Figure 41.

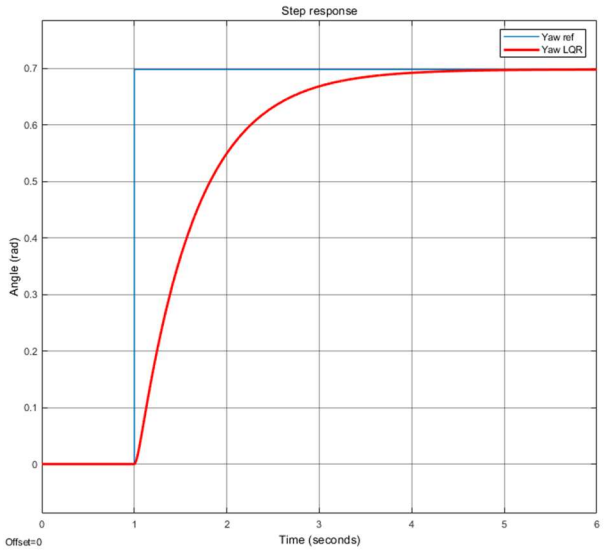


Figure 41 - LQR Step response

In comparison with the projected controller shown before, it is possible to see that this one is slower. In practice show low repeatability and don't achieve the desired direction. The trajectory made by the car (blue plot) is shown in Figure 42.

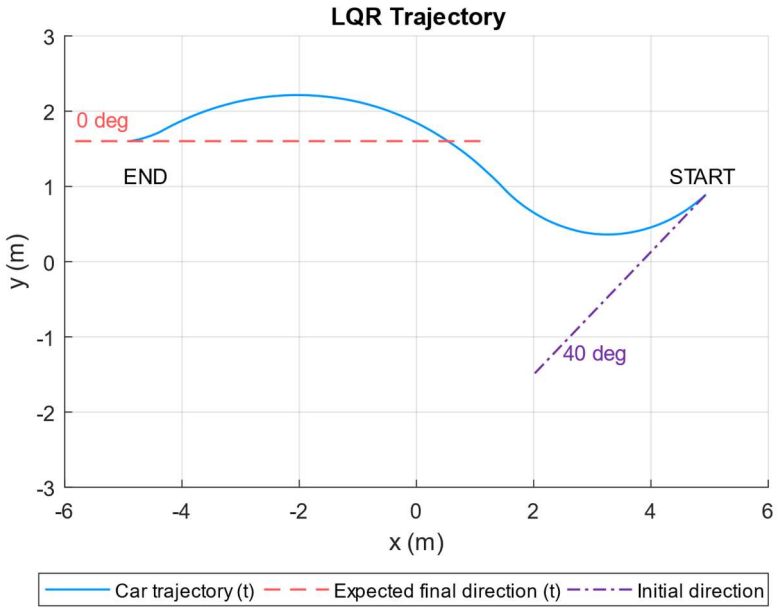


Figure 42 - LQR Trajectory

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

7.2.2 PID implementation

As well as the LQR, the PID had also changed configuration to improve control. For that new values were strategically changed until the results were obtained, and the values were:

$$P = 14$$

$$I = 0$$

$$D = 6$$

The achieved step response is shown in Figure 43.

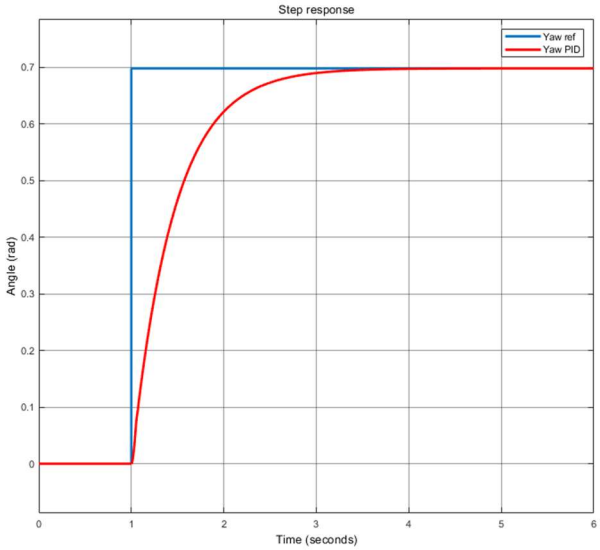


Figure 43 - PID Step response

Evaluating the response, it is possible to see that this response better from a theoretical perspective.

This last sentence could be confirmed considering the results obtained with the car using PID controller. The same test was repeated, and the results are shown in Figure 44.

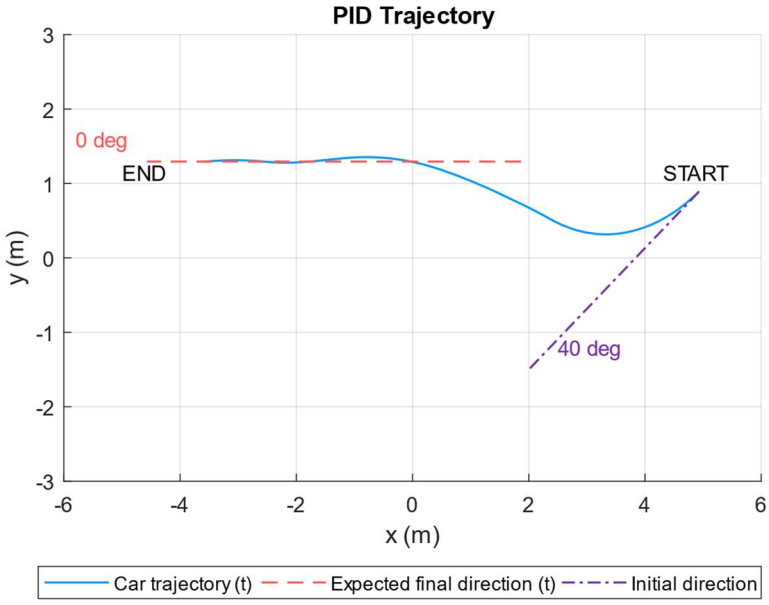


Figure 44 - PID Trajectory

Evaluating the above results, it is possible to understand that the desired direction was successfully obtained because the car was able to follow the same direction since halfway until the end.

7.3 Torque vectoring cases of study

To evaluate the developed code inside the Arduino Due, different cases were studied. The cases were:

Case 1: Steering and yaw moment aligned (no understeering -> apply normal tv)

Case 1.1: Turn right (steering & throttle position as inputs):

Calculate the slip of the outer to the inner wheel, equation (15) written as slip=rpm0/rpm1.

Case 1.1.1: Wheels are not slipping

- Increase torque on outer wheel (ch2 > 1490)
- Increase brake on inner wheel (ch2 < 1490)

Case 1.1.2: Outer Wheel is slipping

- Decrease torque on outer wheel (ch2 > 1490)
- Decrease brake on outer wheel (ch2 < 1490)

Case 1.1.3: Inner Wheel is slipping

- Decrease torque on inner wheel (ch2 > 1490)
- Decrease brake on inner wheel (ch2 < 1490)

Case 1.2: Turn left (steering & throttle position as inputs):

Calculate the slip of the outer to the inner wheel equation (15) written as slip=rpm1/rpm0.

Case 1.2.1: Wheels are not slipping

- Increase torque on outer wheel (ch2 > 1490)
- Increase brake on inner wheel (ch2 < 1490)

Case 1.2.2: Outer Wheel is slipping

- Decrease torque on outer wheel (ch2 > 1490)
- Decrease brake on outer wheel (ch2 < 1490)

Case 1.2.3: Inner Wheel is slipping

- Decrease torque on inner wheel (ch2 > 1490)
- Decrease brake on inner wheel (ch2 < 1490)

Case 2: Steering and yaw moment are not aligned (understeering)

Case 2.1: Turn right (steering & throttle position as inputs):

Do not turn inner wheel

Case 2.2: Turn left (steering & throttle position as inputs):

Do not turn inner wheel

Case 3: No steering or no acceleration and braking, only use hackerman geometry configuration:

Case 3.1: Turn right (steering & throttle position as inputs):

Turn wheels at input speed

Case 3.2: Turn left (steering & throttle position as inputs):

Turn wheels at input speed

7.4 Model validation

To validate the torque vectoring model the code was translated in java to get responses of the code on the console. The model was tested on:

- Symmetry for the right and left wheel
- Influence of the tv-multiplier

- Effect of the steering and throttle input on the tv

Case 1.1.1 & 1.2.1: Wheels are not slipping

Vertical axis: Throttle input increase of the outer wheel in comparison to the inner wheel that remains constant.

Horizontal axis: Steering angle for turning right (0 = 1640) and for turning left (0 = 1340) to the maximum steering angle

The torque applied to the outer wheel depends on the steering angle, the throttle input, and the tv-multiplier. This data are shown in Figure 45.

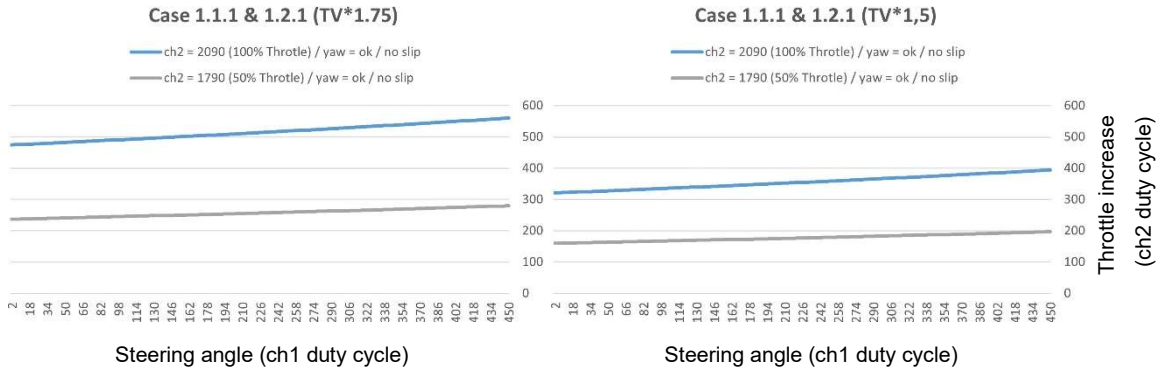


Figure 45 - TV applied to the outer wheel (no slip)

Case 1.1.2 & 1.2.2: Outer Wheel is slipping

Vertical axis: Throttle input increase of the outer wheel in comparison to the inner wheel

Horizontal axis: Steering angle for turning right (0 = 1640) and for turning left (0 = 1340) to the maximum steering angle

If the outer wheel is slipping in the tv operating mode, the torque on the outer wheel is decreased. It is still bigger than the throttle input (effect of traction control can be adapted). This data are shown in Figure 46.

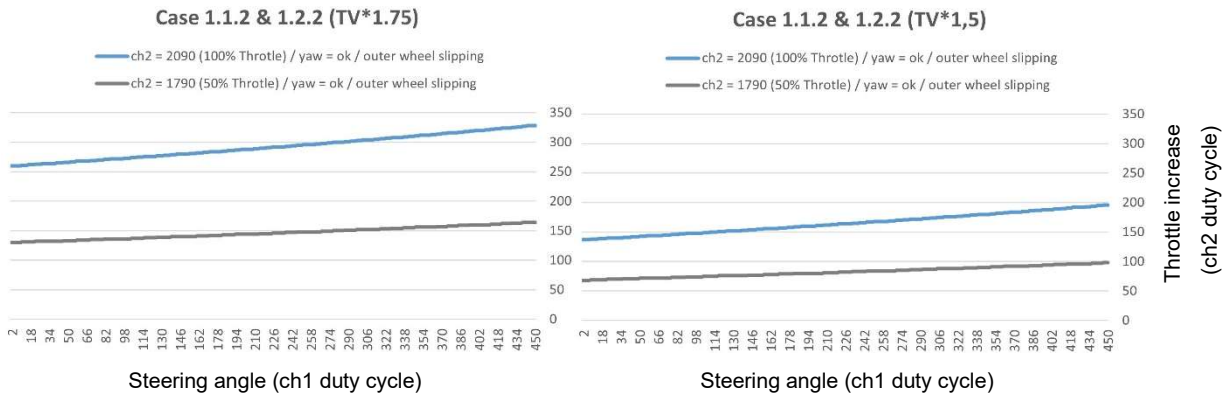


Figure 46 - Traction control applied to the outer wheel

Case 1.1.3 & 1.2.3: Inner wheel is slipping

Vertical axis: Throttle input decrease of the inner wheel in comparison to the outer wheel

Horizontal axis: Steering angle for turning right (0 = 1640) and for turning left (0 = 1340) to the maximum steering angle

If the inner wheel is slipping in the tv operating mode, the torque on the inner wheel is decreased. It is smaller than the throttle input (effect of traction control can be adapted).

Inverting the torque vectoring model for braking the inner wheel increases the braking input (Electronic Stability Control - Esc).

Inverting the traction control model, it can be used as anti-lock braking system if there is slip on inner wheel, this data are shown in Figure 47.

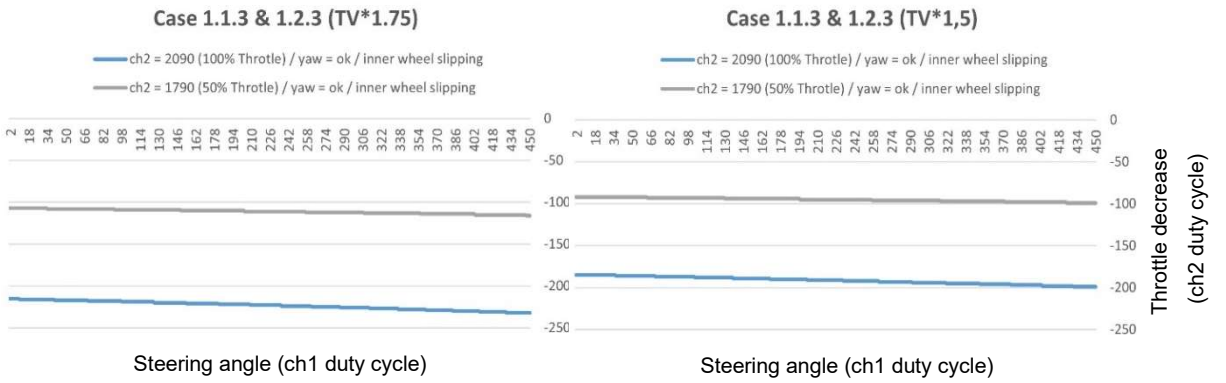
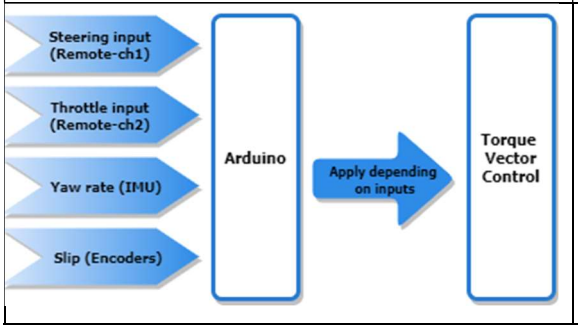
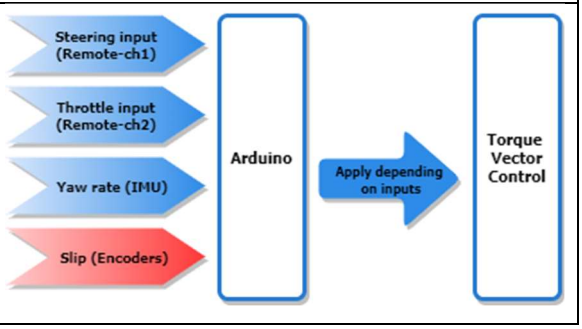


Figure 47 - Traction control applied to the inner wheel

The theoretical model could not fully be implemented on the car (the slip of the wheels cannot be obtained). The diagram is presented on Table 8.

Table 8 – Theoretical model vs implemented model

Theoretical model	Implementation in the car
	

Nevertheless, a torque vectoring model, which uses the steering input and the throttle input of the remote as well as the yaw rate of the IMU, was implemented in the car.

Driving behavior of the car:

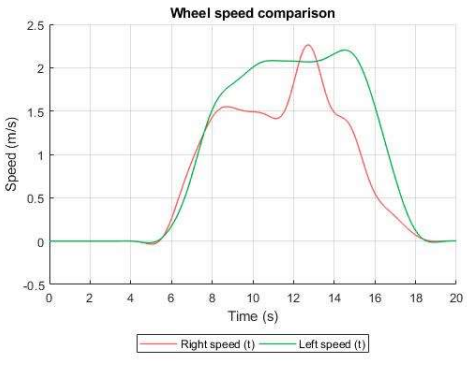
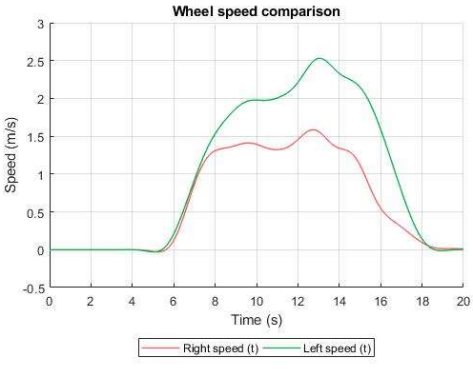
- The driving behavior of the car in torque vectoring mode could be optimized by following aspects:
  - Smaller curve radius
  - Faster curve speed
  - Acceleration:
    - Torque vectoring (increases torque of the outer wheel)
    - Traction control (decreases torque independently for both wheels if there is slip) – theoretical implementation
  - Braking:
    - Electronic stability control (increases the braking of the inner wheel to help the car to turn)
    - Anti-lock braking system (decreases the braking independently for both wheels if there is slip) – theoretical implementation

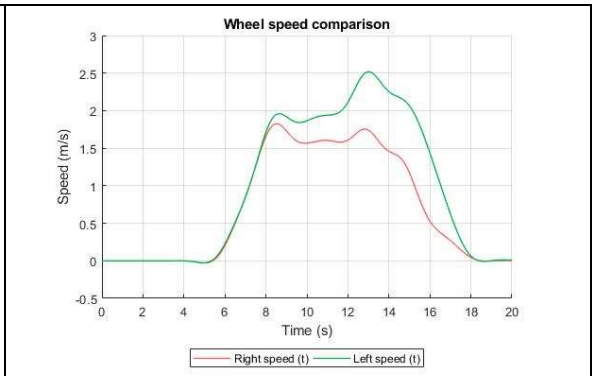
7.5 Tests results

The first step in this part was to check the telemetry measured in the car. To check if it was accurate the time and the delay was considered. Knowing that the trajectory was time based, and the throttle was activated when the time correspond to exactly five seconds, it was possible to compare the commands sent with the telemetry received and conclude that the maximum delay was around one second. Evaluating the overall system, the delay appears to be mostly inside of the MATLAB running program (MATLAB -> DAC) instead of the wireless communication system.

For better understanding of the telemetry, the data received during the tests are shown in tables 9, 10, 11 and 12 explaining each relevant part.

Table 9 - Wheel speed comparison

Wheel Speed	LEFT TURN	RIGHT TURN
NO TV		
TV * 1,5		

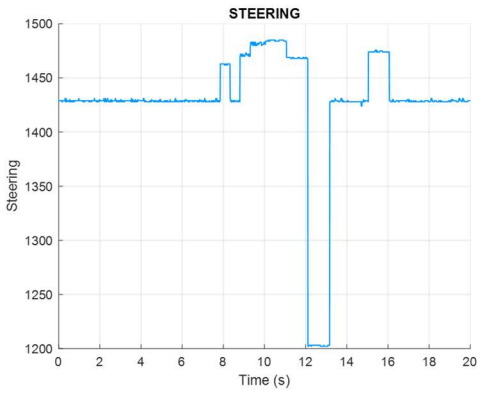
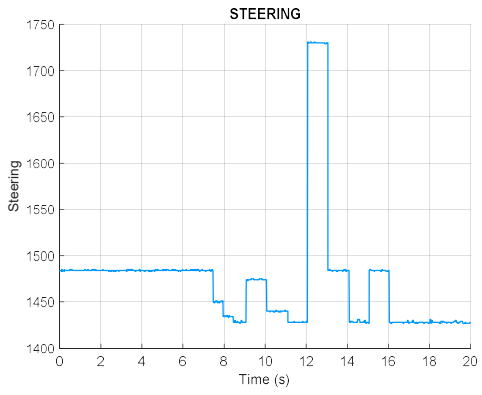
TV * 1,75		

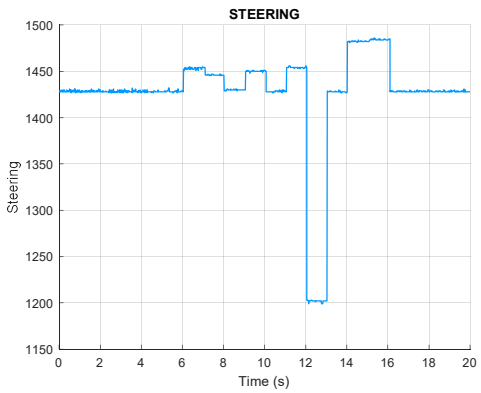
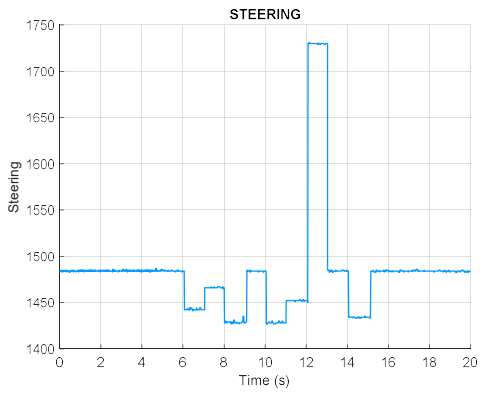
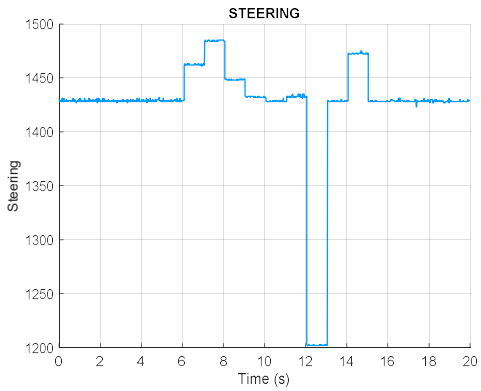
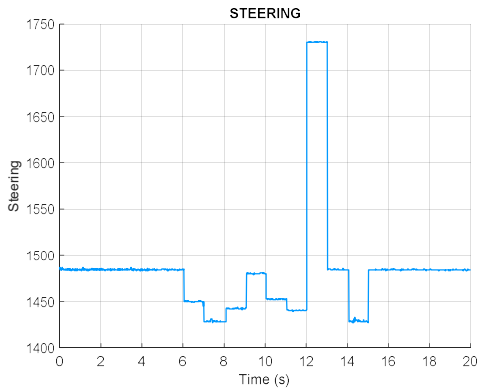
The **left** turn of the car starts at 12s, when the **speed of the right wheel** of the car **increases.**

The **right** turn of the car starts at 12s, when the **speed of the left wheel** of the car **increases.**

The wheel speed obtained shows that in torque vectoring mode the speed of the outer wheel increases more compared to the speed of the inner wheel.

Table 10 - Steering input comparison

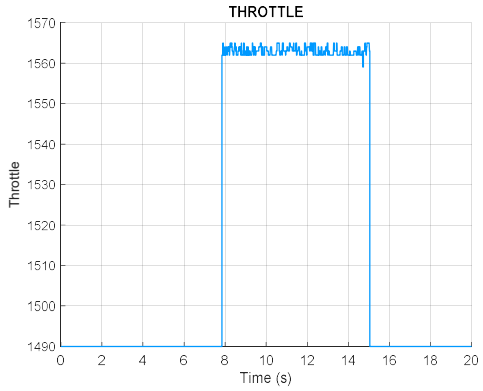
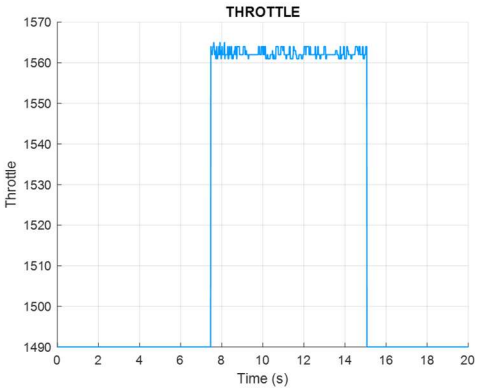
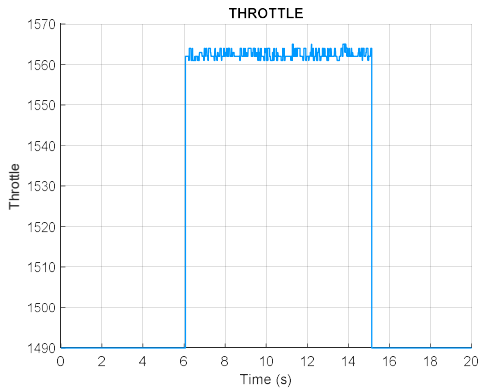
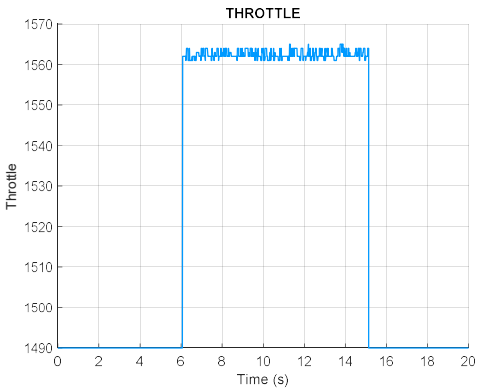
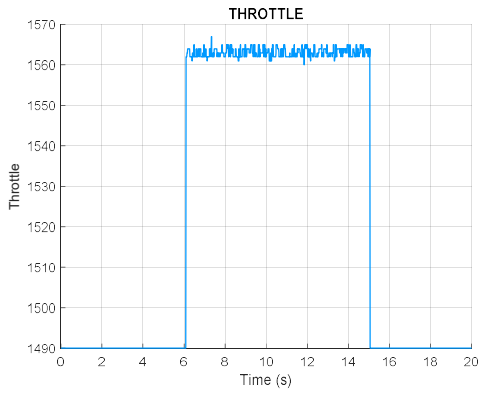
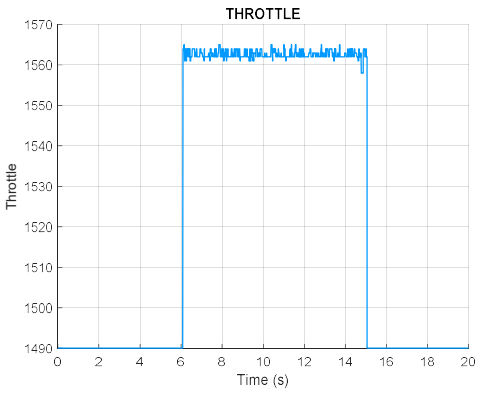
STEERING INPUT	LEFT TURN	RIGHT TURN
NO TV		

TV * 1,5		
TV * 1,75		

The steering of the turn starts at 12s. The steering prior and after to that was performed by PID controller using Qualisys data to drive the car in a straight line.

Although the initial conditions were different for each curve, it was possible to verify that in the three tests carried out on each side the behavior was similar.

Table 11 - Throtle input comparison

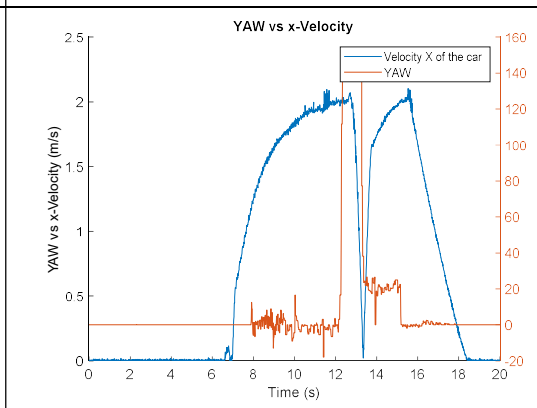
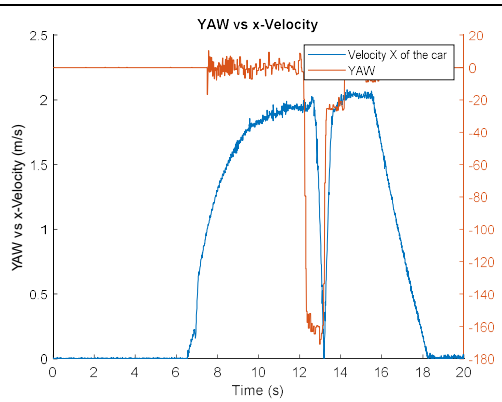
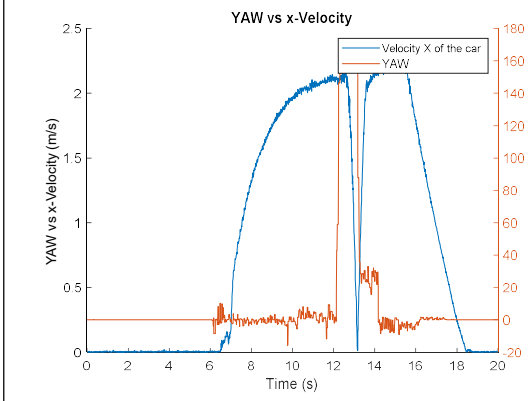
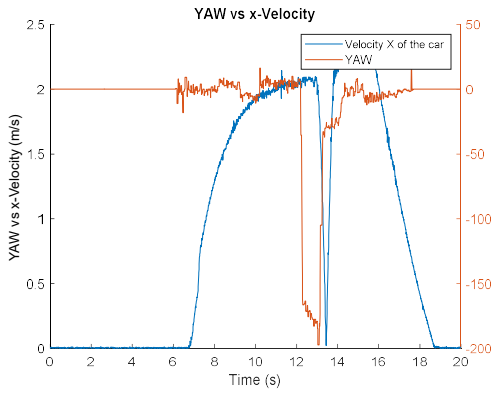
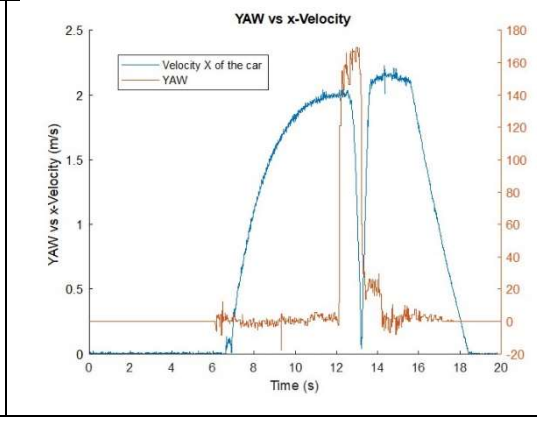
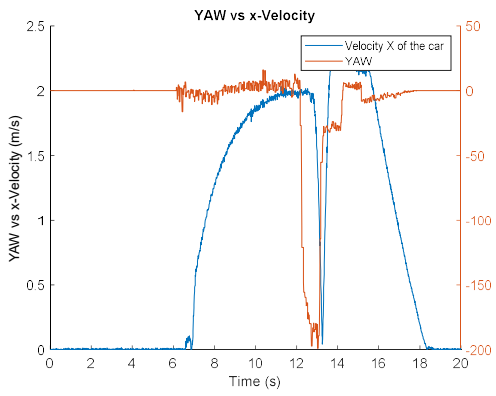
THROTTLE INPUT	LEFT TURN	RIGHT TURN
NO TV		
TV * 1,5		
TV * 1,75		

The steering of the turn starts at 12s and the throttle send to the car was constant as could be seen in the blue plot, leaving the torque management for the controller programed inside the Arduino Due.

Similarly to what happened with the steering, it was possible to verify that the behavior of the throttle send it for the car was exactly the same in each test,

leaving the management of the power for the controller programed inside the Arduino Due.

Table 12 - Yaw and X-speed comparison

YAW vs X-Speed	LEFT TURN	RIGHT TURN
NO TV		
TV * 1,5		
TV * 1,75		

The steering of the turn starts at 12s.

The yaw rate was used to verify whether the car is turning (detect understeering behavior). The dead zone for the yaw moment (<1 for left turns & > -1 for right turns) was defined as sufficient.

The x-speed of the car increases with the torque vectoring mode.

Now that the controller has been chosen and the telemetry has been shown, 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 a straight line and then perform a right or left turn. 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 presented in Figure 48.

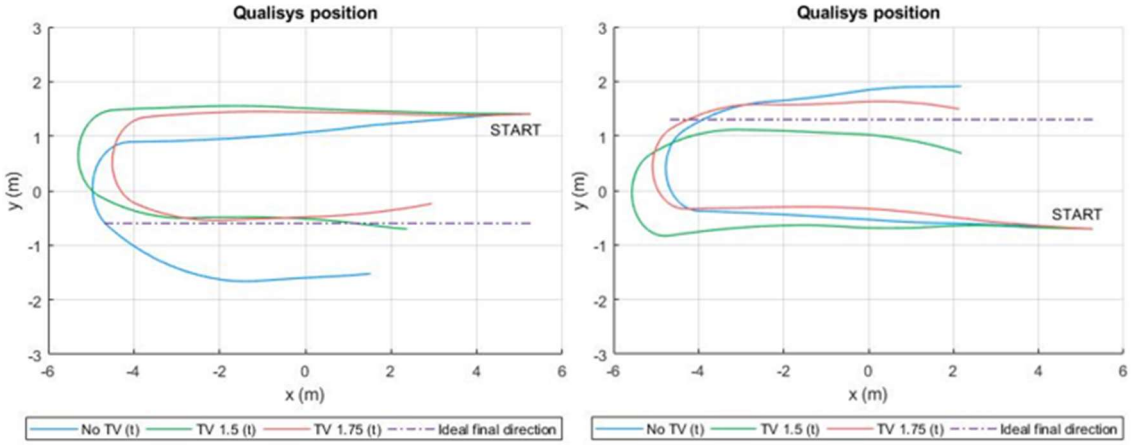


Figure 48 - Left and right turn comparison

In the environment of the laboratorial tests, sending a signal of 0.8V for the steering corresponding to $\delta = 14.29° = 0.249\, rad$ and using equation (41), it was expected to have a left turn with the expected radius $R \approx 1m$. Evaluating Figure 48 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, improving the previous results reaching the expected curve radius and reducing the understeer of the car.

In a limited space like 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 tends 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 part.

Despite all the extra effort associated with communications and hardware, good results were successfully achieved with these controllers.

8. **Conclusions**

Between all the setbacks during this time, the purpose of this work was successfully reached.

This work provides 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 were 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 realise different tests describing the expected trajectory.

The weakest part of all the assembly were probably the ESC's because between all the setbacks, the fact that this part doesn't allow the braking of the wheel left us in a situation that only the rise of power is possible.

The Qualisys system was a very important tool to this dissertation but was noticed that some unexpected delays occur from time to time. In the beginning the calibration was very poor letting the car without reference with the lack of data. A new calibration of the arena proves to be enough to never lose sight of the car inside the arena.

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 proves to be reliable as well avoiding the use of normal batteries. It would be possible to assemble a battery charger inside the remote but once that doesn't affect the results achieved because in the lab the majority of the work is done with the remote connected to a computer and for that no batteries are needed.

In the end the reliability of the state machine proves to be good because the transition from one state to other occur immediately without delay.

One of the setbacks mentioned was the upgrade of the encoders, but the proto-board developed proved to be a good choice because the Pull-up resistor

was easily implemented. On the other hand, the most intricate part were the encoders' mounts' construction and the resizing of the belts.

In general, the objectives were achieved with success despite all the setbacks that are normal in a dissertation like this.

8.1 Future work

To complement the work started here, it would be to upgrade even more the Arduino code to have a better controller to Torque vectoring and maybe using GPS system when it is outdoors.

The upgrade of the ESC's is essential because the results are better as well as this should increase the expected lifetime of the motors.

To test if the Raspberry is a good solution or not, more time need to be invested to understand why so long delays were visualised. One possible solution would be set aside the state machine configuration and use only the Raspberry Pi, but for that other solutions are available already in laboratory and could also be evaluated.

9. **Bibliography**

[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] L. Zhao, “Using UDP Datagram to Realize a Distributed Control Mode at High-Speed Data Communication,” *Phys. Procedia*, vol. 25, pp. 886–891, 2012, doi: https://doi.org/10.1016/j.phpro.2012.03.173.

[4] P. Oliveira, M. Rego, P. Machado, and L. Duarte, “Torque vectoring control,” no. November, 2017.

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

[6] J. Pedro and M. Antunes, “Torque Vectoring for a Formula Student Prototype,” no. June, 2017.

[7] G. Kaiser, Q. Liu, C. Hoffmann, M. Korte, and H. Werner, “Torque Vectoring for an Electric Vehicle Using an LPV Drive Controller and a Torque and Slip Limiter.”

[8] J. Guerra, P. Machado, and L. Duarte, “Trajectory Tracking and Obstacle Bypass with a RC Car,” 2017.

[9] D. Silva and M. Cunha, “OPTIMAL CONTROL RC Car on a Circuit Mechanical Engineering Professor : Paulo Oliveira.”

[10] Mathworks, “Linear-Quadratic Regulator (LQR) design.https://www. mathworks.com/help/control/ref/lqr.html.” .

[11] Muggn, “GY80 arduino libray. url: https://github.com/muggn/GY80.” .

[12] Arduino, “Wire library. url: https://www.arduino.cc/en/Reference/Wire.” .

[13] Espressif, “ESP8266 AT Instruction Set,” p. 70, 2016.

[14] Espressif, “https://www.espressif.com/en/support/download/other-tools.” .

[15] Farrell, “http://farrellf.com/arduino/esp8266_udp_transmitter.ino.” .

[16] 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.