POLITECNICO DI TORINO

Department of Electronics and Telecommunications
Master of Science in Mechatronic Engineering

MASTER'S THESIS

Estimation and Sensor Fusion techniques on an autonomous vehicle

Supervisor:
Prof. Massimo Violante

Candidate:
Antonella Tufiño Di Costanzo
255252

Tutor:
Ing. Giovanni Guida
Brain Technologies S.r.l.

April 2021

A mi mamá, Maria,
que aunque no entienda de que le hablo,
le da gusto oírme hablar de mi trabajo
y me apoya en todas mis decisiones.

A mio padre, Salvatore,
grazie a lui non sarei qui dove sono.

A Guapo, il mio cane,
grazie per farmi prendere le pause quando ne avevo bisogno.

Al Ing. Giovanni Guida,
grazie per la disponibilità e la pazienza.

Al professor Violante,
grazie per i vari insegnamenti nei corsi di laurea magistrale
e per accettare di farmi da relatore tesi.

ABSTRACT

A model is a simplified or partial representation of reality, defined in order to accomplish a task or to reach an agreement. It is based on an original system reflecting the relevant selection of the original system properties. The model is the central artifact of software development, it includes the documentation, static analysis, rapid prototyping, automated testing, refactoring, transformation and code generation.
The increasing complexity of software in engineering applications leads to problems in the software development like production delay, wrong functionalities, software poorly documented or commented.
Modelling allows to increase productivity, efficiency, reusability, improve portability of the software and is easy to test.
This are the main reason why nowadays the model-based method is being used in the complex engineering projects.
In this thesis it will be discussed specifically the model of a skid steering mobile robot, courtesy of Brain Technologies, with ultrasonic sensors in order to define its position with respect to another object and follow the tracked object in an S shape path by means of Matlab and Simulink.
In the end, after testing the single components and the whole model of the robot, the model will be code generated via Embedded coder and deployed in the Arduino board.
The code generated by Simulink used a huge amount of memory and it was needed to reduce and remove some redundant code generated.

CHAPTER 1
Introduction

This thesis refers to the development of autonomous guidance features of a mobile robot, in collaboration with Brain Technologies, a company of engineering and scientific services for industrial projects, head-quartered in Turin, Italy.
The thesis proposal is inside the development program of BAT-MAN projects. In particular, the BAT-MOB project scope is to apply estimation and sensor fusion techniques, developed during previous BAT-MAN projects, to an autonomous vehicle (e.g., odometric problem).
This thesis was carried mostly on smart working with an Arduino board.

Problem overview

"Autonomous systems rely on sensor suites that provide data about the surrounding environment to feed the perception system. These sensors include radars and cameras, which provide detections of objects in their field of view." [1]

In this thesis we will focus on the integration of ultrasonic sensors in order to calculate the pose of a vehicle in front of ours and follow it keeping a desirable safety distance by means of a technic similar to the adaptive cruise controls of the automotive vehicles. In case there is no forward vehicle, my vehicle needs to keep moving forward in an S shaped path.
For the path planning, among all the techniques, it was used the cubic polynomial path using as parameters the desired linear speed of the x and y coordinates, lateral and longitudinal speeds, and limiting the displacement within a certain limit distance.

Objective of the thesis

The scope of the thesis work is to develop some autonomous guidance for a mobile robot in order to make it move forward in case of no vehicle is detected and if detected, then to follow that forward vehicle.
An initial part of the thesis is devoted to the study of kinematic and dynamic equations for the mobile robot, a type of skid steer robot. Then its implementation as a Simulink model using the model-based method.
A part will be devoted to the integration of the ultrasonic sensors which will be used in order to retrieve the pose of the forward vehicle, called R1. Then make it move in an S path either during the follow up of R1 and when moving forward.
In the last part of the thesis, the code generation and a test of it.

Thesis outline

The thesis is developed in 8 chapters:

- Chapter 1: A brief introduction of the thesis and its research topics are presented.
- Chapter 2: A brief description of what does robotics means, in particular an overview of mobile robots and the considerations done for the particular mobile robot used for the thesis, the 4WD Hercules mobile robot.
- Chapter 3: Mathematical description of the Kinematic and Inverse Kinematic models of the mobile robot, with their Simulink test.
- Chapter 4: Mathematical description of the Dynamic model of the mobile robot, with its Simulink test.
- Chapter 5: Considerations of mobile robot path planning and choice of the method in

order to have an S shaped path considering the linear speeds of the vehicle.

- Chapter 6: Integration of Ultrasonic Sensors in the Simulink Model
- Chapter 7: Code Generation and testing
- Chapter 8: Conclusions

CHAPTER 2
Mobile Robots

2.1 Robotics introduction

"Robotics is the science studying the intelligent connection between perception and action. Therefore, a robotic system is a complex system, functionally represented by multiple subsystems." [2]

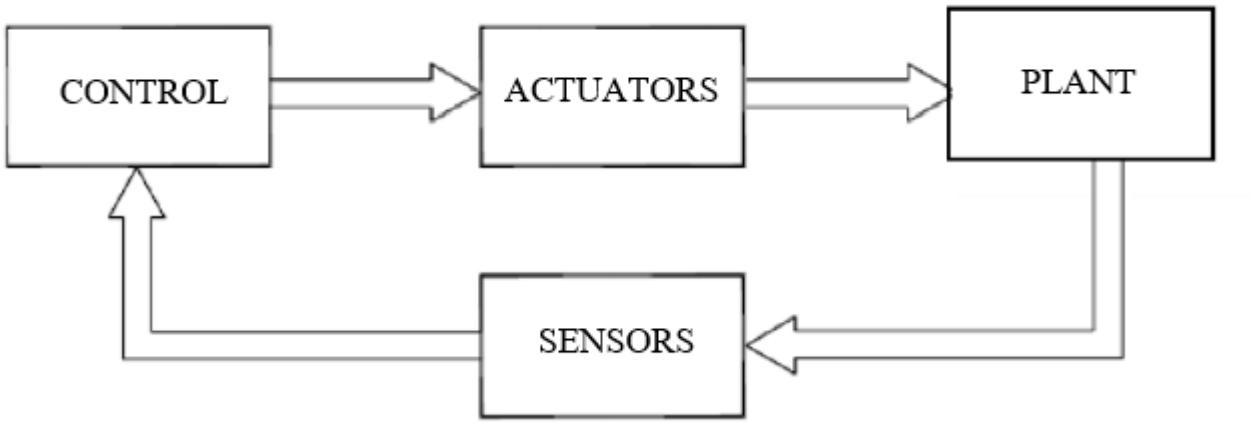


Figure 2.1 Components of a robotic system

The type of robot that will be discussed in this thesis is a mobile robot.

The essential components of a mobile robot system are:

- Control, which is done in this thesis via a Simulink model of the dynamic and kinematic equations of the robot;
- Actuators, which are the encoder motors;
- Plant, the wheels;
- Sensors, which are the ultrasonic sensors.

2.2 4WD Hercules Mobile Robot

The 4WD Hercules Mobile Robotic Platform, is a discontinued bot from the Seed Studio company, it is designed such that one can enter into the robotic world making their own robot mobile platform. It is also Arduino compatible, therefore there could be encountered many open-source applications to test it.

Figure 2.2 Hercules 4WD

2.3 Skid Steer Drive

In order to choose the best vehicle equations, a small debate of which type of drive our vehicle might be considering the number of wheels and that it does not have the steering angle, the one between the longitudinal axis and the steered wheel since the wheel is constrained by the robot chassis.

Differential Drive or Skid Steer Drive?

"Differential drive vehicles, are two-wheeled drive systems with independent actuators for each wheel. The drive wheels are usually placed on each side of the robot and towards the front. Sometimes it has an extra wheel, a non-driven wheel which forms a tripod-like support structure for the body of the robot. Often, the non-driven wheel is a *caster wheel*, a small swivelled wheel used on office furniture. It has two motors, one for each drive wheel.

The advantages are that it is very simple, often the drive wheel is directly connected to the motor, usually a gear motor (a motor with internal gear reduction) because most motors do not have enough torque to drive a wheel directly.

The disadvantages are that it is difficult to move a robot in a straight line. The drive wheels are independent, if they are not turning at exactly the same rate the robot will veer to one side. Making the drive motors turn at the same rate is a challenge due to slight differences in the motors, friction differences in the drive trains, and friction differences in the wheel-ground interface. To ensure that the robot is traveling in a straight line, it may be necessary to adjust the motor RPM very often (many times per second). It may require interrupt-based software and assembly language programming. It is also very important to have accurate information on wheel position usually using odometry sensors." [3]

"Skid Steer locomotion, is used on tracked vehicles such as tanks and bulldozers, but is also used on some four- and six-wheeled vehicles. On these vehicles, the wheels on each side can be driven at various speeds in forward and reverse (all wheels on a side are driven at the same rate). There is no explicit steering mechanism, the name implies steering is accomplished by actuating each side at a different rate or in a different direction, causing the wheels or tracks to slip, or skid, on the ground. Vehicles that use skid-steer usually are off-road types such as construction equipment and tanks, the

reduced friction of a non-paved surface helps to reduce tire/track wear. These disadvantages are offset by the simplicity of the drive system because of the negative effect it has on odometry: wheels that are skidding are not tracking the exact movement of the robot. Skid-steer is not commonly used on robots with sparse sensing (with no video cameras or sonar) that require accurate position determination.
Skid-steer is closely related to the differential drive system, replacing the caster wheel with extra drive wheels. It has the same disadvantage: moving in a straight line requires the wheels on each side to be turning at the same speed, which can be difficult to achieve. The advantage of skid-steer is the increased traction and no "caster wheel effect".
It has two motors, one for each side of the robot." [4]

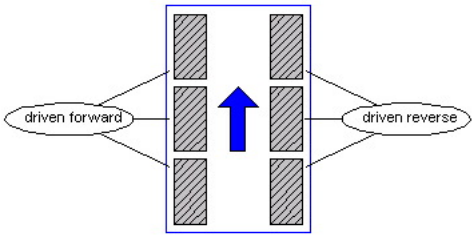
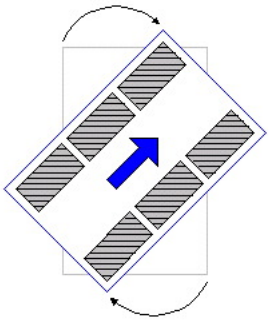


Figure 2.3 Skid-steer mobile robot



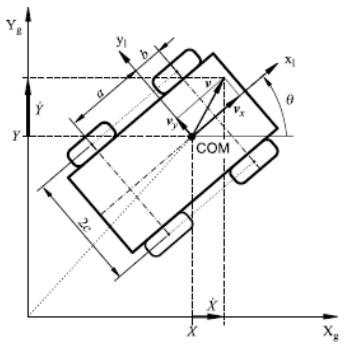


Figure 2.4 SSMR Free body diagram

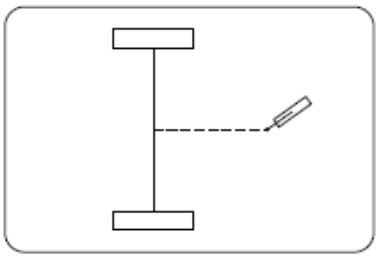


Figure 2.5 Tricycle mobile robot and differential-drive mobile robot

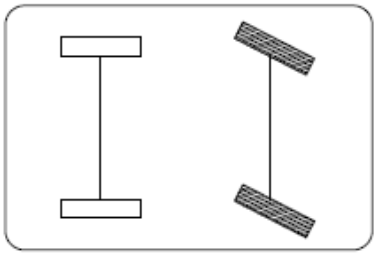


Figure 2.6 Car-like mobile robot

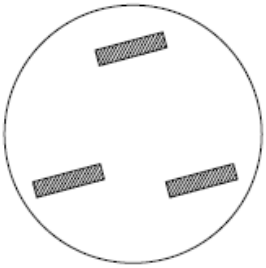


Figure 2.7 Synchro-drive mobile robot

Considering all the aspects described above, the choice of a skid steer robot was done in order to be more compliant with the specifications of Hercules which has no steering wheel but the steer depends on the change of speed on the left and right wheels.

CHAPTER 3
Kinematic Model and Inverse Kinematic Model

"Modelling of mobile robots requires a preliminary analysis of the kinematic constraint imposed by the presence of wheels. Depending on the mechanical structure, such constraints can be integrable or not.
The kinematic model of a mobile robot is essentially the description of the admissible instantaneous motions in respect of the constraints."
"The direct kinematics equation, establishes the functional relationship between the joint variables and the end effector position and orientation. The inverse kinematics consists of the determination of the joint variables corresponding to a given end-effector position and orientation." [2]

"The kinematic motion model is a mathematical model that describe the motion of objects without consideration of forces." [5]

3.1 Model of an SSMR

The model of a SSMR, Skid-Steer Mobile Robot, is similar to the one of the bicycle without a steering angle.
The Kinematic equations extracted from the study done by Krzysztof Kozlowski and Dariusz Pazderski, considering the robot in a plane surface with inertial orthonormal basis (X_g, Y_g, Z_g) and the robot coordinates are (x_l, y_l, z_l), assigned to the robot on its COM, Centre of Mass.
Having as lengths:

- a, the longitudinal distance between the COM and the centres of the backward wheels;
- b, the longitudinal distance between the COM and the centres of the forward wheels;
- c, the lateral distance between the COM and the centre of the left or right wheels.

moving with a linear velocity in the local frame $v = [v_x \ v_y \ 0]^T$ and rotating with an angular velocity $\omega = [0 \ 0 \ \omega]^T = [0 \ 0 \ \dot{\theta}]^T$, $\dot{\theta} = \omega$ holds due to the planar motion.

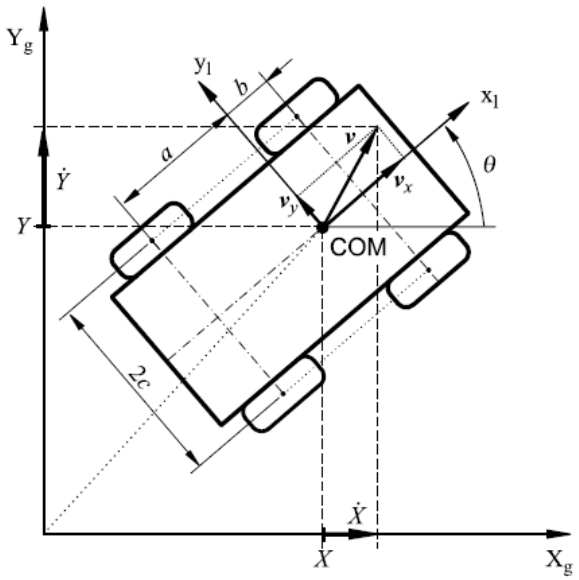


Figure 3.1 Robot Free Body Diagram

The state vector of the robot $q = [X\ Y\ \theta]^T$, describes position and orientation of the robot with respect to the inertial frame.
The velocity vector $\dot{q} = [\dot{X}\ \dot{Y}\ \dot{\theta}]^T$ denotes the vector of generalized velocities, the time variation of the position and orientation, again with respect to the inertial frame.

Putting all together the kinematic model equations can be seen as:

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ \omega\end{bmatrix}. \tag{3.1}$$

Considering the Pacejka Magic Formula tire model for the i-th wheel (i=1,2,3,4):

$$v_{ix} = r_i\omega_i, \tag{3.2}$$

where v_{ix} is the longitudinal component of the total velocity vector v_i of the *i*-th wheel expressed in the local frame and r_i denotes the effective rolling radius of that wheel.
And the geometry of the model we can have the relationship linking the linear velocities, the ICR, instant centre of rotation, the coordinates and the angular robot velocity:

$$\frac{v_x}{y_{ICR}} = -\frac{v_y}{x_{ICR}} = \omega. \tag{3.3}$$

where v_x is the longitudinal velocity and v_y the lateral velocity.

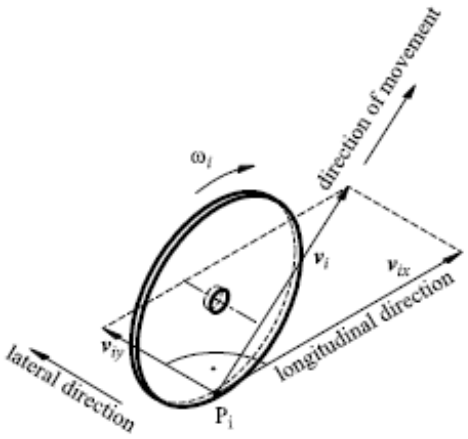


Figure 3.2 Velocities of one wheel

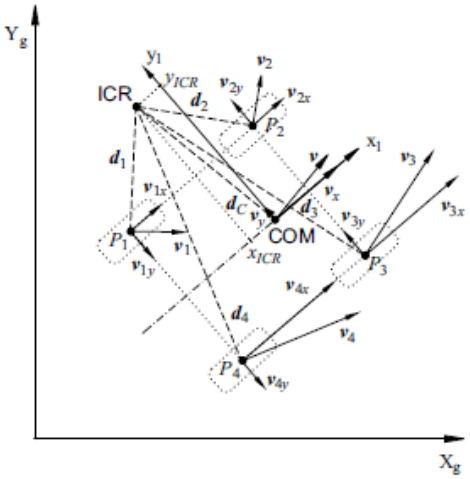


Figure 3.3 Wheel velocities

In brief, the kinematic model, uses as input the measured left and right wheel's angular velocities ω_L and ω_R and the physical parameters of the vehicle, the effective radius of the wheel r and the horizontal distance from the wheels to the centre of mass c to determine the longitudinal speed v_x and the angle of the vehicle with respect the centre of mass θ (=ω).
Thus, we are moving from the wheels towards the operating space, the vehicle generalized velocities and generalized angle with respect the centre of mass: $\dot{X}, \dot{Y}, \theta$.

In order to design a Simulink model which controls the wheels deciding the path the vehicle should follow we need the Inverse Kinematic equations, thus given a generalized pose, retrieve the generalized velocities and so get the angular left and right velocities.
To retrieve the Inverse Kinematic model, I took the Kinematic model and reverse the equations, so we end up with the system:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix}$$
$$\omega_R = \frac{2v_x}{r} - \omega_L \qquad (3.4)$$
$$\omega_L = \frac{v_x - \dot{\theta}c}{r}$$
$$\omega = \dot{\theta}$$

The following Simulink model refers to the test of the direct and inverse kinematics models

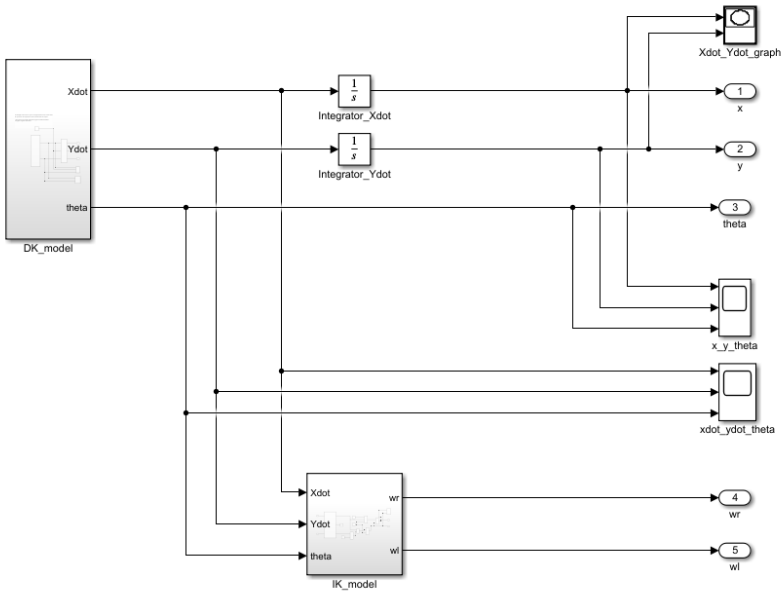


Figure 3.4 Kinematic Test

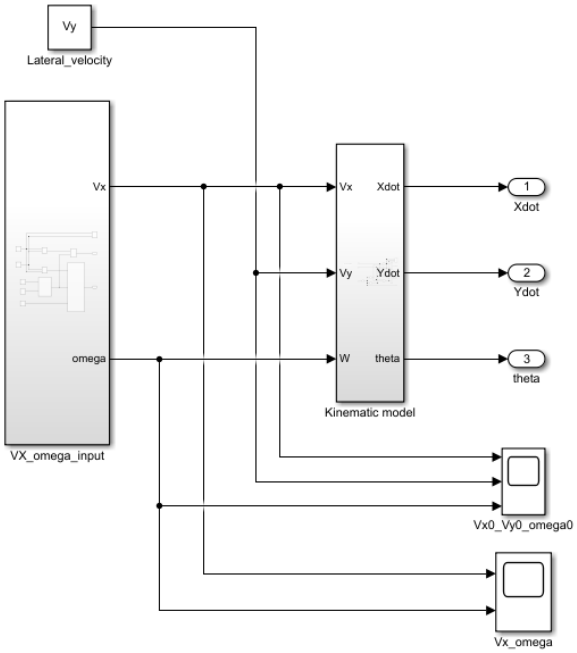


Figure 3.5 DK_model

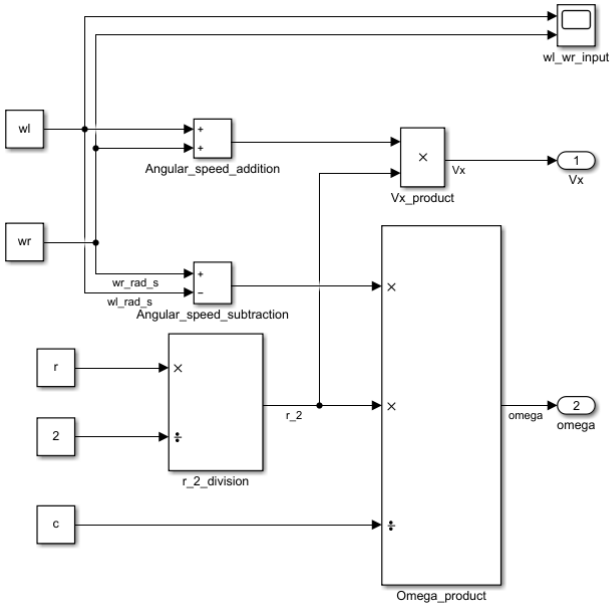


Figure 3.6 VX_omega_input

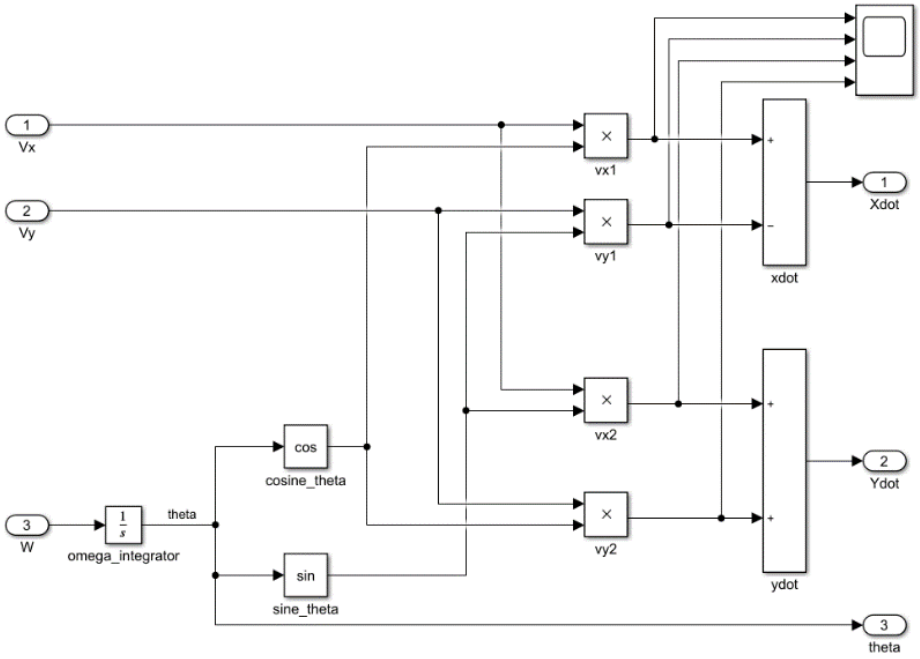


Figure 3.7 Kinematic model

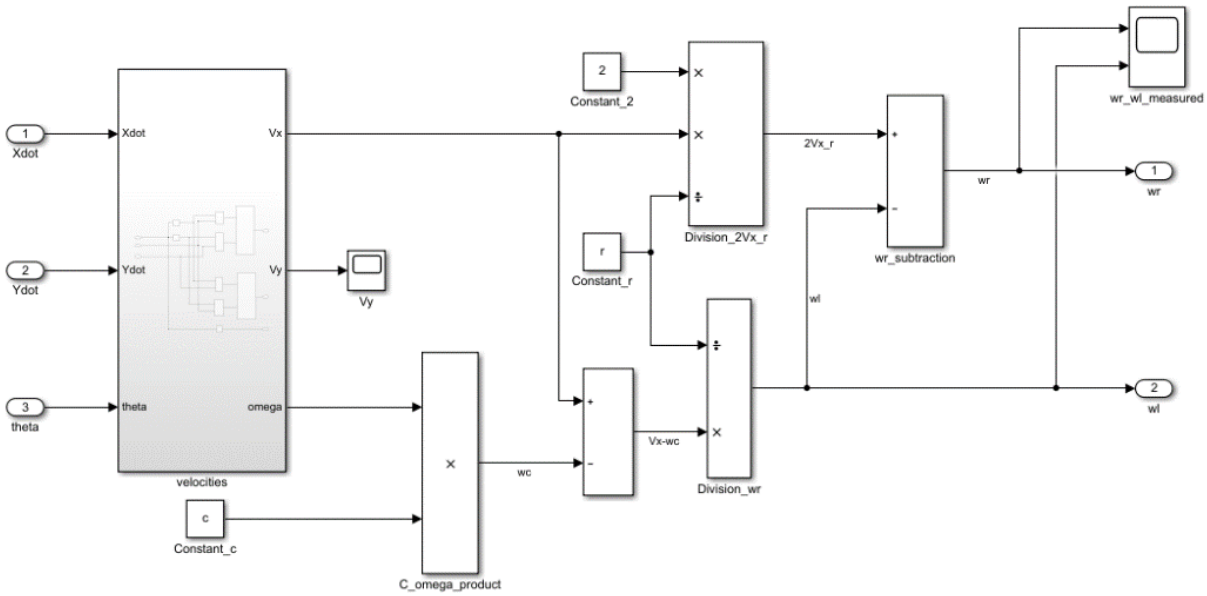


Figure 3.8 IK_model

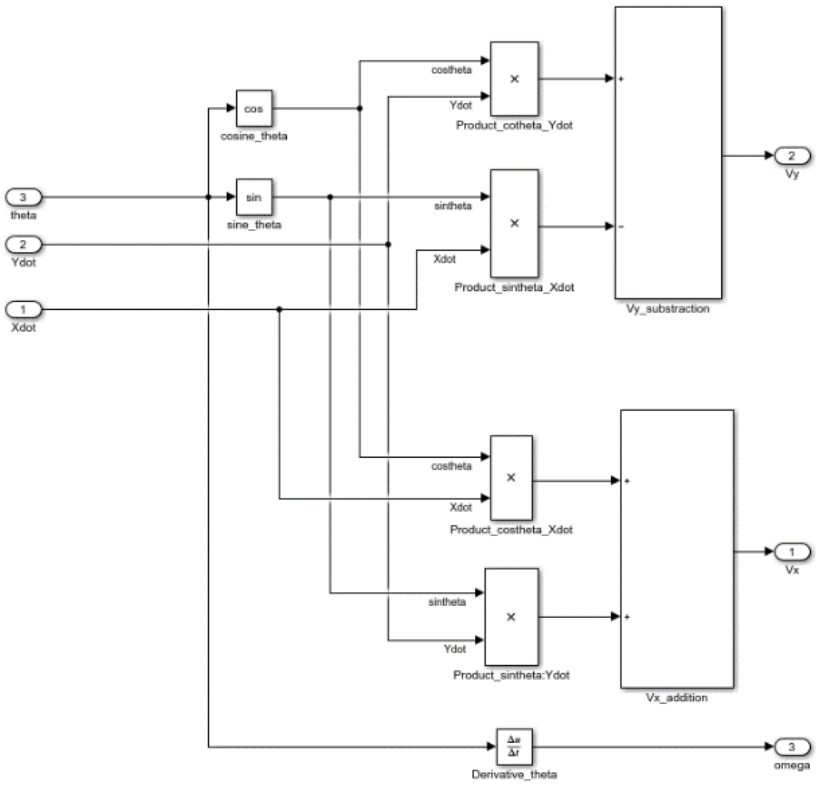


Figure 3.9 velocities

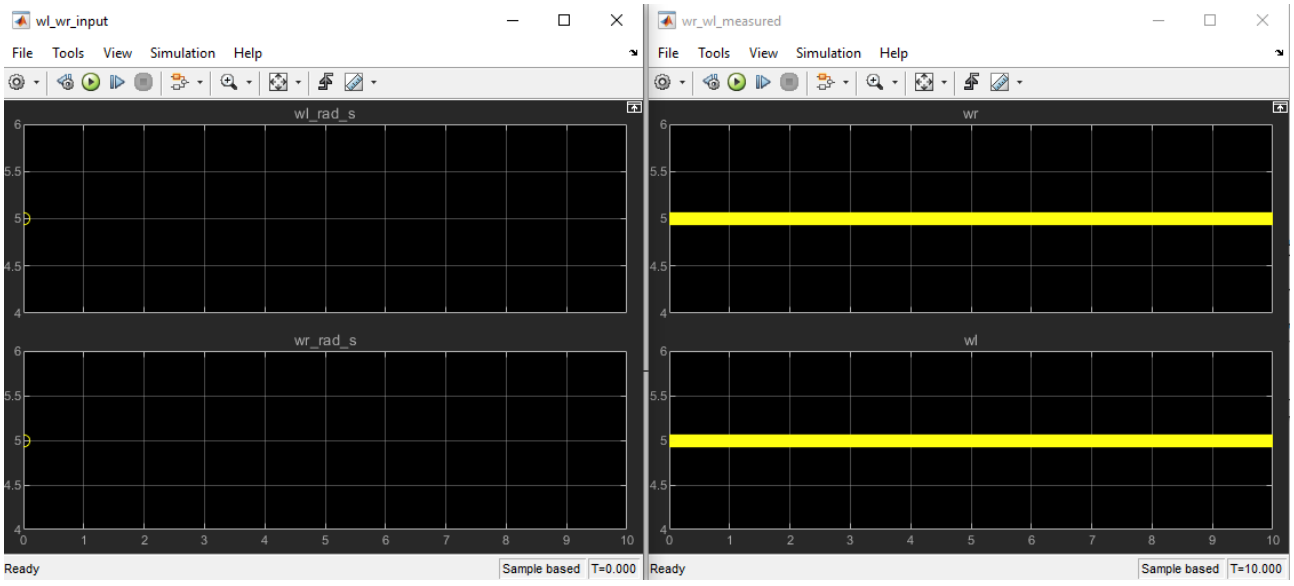


Figure 3.10 Test input and measured angular left and right speeds

From the graphs it can be seen that for the input given to w_r and w_l is of 5 rad/s, entering in the direct kinematics model and the measured values from the inverse kinematic model we have the same values. Thus, the test is successfully completed.

CHAPTER 4
DYNAMIC MODEL

"The dynamic model accounts for the reaction forces and describes the relationship between the motions and the generalized forces acting on the robot." [2]

4.1 SSMR Dynamic model

The dynamic model from the study done by Krzysztof Kozlowski and Dariusz Pazderski, considers as main blocks the active forces F_i and the reactive forces N_i, which are the vertical normal force acting on the wheels.
The first ones are linearly dependent on the wheel torque τ_i and the latter ones are dependent of the gravity g:

$$F_i = \frac{\tau_i}{r}. \tag{4.1}$$

$$\sum_{i=1}^{4} N_i = mg, \tag{4.2}$$

where r denotes the wheel radius and m the vehicle mass.
The symmetry of the vehicle along the longitudinal midline we can have the backward wheels normal forces and the forward normal forces are equal as well, this allows us to further simplify the number of equations:

$$\begin{aligned} N_1 = N_4 &= \frac{b}{2(a+b)}mg, \\ N_2 = N_3 &= \frac{a}{2(a+b)}mg. \end{aligned} \tag{4.3}$$

The dynamic properties of the SSMR are caused by unknown lateral skidding ground interaction forces. The active force F_i and reactive force N_i which are related to the wheel torque τ and gravity g.
The active force F_i is linearly dependent on the wheel control input τ_i

$$F_i = \frac{\tau_i}{r}. \tag{4.4}$$

The vertical normal force N_i, based on the Pacejka Magic Formula, acts from the surface to the wheel.
Considering the four wheels of the vehicle and neglecting additional dynamic properties, the equilibrium equations are:

$$\sum_{i=1}^{4} N_i = mg, \tag{4.5}$$

$$N_1 a = N_2 b, \tag{4.6}$$

$$N_4 a = N_3 b \tag{4.7}$$

where m denotes the vehicle mass and g the gravity acceleration.
Considering the Potential energy equal to zero due to the planar motion, neglecting the energy of the wheels and having homogeneous mass distribution, one can have the longitudinal and lateral active forces and the resistant moment generated by the actuators in the inertial frame around the centre of mass generated by the actuators:

$$F = \begin{bmatrix} F_x \\ F_y \\ M \end{bmatrix} = \frac{1}{r} \begin{bmatrix} \cos\theta \sum_{i=1}^{4} \tau_i \\ \sin\theta \sum_{i=1}^{4} \tau_i \\ c(-\tau_1 - \tau_2 + \tau_3 + \tau_4) \end{bmatrix} \tag{4.8}$$

The torque control input is divided in left and right torques produced by the wheels of the vehicle

$$\tau = \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} = \begin{bmatrix} \tau_1 + \tau_2 \\ \tau_3 + \tau_4 \end{bmatrix}, \tag{4.9}$$

On my Simulink model, the dynamic block has as inputs the right and left torques of the vehicle and the centre of mass angular speed ω and returns as output the generalized velocities of the wheels $\dot{X}, \dot{Y}, \dot{\theta}$ which later on will be converted into wheel angular velocities via an inverse kinematics block and then compared with the desired ones via a PID block.

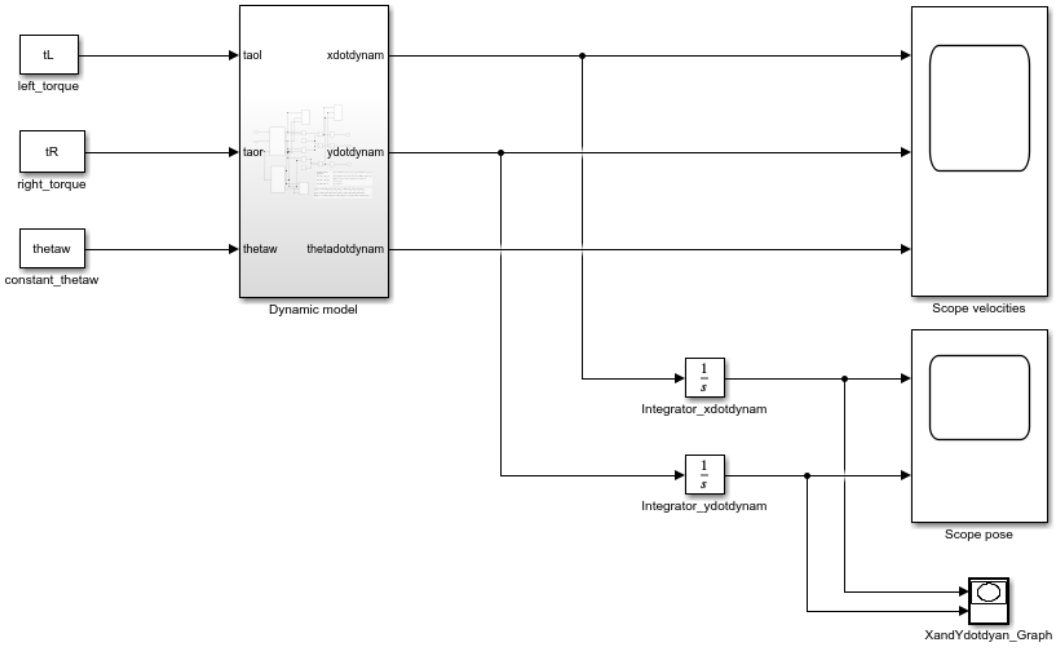


Figure 4.1 Dynamic model test

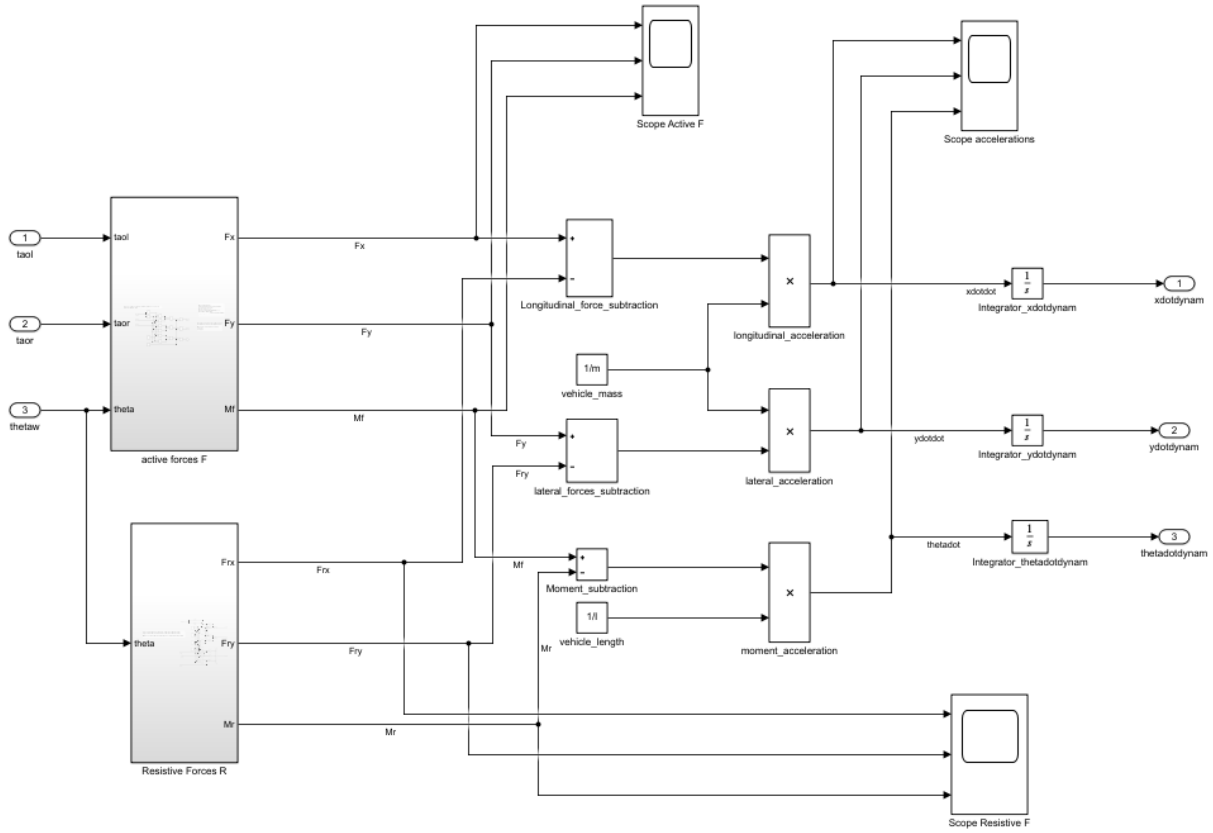


Figure 4.2 Dynamic model

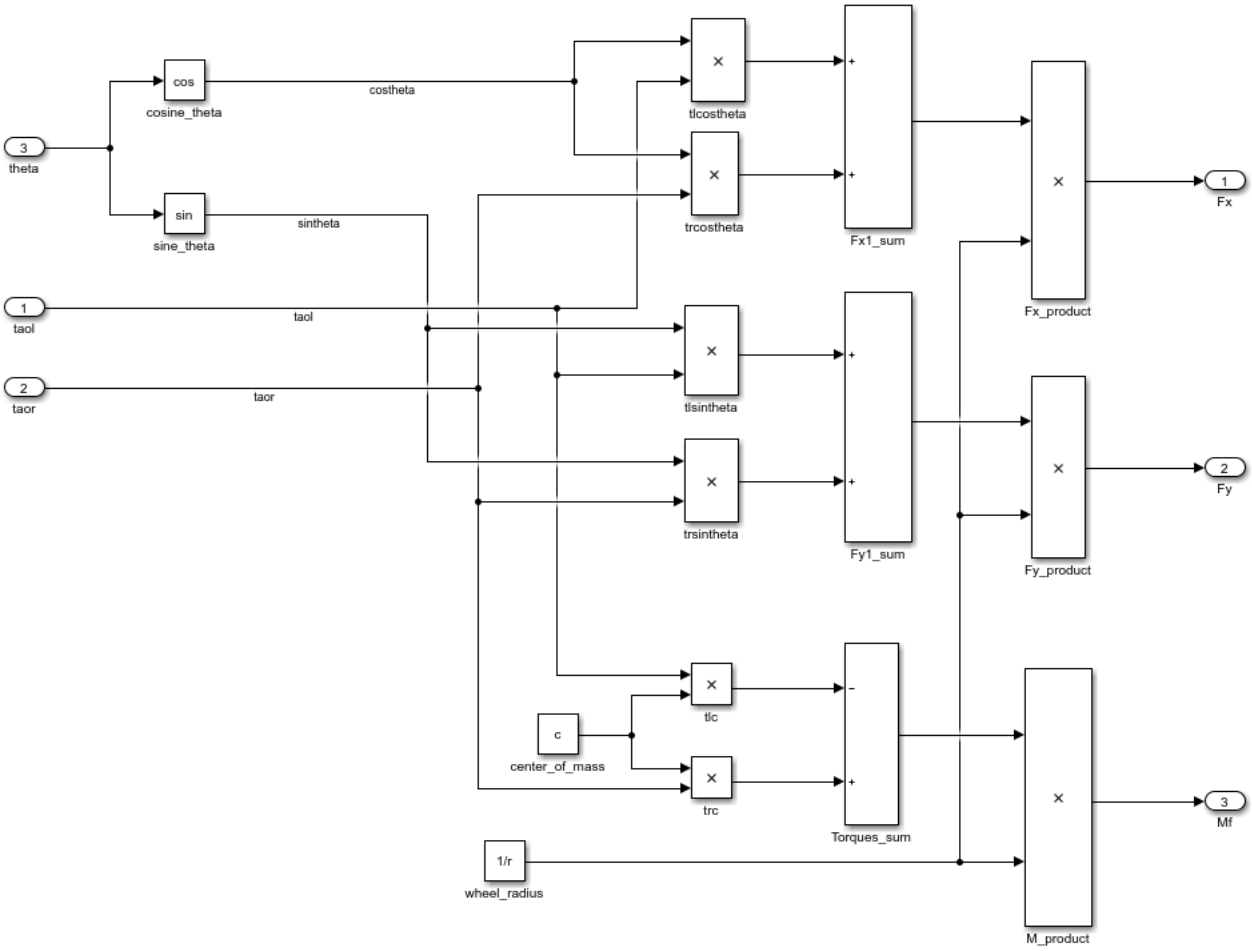


Figure 4.3 Active Forces

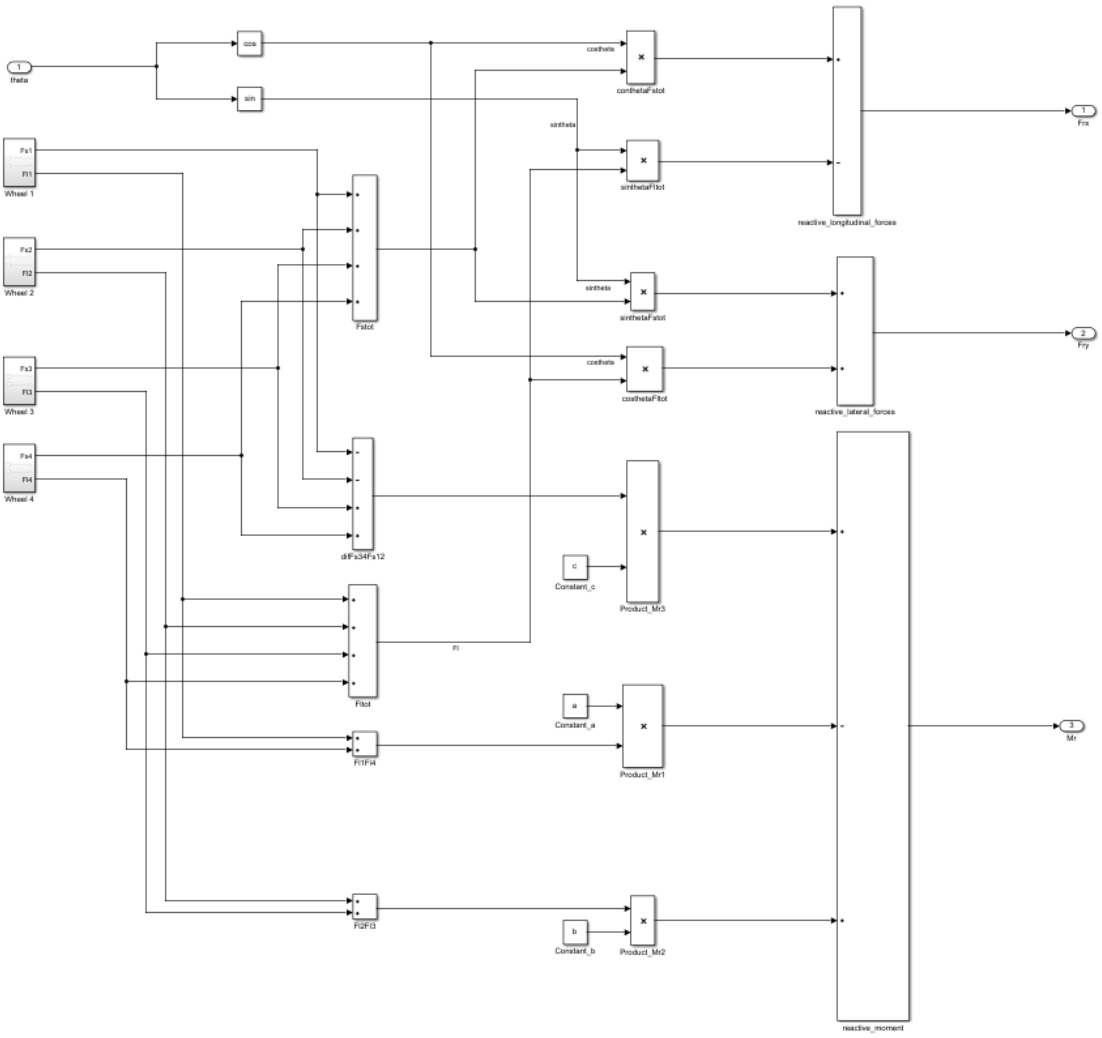


Figure 4.4 Resistive Forces

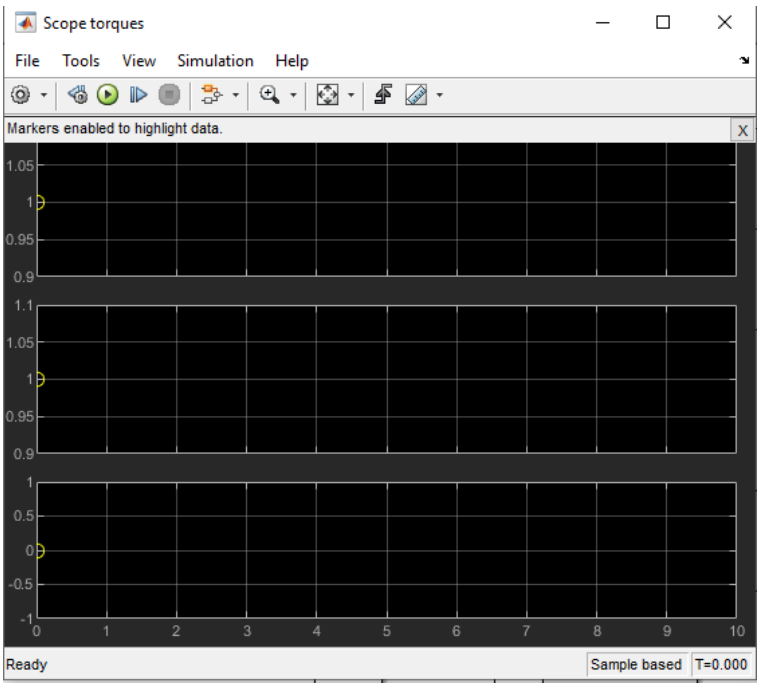


Figure 4.5 Scopes of input torques and angle on the dynamic model

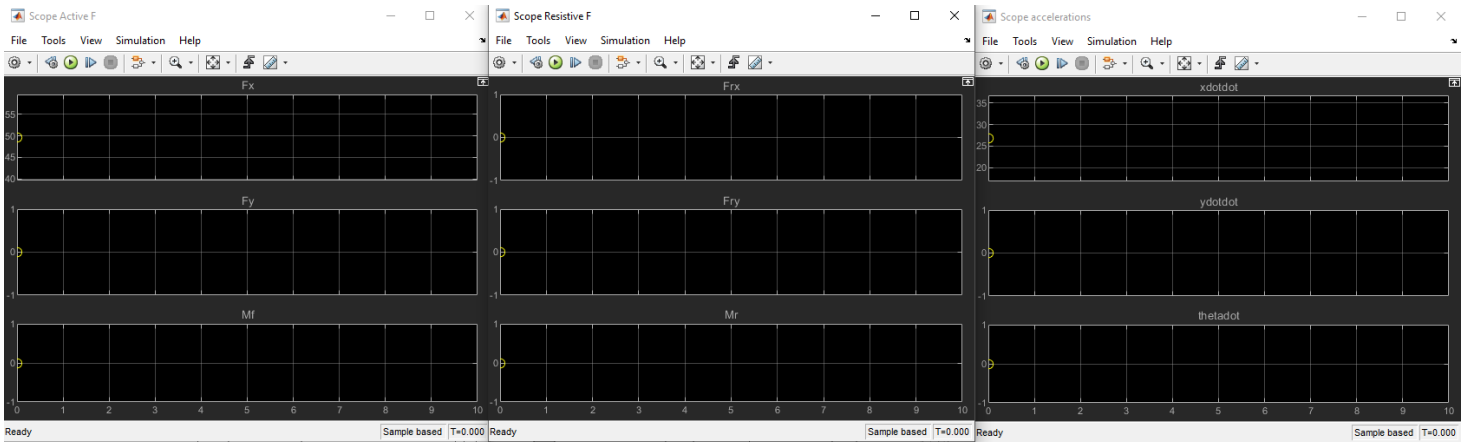


Figure 4.6 Scopes of Active and Reactive Forces and Dynamic accelerations

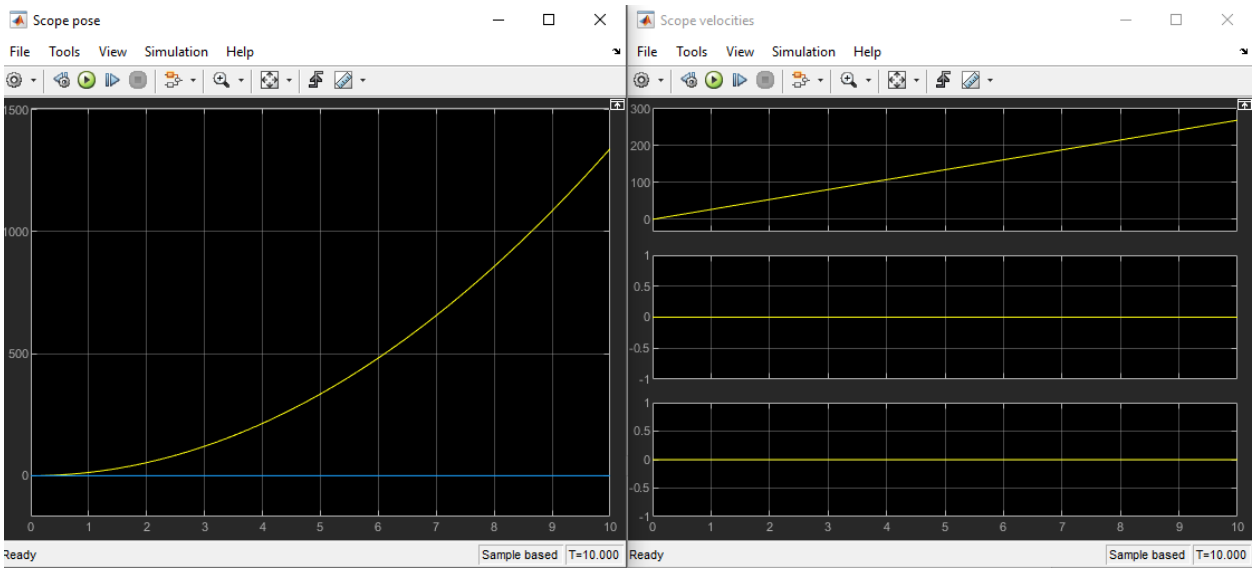


Figure 4.7 Scopes of Dynamic velocities and pose

As can be seen from the graphs, whenever the pose of the robot turns to the left, the right velocity increase in order to make it rotate towards the left.

CHAPTER 5
Trajectory Planning

"Planning a trajectory for a mobile robot can be broken down in finding a path and defining a timing law on the path." [2]

The scope of the thesis is to make the robot to move in an S path whenever a target, robot R1, is found and to follow it keeping a safety distance. In case the target is not detected, the robot should keep moving forward. In particular, the displacement should be linked to the lateral and longitudinal speeds of the robot.

In literature [2] there are three types of algorithms for mobile robots:

- ***Cubic Cartesian polynomials***, based in the interpolation of the initial values x_i, y_i and the final values x_f, y_f of the flat outputs x, y. Furthermore, they can be integrated with some constraints, depending on the initial and final values, but for the purpose of this thesis were substituted with the lateral and longitudinal velocities.
- ***Chained form,*** the path is planned in the chained form coordinates z. Computing the initial and final values z_i and z_f that correspond to q_i and q_f, by using the change of coordinates. Then interpolating the initial and final values of z_1 and z_3 (the flat outputs) with the appropriate boundary conditions on the remaining variable $z_2=z_3'/z_1'$.
- ***Parameterized inputs***, it consists on writing the inputs, rather than the path, in parameterized form, and computing the value of the parameters so as to drive the robot from q_i to q_f.

As anticipated before, the method used in this thesis is the Cubic Cartesian polynomial method due to its simplicity in application. As a start point it was tested an S shaped path from the starting position to the final position as shown in fig.5.1. Later on, the same method will be discussed but integrating the linear speed of the vehicle in the parameters and keeping a safety distance from the vehicle.

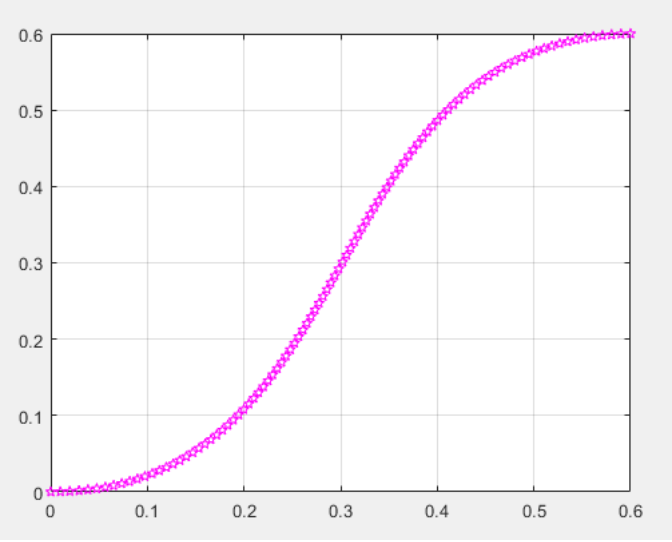


Figure 5.1 S path from initial pose (0, 0) to (0.6, 0.6)

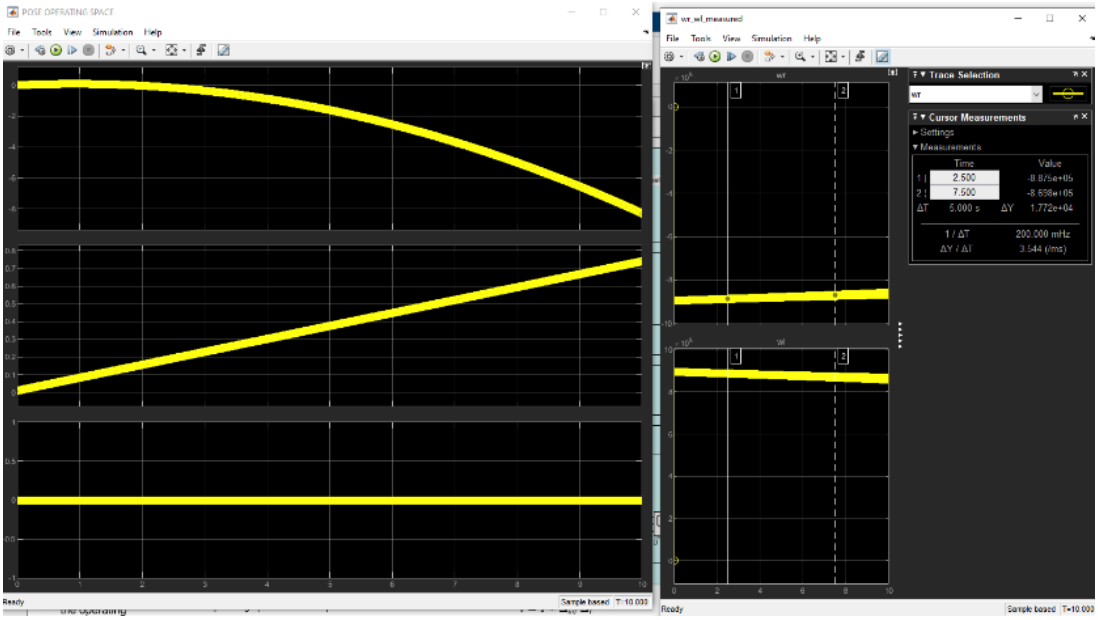


Figure 5.2 Comparison of pose and wheels turning to the right direction

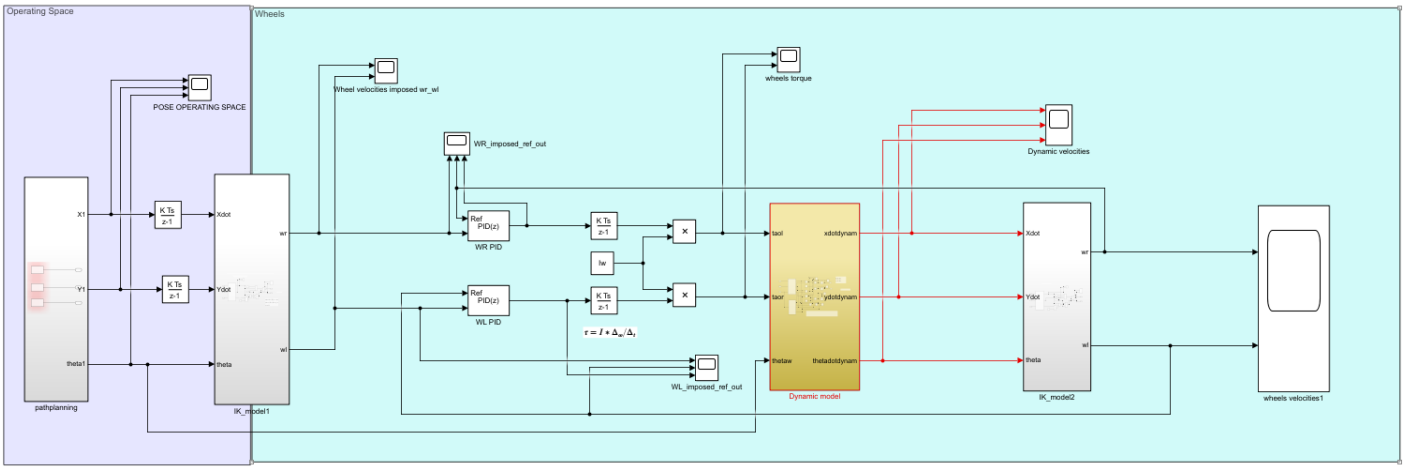


Figure 5.3 Complete model

CHAPTER 6:
Integration of Ultrasonic Sensors in the Simulink Model

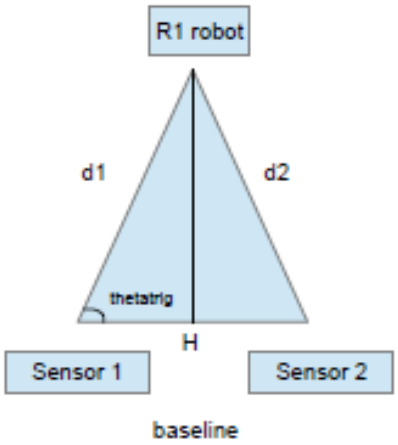


Figure 8.1 Triangulation scheme

8.1 Pose R1

As a first test, it was taken as an example the project Ultrasound Sensor: 2D Tracking with Arduino from Instructables [6] which introduces two ultrasonic sensors and a triangulation method depending on the distance between the sensors and the object detected, robot R1. They can be seen as a triangle.
Pose of R1 is retrieved by the ultrasonic sensors, connected to the Arduino board on pins 4,5,6 and 7, which measures *d1* and *d2* in meters, I converted it in centimetres. The baseline is the distance between the sensors and so we can retrieve the internal angle *thetatrig* via the cosine rule, to identify the distance of the object in front of our vehicle. The code give as output the pose of the forward vehicle in X,Y,θ coordinates.
Once it has been tested in Arduino, it is added to a Matlab function in Simulink on a function called PoseR1_fcn.
In order to avoid NaN values, I have defined a constraint to the internal angle to a fixed point, so that later on whenever my vehicle wants to move forward and there are no objects nearby it can have a reference point to reach at each cycle, until it encounters an object.

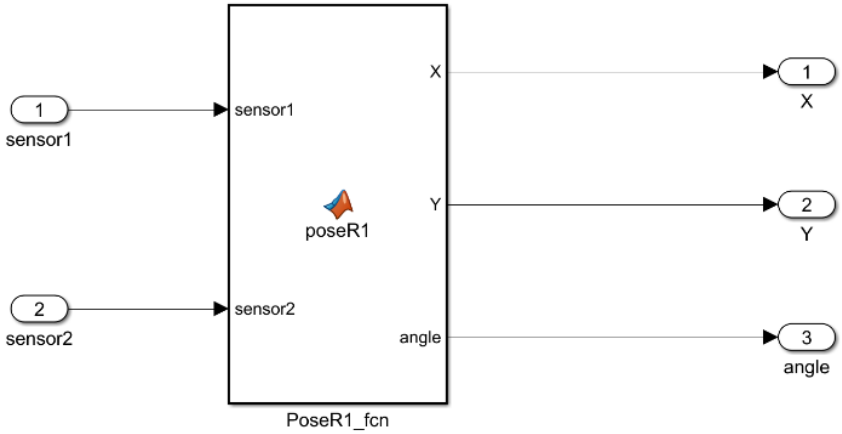


Figure 6.2 PoseR1_fcn

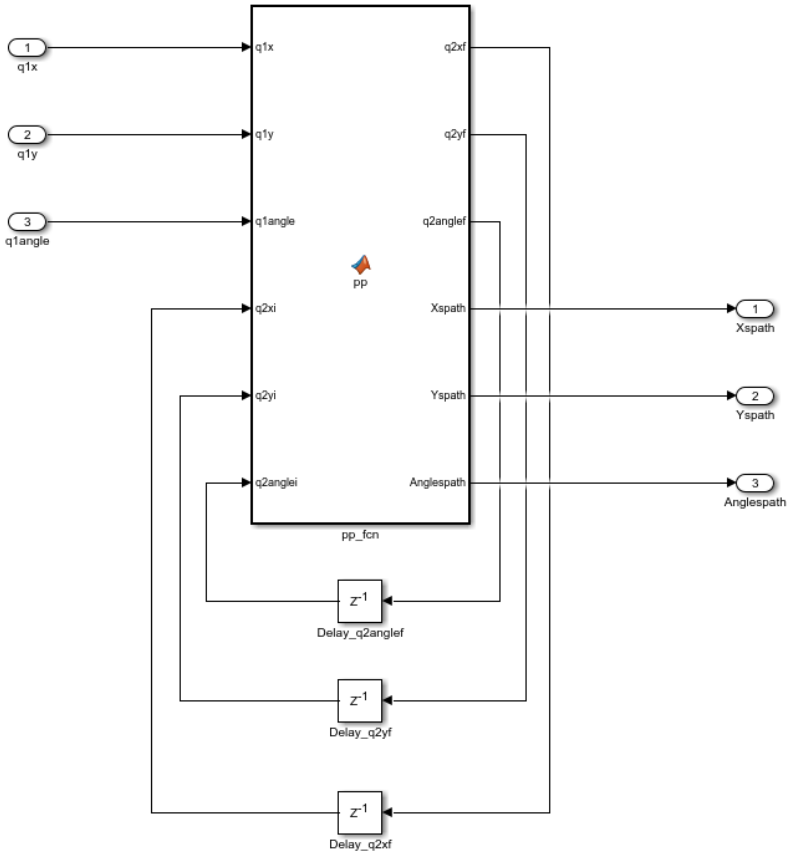


Figure 6.3 pp_fcn

6.2 Path Plan

On the Matlab function pp_fcn we have as input the pose of the forward vehicle and as feedback the previous pose of my vehicle, designed by taking the pose of my vehicle signal going through a delay block. The S path formulas where updated giving velocity constraints in order to move towards the object with a certain speed. In case the vehicle approaches an object, the vehicle needs to reduce the speed proportionally to the distance measured, and in case the object stands still, the vehicle should stop keeping the safety distance.

Arduino meas (m)		internal angle	Pose R1 (cm)		Pose R2 (cm)	
1.2454	0.24988	NaN	100	100	0.00051	1.0012e-05
1.2526	0.24233	NaN	100	100	50	50
1.258	0.2509	NaN	100	100	75	75
1.2454	0.2413	NaN	100	100	87.5	87.5
1.2495	0.2509	NaN	100	100	93.75	93.75
1.2614	0.23358	NaN	100	100	96.875	96.875
0.48706	0.23941	NaN	100	100	96.875	96.875
0.48877	0.28126	NaN	100	100	96.875	96.875
0.284	0.27303	73.346	13.339	27.209	96.875	96.875
0.29035	0.29155	80.356	10.064	28.625	55.107	62.042
0.27697	0.35946	136.35	100	100	32.586	45.333
0.2672	0.50575	NaN	100	100	66.293	72.667
0.27886	0.3502	125.96	100	100	83.146	86.333
0.28246	0.30544	92.629	3.9045	28.216	91.573	93.166
0.28418	0.31487	97.362	1.5587	28.183	47.739	60.691
0.284	0.34574	118.36	-8.2926	24.991	24.649	44.437
0.29292	0.33031	101.82	-0.80228	28.671	8.178	34.714
0.30767	0.34574	102.69	-1.5603	30.015	3.6879	31.692
0.32345	0.47866	NaN	100	100	3.6884	31.692
0.32345	0.53028	NaN	100	100	51.844	65.846
0.33906	0.5109	NaN	100	100	75.922	82.923

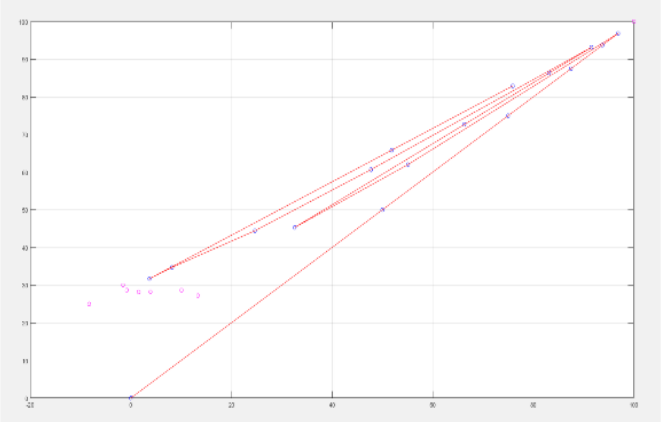


Figure 6.4 Measurements and follow up simulation

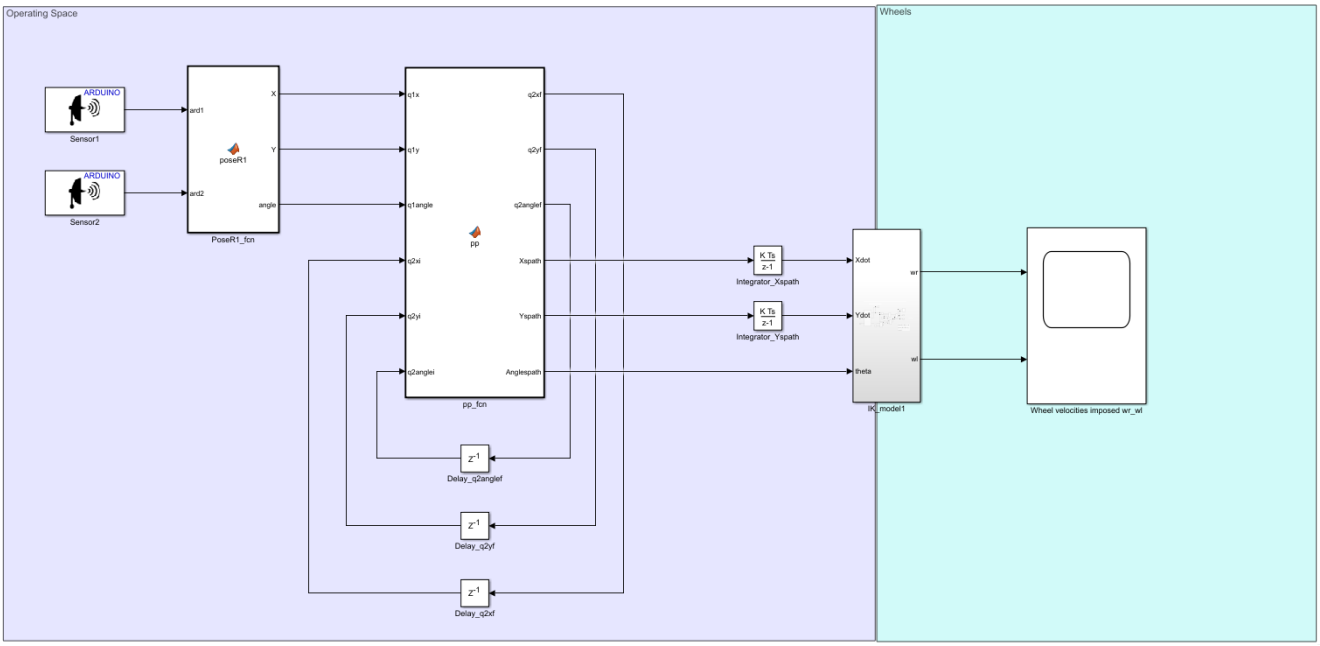


Figure 6.5 Final model integrating poseR1, path plan and Inverse Kinematic equations

CHAPTER 7:
Code Generation

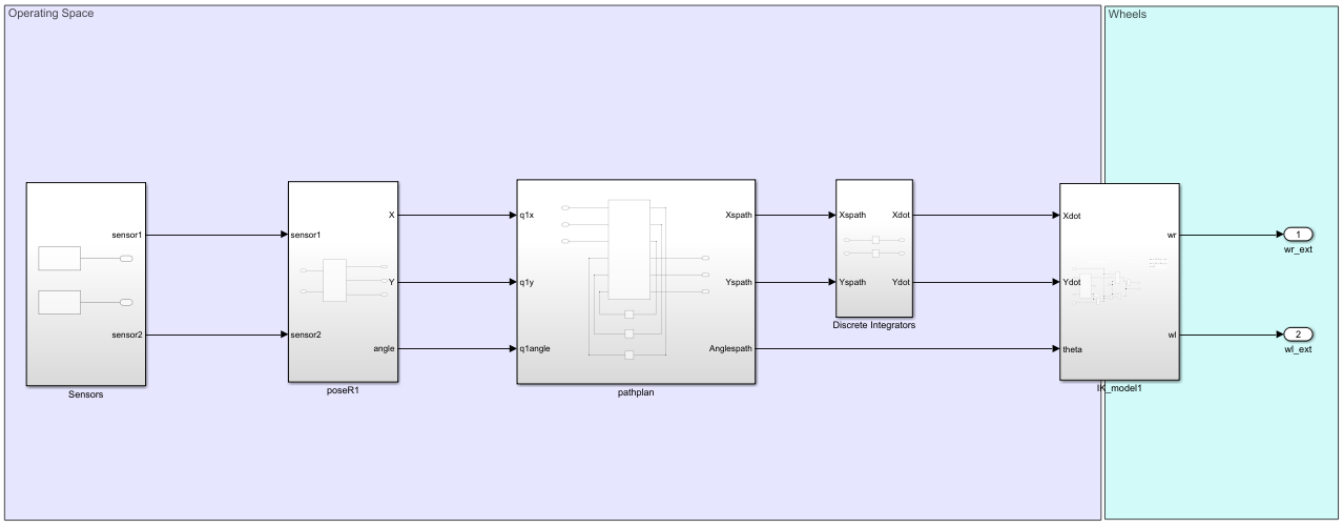


Figure 7.1 Model for code generation

In order to test the model inside the Arduino UNO board, Hardware in the loop method, I have prepared the model for code generation by specifying code generation settings in the Configuration Parameters dialog box.

On the Code Generation section, I choose the System target file ert.tlc (Embedded Coder) and language C. Also ticked the Package code and artifacts in order to have an easier transportability of the code files to deploy the files later on in the Arduino sketch.

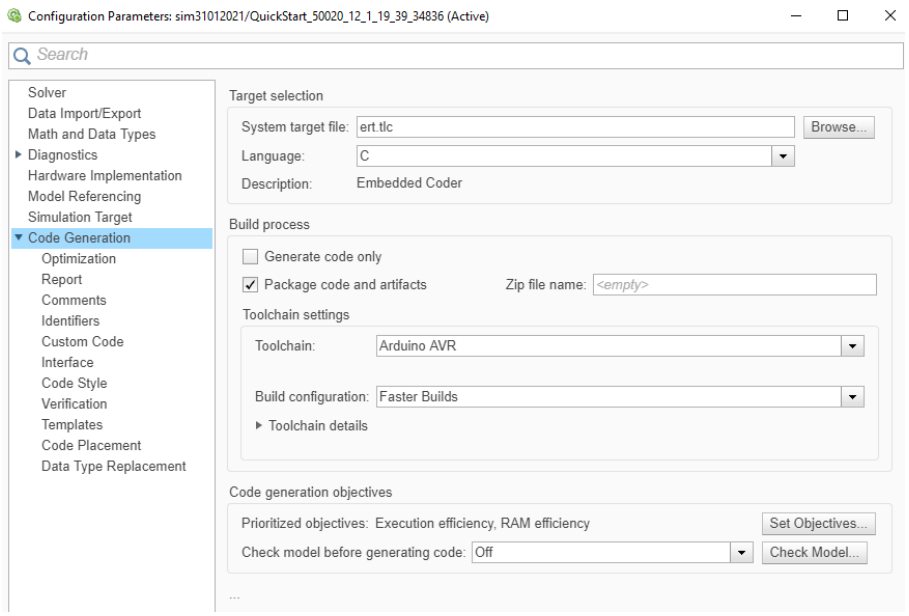


Figure 7.2 Configuration Code Generation

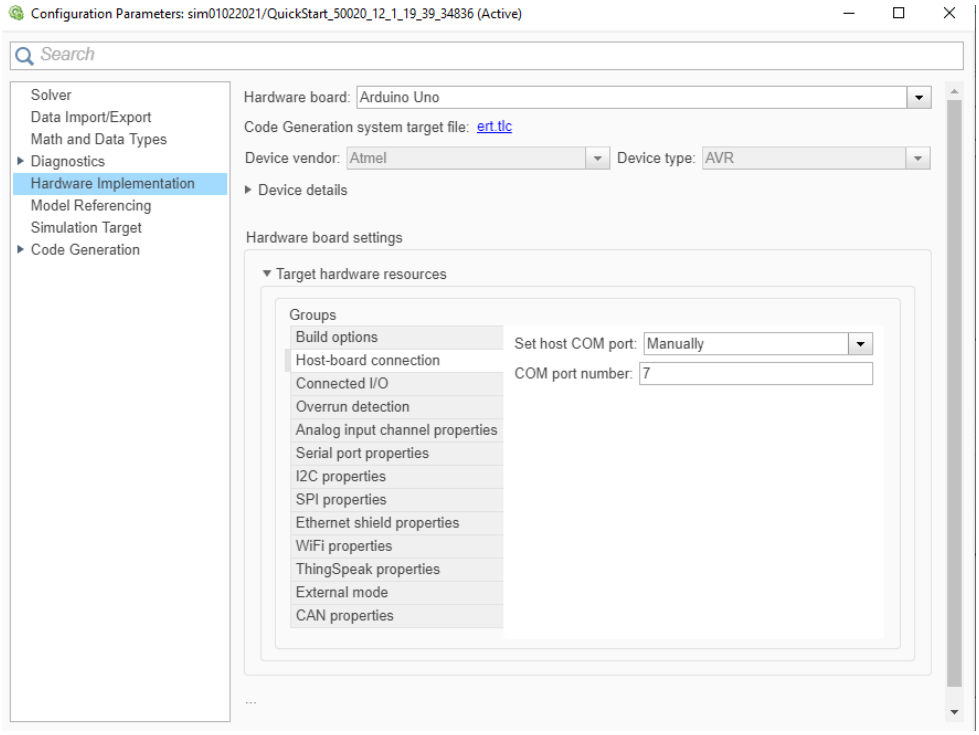


Figure 7.3 Configuration Hardware Implementation

The model blocks are the ones shown in the following pictures

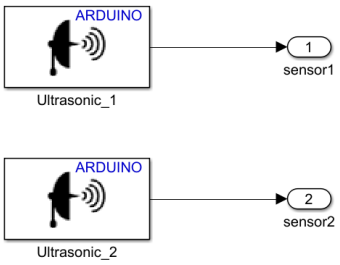


Figure 7.4 Sensors

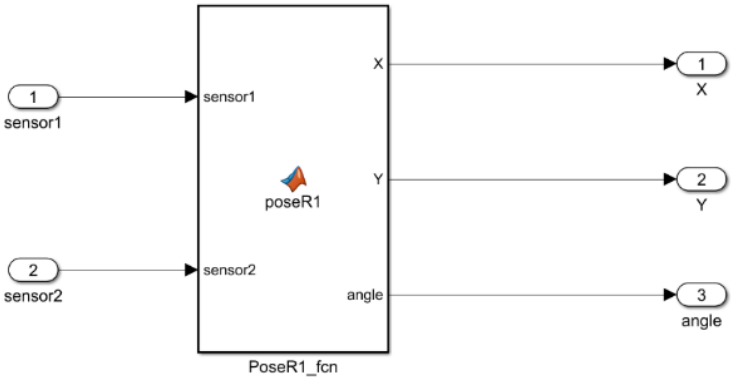


Figure 7.5 PoseR1

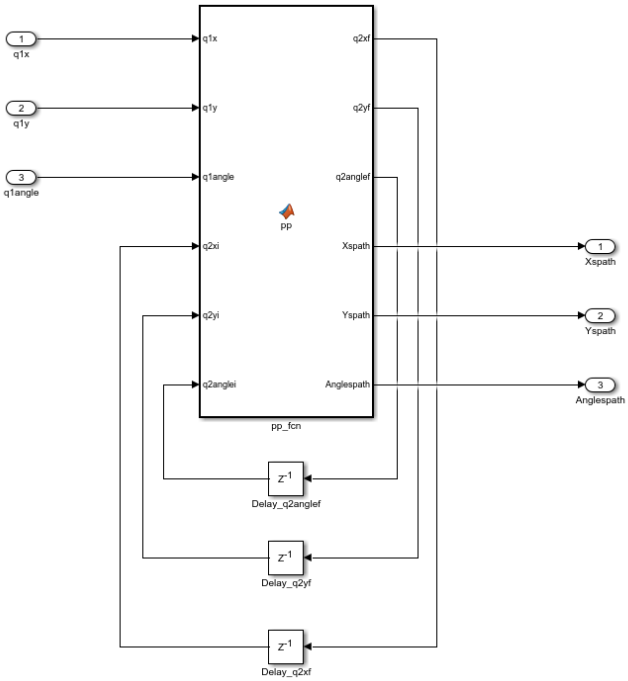


Figure 7.6 Path Plan

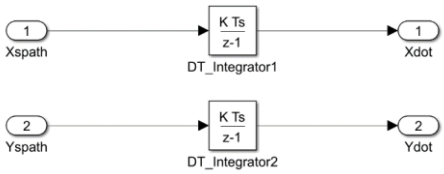


Figure 7.7 Integrators

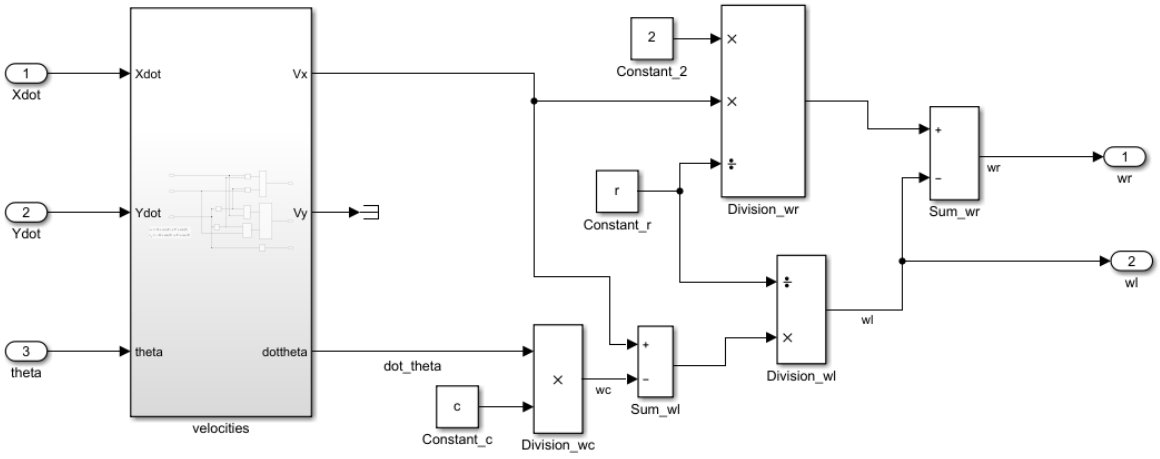


Figure 7.8 Inverse Kinematic model

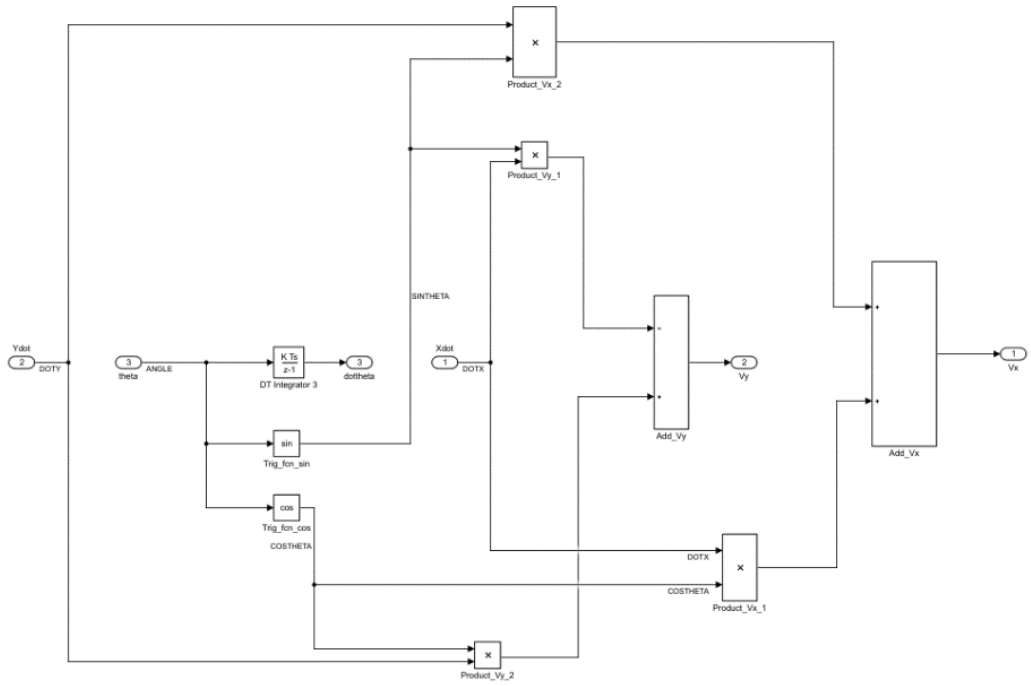


Figure 7.9 Velocities of Inverse Kinematic model

As can be seen the Simulink model comply with the MAAB rules:

- Inport blocks on the left side of the diagram and the Output ones on the right side of the diagram.
- The names of the blocks are below the blocks.
- The Signal lines are drawn with right angles. There are some intersections that I could not avoid. The signals flow moves from left to right.

Whenever one builds a model via the Embedded Coder, one has 3 folders: otherFiles, R2020a and sim31012021 (the model's name folder). On the model's folder there is an extra folder ending with ert_rtw, which stands for embedded real time code in real-time workshop, and inside of it there are some header and source code C files that I used to deploy the model inside Arduino.

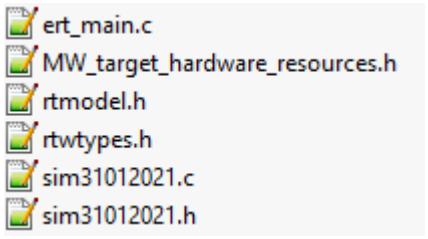


Figure 7.10 Generated files

The ert_main.c file is the testbench for the model and was skipped since is what the Arduino ino sketch file does in practice.
The rtmodel header file was not used since it is just defining the rtmGetStopRequested macros which was not used on the code.
Taking some parts that were inside the source code file of the model and arranging them inside the sketch file, including the header files needed to compile the code.
The header files that were used are described as follows:

- Rtwtypes, which assigns the Arduino hardware-specific data types to the ones used in

Simulink.

- Sim31012021, define the model-specific data types via typedef struct definitions.
- MW_Ultrasonic, this extra header file was added from another folder of Simulink which defines the setup and read of the ultrasonic sensors, in order to use them on the main code without defining trig pin or echo pin explicitly.

The code was then compiled and tested in the robot.

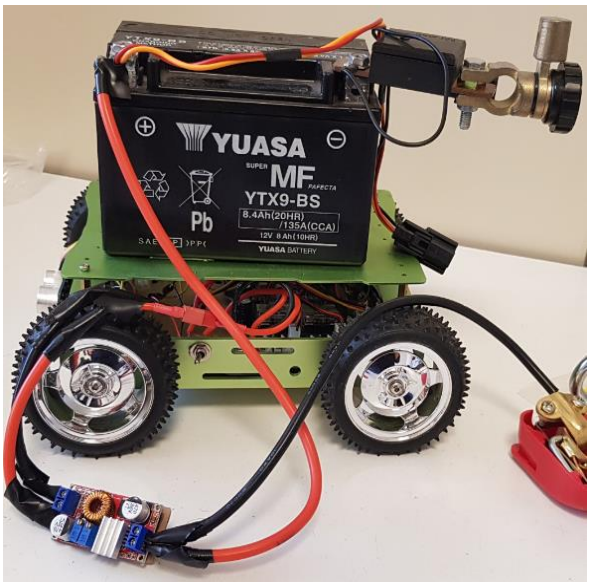


Figure 7.11 Hercules 4WD Brain Technologies

Chapter 8
Conclusions

The scope of the thesis was to make a mobile robot autonomous by integrating ultrasonic sensors as vision system, using that vision system as inputs to define the path plan of the robot and control the vehicle wheels accordingly.
As described above on the previous chapters, all the tests where successfully done, the testing of the whole system was sometimes tedious due to the fact that the generated code model was too big for the Arduino memory, reason why it was necessary to reassemble the code for the Arduino IDE with the functions created by Simulink and connect them all together “manually”.

Further developments can be done on the testing of the mobile robot for more complex environments with more than one object moving, for example.

Bibliography

[1] MathWorks, “White paper: Sensor Fusion and Tracking for Autonomous Systems,” 2019.

[2] L. S. L. V. G. O. Bruno Siciliano, “Robotics Modelling, Planning and Control,” in *Robotics Modelling, Planning and Control*, Springer, 2009.

[3] D. Kohanbash, “Robots for Roboticists,” 25 July 2016. [Online]. Available: http://robotsforroboticists.com/drive-selection. [Accessed 25 February 2021].

[4] K. Kotay, “ROBO-Rats,” 04 April 2001. [Online]. Available: https://groups.csail.mit.edu/drl/courses/cs54-2001s/skidsteer.html. [Accessed 25 February 2021].

[5] C. Bregler, “Springer,” Springer, 05 February 2016. [Online]. Available: https://doi.org/10.1007/978-0-387-31439-6_587. [Accessed 26 February 2021].

[6] abhilash_patel, “Instructables,” [Online]. Available: https://www.instructables.com/Ultrasound-Sensor-2D-Tracking-With-Arduino/. [Accessed 01 September 2020].

[7] Wikipedia, “Wikipedia,” 09 Febbraio 2021. [Online]. Available: https://en.wikipedia.org/wiki/Sensor. [Accessed 24 Febbraio 2021].

[8] Merriam-Webster, “Merriam-Webster,” [Online]. Available: https://www.merriam-webster.com/dictionary/sensor. [Accessed 24 02 2021].

[9] Vidya.M, “Electrical Technology,” [Online]. Available: https://www.electricaltechnology.org/2018/11/types-sensors-applications.html. [Accessed 24 Febbraio 2021].

[10] D. Jost, “Fierce Electronics,” 7 October 2019. [Online]. Available: https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor. [Accessed 25 02 2021].

[11] B. Zuo, “Seeedstudio,” [Online]. Available: https://wiki.seeedstudio.com/Skeleton_Bot-4WD_hercules_mobile_robotic_platform. [Accessed July 2020].

[12] D. P. Krystof Kozlowki, “Modeling and Control of a 4-wheel skid-steering mobile robot,” 2004. [Online]. Available: http://matwbn.icm.edu.pl/ksiazki/amc/amc14/amc1445.pdf. [Accessed 16 July 2020].

[13] Arduino.cc, “Arduino,” 05 February 2008. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction. [Accessed September 2020].

[14] Arduino.cc, “Arduino,” [Online]. Available: https://store.arduino.cc/arduino-uno-rev3. [Accessed September 2020].

Appendix

A. Sensors

“A sensor is a device, module, machine, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor.” [7]

“A device that responds to a physical stimulus (such as heat, light, sound, pressure, magnetism, or a particular motion) and transmits a resulting impulse (as for measurement or operating a control).” [8]

“Autonomous systems rely on sensor suites that provide data about the surrounding environment to feed the perception system. These sensors include radars and cameras, which provide detections of objects in their field of view.
They also include LiDAR sensors, which provide point clouds of returns from obstacles in the environment, and in some cases, ultrasound and sonar sensors. Autonomous systems must also be able to estimate their position to maintain self-awareness. For this, sensors such as Inertial Measurement Unit (IMU) and Global Positioning System (GPS) receivers are used.” [1]

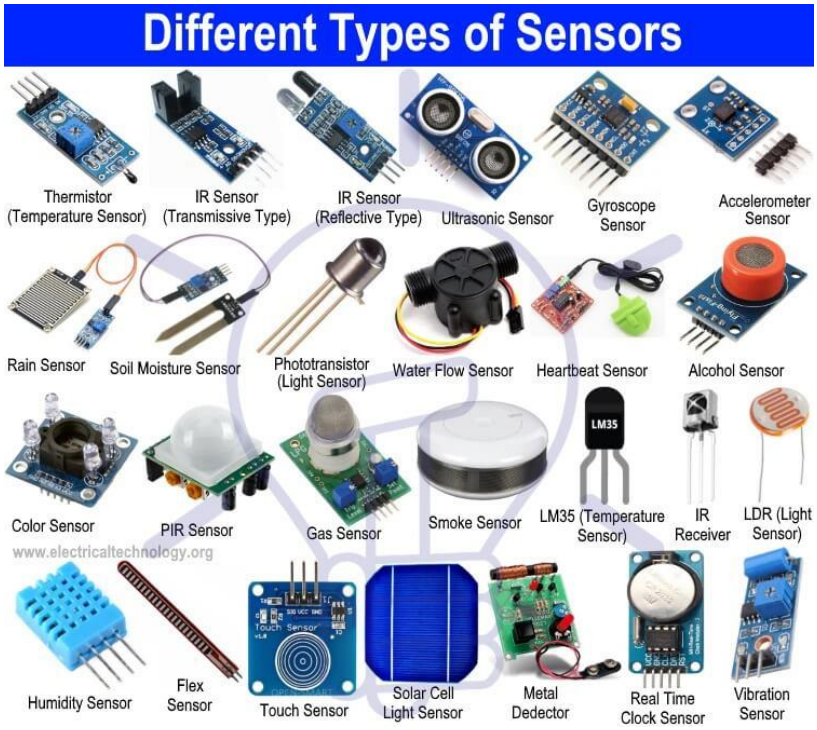


Figure 2 Types of Sensors [9]

Ultrasonic sensors: They are used primarily as proximity sensors. They can be found in automobile self-parking technology and anti-collision safety system. They are also used in robotic obstacle detection systems as well as manufacturing technology. In comparison to infrared (IR) sensors in proximity sensing applications, ultrasonic sensors are not as susceptible to interference of smoke, gas, and other airborne particles (through the physical components are still affected by variables such as heat, in some cases one can use a temperature).
Ultrasonic sensors are also used as level sensors to detect, monitor, and regulate liquid levels in closed containers (such as vats in chemical factories). Most notably, ultrasonic technology has

enabled the medical industry to produce images of internal organs, identify tumours, and ensure the health of babies in the womb. [10]
In order to make more accurate measurements in hot environments one can use the temperature sensor LM35 combined with the Ultrasonic sensors.
Robotics

The essential component of a robot is the mechanical system endowed, in general, with a locomotion apparatus (wheel, crawlers, mechanical legs) and a manipulation apparatus (mechanical arms, end-effectors, artificial hands).
The capability to exert an action, both locomotion and manipulation, is provided by an actuation system which animates the mechanical components of the robot. The concept of such a system refers to the context of motion control, dealing with servomotors, drives and transmissions.

The capability for perception is entrusted to a sensory system which can acquire data on the internal status of the mechanical system (proprioceptive sensors, such as position transducers) as well as on the external status of the environment (exteroceptive sensors, such as force sensors and cameras). The realization of such a system refers to the context of materials properties, signal conditioning, data processing, and information retrieval.
The capability for connecting action to perception in an intelligent fashion is provided by a control system which can command the execution of the action in respect to the goals set by a task planning technique, as well as of the constraints imposed by the robot and the environment. The realization of such a system follows the same feedback principle devoted to control of human body functions, possibly exploiting the description of the robotic system's components (modelling). The context is that of cybernetics, dealing with control and supervision of robot motions, artificial intelligence and expert systems, the computational architecture and programming environment. [2]

B. Mobile robots

"The main feature of mobile robots is the presence of a mobile base which allows the robot to move freely in the environment. From a mechanical point of view, a mobile robot consists of one or more rigid bodies equipped with a locomotion system." [2]

There are two main classes of mobile robots:
Wheeled mobile robots, which consist of a rigid body and a system of wheels which provide motion with respect to the ground.
Legged mobile robots, which are made of multiple rigid bodies, interconnected by prismatic joints or by revolute joints.

Here we are going to deal with wheeled robots.
There are three types of conventional wheels:

1) The **fixed wheel**, it can rotate about an axis that goes through the centre of the wheel and is orthogonal to the wheel plane. The wheel is rigidly attached to the chassis, whose orientation with respect to the wheel is therefore constant.
2) The **steerable wheel**, it has two axes of rotation. The first is the same as a fixed wheel, while the second is vertical and goes through the centre of the wheel. This allows the wheel to change its orientation with respect to the chassis.
3) The **caster wheel**, it has two axes of rotation, but the vertical axis does not pass through the centre of the wheel, from which it is displaced by a constant offset. Such an arrangement causes the wheel to swivel automatically, rapidly aligning with the direction of motion of the chassis. This type of wheel is introduced to provide a supporting point for static balance without affecting the mobility of the base.

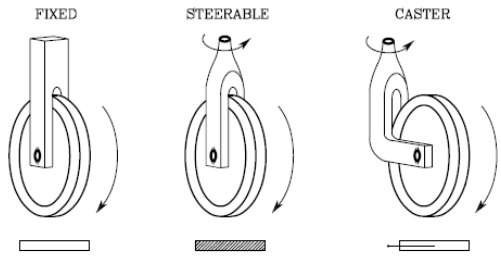


Figure 3.2 Conventional wheels

In a differential-drive vehicle there are two fixed wheels with a common axis of rotation, and one or more caster wheels, typically smaller, whose function is to keep the robot statically balanced. The two fixed wheels are separately controlled, in that different values of angular velocity may be arbitrarily imposed, while the caster wheel is passive. Such a robot can rotate on the spot, provided that the angular velocities of the two wheels are equal and opposite.
In this thesis it will be used a subsystem of the differential-drive vehicle called Skid Steer Drive, technically speaking it does not have the steering angle.

There exists another vehicle with similar mobility using a synchro-drive kinematic arrangement. This robot has three aligned steerable wheels which are synchronously driven by only two motors through a mechanical coupling, e.g., a chain or a transmission belt. The first motor controls the rotation of the wheels around the horizontal axis, thus providing the driving force (traction) to the vehicle. The second motor controls the rotation of the wheels around the vertical axis, hence affecting their orientation.

There is also a tricycle vehicle where there are two fixed wheels mounted on a rear axle and a steerable wheel in front. The fixed wheels are driven by a single motor which controls their traction, while the steerable wheel is driven by another motor which changes its orientation, acting then as a steering device. Alternatively, the two rear wheels may be passive and the front wheel may provide traction as well as steering.

A car-like vehicle has two fixed wheels mounted on a rear axle and two steerable wheels mounted on a front axle. One motor provides traction while the other changes the orientation of the front wheels with respect to the vehicle. In order to avoid slippage, the two front wheels must have a different orientation when the vehicle moves along a curve; in particular, the internal wheel is slightly more steered with respect to the external one. This is guaranteed by the use of a specific device called Ackermann steering. [2]

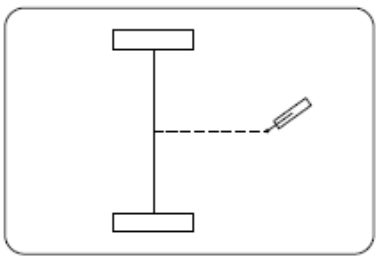


Figure 3.3 Tricycle mobile robot and differential-drive mobile robot

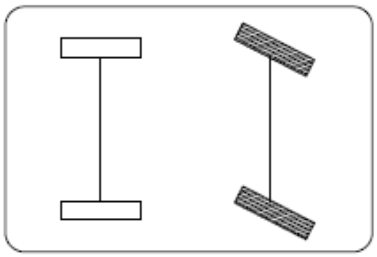


Figure 3.3 Car-like mobile robot

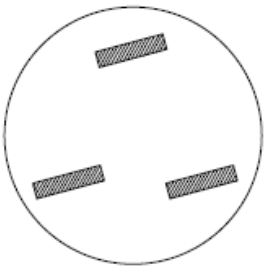


Figure 3.4 Synchro-drive mobile robot

C. 4WD Hercules Mobile Robot

The 4WD Hercules Mobile Robotic Platform, is a discontinued bot from the Seed Studio company, it is designed such that one can enter into the robotic world making their own robot mobile

platform.
The name Hercules comes from the Titan in Greek mythology, known for his strength and spirit of adventure.

It consists of the component of Hercules motor controller, Hercules skeleton, gear motor etc. The Hercules motor controller can consistently support the current up to 15A and driving voltage in the range of 6V-20V, thus, it can supply strong motive power to the whole platform. The Hercules skeleton is made up of aluminium alloy plate with firm but pliable texture which can carry, display and connect sorts of accessories in the project. Its 4 powerful gear motors, especially a couple of them from Seeed's original encipheror (encoder) can monitor the running speed of your platform and amend the process parameter by the shaped closed-loop control. This provides a possibility to control precise process. Besides, other accessories such as wheels, cooper cylinder and acrylic guard plate that make the platform to be a completely mobile platform.

Hercules is a suite of open-platform. It is convenient for users to install all kinds of components through the hole sites on the board. In addition, the Grove connector reserved on the board can help the inventor input several sorts of sensor data into system. What's more, Hercules controller is Arduino compatible, so you can freely alter the device drives and programs. [11]

D. Kinematic Model

D.1 Nonholonomic Constraints

There are some nonlinear behaviours typical of the mobile robots and are the non-holonomic constraints which develop on the wheels as a kinematic constraint that reduce in general its local mobility, while leaving intact the possibility of reaching arbitrary configurations by appropriate manoeuvres.
For example, consider a mechanical system whose configuration $q \in C$ is described by a vector of generalized coordinates, and assume that the *configuration space* C coincides with $\mathbb{R}^n$.
The motion of the system that is represented by the evolution of $\boldsymbol{q}$ over time may be subjected to constraints that can be classified under various criteria, like being expressed as equalities (bilateral) or inequalities (unilateral), and may depend explicitly on time (rheonomic) or not (scleronomic).

The holonomic or integrable constraints are of the form:

$$h_i(q) = 0 \qquad i = 1, \ldots, k < n \tag{D.1.1}$$

Assuming that the functions $h_i: C \to \mathbb{R}$ are of class C^∞ smooth and independent.
The effect of holonomic constraints is to reduce the space of accessible configurations to a subset of C with dimension *n-k*. A mechanical system for which all the constraint can be expressed in the form of (D.1.1) is called *holonomic*.
In the presence of holonomic constraints, the implicit function theorem can be used in principle to solve the equations in (D.1.1) by expressing *k* generalized coordinates as a function of the remaining *n-k*, so as to eliminate them from the formulation of the problem. In general, this procedure is only valid locally, and may introduce singularities.

Holonomic constraints are generally the result of mechanical interconnections between the various bodies of the system.

Constraints that involve generalized coordinates and velocities

$$a_i(q, \dot{q}) = 0 \qquad i = 1, \ldots, k < n \tag{D.1.2}$$

are called *kinematic*. They constrain the instantaneous admissible motion of the mechanical system by reducing the set of generalized velocities that can be attained at each configuration. Kinematic constraints are generally expressed in Pfaffian form, i.e., they are linear in the generalized velocities in vector or matrix form:

$$a_i^T(q)\dot{q} = 0 \qquad i = 1, \ldots, k < n, \tag{D.1.3}$$

$$A^T(q)\dot{q} = 0 \tag{D.1.4}$$

Vectors $a_i : C \to \mathbb{R}^n$ are assumed to be smooth as well as linearly independent.
The existence of *k* holonomic constraints ($D.1.1$) implies that of an equal number of kinematic constraints:

$$\frac{dh_i(q)}{dt} = \frac{\partial h_i(q)}{\partial t}\dot{q} = 0 \quad i = 1, \ldots, k \tag{D.1.5}$$

A system of kinematic constraints in the form (D.1.4) may or may not be integrable to the form of (D.1.1). In the negative case, the kinematic constraints are said to be non-holonomic or non-integrable. A mechanical system that is subject to at least one such constraint is called nonholonomic.
The nonholonomic constraints reduce the mobility of the mechanical system in a completely different way with respect to holonomic constraints.
Considering a single Pfaffian constraint

$$a^T(q)\dot{q} = 0 \tag{D.1.6}$$

If the constraint is holonomic, it can be integrated and written as

$$h(q) = c, \tag{D.1.7}$$

where $\frac{\partial h}{\partial q} = \gamma(q)a^T(q)$, with $\gamma(q) \neq 0$ an *integrating factor* and *c* an *integration constant*.
Therefore, there is a loss of accessibility in the configuration space, because the motion of the mechanical system in *C* is confined to a particular *level surface* of the scalar function *h*. This surface, which depends on the initial configuration q_0 through the values of $h(q_0) = c$, has dimension *n-1*.
Assuming that the constraint (D.1.6) is nonholonomic, then the generalized velocities are indeed constrained to belong to a subspace of dimension *n-1*, i.e., the null space of matrix $a^T(q)$.
Nevertheless, the fact that the constraint is non-integrable means that there is no loss of accessibility in *C* for the system. While the number of DOFs decreases to *n-1* due to the constraint, the number of generalized coordinates cannot be reduced, not even locally.
An *n*-dimensional mechanical system subject to *k* nonholonomic constraints can access its whole configuration space *C*, although at any configuration its generalized velocities must belong to an (*n-k*) -dimensional subspace.

D.2 Integrability Conditions

In the presence of Pfaffian kinematic constraints, integrability conditions can be used to decide whether the system is holonomic or nonholonomic.
Considering the case of a single Pfaffian constraint:

$$a^T(q)\dot{q} = \sum_{j=1}^{n} a_j(q)\dot{q}_J = 0. \tag{D.1.8}$$

For this constraint to be integrable, there must exist a scalar function $h(q)$ and an integrating factor

$\gamma(q) \neq 0$ such that the following condition holds:

$$\gamma(q)a_j(q) = \frac{\partial h(q)}{\partial q_j} \qquad j = 1, \ldots, n. \qquad (D.1.9)$$

the converse is also true: if there exists and integrating factor $\gamma(q) \neq 0$ such that $\gamma(q)a(q)$ is the gradient of a scalar function $h(q)$, constraint (D.1.8) is integrable, by using Schwarz theorem on the symmetry of second derivatives, the integrability condition (D.1.9) may be replaced by the following system of partial differential equations:

$$\frac{\partial(\gamma a_k)}{\partial q_j} = \frac{\partial(\gamma a_j)}{\partial q_k} j, k = 1, \ldots, n, j \neq k, \qquad (D.1.10)$$

that does not contain the unknown function $h(q)$. This last equation implies that Pfaffian constrain with constant coefficients a_j is always holonomic. [2]

The system of *k* Pfaffian constraints (D.1.4) entails that the admissible generalized velocities at each configuration *q* belong to the (*n-k*)-dimensional null space of matrix $A^T(q)$. Denoting by $\{g_1(q), \ldots, g_{n-k}(q)\}$ a basis of $N(A^T(q))$, the admissible trajectories for the mechanical system can then be characterized as the solutions of the nonlinear dynamic system

$$\dot{q} = \sum_{j=1}^{m} g_i(q)u_j = G(q)u \qquad m = n - k, \qquad (D.1.11)$$

where $q \in \mathbb{R}^n$ is the state vector and $u = [u_1 \ldots u_m]^T \in \mathbb{R}^m$ is the input vector. The system (D.1.11) is said to be driftless because one has $\dot{q} = 0$ if the input is zero.
The choice of the *input vector fields* $g_1(q), \ldots, g_m(q)$ in (D.11) is not unique. The components of *u* may have different meanings. In general, it is possible to choose the basis of $N(A^T(q))$ in such a way that the $u_j s$ have a physical interpretation. In any case vector u may not be directly related to the actual control inputs, that are in general forces and/or torques. That's why the equation (D.1.11) is referred as the kinematic model of the constrained mechanical system.
The holonomy or non-holonomy of constraints (D.1.4) can be established by analysing the controllability properties of the associated kinematic model (D.1.11).

Two cases are possible:

1. If system (D.1.11) is controllable, given two arbitrary configurations q_i and q_f in C, there exist a choice of $u(t)$ that steers the system from q_i to q_f, i.e., there exist a trajectory that joins the two configurations and satisfies the kinematic constraints (D.1.4). Therefore, these do not affect in any way the accessibility of C, and they are nonholonomic.
2. If system (D.1.11) is not controllable, the kinematic constraints (D.1.4) reduce the set of accessible configurations in C. Hence, the constraints are partially or completely integrable depending on the dimension $v < n$ of the accessible configuration space. In particular:

2a. If $m < v < n$, the loss of accessibility is not maximal, and thus constraints (D.1.4) are only partially integrable. The mechanical system is still nonholonomic.
2b. If $v = m$, the loss of accessibility is maximal, and constraints (D.1.4) are completely integrable. Therefore, the mechanical system is holonomic. [2]

In brief, the holonomic motion is whenever a vehicle can go in all x and y directions and it is not constrained. The nonholonomic motion is whenever a vehicle is constrained to only certain motions.

D.2.1 Unicycle mobile robot

A unicycle is a vehicle with a single orientable wheel. Its configuration is completely described by $q = [x\ y\ \theta]^T$, where (x, y) are the Cartesian coordinates of the contact point of the wheel with the ground (or equivalently, of the wheel centre) and θ is the orientation of the wheel with respect to the x axis.

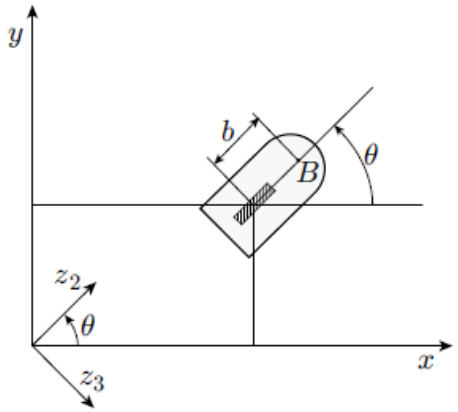


Figure 4.1 Generalized coordinates for a unicycle

The pure rolling constraint for the wheel is expressed as

$$\dot{x}sin\theta - \dot{y}cos\theta = [sin\theta - cos\theta\ 0]\dot{q} = 0, \qquad (D.1.12)$$

entailing that the velocity of the contact point is zero in the direction orthogonal to the sagittal axis of the vehicle. The line passing through the contact point and having such direction is therefore called *zero motion line.*
Consider the matrix

$$G(q) = [g_1(q) \quad g_2(q)] = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix}, \qquad (D.1.13)$$

whose columns $g_1(q)$ and $g_2(q)$ are, for each q, a basis of the null space of the matrix associated with the Pfaffian constraint. All the admissible generalized velocities at q are therefore obtained as a linear combination of $g_1(q)$ and $g_2(q)$. The kinematic model of the unicycle is then

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta \\ sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w, \qquad (D.1.14)$$

where the inputs v and w have a clear physical interpretation. Where v is the driving velocity, i.e., the modulus (with sign) of the contact point velocity vector, it's given by the angular speed of the wheel around its horizontal axis multiplied by the wheel radius, whereas the steering velocity w is the wheel angular speed around the vertical axis.

A unicycle, a vehicle equipped with a single wheel, is a robot with a serious problem of balance in static conditions. There exist vehicles that are kinematically equivalent to a unicycle but more stable from a mechanical viewpoint. The most important ones are the differential drive and the synchro drive vehicles.

For the differential drive mobile robot, denote by (x, y) the Cartesian coordinates of the midpoint of the segment joining the two-wheel centres, and by θ the common orientation of the fixed wheels, thus the vehicle body. Then, the kinematic model of the unicycle also applies to the

differential drive vehicle, provided that the driving and steering velocities v and w are expressed as a function of the actual velocity inputs, i.e., the angular speed w_R and w_L of the right and left wheel, respectively. There is a one-to-one correspondence between the two sets of inputs:

$$v = \frac{r(w_R + w_l)}{2} \quad w = \frac{r(w_R - w_l)}{2}, \tag{D.1.15}$$

where r is the radius of the wheels and d is the distance between their centres.
The equivalence with the kinematic model is even more straight forward for the synchro drive mobile robot, whose control inputs are indeed the driving velocity v and the steering velocity w, that are common to the three orientable wheels. The Cartesian coordinates (x, y) may represent in this case any point of the robot, while θ is the common orientation of the body of a synchro drive vehicle never changes, unless a third actuator is added for this specific purpose.
(Robotics-Modelling, Planning and Control Chapter 11 page 479)

D.2.2 Bicycle mobile robot

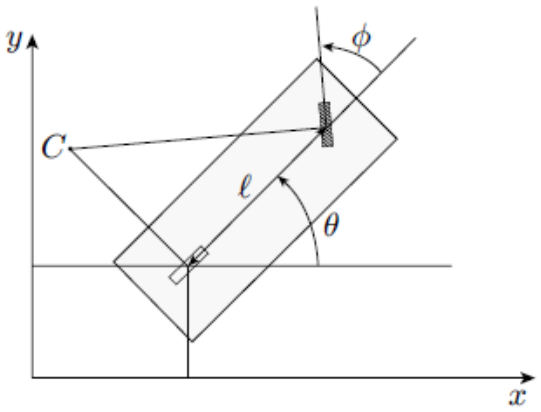


Figure 4.2 Generalized coordinates and instantaneous centre of rotation for a bicycle

The bicycle is a vehicle having an orientable wheel and a fixed wheel arranged. A possible choice for the generalized coordinates is $q = [x\ y\ \theta\ \phi]^T$, where (x, y) are the Cartesian coordinates of the contact point between the rear wheel and the ground, θ is the orientation of the vehicle with respect to the x axis, and ϕ is the steering angle of the front wheel with respect to the vehicle.
The motion of the vehicles is subject to two pure rolling constraints, one for each wheel:

$$\dot{x_f}\sin(\theta + \phi) - \dot{y_f}\cos(\theta + \phi) = 0 \tag{D.1.16}$$

$$\dot{x}sin\theta - \dot{y}cos\theta = 0 \tag{D.1.17}$$

where (x_f, y_f) is the Cartesian position of the centre of the front wheel. The geometric meaning of these constraints is obvious: the velocity of the centre of the from wheel is zero in the direction orthogonal to the wheel itself, while the velocity of the centre of the rear wheel is zero in the direction orthogonal to the sagittal axis of the vehicle. The zero motion lines of the two wheels meet at a point C called *instantaneous centre of rotation* (Fig. 4.2), whose position depends only on (and changes with) the configuration q of the bicycle. Each with centre in C.
Using the rigid body constraints

$$\begin{aligned} x_f &= x + lcos\theta \\ y_f &= y + lsin\theta, \end{aligned} \tag{D.1.18}$$

where l is the distance between the wheels, constraint $(D.1.16)$ can be rewritten as

$$\dot{x}\sin(\theta+\phi)-\dot{y}\cos(\theta+\phi)-l\dot{\theta}cos\phi=0 \qquad (D.1.19)$$

The matrix associated with the Pfaffian constraints $(D.1.17)$, $(D.1.19)$ is then

$$A^t(q)=\begin{bmatrix} sin\theta & -cos\theta & 0 & 0 \\ \sin(\theta+\phi) & -\cos(\theta+\phi) & -lcos\phi & 0 \end{bmatrix}, \qquad (D.1.20)$$

with constant rank k=2. The dimension of its null space is n-k=2, and all the admissible velocities at q may be written as a linear combination of a basis of $N(A^T(q))$, for example

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix}=\begin{bmatrix} cos\theta cos\phi \\ sin\theta sin\phi \\ sin\phi/l \\ 0 \end{bmatrix}u_1+\begin{bmatrix}0\\0\\0\\1\end{bmatrix}u_2. \qquad (D.1.21)$$

Since the front wheel is orientable, it is immediate to set $u_2=\omega$, where ω is the steering velocity. The expression of u_1 depends instead on how the vehicle is driven.

If the bicycle has front-wheel drive, one has directly $u_1=v$, where v is the driving velocity of the front wheel. The corresponding kinematic model is

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix}=\begin{bmatrix} cos\theta cos\phi \\ sin\theta sin\phi \\ sin\phi/l \\ 0 \end{bmatrix}v+\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\omega. \qquad (D.1.22)$$

Denoting by $g_1(q)$ and $g_2(q)$ the two input vector fields, simple computations give

$$g_3(q)=[g_1,g_2](q)=\begin{bmatrix} cos\theta cos\phi \\ sin\theta sin\phi \\ sin\phi/l \\ 0 \end{bmatrix} \quad g_4(q)=[g_1,g_3](q)=\begin{bmatrix} -sin\theta/l \\ cos\theta/l \\ 0 \\ 0 \end{bmatrix}, \qquad (D.1.23)$$

both linearly independent from $g_1(q)$ and $g_2(q)$. Hence, the iterative procedure for building the accessibility distribution Δ_A ends with

$$dim\Delta_A=dim\Delta_3=\dim\ span\{g_1,g_2,g_3,g_4\}=4.$$

The front-wheel drive bicycle is controllable with degree of nonholonomy k=3, and constraints (4.17), (4.19) are nonholonomic.

If the bicycle with rear-wheel drive can be derived by noting that in this case the first two equations must coincide with those of the unicycle model (5.4). It is then sufficient to set $u_1=v/cos\phi$ to obtain

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\end{bmatrix}=\begin{bmatrix} cos\theta \\ sin\theta \\ \tan\phi/l \\ 0 \end{bmatrix}v+\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\omega, \qquad (D.1.24)$$

Where v is the driving velocity of the rear wheel.
In this case, one has

$$g_3(q) = [g_1, g_2](q) = \begin{bmatrix} 0 \\ 0 \\ -1 \\ \hline lcos^2\phi \\ 0 \end{bmatrix} g_4(q) = [g1, g3](q) = \begin{bmatrix} \frac{-sin\theta}{lcos^2\phi} \\ \frac{cos\theta}{lcos^2\phi} \\ 0 \\ 0 \end{bmatrix}, \qquad (D.1.25)$$

and it is linearly independent from $g_1(q)$ and $g_2(q)$. Hence, the rear-wheel drive bicycle is also controllable with degree of nonholonomy *k*=3.

Like the unicycle, the bicycle is also unstable in static conditions. [2]

To consider the kinematic model of a SSMR, it is assumed that the robot is placed on a plane surface with the inertial orthonormal basis (X_g, Y_g, Z_g), the robot has a local coordinate frame denoted as (x_l, y_l, z_l) assigned to the robot at its centre of mass. Considering the *Z*-coordinate constant. The robot moves on a plane with linear velocity in the local frame as $v = [v_x \; v_y \; 0]^T$ and rotates with an angular velocity $\omega = [0 \;\; 0 \;\; \omega]^T$.
Considering the state vector as $q = [X \; Y \; \theta]^T$, which describes the state vector generalized coordinates of the robot, position and orientation of the robot with respect to the inertial frame, then $\dot{q} = [\dot{X} \; \dot{Y} \; \dot{\theta}]^T$ denotes the vector of generalized velocities.
The variables $\dot{X}$ and $\dot{Y}$ are related to the coordinates of the local velocity vector (v_x, v_y):

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}. \qquad (D.1.26)$$

and due to the planar motion one can write $\dot{\theta} = \omega$.
So, we can get the general kinematic model equation as:

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}. \qquad (D.1.27)$$

The equation $(D.1.26)$ does not impose any restrictions on the SSMR plane movement, since it describes free-body kinematics only. To add these constraints, it is considered the analysis of the relationship between wheel velocities and local velocities.
Supposing that the *i*-th wheel rotates with an angular velocity $\omega_i(t)$, where *i*=1, 2, 3, 4, which can be seen as a control input. For simplicity the thickness of the wheel is neglected and is assumed to be in contact with the plane at point Pi as illustrated in Fig.4.3. In contrast to most wheeled vehicles, the lateral velocity of the SMRR, v_{iy} is generally non-zero. This property comes from the mechanical structure of the SSMR that makes lateral skidding necessary if the vehicle changes its orientation. Therefore, the wheels are tangent to the path only if $\omega = 0$, i.e., when the robot moves along a straight line.
Considering only a simplified case of the SSMR movement for which the longitudinal slip between the wheels and the surface can be neglected. Based on the Pacejka Magic Formula tire model, we get the relation:

$$v_{ix} = r_i \omega_i, \qquad (D.1.28)$$

where v_{ix} is the longitudinal component of the total velocity vector v_i of the *i*-th wheel expressed in the local frame and r_i denotes the effective rolling radius of that wheel.

To develop a kinematic model, it is necessary to take into consideration all wheels together.
The radius vectors $d_i = [d_{ix}\ d_{iy}]^T$ and $d_C = [d_{Cx}\ d_{Cy}]^T$ are defined with respect to the local frame from the instantaneous centre of rotation (ICR). Based on the geometry, the following expression can be deduced:

$$\frac{\| v_i \|}{\| d_i \|} = \frac{\| v \|}{\| d_C \|} = | \omega | \qquad (D.1.29)$$

where the symbol ||.|| denotes the Euclidean norm.
And in the detailed form,

$$\frac{v_{ix}}{-d_{iy}} = \frac{v_x}{-d_{Cy}} = \frac{v_{iy}}{d_{ix}} = \frac{v_y}{d_{Cx}} = \omega, \qquad (D.1.30)$$

Defining the coordinates of the ICR in the local frame as

$$ICR = (x_{ICR}, y_{ICR}) = \left(-d_{Cx}, -d_{Cy}\right) \qquad (D.1.31)$$

lead to the equivalence:

$$\frac{v_x}{y_{ICR}} = -\frac{v_y}{x_{ICR}} = \omega. \qquad (D.1.32)$$

The coordinates of vectors d_i satisfy the following relationships:

$$\begin{aligned} d_{1y} &= d_{2y} = d_{Cy} + c, \\ d_{3y} &= d_{4y} = d_{Cy} - c, \\ d_{1x} &= d_{4x} = d_{Cx} - a, \\ d_{2x} &= d_{3x} = d_{Cx} + b, \end{aligned} \qquad (D.1.33)$$

where a, b and c are positive kinematic parameters of the robot. After combining $(D.1.30)$ and $(D.1.33)$
one can get the relationships between wheel velocities:

$$\begin{aligned} v_L &= v_{1x} = v_{2x}, \\ v_R &= v_{3x} = v_{4x}, \\ v_F &= v_{2y} = v_{3y}, \\ v_B &= v_{1y} = v_{4y}, \end{aligned} \qquad (D.1.34)$$

where v_L and v_R denote the longitudinal coordinates of the left and right wheel velocities, v_F and v_B are the lateral coordinates of the velocities of the front and rear wheels, respectively.
Combining from $(D.1.30)$ to $(D.1.34)$ it is possible to obtain the transformation describing the relationship between the wheel velocities and the velocity of the robot:

$$\begin{bmatrix} v_L \\ v_R \\ v_F \\ v_B \end{bmatrix} = \begin{bmatrix} 1 & -c \\ 1 & c \\ 0 & -x_{ICR} + b \\ 0 & -x_{ICR} - a \end{bmatrix} \begin{bmatrix} v_x \\ \omega \end{bmatrix}. \qquad (D.1.35)$$

In accordance with $(D.1.28)$ and $(D.1.34)$, assuming that the effective radius is $r_i = r$ for each wheel, it can be written

$$\omega_w = \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \frac{1}{r}\begin{bmatrix} v_L \\ v_R \end{bmatrix}, \quad (D.1.36)$$

where ω_L and ω_R are the angular velocities of the left and right wheels, respectively.
Combining $(D.1.35)$ and $(D.1.36)$, it can be found the relation between the angular wheel velocities and the velocities of the robot:

$$\eta = \begin{bmatrix} v_x \\ \omega \end{bmatrix} = r\begin{bmatrix} \frac{\omega_L + \omega_R}{2} \\ \frac{-\omega_L + \omega_R}{2c} \end{bmatrix}, \quad (D.1.37)$$

where η is a new control input introduced at the kinematic level.
It can be noticed that the pair of velocities ω_L and ω_R can be treated as a control kinematic input signal as well as velocities v_x and ω. The accuracy of $(D.1.37)$ depends mostly on the longitudinal slip and can be valid only if this phenomenon is not dominant. In addition, the parameters r and c may be identified experimentally to ensure a high validity of the determination of the angular robot velocity with respect to the angular velocities of the wheels.

From $(D.1.33)$, the velocity constraint introduced in (Caracciolo et al., 1999) can be considered:

$$v_y + x_{ICR}\dot{\theta} = 0. \quad (D.1.38)$$

This equation is not integrable. Thus, it describes a nonholonomic constraint which can be rewritten in the Pfaffian form:

$$[-\sin\theta \;\; \cos\theta \;\; x_{ICR}]\left[\dot{X} \;\; \dot{Y} \;\; \dot{\theta}\right]^T = A(q)\dot{q} = 0, \quad (D.1.39)$$

since the generalized velocity$\dot{q}$us always in the null space of A, we can write

$$\dot{q} = S(q)\eta, \quad (D.1.40)$$

where

$$S^T(q)A^T(q) = 0 \quad (D.1.41)$$

and

$$S(q) = \begin{bmatrix} \cos\theta x_{ICR}\sin\theta \\ \sin\theta - x_{ICR}\cos\theta \\ 01 \end{bmatrix}. \quad (D.1.42)$$

It should be noted that since dim(η)=2 < dim(q)=3, the equation $(D.1.40)$ describes the kinematics of the robot, which is underactuated. Additionally, this is a nonholonomic system because of the constraint described by $(D.1.38)$.
From $(D.1.32)$ and $(D.1.35)$ it can be seen that the control of the v_y and v_{yi} velocity coordinates is not possible without the knowledge of the x_I-axis projection of the ICR. Therefore, considering the linear velocity v_x and the angular velocity w as control signals seems to have an advantage over other propositions where instead of w, the velocity v_y was used. [12]

E. Dynamic model definition

A consequence of nonholonomy in the mobile robots is that there is no exact linearization of the dynamic model via feedback. That's why the Lagrange formulation is used to obtain the dynamic model of an n-dimensional mechanical system subject to $k < n$ kinematic constraints, in the form $(D.1.4)$, which can be partially linearized via feedback.

The Lagrangian $\boldsymbol{\mathcal{L}}$ of the mechanical system is defined as the difference between its kinetic energy $\boldsymbol{T}$ and potential energy $\boldsymbol{U}$:

$$\mathcal{L}(q,\dot{q}) = T(q,\dot{q}) - U(q) = \frac{1}{2}\ \dot{q}^T B(q)\dot{q} - U(q), \qquad (E.1.1)$$

where *B(q)* is the symmetric and positive definite inertia matrix of the mechanical system. The Lagrange equations are

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right)^T = S(q)\tau + A(q)\lambda, \qquad (E.1.2)$$

where S(q) is an $(n \times m)$ matrix mapping the $m = n - k$ external inputs τ to generalized forces performing work on q, A(q) is the transpose of the $(k \times n)$ matrix characterizing the kinematic constraints, and$\lambda \in \mathbb{R}^m$is the vector of Lagrange multipliers. The term A(q)λ represents the vector of reaction forces at the generalized coordinate level. It has been assumed that the number of available inputs matches the number of DOFs (*full actuation*), that is, equal to the number *n* of generalized coordinates minus the number *k* of constraints.
Using $(E.1.1)$, $(E.1.2)$, the *dynamic model* of the constrained mechanical system is expressed as

$$B(q)\ddot{q} + n(q,\dot{q}) = S(q)\tau + A(q)\lambda \qquad (E.1.3)$$

$$A^T(q)\dot{q} = 0, \qquad (E.1.4)$$

where

$$n(q,\dot{q}) = \dot{B}(q)\dot{q} - \frac{1}{2}\left(\frac{\partial}{\partial q}(\dot{q}^T B(q)\dot{q})\right)^T + \left(\frac{\partial U(q)}{\partial q}\right)^T. \qquad (E.1.5)$$

Consider now a matrix *G(q)* whose columns are a basis for the null space of $A^T(q)$, so that $A^T(q)G(q) = 0$. One can replace the constraint given by $(E.1.4)$, with the kinematic model

$$\dot{q} = G(q)v = \sum_{i=1}^{m} g_i(q)v_i, \qquad (E.1.6)$$

Where $v \in \mathbb{R}^m$ is the vector of pseudo-velocities, the actual (generalized) velocities of the mechanical system are referred as $\dot{q}$; for example, in the case of unicycle the components of this vector are the driving velocity *v* and the steering velocity *ω*.
Moreover, the Lagrange multipliers in $(E.1.3)$ can be eliminated premultiplying both sides of the equation by $G^T(q)$. This leads to the reduced dynamic model

$$G^T(q)\big(B(q)\ddot{q} + n(q,\dot{q})\big) = G^T(q)S(q)\tau, \qquad (E.1.7)$$

a system of *m* differential equations.
Differentiation of $(E.1.6)$ with respect to time gives

$$\ddot{q} = \dot{G}(q)v + G(q)\dot{v}. \qquad (E.1.8)$$

Premultiplying $(E.1.8)$ by $G^T(q)B(q)$ and using the reduced dynamic model $(E.1.7)$, one obtains

$$M(q)\dot{v} + m(q,v) = G^T(q)S(q)\tau, \qquad (E.1.9)$$

where

$$M(q) = G^T(q)B(q)G(q) \qquad (E.1.10)$$

$$m(q,v) = G^T(q)B(q)\dot{G}(q)v + G^T(q)n(q,G(q)v), \qquad (E.1.11)$$

With $M(q)$ positive definite and

$$\dot{G}(q)v = \sum_{i=1}^{m} \left(v_i \frac{\partial g_i}{\partial q}(q) \right) G(q)v. \qquad (E.1.12)$$

This finally leads to the state-space reduced model

$$\dot{q} = G(q)v \qquad (E.1.13)$$
$$\dot{v} = M^{-1}(q)m(q,v) + M^{-1}(q)G^T(q)S(q)\tau, \qquad (E.1.14)$$

that represents in a compact form the kinematic and dynamic models of the constrained system as a set of $n + m$ differential equations.
Suppose that we have the following assumption on the 'control availability'

$$det(G^T(q)S(q)) \neq 0. \qquad (E.1.15)$$

The it is possible to perform a partial linearization via feedback of $(E.1.13)$ and $(E.1.14)$ by letting

$$\tau = (G^T(q)S(q))^{-1}(M(q)a + m(q,v)), \qquad (E.1.16)$$

where $a \in \mathbb{R}^m$ is the pseudo-acceleration vector. The resulting system is

$$\dot{q} = G(q)v \qquad (E.1.17)$$
$$\dot{v} = a. \qquad (E.1.18)$$

Note that the structure of this system: the first *n* equations are the kinematic model, of which the last integrator *m* on the input channels, are a dynamic extension. If the system is unconstrained and fully actuated, it is $G(q) = S(q) = I_n$; then, the feedback law $(E.1.16)$ simply reduces to an inverse dynamics control, and correspondingly the closed-loop system is equivalent to *n* decoupled double integrators.
The implementation of the feedback control $(E.1.16)$ requires the measurement of *v*, and it could not be available, but the pseudo velocities can be computed via the kinematic model as

$$v = G^{\dagger}(q)\dot{q} = \left(G^T(q)G(q)\right)^{-1} G^T(q)\dot{q}, \qquad (E.1.19)$$

provided that q and $\dot{q}$ are measured.
By defining the state $x = (q, v) \in \mathbb{R}^{n+m}$ and the input $u = a \in \mathbb{R}^m$, system $(E.1.17)$ and $(E.1.18)$ can be expressed as

$$\dot{x} = f(x) + G(x)u = \begin{bmatrix} G(q)v \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I_m \end{bmatrix} u, \qquad (E.1.20)$$

i.e., a nonlinear system with drift also known as the second-order kinematic model of the constrained mechanical system. The equation $(E.1.17)$ is also called first-order kinematic model. Its controllability guarantees the controllability of the system $(E.1.20)$.
In brief, in nonholonomic mechanical systems, it is possible to 'cancel' the dynamic effects via nonlinear state feedback, provided that the dynamic parameters are exactly known and the complete state of the system, the generalized coordinates and velocities q and $\dot{q}$, are measured.
Under these assumptions, the control problem can be addressed at a pseudo-velocity level, like choosing *v* in such a way that the kinematic model

$$\dot{q} = G(q)v \qquad (E.1.21)$$

behaves as desired. From v, it is possible to derive the actual control inputs at the generalized force level through $(E.1.16)$. Since $a = \dot{v}$ appears in this equation, the pseudo-velocities v must be differentiable with respect to time. [2]

The components described above are:
$G(q)$ input vectors matrix
u is the input vector
q is the generalized state vector
I is the identity matrix
n mechanical system, number of generalized coordinates
k kinematic constraints
m difference between mechanical system and kinematic constrains, thus DOF
λ Lagrange multipliers
$S(q)$ maps the external inputs τ to generalized forces performing work on q
$A(q)$ transpose of the matrix characterizing the kinematic constrains
$\mathcal{L}$ Lagrangian
$B(q)$ Inertia matrix

Due to symmetry along the longitudinal midline, we obtain

$$N_1 = N_4 = \frac{b}{2(a+b)}mg,$$
$$N_2 = N_3 = \frac{a}{2(a+b)}mg. \qquad (E.1.24)$$

The vector F_{si} results from the rolling resistant moment τ_{ri} and the lateral reactive force F_{li}. These reactive forces can be regarded as friction ones, but it is important to note that friction modelling is complex since it is highly nonlinear and depends on many variables.
Here we are going to consider a simplified approximation describing the friction F_f depending on the linear velocity σ, the force perpendicular to the surface N, the Coulomb coefficient μ_c and viscous friction μ_v.
It can be written as:

$$F_f(\sigma) = \mu_c N sgn(\sigma) + \mu_v \sigma, \qquad (E.1.25)$$

Since for the SSMR the velocity σ is relatively low, especially during lateral slippage, the relation $\mu_c N \gg |\mu_v \sigma|$ is valid, which allows to neglect the term $\mu_v\sigma$ to simplify the model. The function $(E.1.25)$ is not smooth when the velocity σ equals to zero, due to the sign function sgn(σ). Hence, it is not differentiable at σ=0. So, we need an approximation of this function like:

$$\widehat{sgn(\sigma)} = \frac{2}{\pi}\arctan(k_s\sigma), \qquad (E.1.26)$$

where k_s>>1 is a constant which determines the approximation accuracy according to the relation

$$\lim_{k_s\to\infty} \frac{2}{\pi}\arctan(k_s\sigma) = sgn(x). \qquad (E.1.27)$$

So, the friction forces for each wheel are longitudinal and lateral and depend on the coefficients of each force, mass, gravity acceleration and the sign of the linear speeds.

$$F_{li} = \mu_{lci} mg\widehat{sgn}(v_{yi}), \quad (E.1.28)$$
$$F_{si} = \mu_{sci} mg\widehat{\text{sgn}}(v_{xi}), \quad (E.1.29)$$

where μ_{lci} and μ_{sci} denote the coefficients of the lateral and longitudinal forces.
By means of Lagrange-Euler formula with the Lagrange multipliers including the nonholonomic constraint (D.1.38) it's possible to find the dynamic equation of the robot.
Assuming the potential energy *PE(q)*=0 due to the planar motion.

The Lagrangian $\mathcal{L}$ of the system equals the kinetic energy:

$$\mathcal{L}(q, \dot{q}) = T(q, \dot{q}) \quad (E.1.30)$$

Considering the kinetic energy of the vehicle and neglecting the energy of rotating wheels:

$$T = \frac{1}{2} m v^T v + \frac{1}{2} I \omega^2, \quad (E.1.31)$$

where *m* is the mass of the robot and *I* is the moment of inertia of the robot about the centre of mass.
Assuming homogeneous mass distribution:

$$v^T v = v_x^2 + v_y^2 = \dot{X}^2 + \dot{Y}^2 \quad (E.1.32)$$
$$\omega = \dot{\vartheta} \quad (E.1.33)$$
$$T = \frac{1}{2} m\left(\dot{X}^2 + \dot{Y}^2\right) + \frac{1}{2} I \dot{\theta}^2. \quad (E.1.34)$$

Calculating the partial derivative of kinetic energy and its time-derivative to get the inertial forces which causes the dissipation of energy.

$$\frac{d}{dt}\left(\frac{\partial E_k}{\partial \dot{q}}\right) = \begin{bmatrix} m\ddot{X} \\ m\ddot{Y} \\ I\ddot{\theta} \end{bmatrix} = M\ddot{q}, \quad (E.1.35)$$

Where the mass matrix *M* is

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}. \quad (E.1.36)$$

So, the active forces generated by the actuators can be expressed in the inertial frame,

$$F_{rx}(\dot{q}) = \cos\theta \sum_{i=1}^{4} F_{si}(v_{xi}) - \sin\theta \sum_{i=1}^{4} F_{li}(v_{yi}), \quad (E.1.37)$$
$$F_{ry}(\dot{q}) = \sin\theta \sum_{i=1}^{4} F_{si}(v_{xi}) + \cos\theta \sum_{i=1}^{4} F_{li}(v_{yi}). \quad (E.1.38)$$

The resistant moment around the centre of mass M_r can be obtained as

$$M_r(\dot{q}) = -a \sum_{i=1,4} F_{li}(v_{yi}) + b \sum_{i=2,3} F_{li}(v_{yi}) + c\left[-\sum_{i=1,2} F_{si}(v_{xi}) + \sum_{i=3,4} F_{si}(v_{xi})\right]. \quad (E.1.39)$$

To define generalized resistive forces, it is used the vector

$$R(\dot{q}) = \left[F_{rx}(\dot{q})F_{ry}(\dot{q})M_r(\dot{q})\right]^T \qquad (E.1.40)$$

The active forces and torque around the centre of mass generated by the actuators, can be expressed in torque equations considering all the wheels have the same radius:

$$F_x = cos\vartheta \sum_{i=1}^{4} F_i = \frac{1}{r} cos\vartheta \sum_{i=1}^{4} \tau_i \qquad (E.1.41)$$

$$F_y = sin\vartheta \sum_{i=1}^{4} F_i = \frac{1}{r} sin\vartheta \sum_{i=1}^{4} \tau_i \qquad (E.1.42)$$

$$M = c(-F_1 - F_2 + F_3 + F_4) = \frac{c}{r}(-\tau_1 - \tau_2 + \tau_3 + \tau_4). \qquad (E.1.43)$$

As a consequence, the vector *F* of active forces has the following form:

$$F = \begin{bmatrix} F_x \\ F_y \\ M \end{bmatrix} = \frac{1}{r} \begin{bmatrix} \cos\theta \sum_{i=1}^{4} \tau_i \\ \sin\theta \sum_{i=1}^{4} \tau_i \\ c(-\tau_1 - \tau_2 + \tau_3 + \tau_4) \end{bmatrix}. \qquad (E.1.44)$$

The torque control input is divided in left and right torques produced by the wheels of the vehicle

$$\tau = \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} = \begin{bmatrix} \tau_1 + \tau_2 \\ \tau_3 + \tau_4 \end{bmatrix}, \qquad (E.1.45)$$

Combining ($E.1.44$) and ($E.1.45$), we get

$$F = B(q)\tau, \qquad (E.1.46)$$

where B is the input transformation matrix defined as

$$B(q) = \frac{1}{r} \begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ -c & c \end{bmatrix}. \qquad (E.1.47)$$

The dynamic model describing the dynamics of a free body only and does not include the non-holonomic constraint of the wheels.

$$M(q)\ddot{q} + R(\dot{q}) = B(q)\tau \qquad (E.1.48)$$

In case one need to add such constraint, the equation will be integrated with Lagrange multipliers λ.

$$M(q)\ddot{q} + R(\dot{q}) = B(q)\tau + A^T(q)\lambda. \qquad (E.1.49)$$

For control purposes it would be more suitable to express it in terms of the internal velocity vector η. Therefore, the equation is multiplied from the left by $S^T(q)$, which results in

$$S^T(q)M(q)\ddot{q} + S^T(q)R(\dot{q}) = S(q)^T B(q)\tau + S^T(q)A^T(q)\lambda. \qquad (E.1.50)$$

After taking the time derivative of (4.39), we obtain

$$\ddot{q} = \dot{S}(q)\eta + S(q)\dot{\eta}. \qquad (E.1.51)$$

[12]

F. Path and Timing Law

Whenever one wants to plan a trajectory *q(t)*, for $t \in [t_i, t_f]$, which leads the mobile robot from an initial configuration $q(t_i) = q_i$ to a final configuration $q(t_f) = q_f$ in the absence of obstacles. The trajectory $q(t)$ can be broken down into a geometric path $q(s)$, with $\frac{dq(s)}{ds} \neq 0$ for any value of *s*, and $s(t_f) = s_f$ in a monotonic fashion, i.e., with $\dot{s}(t) \geqslant 0,\ for\ t \in [t_i, t_f]$.
A possible choice for s is the arc length along the path.
If we consider $s_i = 0$ and $s_f = L$, where *L* is the length of the path.
The space-time separation implies that

$$\dot{q} = \frac{dq}{dt} = \frac{dq}{ds}\dot{s} = q'\dot{s}, \qquad (F.1.1)$$

where the prime symbol denotes differentiation with respect to *s*. The generalized velocity vector is then obtained as the product of the vector *q'*, which is directed as the tangent to the path in configuration space, by the scalar $\dot{s}$, that varies its modulus. Note that the vector $q' = [x'\ \ y']^T \in \mathbb{R}^2$ is directed as the tangent to the Cartesian path, and has unit norm if *s* is the Cartesian arc length. The nonholonomic constraints of the form (D.1.4) can be rewritten as

$$A(q)\dot{q} = A(q)q'\dot{s} = 0 \qquad (F.1.2)$$

If $\dot{s}(t) > 0$, for $t \in [t_i, t_f]$, one has

$$A(q)q' = 0. \qquad (F.1.3)$$

This condition, that must be verified at all points by the tangent vector on the configurations space path, characterizes the notion of geometric path admissibility induced by the kinematic constraint (D.1.4) that actually affects generalized velocities. Geometrically admissible paths can be explicitly defined as the solutions of the nonlinear system

$$q' = G(q)\tilde{u}\,, \qquad (F.1.4)$$

Where $\tilde{u}$ is a vector of geometric inputs that are related to the velocity inputs *u* by the relationship $u(t) = \tilde{u}\,(s)\dot{s}(t)$. Once the geometric inputs $\tilde{u}\,(s)$ are assigned for $s \in [s_i, s_f]$, the path of the robot in configuration space is uniquely determined. The choice of a timing law $s = s(t)$, for $t \in [t_i, t_f]$, will then identify a particular trajectory along this path.

F.2 Flat Outputs

Kinematic models of mobile robots exhibit a property known as *differential flatness*.
A nonlinear dynamic system $\dot{x} = f(x) + G(x)u$ is differential flat if there exists a set of outputs *y*, called flat outputs, such that the state *x* and the control inputs *u* can be expressed algebraically as a function of *y* and its time derivatives up to a certain order:

$$x = x\left(y, \dot{y}, \ddot{y}, \ldots, y^{(r)}\right) \qquad (F.1.5)$$
$$u = u\left(y, \dot{y}, \ddot{y}, \ldots, y^{(r)}\right). \qquad (F.1.6)$$

As a consequence, once an output trajectory is assigned for *y*, the associated trajectory of the state *x* and history of control inputs *u* are uniquely determined.
Since the Cartesian coordinates are indeed flat outputs, given the Cartesian path $(x(s)y(s))$, the associated state trajectory is $q(s) = [x(s)y(s)\vartheta(s)]^T$ where $\vartheta(s) = atan2(y'(s), x'(s)) + k\pi$ with (*k=0*) for forward motion and (*k=1*) for backward motion.

F.3 Path Planning

Whenever a mobile robot admits a set of flat outputs *y*, there may be exploited to solve planning problems efficiently. One may use any interpolation scheme to plan the path of *y* in such a way as to satisfy the appropriate boundary conditions. The evolution of the other configuration variables, together with the associated control inputs, can then be computed algebraically from *y(s)*. The resulting configuration space path will automatically satisfy the nonholonomic constraints (F.1.3).

There are three ways to do a path plan for a mobile robot:

Cubic Cartesian polynomials, based in the interpolation of the initial values x_i, y_i and the final values x_f, y_f of the flat outputs *x*, *y*. Letting $s_i = 0$ and $s_f = 1$, using the cubic polynomials:

$$x(s) = s^3 x_f - (s-1)^3 x_i + \alpha_x s^2(s-1) + \beta_x s(s-1)^2 \qquad (F.1.7)$$
$$y(s) = s^3 y_f - (s-1)^3 y_i + \alpha_y s^2(s-1) + \beta_y s(s-1)^2 \qquad (F.1.8)$$

which automatically satisfy the boundary conditions on *x* and *y*.
The orientation at each point is related to x' and y' and the additional boundary conditions are:

$$x'(0) = k_i \cos(\vartheta_i) \qquad x'(1) = k_f \cos\left(\vartheta_f\right) \qquad (F.1.9)$$
$$y'(0) = k_i \sin(\vartheta_i) \qquad y'(1) = k_f \sin\left(\vartheta_f\right) \qquad (F.1.10)$$

The parameters $k_i \neq 0, k_f \neq 0$ are free parameters that must have the same sign, so that the robot arrives in q_f with the same kind of motion (forward or backwards) with which it leaves q_i, and since *x(s)* and *y(s)* are cubic polynomials, the Cartesian path does not contain motion inversions in general.
For $k_i = k_f = k > 0$, one obtains α and β of the form:

$$\alpha_x = k\cos\left(\vartheta_f\right) - 3\mathrm{x}_f \qquad \alpha_y = k\sin\left(\vartheta_f\right) - 3\mathrm{y}_f \qquad (F.1.11)$$
$$\beta_x = k\cos(\vartheta_i) + 3\mathrm{x}_i \qquad \beta_y = k\sin(\vartheta_i) + 3\mathrm{y}_i \qquad (F.1.12)$$

Chained form, the path is planned in the chained form coordinates *z*. First it is necessary to compute the initial and final values z_i and z_f that correspond to q_i and q_f, by using the change of coordinates.
Then it is enough to interpolate the initial and final values of z_1 and z_3 (the flat outputs) with the appropriate boundary conditions on the remaining variable $z_2 = z_3'/z_1'$.
It's possible to adopt a cubic polynomial to solve the problem. As an alternative, one may use polynomials of different degree for *x* and *y* in order to reduce the number of unknown coefficients to be computed.
Once the path has been planned for the chained form, the path *q(s)* in the original coordinates and

the associated geometric inputs $\tilde{u}(s)$ are reconstructed by inverting the change of coordinates and of inputs.

Parameterized inputs, it consists on writing the inputs, rather than the path, in parameterized form, and computing the value of the parameters so as to drive the robot from q_i to q_f. It is convenient to work on the chained form, whose equations are easily integrable in closed form under appropriate inputs. This method does not make explicit use of the flat outputs, but relies on the closed-form integrability of the chained form, whose existence is equivalent to differential flatness. [2]

G. Microcontroller ARDUINO

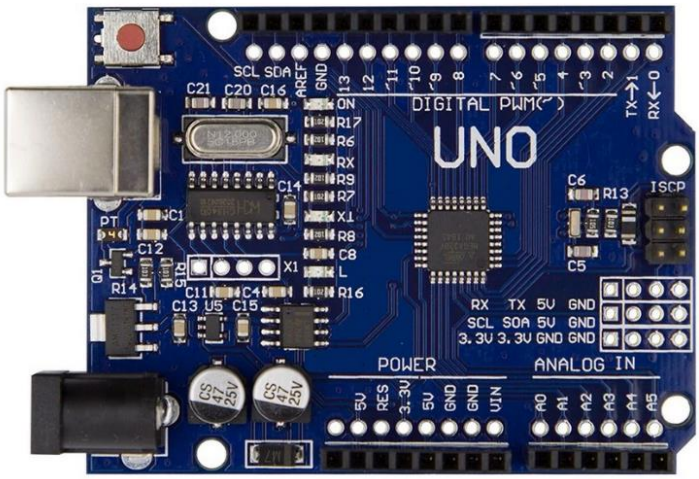


Figure 7.1 Arduino Uno board

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. One can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so one use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Some reasons to use Arduino are:

- It is inexpensive, compared to other microcontroller platforms.
- It is a cross-platform, thus it can be run on Windows, Macintosh OSX and Linux operating systems.

- It is simple and clear programming environment, the IDE is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.
- It is open source and has an extensible software that can be expanded through C++ libraries.
- It has an extensible hardware as well, since the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. [13]

G.1 Arduino Uno

Arduino Uno is a microcontroller board based on the Atmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, at 16MHz ceramic resonator, a USB connection, a power jack, and ICSP header and a reset button. It contains everything needed to support the microcontroller.
It has its own Integrated Development environment, IDE, where one can write their own programs, called sketch, which will be executed by the microcontroller.
The development board is open source in both Hardware and Software. It is cheap, easy to retrieve and can be expanded by means of shields.

Arduino can be power up either using power supplied from the computer via a USB cable and/or by using external power sources like batteries, wall-warts or solar panels. [14]

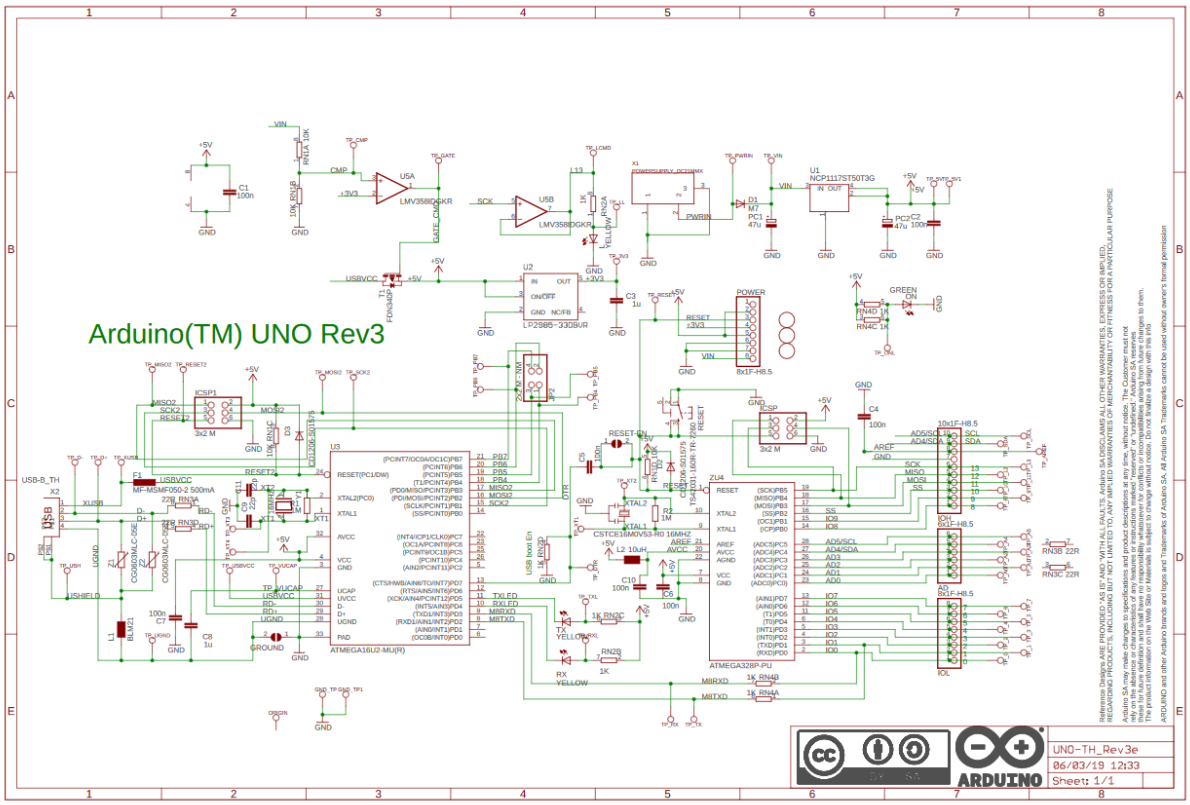


Figure 7.2 Schematics of Arduino Uno rev.3 board