Politecnico di Torino

Master course in Mechatronic Engineering

Master thesis project

Dependability in a Mission Critical Scenario: D.I.A.N.A. Mars Rover System Analysis

Supervisors:
Massimo Violante
Jacopo Sini

Author:
Andrea Passarino

Accademic year 2020

DIANA

Contents

Part I
Introduction

"Space: the final frontier. These are the voyages of the starship Enterprise. Its five-year mission: to explore strange new worlds. To seek out new life and new civilizations. To boldly go where no man has gone before!"

This sentence starts the exploration of the starship Enterprise in the Star Trek series created in 1966. This words described a journey in a fantasy universe but they represent the will of the humanity to known better the heavenly bodies around us.
In that period borne the concept of "space exploration" because humanity can be able to send probe outside the Earth atmosphere.
Many missions started since the '60 in order to analyse heavenly bodies of our solar system but they produced few results mainly caused by the limit of the technology. The first mission was "Moon 2", sent by the RKA (Russian Space Agency) in 1959 to analyse and take pictures of the Moon surface. A few years later borne the Apollo program[1] of NASA (United State space agency) that brought humans on the surface of our satellite for the first time on 20 July 1969.
Parallel with the exploration of the Moon, URSS and USA attempted to orbit around Mars. Unfortunately, the first six mission failed, only in 1964 "Mariner 4"[2], a USA probe, sent some photos of the martian surface.

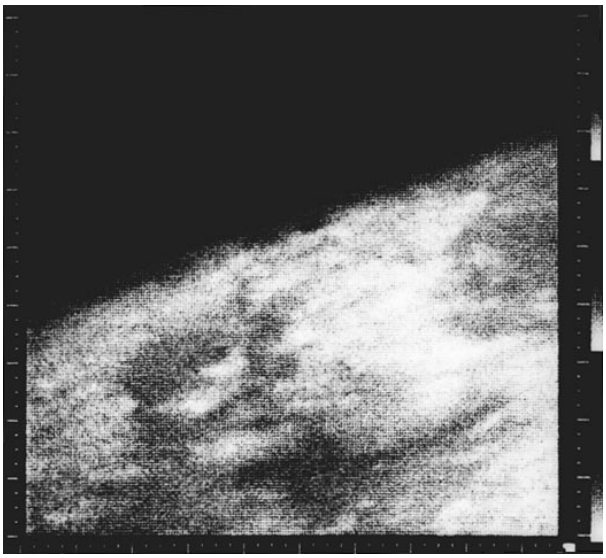


Figure 1: First image of Mars given by Mariner 4 [NASA]

Since this moment the interest to the Red planet is not reduced, rather other space agencies joined to the exploration.
This was caused because Mars is very similar to Earth and knowing its evolution and how it is coming can improve the knowledge of our planet. Also, Mars respect to the Moon has an atmosphere that can protect possible colonists from the solar radiation and makes the temperature more "warm" (Mars temperature is between -140 Celsius and +20 celsius)[3].
Another important factor for possible colonization is the presence of the water in the solid or liquid states. Due to the low temperature and pressure (about 6 mPa, on Earth it is about 1000 mPa) on Mars water can be only in solid state and it is present mainly in the northern hemisphere[4].

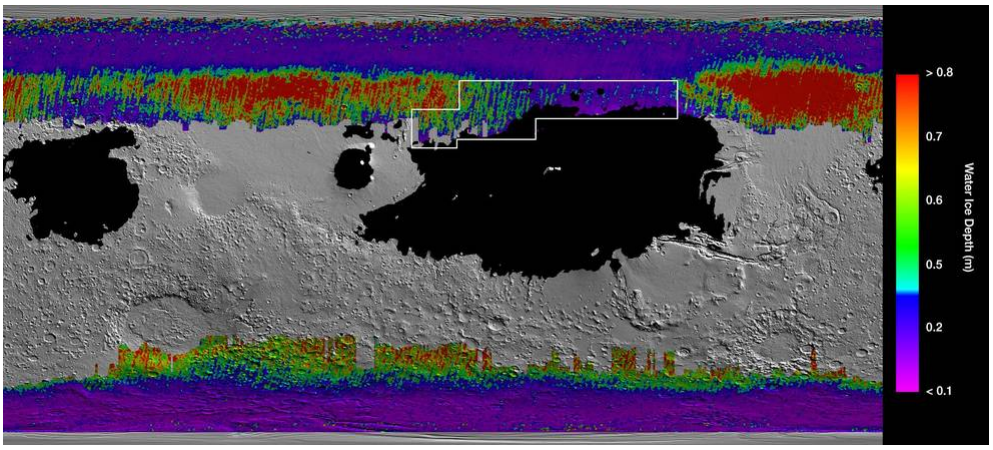


Figure 2: Distribution of the ice in the Mars surface [NASA]

In the previous figure it is represented the distribution of the ice on Mars, colors represent how much it is deep in the terrain and it is the merge of the data given by *Mars Reconnaissance Orbiter* (NASA) and *Mars Odyssey* (NASA).

The probes used to analyse a heavenly body can be divided into three types: orbiter, lander and rover.
Orbiter is used to keep information about the atmosphere and surface (if the atmosphere is quite clear) from the space. This type of probes, like its name suggests, orbits around the planet without lands on the surface. Same working orbiter around Mars are:

- *Mars Odyssey*[5] (NASA,2001)
- *Mars Express*[6, 7] (ESA,2003)
- *Mars Reconnaissance Orbiter*[8] (NASA,2005)

- *Mars Orbiter Mission*[9] (ISRO,2013)
- *MAVEN*[10] (NASA,2013)
- *ExoMars*[11] (ESA,2016)

The lander is a static probe that lands on the heavenly body surface but it can't be able to move itself. It can analyses the atmosphere, temperature on the surface and it can operate some experiments on the terrain. Some lander on Mars were:

- *Beagle 2*[12] (NASA,2003)
- *Phoenix Mars lander*[13, 14, 15] (NASA,2007)
- *Schiaparelli*[16, 17] (ESA,2016)
- *InSight*[18, 19] (NASA,2018)

The rovers are probes that operate on the surface of the heavenly body and thanks to its mobility system can travel in order to explore not only the landing site. Also, this type of probes is equipped with an arm and/or a driller in order to bring rocks or a portion of terrain that can be analysed by the scientific system on its. Same martian rover are:

- *Spirit*[20] (NASA,2003)
- *Opportunity*[21] (NASA,2003)
- *Curiosity*[22, 23] (NASA,2011)

The first rover *Lunokhod 1*[24, 25] was built by RKA (Russian space agency) and it landed on the surface of the Moon in 1970.

Since its creation, the rover, is a good substituted of the human during the exploration of an unknown heavenly body because it is able to travel on the surface. Many researches analyse the best design of the rover(functions, autonomy ex.)[26, 27] and in particular of mobility system[28, 29, 30]. The most used are the design of 6 traction wheels with 4 (NASA's Curiosity rover) or 6 (ESA's Exomars rover[31, 32, 33]) of them steerable. The most used suspension system is NASA's Rocker-bogie system.
The Rocker-bogie system is a passive suspension divided in two part, the rocker that is the bigger and the bogie that is the smaller[34, 35, 36].

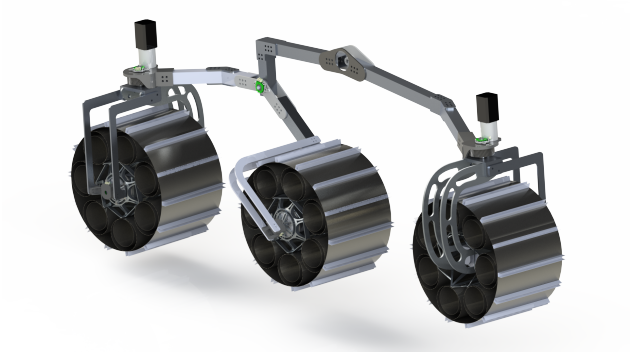


Figure 3: Rocker-Bogie structure (right view)

The parts are connected to each other with a passive rotational joint and the rocker is connected to the chassis with another passive rotational joint. With the terminology "passive" it is meaning that there is any motor that actuated the joint. In other words, the structure follows the terrain shape without any correction given by the suspension.

In order to connect the dynamics of the two legs of the rover, it is possible to put a torsion bar or a differential system between the slewing ring of the rockers mounted in the chassis.

Other important parts during the design of the rover's mobility system are the mobility controller and the operational safe.
The mobility controller must avoid the slipping of the wheels to have a desired dynamic of the rover and to reduce the degradation of the wheels thread.[37, 38, 39, 40]
Also off-road terrain the Rocker-bogie system can go in a not proper configuration, called singularity of the rocker-bogie system, when the rover overpass an obstacle or a hill.[41] When the system goes in singularity the front wheel of the leg, that is overpassing the obstacle, reduces its velocity, then the central wheel goes upwards and rotate the bogie joint. After that, if the central wheel will collide with the structure of the rocker and the two rockers are connected by a torsion bar the rover over tips. The mobility controller must avoid also this situation. The operational safe must be guaranteed otherwise the rover can be damaged. In order to guarantee its a failure analysis is needed and also a mitigation of them.[42]

In this thesis thanks to the collaboration with the student team D.I.A.N.A of Polytechnic of Turin, that design and build rover aimed to support possible martian colonists, it is underlined all of the previous design problems in real project work.

The aim of this thesis work is to improve the driveability of the rover by the user and to increase its dependability in a possible mission-critical scenarios. For what concern the drivability a mobility controller is developed. Thanks to this controller, the driver controls the rover as if it is on its Center of Mass.
In order to increase the dependability of the rover a Failure Mode and Effect Analysis (FMEA)[43, 44, 45, 46] is applied to the rover design. After underlined the possible failures and their effect it is important to try a diagnostic and mitigation, Failure Mode Effect and Diagnostic Analysis (FMEDA)[47, 48, 49, 50] of them to avoid incorrect behaviour of the rover.
The FMEDA is applied only on Mobility system even if the FMEA is applied on the entire rover.
After that we have obtained all the reliability requirements needed by the system, we developed the mobility system software to reach them. Of course, we need to validate against the requirement, hence we decided to adopt a simulation-based approach. Thanks to CoppeliaSim we obtained a virtual rover, where we tested the controller and diagnostic routines and verified that they are able to work properly.

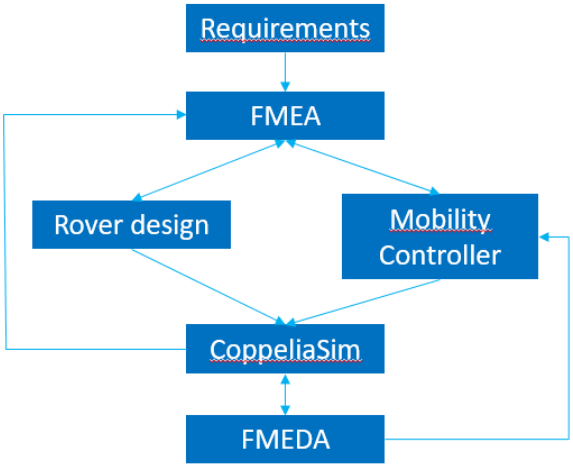


Figure 4: Simple thesis work-flow chart

The thesis structure is the following:

- Part 1: Team D.I.A.N.A.. It is a little paragraph about the history of the student team and its works.

- Part 2: Ardito requirements. In this part are underlined all the design requirements takes by the international competitions in which the team participated.
- Part 3: Failure Mode Effect Analysis. It is the complete analysis of the rover common design between the competitions.
- Part 4: Mobility Controller. In this part is underlined the structure of the Mobility controller algorithm, the model used and the simulation results.
- Part 5: Failure Mode Effect and Diagnostic Analysis. In this part is showed the main failure of the Mobility system, their detection and mitigation structure algorithm and the results from the co-simulation with Coppeliasim.
- Part 6: Results. It is a recap of the results of the previous part and the possible improvement of the current work.

Part II
Team D.I.A.N.A

Team D.I.A.N.A, the acronym of Ducti Ingenio Accipimus Naturam Astrorum, is a student team of Polytechnic of Turin. It was created in 2008, by professor Giancarlo Genta, in order to participate at the Google Lunar X Prize, a competition opened only to private teams. The goal of the contest was the launch of rover on the Moon that send images and data of the surface to the Earth control station.
For this competition, team D.I.A.N.A collaborated with student teams of other italian universities in order to create its first rover called A.M.A.L.I.A., the acronym of *Ascensio Machinae Ad Lunam Italica Arte*.
The main characteristics of Amalia rover were:

- four elastic wheels;
- active suspension;
- solar panels;
- TOF camera (3D Lidar);

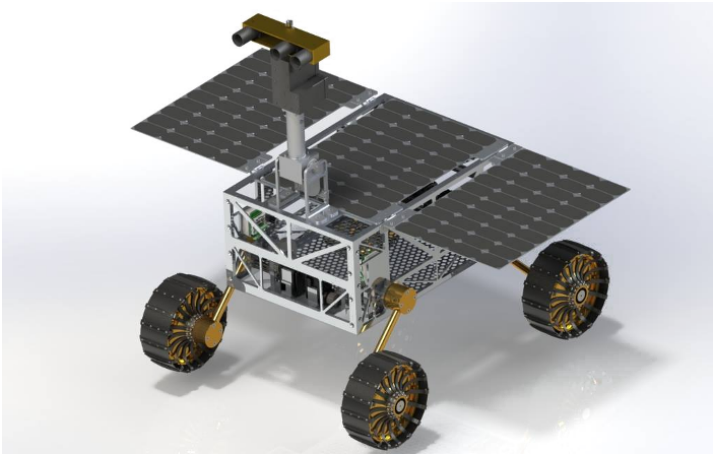


Figure 5: A.M.A.L.I.A render

This project was developed until 2016. At that time it was decided to change the team's research goal. A.M.A.L.I.A was built as an exploration rover, in other words, the rover will be alone on the planet and it will be controlled by Earth. It had unlimited power supply thanks to the solar panels and it could not bring objects because it didn't have an arm with a gripper.

The new goal, that is already followed, is the building of a support rover for a possible space colonist. Many competitions in the worlds are focused on this goal, for example, European Rover Challenge (ERC) or University Rover Challenge (URC). In this case, the tasks have a limited amount of time, for this reason the solar panels were removed and it was improved the capacity of batteries power supply. Also, it's was added a 6DOF (degrees of freedom) robotic arm with a gripper or a scoop to bring an object or to take away a portion of terrain.

The new designed rover was bigger respect A.M.A.L.I.A. because the arm needed a bigger distance between the wheels, otherwise, the rover can tip over when the arm was totally extended. It is decided to change the type of suspensions; active suspensions were powerful but the design and control were too complex for the size of the rover, rocker-bogie was more adept. Rocker-bogie system is a passive suspension that can have independent or dependent legs movement.

Our second rover, designed and produced between 2016 and 2018, is T0R0. Its main characteristics are:

- six wheels;
- rocker-bogie system with torsion bar;
- 6DOF robotic arm;
- slip-steering mobility.

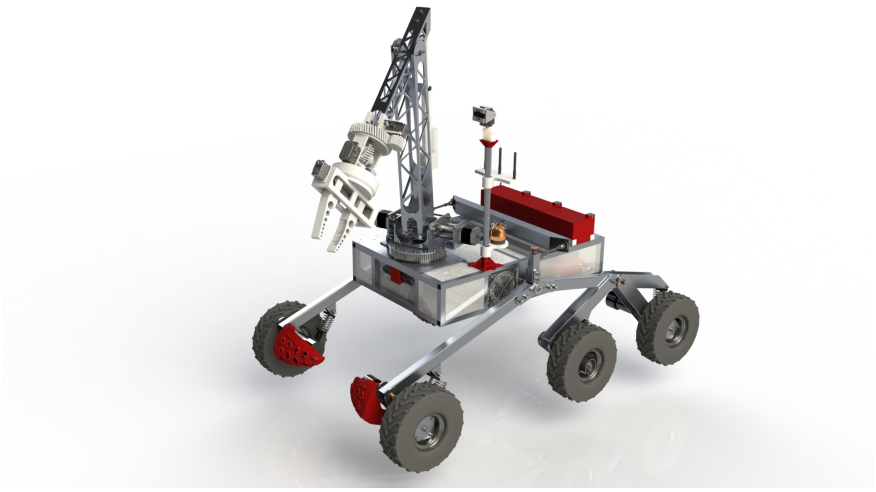


Figure 6: T0R0 render

After the presence on ERC competition in 2018, the work to improve the rover didn't stop and only in one year it was built a new rover, Trinity. Trinity's main difference from T0R0 is the steering system and elastic wheels. During the competition, it was highlight that slip-steering was less precise than steering wheels to follow the desired trajectory. Then it is decided to make the rear and front wheels steerable.

Figure 7: Trinity render

The least rover, Ardito, is an improvement of Trinity. With Ardito design, lateral tip over is avoided thanks to an improvement of wheelbase. Also both steering and wheels' motors are changed to increase the torque and their control precision.
The weight of the rover is the same even if the volume of the chassis is increased because the arm structure is optimized.

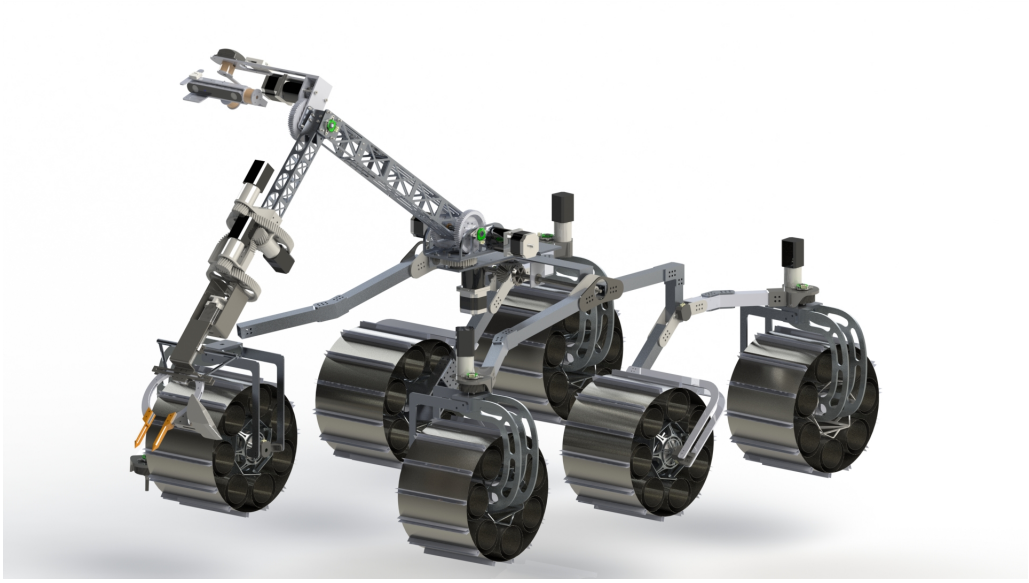


Figure 8: Ardito render (without payload)

Unfortunately, in 2020 Ardito is not built caused by Covid-19 lockdown and also URC competition was cancelled.
In this period the team worked on a new competition type, Indian Rover Design Challenge (IRDC), that is based on the design of an assistance rover with the capability to work on Mars surface for 15 sol (about 15 days). This competition was very important for our project because underlined the space-grade structure of Ardito rover and also the modularity of it. In fact, the IRDC designed rover was our rover Ardito changed to work with the temperature and atmosphere characteristics of Mars. It is added 2 powerful batteries, a thermal control system, a solar charge system and an improved URC scientific system.

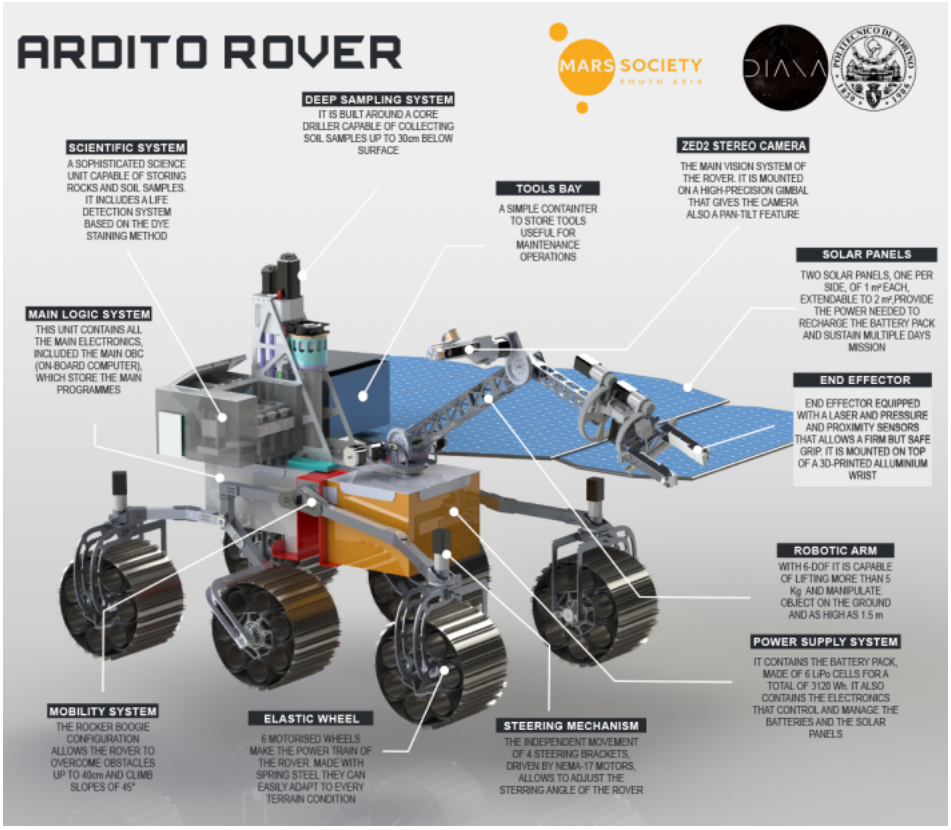


Figure 9: Ardito IRDC render

Part III
Ardito requirement

The requirements and constraints of our project are strictly depending on the competition rules. But in order to design a good project we take a look not only of these requirements but also the characteristics of working space exploration probes.
With this approach, it is decided to build a rover bigger and more complex respect the other rivals because it guaranteed better stability and also an improvement of the capability to do the tasks. For example, a bigger wheels base improves stability, consequently, the arm can be more extended and it can have a bigger operation range.

This year we are interested to participate in three different challenges: European Rover Challenge (ERC), University Rover Challenge (URC) and Indian Rover Design Challenge (IRDC). Following the common requirements between ERC and URC.

1 General requirements

- *Rover shall be a stand-alone platform*:
  Our approach was to design the rover self-sufficient and as modular as possible. Each subsystem was designed to be easily interchangeable and maintainable.

- *Rover suggested weight is 50 Kg in every configuration:*
  Unfortunately, due to its dimension, our rover in some configuration exceeds the suggested weight.

- *Rover maximum dimension must be under 1.2 x 1.2 m:*
  Our design takes care of this limit then the maximum size of our rover is about 1 x 1 m.

- *Rover shall utilize power and propulsion systems that are applicable to operations on Mars. Also, air-breathing system and any power or propulsion system that ingest ambient air are not permitted:*
  Our rover has an electric power supply dimensioned to guaranteed energy for all the time of the task. The power supply can change depending on the competition. No solar panels are equipped.

- *Rover maximum speed can not exceed 0.5 m/s (1.8 Km/h) for ERC:*
  The maximum speed bounded is guaranteed by the mobility controller. It's decided to impose a limit of 1 m/s of the rover velocity for URC

otherwise it can be unstable. The maximum speed bound is guaranteed by the mobility controller. It's decided to impose a limit of 1 m/s of the rover velocity for URC otherwise, it can become unstable. Also, it adds a signal thanks to which it is possible to set the maximum velocity equal to one of the two limits. The limit of the velocity can be changed during the work to have a slower dynamics during URC.

- *Rover shall operate between +10 and +30 Celsius degree:*
  Our rover must operate in Earth condition, then we didn't analyse the problem of thermal control. The only critical condition was the design of a cooling system for the logics and an insulation system in order to block dust particles or foggy.

2 Autonomous operation requirements

- *Rover autonomy is highly recommended:*
  Our rover can work in autonomy or it can help the operator during every task, increasing the situational awareness.

- *Controllers can not be touched and telemetry data should be monitored during autonomous operation attempt:*
  When the rover is in autonomous mode it sends telemetry to the base station, therefore the user has time by time the situation under control.

- *In autonomous mode, extra safety precaution should be taken:*
  When the rover is in autonomous mode specific diagnostic software monitors the telemetry and in case of critical failure, it takes the rover in a safe state and returns the command to the user.

- *Any autonomous or automatic operation should start with a delay of at least 5 seconds after activation:*
  The rover, before the activation of autonomous mode, activates the flashing and waiting for 5 seconds.

- *Rover should be able to take samples in autonomous mode during the science task:*
  Arm autonomy shall be implemented using inverse kinematics approach.

- *During maintenance task rover should be able to automatically detect, approach and manipulate elements on the panel:*
  Arm autonomy and Object recognition algorithms shall be implemented.

- *During collection task, rover should be able to perform automatic detection, automatic approach and automatic pickup of caches:*

The rover shall be able to recognise, using Object recognition algorithm, approach, using auto navigation algorithm and collect, using arm autonomy, the caches. All of the action must be confirmed by the user before their activations.

3 Safety requirements

- *The rover shall be equipped with an easily accessible red emergency stop button:*
  For design, a red emergency stop button is installed on the top of the rover chassis. Also, a virtual emergency button shall be developed in the web-app.

- *The emergency stop button shall isolate the batteries from the system until the reset procedure:*
  The insulation is due to the BMS connected to the emergency button.

- *Button mounted should withstand hard hit and should be attached to a stiff element of the rover's body:*
  The button is mounted on the chassis thanks to a rigid connection.

- *Rover should be equipped with indicator lamp:*
  Beacon lamp shall be equipped.

- *Indicator should be clearly visible from at least 10 m:*
  It is chosen as a beacon lamp with a powerful LED.

- *Activity indicator lamp should be active for 5 seconds before any rover operation:*
  Software installed in the main CPU does this functionality.

4 Tasks requirements

4.1 Maintenance task requirements

- *Controllers should provide feedback of all operations (forces, speed, and position of the manipulator):*
  The robotic arm is equipped with several sensors in order to track motion and other valuable information.

- *The end effectors architecture should be a multiple exchangeable tool system:*
  The maintenance end-effector can be exchanged depending on how the arm shall be brought.

- *Manipulator shall be able to manipulate switch, buttons, measure voltage (with external tools), etc:*
  Rover's gripper end-effector can bring objects with precision thanks to the control software. Also, the rover can detect specific objects using a computer vision algorithm.

4.2 Collection task requirements

- *The rover shall be able to reach caches and to search the caches in the area, approach them and take pictures:*
  The autonomous system assists the operator during search tasks implementing object recognition algorithm.

- *The rover shall be equipped with a manipulator device able to pick up the cache and place it into an on-board container:*
  Design of specific end effector with fingers modelled on cache shape for better grip.

- *The rover system should able to deliver container with caches from rover to designated place:*
  Cache container shall be a modular and removable box that can be manipulated autonomously by the robotic arm.

4.3 Traverse task requirements

- *The rover mobility system should be able to drive over challenging terrain in conditions described:*
  A robust and large rocker-bogie architecture with custom elastic wheels it is designed.

- *The rover could be helped with landmarks, placed on team request:*
  The autonomous system shall be able to recognise landmarks and place them in the reference frame.

- *The rover can use rough height map of the arena provided by organizers, solutions working without a predefined map will score extra points:*
  The autonomous system shall be able to build a map of the local environment using only topography information from onboard sensors.

- *Use of GNSS receivers is not allowed. Any other type of sensor (camera, lidar, IMU, odometer, sonar,etc.) can be used for onboard processing (only in ERC competition):*
  Rover uses only IMU in order to localize its position. In the rover there are 2 different IMU one inside the stereo camera and the other in the centre of mass of the rover.

- *The rover should be able to plan an optimal path based on a given map and way-points coordinates:*
  The autonomous system shall be able to build a map of the environment using only topography information from onboard sensors.

- *The rover can be teleoperated but only with position and orientation estimate available:*
  The autonomous system shall implement a specific state that can accept mobility controls from the operator.

Part IV
Failure Mode Effect Analysis

5 Introduction

The aim of Failure Mode Effect Analysis (FMEA) is to determine and mitigate the effect, or ideally to remove the causes, of all the failure in a system to improve its reliability. In other words, it helps the designer to understand the failure chain, that is the relation between effect, mode and cause of the failure.
A component fails if its behaviour is different respect the nominal. The FMEA analyses both the design and production process phases in order to cover all the requirements of the customer. In our case, only the Design phase is improved because the Process phase is only related to the software and cable management.

The analysis is based on the new manual[43] released by Automotive Industry Action Group (AIAG) and Verband der Automobilindustrie (VDA) in June 2019. This manual divided the FMEA process into 7 steps:

1. Planning and Preparation
2. Structure analysis
3. Function analysis
4. Failure analysis
5. Risk analysis
6. Optimization
7. Documentation of the results

Steps 1, 2 and 3 analyse the system, the functions of the components and their connections. Steps 4, 5 and 6 analyse the failures and their possible mitigations. The failure analysis underlines the effect, the mode and the cause of the failure and also its Severity.
Risk analysis underlines the Occurrency and the Detection index taking into account the current detection action if it is present.
In the Optimization, using the Action Priority index that is the merge of the previous indexes, it is decided the correct detection action that reduces the Action Priority of the failure. Step 7 resume the results and all the mitigation to underline possible changes in the design.

The three main indexes in this analysis are Severity, Occurrency and Detection index. The following definitions are taken by the FMEA manual guideline.

- ***Severity:***
  The Severity is a value in the range 1 to 10 where 1 means that this failure has no action on the plant and 10 means that the failure produce a dangerous behaviour of the plant that can be also dangerous for the user.
  The complete list[43] is the following:

  1. *Very low*: No discernible effect.
  2. *Low*: Slightly objectionable appearance, sound, vibration, harshness or haptics.
  3. *Low*: Moderately objectionable appearance, sound, vibration, harshness or haptics.
  4. *Moderate*: Very objectionable appearance, sound, vibration, harshness or haptics.
  5. *Moderate*: Degradation of secondary vehicle function.
  6. *Moderate*: Loss of secondary vehicle function.
  7. *High*: Degradation of primary vehicle function necessary for normal driving during expected service life.
  8. *High*: Loss of primary vehicle function necessary for normal driving during expected service life.
  9. *Very High*: Non-compliance with regulation.
  10. *Very High*: Affects safe operation of the vehicle and/or other vehicles, the health of driver of passengers or road users or pedestrians.

- ***Occurrency:***

  The Occurrency is a value between 1 and 10 that measure how frequent the failure is present. The value 1 means that the failure is not present and 10 means that the failure is present 1 in 10 of the time.

  The complete list[43] is the following:

  1. *Extremely low*: Prevention controls eliminate failure.
  2. *Very low*: 1 in 1000000 or less than once per year.
  3. *Low*: 1 in 100000 or once per year.
  4. *Moderate*: 1 in 10000 or more than once per year.
  5. *Moderate*: 1 in 2000 or more than once per month.

6. *High*: 1 in 500 or more than once per week.
7. *High*: 1 in 100 or more than once per day.
8. *Very High*: 1 in 50 or more than once per shift.
9. *Very High*: 1 in 20 or almost every time.
10. *Extremely High*: 1 in 10 or every time.

- ***Detection index:***
  The Detection index is a value in the range 1 to 10 and represents how much the detection action can find and mitigates that failure. The value 1 means that the problem is eliminated through the prevention control and 10 means that there isn't any control for this failure.
  The complete list[43] is the following:

  1. *Very High*: Failure mode cannot be physically produced as-designed or processed of detection methods prove to always detect the failure mode or failure cause.
  2. *High*: Machine-based detection method that will detect the cause and prevent the failure mode.
  3. *High*: Machine-based automated detection that will detect the failure mode instation prevent further processing or system identifies the product discrepancy.
  4. *High*: Machine-based automated detection method that will detect the failure mode downstream prevent further processing or system will identify the product.
  5. *Moderate*: Machine-based detection.
  6. *Moderate*: Human inspection or use of manual gauging that will detect the failure mode or failure cause.
  7. *Low*: Machine-based detection or use of inspection equipment such as a coordinate measuring machine that should detect failure mode and failure cause.
  8. *Low*: Human inspection or use of manual gauging that should detect the failure mode and failure cause.
  9. *Very Low*: The failure mode is not easily detected through random or sporadic audits.
  10. *Very Low*: The failure mode will or cannot be detected.

The **Action Priority** is related to the Severity, Occurrency and Detection indexes. Its possible values are:

- H: high priority.
- M: medium priority.
- L: low priority.

In our analysis in more cases even if the failure is dangerous for the rover it is decided to use a Severity equal to 8 and not 10 because there isn't a human driver inside them.
After the analysis is essential to have all the H priority reduced to M or L.

Important notes:

- due to budget limitation imposed by the competitions and to reduce the development time, not all the detection actions are implemented on the rover.
- The FMEA analyses it is applied to the general design of the rover, neglecting the scientific part because it is depending on the competition.

6 System division

To have a complete analysis of the components on the rover it is decided to divide they into:

- Mobility system
- Arm system
- Logic system

The Mobility system contain all the components, both mechanical and electrical, that work during the rover movement. Its main elements are:

- Wheels system
- Steering system
- Rocker-Bogie system

The Arm system is composed by the robotic arm that can interact with the environment collecting object or doing actions, for example press a button or turn a lever.

The Logic system collects all the logic in the rover. It's divided in some components:

- Mobility/Arm logic
- Generic logic
- Rover position sensor
- Transmission logic
- Rover camera system

7 Failure Mode Effect Analysis

In this section is reported the complete FMEA analysis, underlined the main characteristics of the component and its failures.
To help the reader in case of repetition of a component but posed in different position, for example the stepper motor is used in 3 different joints in the arm, the failure analysis is rewritten.

7.1 (Mobility system)Wheel system

The wheel, taking into account, is an elastic system composed by harmonic steel and polymeric parts.
The components analysed in this section are:

- Brushless motor
- Clutch
- Gear reduction
- Grouser
- Main grouser
- External skin
- External and internal cylinder
- Wheel's hub
- Slewing ring
- Motor encoder (incremental encoder)
- Motor hall sensors
- Reduction encoder (incremental encoder)

Figure 10: Wheel and Steering system render

Following the analysis for each components.

7.1.1 Brushless motor

The brushless motor converts electrical power into mechanical power. The failure analysis for this component is:

- *Failure mode*: burning of the insulation painting of the coils wires.
- *Failure cause*: overheats due to excessive absorbed current.
- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stuck preventing the connected wheel to rotate flowing and it generates an inspected behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: the continuous monitoring of the absorbed current.
- *Action Priority*: L
- *Suggested prevention/detection action*: install temperature sensors inside the motor, near the coils, to check the temperature and cut off the motor if its temperature overpass the limit.

7.1.2 Clutch

The clutch is an electro-mechanical part that connects the motor shaft to the gear reduction shaft. It can connect or not the two parts depending on an input signal.
The failure analysis for this component is:

- *Failure mode*: the two parts of the clutch remain always disconnected.
- *Failure cause*: electronic failure.
- *Failure effect*: the motor and its gear reduction remain disconnected and the wheel hasn't any traction (freewheel). It can produce an incorrect behaviour of the rover.
- *Severity*: 7
- *Occurency*: 2

- *Detection index*: 2
- *Current detection action*: continuous monitoring of the difference between the feedbacks given by the motor and gear reduction.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is needed.

7.1.3 Gear reduction

The gear reduction transmits mechanical power from a motor to a load. It maintains the power constant changing the ratio between torque and speed. This ratio is depending on the number of teeth of the main and the second gear.
The failure analysis of this component is:

- *Failure mode*: bearings or gear lose mobility and coupling.
- *Failure cause*: over torque or too high rotation speed.
- *Failure effect*: the wheel is blocked or rotates with a jerk that produces incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 4
- *Current detection action*: compare angular velocity feedback of the motor with the feedback of the reduction, if they are not correlated with the gear ratio the gear reduction fails or will fail.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is needed.

7.1.4 Grouser

The grouser transfers torque by the wheel to the terrain, generating longitudinal movement thanks to the friction force between them. The number of these elements for each wheel is 14. This component has 2 different failure analysis depending on their dangerous. The first failure is:

- *Failure mode*: grouser loses contact with the terrain.

- *Failure cause*: too lower friction between grouser and terrain, especially in hard terrain.
- *Failure effect*: the wheel slips and produces an unpredictable behaviour of the rover.
- *Severity*: 6 (because the wheel slips but not destroys itself).
- *Occurency*: 10
- *Detection index*: 2
- *Current detection action*: compare the actual angular velocity of the wheel with the desired and tune the current on the motor phases to change the speed.
- *Action Priority*: M
- *Suggested prevention/detection action*: equipped the grousers with a rubber skin to increase adherence between wheels and hard terrain.

The second failure is:

- *Failure mode*: grouser leaves the connection with the external skin.
- *Failure cause*: too bigger longitudinal force applied to it. An example of this condition is when the wheel attempts to pass (but it's stuck) an obstacle and all the toque is applied on a single grouser.
- *Failure effect*: the wheel's thread is damaged and produces an incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: analyse the angular velocity of the wheels. If a wheel has a velocity closed to 0 and the other not, increase the traction of the others to avoid the stucking of its.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

Important note: *in this case the analysis is quite complex, because it is assumed that in normal condition the behaviour of a wheel is not depending on the others. In the real case, the dynamic of the system can help the unstucking or the wheel without any detection action, because there is a balancing of the torque between the wheels.*

7.1.5 Main grouser

The main grouser has the same function as a normal grouser but also close the external skin. The failure of this component can be different: The first failure is:

- *Failure mode*: grouser loses contact with the terrain.
- *Failure cause*: too lower friction between grouser and terrain, especially in hard terrain.
- *Failure effect*: the wheel slips and produces an unpredictable behaviour of the rover.
- *Severity*: 6 (because the wheel slips but not destroys itself).
- *Occurency*: 10
- *Detection index*: 2
- *Current detection action*: compare the actual angular velocity of the wheel with the desired and tune the current on the motor phases to change the speed.
- *Action Priority*: M
- *Suggested prevention/detection action*: equipped the grousers with a rubber skin to increase adherence between wheels and hard terrain.

The second failure is:

- *Failure mode*: grouser leaves the connection with the external skin.
- *Failure cause*: too bigger longitudinal force applied to it. An example of this condition is when the wheel attempts to pass (but it's stuck) an obstacle and all the toque is applied on a single grouser.
- *Failure effect*: the wheel's thread is damaged (eccentric thread) and produces an incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10 (because it's not possible to detect if the wheel is eccentric or not).
- *Current detection action*: analyse the angular velocity of the wheels. If a wheel has a velocity closed to 0 and the other not, increase the traction of the others in order to avoid the stucking of its.

- *Action Priority*: M
- *Suggested prevention/detection action*: no other action can be possible.

7.1.6 External skin

The external skin connects the grousers thread to the elastic cylinders. The failure of this component is:

- *Failure mode*: plastic deformation of the elastic steel.
- *Failure cause*: too bigger radial force applied on it.
- *Failure effect*: wheel's structured is deformed, eccentric wheel, that produce vibration and instability of the rover.
- *Severity*: 4
- *Occurency*: 2
- *Detection index*: 10 (because it's not possible to detect if the wheel is eccentric or not).
- *Current detection action*: no action.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

7.1.7 External and internal cylinder

The external and internal cylinder is the main elastic part of the wheel connected between the hub of the wheel and the external skin. They are composed of elastic steel and they give the stiffness of the wheel. The failure analysis for this component is:

- *Failure mode*: permanent deformation of the elastic steel.
- *Failure cause*: too bigger radial force applied to them.
- *Failure effect*: wheel structured is deformed, eccentric wheel, that produces vibration and instability of the rover.
- *Severity*: 7
- *Occurency*: 3
- *Detection index*: 10

- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: measure the vibration transferred by the wheels to the rover structure usable to understand if something in the wheel is wrong. But using this detection it is not possible to determine the type of failure.
- *Action Priority after suggested prevention/detection action:* L

Note: *this problem is partially avoided by the new release of the elastic wheel, because the two cylinders, one inside the other, generate a no linear stiffness. When the first cylinder gives in contact with the second the wheel became more rigid and reduce this deformation.*

7.1.8 Wheel's hub

The wheel's hub hosts the motor and connects it to the slewing ring and the elastic part of the wheel. The failure analysis for this component is:

- *Failure mode*: break of the polymeric part.
- *Failure cause*: the excessive radial force applied to it.
- *Failure effect*: the elastic cylinders leaves from the hub. The result is the deformation of the wheel and incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: measure the vibration transferred by the wheels to the rover structure usable to understand if something in the wheel is wrong. But using this detection it is not possible to determine the type of failure.
- *Action Priority after suggested prevention/detection action:* L

7.1.9 Slewing ring

The slewing ring connects the wheel's hub to the steering bracket. The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: the excessive radial force applied to it.
- *Failure effect*: the hub and the elastic part leave the steering bracket. The result is the left of the whole wheel by the rover and consequently incorrect behaviour of it.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: measure the vibration transferred by the wheels to the rover structure usable to understand if something in the wheel is wrong. But using this detection it is not possible to determine the type of failure.
- *Action Priority after suggested prevention/detection action:* L

7.1.10 Motor encoder (incremental encoder)

The motor encoder is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic reads incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration or misalignment of the rotation shaft.
- *Failure effect*: incorrect feedback velocity of the wheel send by the controller logic.
- *Severity*: 5
- *Occurency*: 4

- *Detection index*: 2
- *Current detection action*: compare the feedback velocities given by motor sensors with gear reduction feedback. If they are different it is possible to select the better between them using an inverse Ackerman model.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism break.
- *Failure cause*: direct hit or excessive velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder doesn't produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: compare the feedback velocities given by motor sensors with gear reduction feedback. In this case, it is possible to select the other encoder value but only if it is coherent with the other feedback wheels, using an inverse Ackermann model.
- *Action Priority*: L
- *Suggested prevention/detection action*:no other action is required.

7.1.11 Motor hall sensors

The motor hall sensors is an electro-mechanical component that measures the magnetic field inside the motor and calculates the corresponding angular velocity. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic read incorrectly the signal produced by hall sensors.
- *Failure cause*: excessive vibration or misalignment of the rotation shaft.

- *Failure effect*: incorrect feedback velocity of the wheel send by the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 2
- *Current detection action*: compare the feedback velocities given by motor sensors with gear reduction feedback. If they are different it's possible to select the better between them using an inverse Ackerman model.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

The analysis of the second failure is:

- *Failure mode*: the internal logic or sensors break.
- *Failure cause*: direct hit or tool low velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the hall sensors do not produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: compare the feedback velocities given by motor sensors with gear reduction feedback. In this case, it's possible to select the other encoder value but only if it is coherent with the other feedback wheels, using an inverse Ackermann model.
- *Action Priority*: L
- *Suggested prevention/detection action*:no other action is required.

7.1.12 Gear reduction encoder (incremental encoder)

The gear reduction encoder it is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic read incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration or misalignment of the rotation shaft.
- *Failure effect*: incorrect feedback velocity of the wheel send by the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 2
- *Current detection action*: compare the feedback velocities given by motor sensors with gear reduction feedback. If they are different it's possible to select the better between them using an inverse Ackermann model.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism break.
- *Failure cause*: direct hit or excessive velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder doesn't produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: compare the feedback velocities given by motor sensors with gear reduction feedback. In this case, it's possible to select the other encoder value but only if it is coherent with the other feedback wheels, using an inverse Ackermann model.

- *Action Priority*: L
- *Suggested prevention/detection action*:no other action is required.

7.2 (Mobility system) Steering system

The steering system is positioned in the front and rear wheels. It is composed of the following components:

- Bracket
- Stepper motor
- Gear reduction
- Motor encoder (incremental encoder)
- Reduction shaft encoder (absolute encoder)
- Steering support on Rocker-Bogie

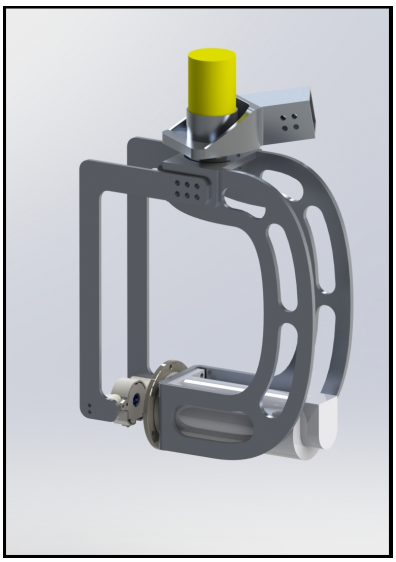


Figure 11: Steering system render

Following the analysis for each component:

7.2.1 Bracket

The steering bracket connects the shaft of the stepper motor's gear reduction with the wheel's hub. The failure of this component can be different.
The analysis of the first failure is:

- *Failure mode*: the steel structure of the bracket is permanent deformed.
- *Failure cause*: higher longitudinal or lateral force.
- *Failure effect*: misalignment of the wheel respect the steer rotation axis that produces an incorrect behaviour of the rover.
- *Severity*: 7
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: compare the feedback steer velocity measured by the IMU with the steering velocity calculate by the inverse dynamic model. But the capability to find the failure is very low.
- *Action Priority after suggested prevention/detection action:* M

The analysis of the second failure is:

- *Failure mode*: the steel structure of the bracket is very permanent deformed.
- *Failure cause*: higher longitudinal or lateral force .
- *Failure effect*: very significant misalignment of the wheel respect the steer rotation axis that produces an incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M

- *Suggested prevention/detection action*: compare the feedback steer velocity measured by the IMU with the steering velocity calculate by the inverse dynamic model. But the capability to find the failure is very low.
- *Action Priority after suggested prevention/detection action:* M

7.2.2 Stepper motor

The stepper motor converts electrical power into mechanical power. The failure analysis for this component is:

- *Failure mode*: burning of the insulation painting of the coils wires.
- *Failure cause*: overheats due to excessive absorbed current.
- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stuck that prevents the connected wheel to steer flowing and generates an unexpected behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: the continuous monitoring of the absorbed current.
- *Action Priority*: L
- *Suggested prevention/detection action*: install temperature sensors inside the motor, near the coils, to check the temperature and cut off the motor if its temperature overpass the limit.

7.2.3 Gear reduction

The gear reduction transmits mechanical power from a motor to a load. It maintains the power constant changing the ratio between torque and speed. This ratio is depending on the number of teeth of the main and the second gear.
The failure analysis of this component is:

- *Failure mode*: bearings or gear lose mobility and coupling.
- *Failure cause*: over torque or too high rotation speed.

- *Failure effect*: the steer is blocked or rotates with a jerk that produces incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 4
- *Current detection action*: compare the feedback angle of the motor with the feedback of the reduction, if they are not correlated with the gear ratio the gear reduction fails or will fail.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is needed.

7.2.4 Motor encoder (incremental encoder)

The motor encoder is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic read incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration or misalignment of the rotation shaft.
- *Failure effect*: incorrect feedback position of the wheel's steer send to the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 2
- *Current detection action*: compare the feedback positions given by the motor's and gear reduction's encoders. If they are different it's possible to select the better between them using an inverse Ackermann model.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism break.
- *Failure cause*: direct hit or excessive velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder doesn't produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: compare the feedback positions given by the motor's and gear reduction's encoders. In this case, it's possible to select the other encoder value but only if it is coherent with the other feedback wheels, using an inverse Ackermann model.
- *Action Priority*: L
- *Suggested prevention/detection action*:no other action is required.

7.2.5 Reduction shaft encoder (incremental encoder)

The gear reduction encoder it is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic read incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration or misalignment of the rotation shaft.
- *Failure effect*: incorrect feedback velocity of the wheel send to the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 2
- *Current detection action*: compare the feedback positions given by the motor's and gear reduction's encoders. If they are different it's possible to select the better between them using an inverse Ackermann model.

- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism break.
- *Failure cause*: direct hit or excessive velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder doesn't produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: compare the feedback positions given by the motor's and gear reduction's encoders. In this case, it's possible to select the other encoder value but only if it is coherent with the other feedback wheels, using an inverse Ackermann model.
- *Action Priority*: L
- *Suggested prevention/detection action*:no other action is required.

7.2.6 Steering support on Rocker-Bogie

The steering support on Rocker-Bogie is the mechanical component that connects the steering system to the Rocker-Bogie system. The failure analysis for this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: bigger lateral torsion or direct hit on the structure.
- *Failure effect*: not strongly connection between the steering system and Rocker-Bogie system that can produce an incorrect and unpredictable behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10

- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*:no other action is required because it is assumed that the component is well designed.

7.3 (Mobility system) Rocker-Bogie system

The Rocker-Bogie system is the passive suspension of the rover. It is composed of 4 main structure: 2 rockers connected to the chassis by the slewing ring, and 2 independent bogies connected to the rocker. A torsion bar connects the movements of the 2 rockers.

The components analysed in this section are:

- Torsion bar encoder (absolute encoder)
- Bogie joint encoder (absolute encoder)
- Slewing ring of the hub rocker chassis
- Hub rocker chassis
- Rocker arm
- Torsion bar
- Slewing ring torsion bar

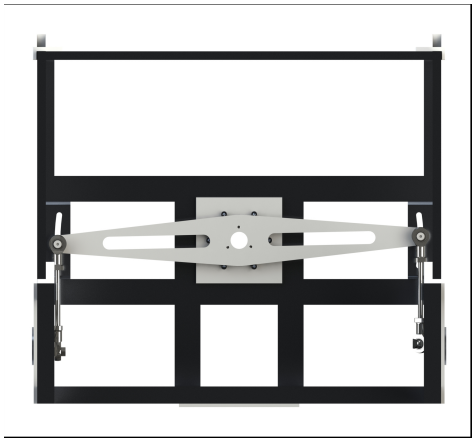


Figure 12: Torsion bar and chassis render (top view)

Following the analysis for each component:

7.3.1 Torsion bar encoder (absolute encoder)

The torsion bar encoder is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. It measures the rotation angle of the torsion bar to understand the angle of rocker joints. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic read incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration.
- *Failure effect*: incorrect feedback angle of the rockers send to the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible because this feedback is strictly related to the terrain shape, than it is not predictable. It is needed another redundant encoder.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism break.
- *Failure cause*: direct hit, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder is not able to send any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 5
- *Current detection action*: check if the feedback is recently or not. If it isn't, stop the rover.
- *Action Priority*: M

- *Suggested prevention/detection action*: no other action is possible because this feedback is strictly related to the terrain shape, then it is not predictable. It is needed another redundant encoder.

7.3.2 Bogie joint encoder (absolute encoder)

The torsion bar encoder is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. It measures the rotation angle of the bogie. The failure can be different. The analysis of the first failure is:

- *Failure mode*: the internal logic read incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration.
- *Failure effect*: incorrect feedback angle of the rockers send to the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible because this feedback is strictly related to the terrain shape, than it is not predictable. It is needed another redundant encoder.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism break.
- *Failure cause*: direct hit, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder is not able to send any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 5

- *Current detection action*: check if the feedback is recently or not. If it isn't, stop the rover.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible because this feedback is strictly related to the terrain shape, then it is not predictable. It is needed another redundant encoder.

7.3.3 Slewing ring of hub rocker chassis

The slewing ring of the hub rocker chassis connects the rocker with the hub of the chassis.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: higher radial force.
- *Failure effect*: the rocker leaves the chassis. The rover is not able to work in this condition because lose half mobility system.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it hasn't a high overload.

7.3.4 Hub rocker chassis

The hub of the rocker chassis connects the slewing ring of the rocker with the chassis.
The failure analysis of this component is:

- *Failure mode*: the structure losses stiffness.
- *Failure cause*: incorrect design.
- *Failure effect*: too mechanical play between the rocker and the chassis that produces possible incorrect behaviour of the rover.

- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it hasn't an high overload.

7.3.5 Rocker arm

The rocker arm is the structure that integrates the movement of the rockers. The failure analysis of this component is:

- *Failure mode*: deformation of the steel structure.
- *Failure cause*: buckling and incorrect design.
- *Failure effect*: non-integral movement of the leg of the rockers that produces incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it hasn't an high overload.

7.3.6 Torsion bar

The torsion bar transfers the weight and the cinematic between the two sides of rocker-bogie system.
The failure analysis of this component is:

- *Failure mode*: break of the structure in the middle because is the most stressed section.

- *Failure cause*: excessive load.
- *Failure effect*: the chassis rotates forward because it is not integrated to the rocker-bogie.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it can't have an overload.

7.3.7 Slewing ring torsion bar

The slewing ring torsion bar connects the torsion bar with the chassis.
The failure analysis of this component is:

- *Failure mode*: break of the structure.
- *Failure cause*: excessive load.
- *Failure effect*: the chassis rotates forward because it is not integrated to the rocker-bogie.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it can't have an overload.

7.4 Arm system

The arm is rover's system that can interact with the environment. It can bring objects or take a portion of terrain.
In this section the components analysed are:

- Slewing ring base
- Gearwheel base
- Support plate of the stepper motor (shoulder)
- Master and slave pinion gears (shoulder)
- Arm structure
- Hub elbow
- Forearm
- Wrist hub
- Wrist centre part
- Wrist-tool interface
- Base stepper motor
- Base stepper motor gear reduction
- Base stepper motor encoder (incremental encoder)
- Shoulder stepper motor
- Shoulder stepper motor gear reduction
- Shoulder stepper motor encoder (incremental encoder)
- Elbow stepper motor
- Elbow stepper motor gear reduction
- Elbow stepper motor encoder (incremental encoder)
- First degree of wrist rotation servomotor
- Second degree of wrist rotation servomotor
- Third degree of wrist rotation servomotor
- Plier's brush motor
- Plier's lead screw support

- Plier's structure
- Grabber's brush motor
- Grabber's lead screw support
- Grabber's structure

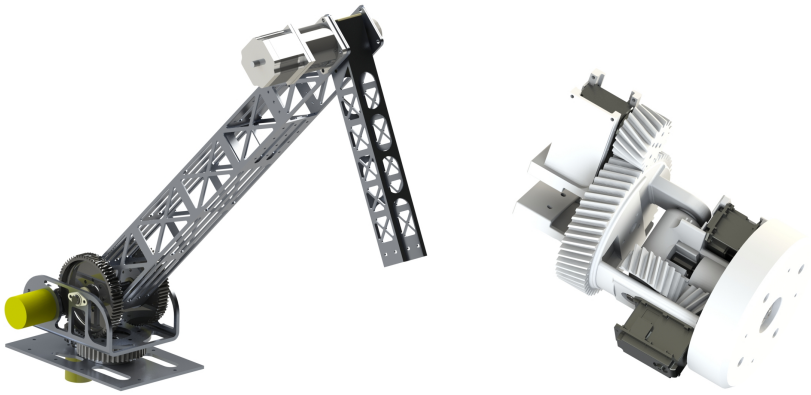


Figure 13: Arm and wrist render

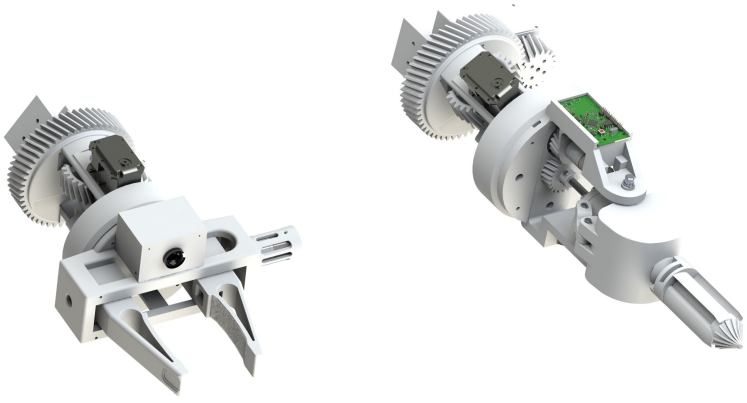


Figure 14: Plier and grabber render

Following the analysis for each component:

7.4.1 Slewing ring base

The slewing ring base connects the arm base with the chassis.
The failure analysis of this component is:

- *Failure mode*: break of the steel structure.
- *Failure cause*: excessive bending moment.
- *Failure effect*: separation of the arm from the rover's chassis.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible because no sensors can be installed, but it is assumed that the component is well designed and it can't have an overload.

7.4.2 Gearwheel base

The gearwheel base transmits torque and speed from the base stepper motor to the base of the arm structure.
The failure analysis of this component is:

- *Failure mode*: break or rounding of the gearwheel's teeth.
- *Failure cause*: the excessive tangential force applied to it.
- *Failure effect*: the arm loses a degree of freedom because the arm base is stucked.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible because no sensors can be installed, but it is assumed that the component is well designed and it can't have an overload.

7.4.3 Support plate of the stepper motor (shoulder)

The support plate of the stepper motor connects the motor of the shoulder with the slewing ring of the base.
The failure analysis of this component is:

- *Failure mode*: break of the steel support structure.
- *Failure cause*: the excessive force applied to it.
- *Failure effect*: separation of the rover arm to the chassis.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible because no sensors can be installed, but it is assumed that the component is well designed and it can't have an overload.

7.4.4 Master and slave pinion gears (shoulder)

The master and slave pinion gears of the shoulder transmits torque and speed with a specified ratio depending on their number of teeth.
The failure analysis of this component is:

- *Failure mode*: break or rounding of the gears' teeth.
- *Failure cause*: the excessive tangential force applied to it due to excessive torque.
- *Failure effect*: the shoulder is blocked and the arm loses a degree of freedom.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M

- *Suggested prevention/detection action*: no other action is possible because no sensors can be installed, but it is assumed that the component is well designed and it can't have an overload.

7.4.5 Arm structure

The arm structure is the structural component that connects the shoulder with the elbow.
The failure analysis of this component is:

- *Failure mode*: deformation of the steel structure.
- *Failure cause*: excessive load apply on it.
- *Failure effect*: the alignment of the arm is not the nominal caused by the deformation.
- *Severity*: 7
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible because no sensors can be installed, but it is assumed that the component is well designed and it can't have an overload.

7.4.6 Hub elbow

The hub elbow connects the motor shaft (stepper motor of the elbow) with the arm structure.
The failure analysis of this component is:

- *Failure mode*: break of the steel component.
- *Failure cause*: excessive torque apply on it.
- *Failure effect*: the elbow is stucked even if the motor is running. The arm loses a degree of freedom.
- *Severity*: 8
- *Occurency*: 2

- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it can't have an overload.

7.4.7 Forearm

The forearm connects rigidly the elbow with the wrist hub.
The failure analysis of this component is:

- *Failure mode*: permanent deformation of the steel structure.
- *Failure cause*: excessive load apply on it.
- *Failure effect*: the alignment of the arm is not the nominal caused by the deformation.
- *Severity*: 7
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it can't have an overload.

7.4.8 Wrist hub

The wrist hub connects the wrist to the forearm.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: excessive load apply on it.
- *Failure effect*: wrist leaves the arm.
- *Severity*: 8

- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it can't have an overload.

7.4.9 Wrist center part

The wrist center part manages the degrees of freedom of the wrist.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: excessive torque apply on it.
- *Failure effect*: wrist loses one or more degrees of freedom.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it can't have an overload.

7.4.10 Wrist-tool interface

The wrist-tool interface allows the connection between the tool and the wrist.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: excessive load apply on it.
- *Failure effect*: separation of the tool from the wrist that produces the inability of the arm to interact with the environment.

- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is possible, but it is assumed that the component is well designed and it can't have an overload.

7.4.11 Base stepper motor

The stepper motor converts electrical power into mechanical power. The failure analysis for this component is:

- *Failure mode*: burning of the insulation painting of the coil's wires.
- *Failure cause*: overheats due to excessive absorbed current.
- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stucked that loses the arm of a degree of freedom.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: the continuous monitoring of the absorbed current.
- *Action Priority*: L
- *Suggested prevention/detection action*: install temperature sensors inside the motor, near the coils, in order to check the temperature and cut off the motor if its temperature overpass the limit.

7.4.12 Base stepper motor gear reduction

The gear reduction transmits mechanical power from a motor to a load. It maintains the power constant changing the ratio between torque and speed. This ratio is depending on the number of teeth of the main and the second gear.
The failure analysis of this component is:

- *Failure mode*: bearings or gear lose mobility and coupling.
- *Failure cause*: over torque or too high rotation speed.
- *Failure effect*: the arm loses a degree of freedom.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install an encoder in the reduction shaft to check its feedback value with the feedback of the motor encoder scaled by the gear ratio.

7.4.13 Base stepper motor encoder (incremental encoder)

The motor encoder is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic reads incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration or misalignment of the rotation shaft.
- *Failure effect*: incorrect feedback position of the wheel's steer send to the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 10

- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install a redundant encoder in the motor or in the reduction shaft in order to check if the value is wrong or not.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism breaks.
- *Failure cause*: direct hit or excessive velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder doesn't produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install a redundant encoder in the motor or in the reduction shaft in order to check if the value is wrong or not.

7.4.14 Shoulder stepper motor

The stepper motor converts electrical power into mechanical power. The failure analysis for this component is:

- *Failure mode*: burning of the insulation painting of the coil's wires.
- *Failure cause*: overheats due to excessive absorbed current.
- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stucked and the arm loses a degree of freedom.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 2

- *Current detection action*: the continuous monitoring of the absorbed current.
- *Action Priority*: L
- *Suggested prevention/detection action*: install temperature sensors inside the motor, near the coils, to check the temperature and cut off the motor if its temperature overpass the limit.

7.4.15 Shoulder stepper motor gear reduction

The gear reduction transmits mechanical power from a motor to a load. It maintains the power constant changing the ratio between torque and speed. This ratio is depending on the number of teeth of the main and the second gear.
The failure analysis of this component is:

- *Failure mode*: bearings or gear lose mobility and coupling.
- *Failure cause*: over torque or too high rotation speed.
- *Failure effect*: the arm loses a degree of freedom.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install an encoder in the reduction shaft to check its feedback value with the feedback of the motor encoder scaled by the gear ratio.

7.4.16 Shoulder stepper motor encoder (incremental encoder)

The motor encoder is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic reads incorrectly the signal produced by mechanical part.

- *Failure cause*: excessive vibration or misalignment of the rotation shaft.
- *Failure effect*: incorrect feedback position of the wheel's steer send to the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install a redundant encoder in the motor or in the reduction shaft to check if the value is wrong or not.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism breaks.
- *Failure cause*: direct hit or excessive velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder doesn't produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install a redundant encoder in the motor or in the reduction shaft to check if the value is wrong or not.

7.4.17 Elbow stepper motor

The stepper motor converts electrical power into mechanical power. The failure analysis for this component is:

- *Failure mode*: burning of the insulation painting of the coil's wires.
- *Failure cause*: overheats due to excessive absorbed current.
- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stucked and the arm loses a degree of freedom.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: the continuous monitoring of the absorbed current.
- *Action Priority*: L
- *Suggested prevention/detection action*: install temperature sensors inside the motor, near the coils, to check the temperature and cut off the motor if its temperature overpass the limit.

7.4.18 Elbow stepper motor gear reduction

The gear reduction transmits mechanical power from a motor to a load. It maintains the power constant changing the ratio between torque and speed. This ratio is depending on the number of teeth of the main and the second gear.
The failure analysis of this component is:

- *Failure mode*: bearings or gear lose mobility and coupling.
- *Failure cause*: over torque or too high rotation speed.
- *Failure effect*: the arm loses a degree of freedom.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M

- *Suggested prevention/detection action*: install an encoder in the reduction shaft to check its feedback value with the feedback of the motor encoder scaled by the gear ratio.

7.4.19 Elbow stepper motor encoder (incremental encoder)

The motor encoder is an electro-mechanical component that converts the rotation angle of the shaft into an analog signal readable by the logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the internal logic reads incorrectly the signal produced by mechanical part.
- *Failure cause*: excessive vibration or misalignment of the rotation shaft.
- *Failure effect*: incorrect feedback position of the wheel's steer send to the controller logic.
- *Severity*: 5
- *Occurency*: 4
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install a redundant encoder in the motor or in the reduction shaft to check if the value is wrong or not.

The analysis of the second failure is:

- *Failure mode*: the internal logic or the rotative mechanism break.
- *Failure cause*: direct hit or excessive velocity, for the mechanical part, and too much absorbed current or voltage, for the electrical part.
- *Failure effect*: the encoder doesn't produce any feedback value readable by the controller logic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10

- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: install a redundant encoder in the motor or in the reduction shaft to check if the value is wrong or not.

7.4.20 First degree of wrist rotation servomotor

The servomotor related to wrist's first degree of freedom is an electromechanics component that converts electrical power into mechanical power using an internal reduction system.
The failure analysis for this component can be different:

Analysis of the first failure:

- *Failure mode*: burning of the insulation painting of the coil's wires.
- *Failure cause*: overheats due to excessive absorbed current.
- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stucked and the wrist loses the first degree of freedom.
- *Severity*: 6
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: the continuous monitoring of the motor temperature, reading the bit number 2 of the error array. Also, it is possible to read the value of the alarm on address 17 and 18 in the same array.
- *Action Priority*: L
- *Suggested prevention/detection action*: monitoring the input voltage reading bit number 0 of the error array.

Analysis of the second failure:

- *Failure mode*: the polymeric part of the gear reduction breaks.
- *Failure cause*: excessive torque or uncorrect working condition.
- *Failure effect*: the motor is not more able to generate torque and the wrist loses the first degree of freedom.

- *Severity*: 6
- *Occurency*: 3
- *Detection index*: 5
- *Current detection action*: monitor the torque reading the bit number 5 of the error array (overload).
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

Analysis of the third failure:

- *Failure mode*: the MCU or the encoder breaks.
- *Failure cause*: incorrect power source or direct hit.
- *Failure effect*: the motor is not more able to generate torque and the wrist loses the first degree of freedom.
- *Severity*: 6
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: monitor the status of the servomotor, reading bit in address 16 that must be different from 0 otherwise there is a problem.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

7.4.21 Second degree of wrist rotation servomotor

The servomotor related to wrist's first degree of freedom is an electromechanics component that converts electrical power into mechanical power using an internal reduction system.
The failure analysis for this component can be different:

Analysis of the first failure:

- *Failure mode*: burning of the insulation painting of the coil's wires.
- *Failure cause*: overheats due to excessive absorbed current.

- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stucked and the wrist loses the second degree of freedom.
- *Severity*: 6
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: the continuous monitoring of the motor temperature, reading the bit number 2 of the error array. Also, it's possible to read the value of the alarm on address 17 and 18 in the same array.
- *Action Priority*: L
- *Suggested prevention/detection action*: monitoring the input voltage reading bit number 0 of the error array.

Analysis of the second failure:

- *Failure mode*: the polymeric part of the gear reduction breaks.
- *Failure cause*: excessive torque or incorrect working condition.
- *Failure effect*: the motor is not more able to generate torque and the wrist loses the second degree of freedom.
- *Severity*: 6
- *Occurency*: 3
- *Detection index*: 5
- *Current detection action*: monitor the torque reading the bit number 5 of the error array (overload).
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

Analysis of the third failure:

- *Failure mode*: the MCU or the encoder breaks.
- *Failure cause*: incorrect power source or direct hit.
- *Failure effect*: the motor is not more able to generate torque and the wrist loses the second degree of freedom.

- *Severity*: 6
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: monitor the status of the servomotor, reading a bit in address 16 that must be different from 0 otherwise there is a problem.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

7.4.22 Third degree of wrist rotation servomotor

The servomotor related to wrist's first degree of freedom is an electromechanics component that converts electrical power into mechanical power using an internal reduction system.
The failure analysis for this component can be different:

Analysis of the first failure:

- *Failure mode*: burning of the insulation painting of the coil's wires.
- *Failure cause*: overheats due to excessive absorbed current.
- *Failure effect*: the motor is not more able to generate torque or, in even worst case, it remains stucked and the wrist loses the third degree of freedom.
- *Severity*: 6
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: the continuous monitoring of the motor temperature, reading the bit number 2 of the error array. Also, it is possible to read the value of the alarm on address 17 and 18 in the same array.
- *Action Priority*: L
- *Suggested prevention/detection action*: monitoring the input voltage reading bit number 0 of the error array.

Analysis of the second failure:

- *Failure mode*: the polymeric part of the gear reduction breaks.
- *Failure cause*: excessive torque or incorrect working condition.
- *Failure effect*: the motor is not more able to generate torque and the wrist loses the third degree of freedom.
- *Severity*: 6
- *Occurency*: 3
- *Detection index*: 5
- *Current detection action*: monitor the torque reading the bit number 5 of the error array (overload).
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

Analysis of the third failure:

- *Failure mode*: the MCU or the encoder breaks.
- *Failure cause*: incorrect power source or direct hit.
- *Failure effect*: the motor is not more able to generate torque and the wrist loses the third degree of freedom.
- *Severity*: 6
- *Occurency*: 2
- *Detection index*: 2
- *Current detection action*: monitor the status of the servomotor, reading bit in address 16 that must be different from 0 otherwise there is a problem.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

7.4.23 Pliers' brush motor

The pliers' brush motor is an electromechanics component that moves the two part of the pliers.
The failure analysis of this component is:

- *Failure mode*: the insulation of the coil's wire burn.
- *Failure cause*: excessive absorbed current.
- *Failure effect*: the arm is unable to clamp objects during the tasks.
- *Severity*: 8
- *Occurency*: 3
- *Detection index*: 3
- *Current detection action*: continuous monitoring of the absorbed current by the motor.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

7.4.24 Pliers' lead screw support

The pliers' lead screw support holds up the lead screw that moves the two part of the pliers.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: excessive stress on the structure.
- *Failure effect*: the arm can not clamp objects during the task.
- *Severity*: 8
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no action it is possible.

7.4.25 Pliers' structure

The pliers' structure is the part that clamps the objects.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: excessive stress on the structure tool.
- *Failure effect*: the arm can not clamp objects during the task.
- *Severity*: 8
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no action it is possible.

7.4.26 Grabber's brush motor

The grabber's brush motor is an electromechanics component that moves the two part of the grabber.
The failure analysis of this component is:

- *Failure mode*: the insulation of the coil's wire burn.
- *Failure cause*: excessive absorber current.
- *Failure effect*: the arm is unable to grab portion of terrain during the task.
- *Severity*: 8
- *Occurency*: 3
- *Detection index*: 3
- *Current detection action*: continuous monitoring of the absorbed current by the motor.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

7.4.27 Grabber's lead screw support

The grabber's lead screw support holds up the lead screw that moves the two part of the grabber.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: excessive stress on the structure.
- *Failure effect*: the arm can not grab portion of terrain during the task.
- *Severity*: 8
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no action it is possible.

7.4.28 Grabber's structure

The grabber's structure is the part that grabs the terrain.
The failure analysis of this component is:

- *Failure mode*: break of the polymeric structure.
- *Failure cause*: excessive stress on the structure tool.
- *Failure effect*: the arm can not grab portion of terrain during the task.
- *Severity*: 8
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no action it is possible.

7.5 (Logic system) Mobility/arm logic

In this section the components analyse are:

- Driver encoder logic
- Motor driver
- Mobility controller
- Arm controller

Following the analysis for each components:

7.5.1 Driver encoder logic

The driver encoder logic receives feedback from both steering and wheels system and transmits them mainly to the mobility logic. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the logic sends incorrect values to the upper logic level.
- *Failure cause*: electronic failure.
- *Failure effect*: the feedbacks receives by the mobility controller are wrong and they produce incorrect behaviour of the rover because the controller can not work properly.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: knowing the desired trajectory and using the inverse dynamic model of the rover it is possible to check the correctness of the feedback output of the subsystem.
- *Action Priority after suggested prevention/detection action:* L

The analysis of the second failure is:

- *Failure mode*: the logic does not send any feedback values to the upper level logic .
- *Failure cause*: electrical failure .

- *Failure effect*: the mobility controller does not receive important data from the dynamics of the rover, then it can not work properly. The rover has an incorrect behaviour.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: check if the feedbacks values stored in the controller are recent or not; if they aren't stop the rover.
- *Action Priority after suggested prevention/detection action:* L

7.5.2 Motor driver

The motor driver controls the brushless motor of the wheels using feedbacks given by the sensors. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the logic sends incorrect values to the motor.
- *Failure cause*: electronic failure.
- *Failure effect*: the motor receives an incorrect command then the rover has an incorrect and not predictable behaviour.
- *Severity*: 7
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: check the correctness between the command value, sends by the mobility controller, and the feedback given by the sensor. If they are not coherent it means that something is wrong then stop the rover. Using this detection it is not possible to understand the type of failure and what component fails.
- *Action Priority after suggested prevention/detection action:* L

The analysis of the second failure is:

- *Failure mode*: the logic does not send any value to the motor.
- *Failure cause*: electrical failure.
- *Failure effect*: the motor does not receive any command then it can't produce torque. Due to this, the rover can have incorrect behaviour.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: check the correctness between the command value sends by the mobility controller and the feedback given by the sensor. If they are not coherent it means that something is wrong then stop the rover. Using this detection it is not possible to understand the type of failure and what component fails.
- *Action Priority after suggested prevention/detection action:* L

7.5.3 Mobility controller

The mobility controller controls the rover using as input the velocity of the centre of mass in polar coordinate and the feedbacks given by the wheels and steering motors. The output of the controller is the desired wheels angular velocity and steering angle to reach the desired trajectory. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the logic doesn't send any value to the motor drivers and to the main CPU.
- *Failure cause*: electronic failure.
- *Failure effect*: if the drivers doesn't receive any values from the controller the rover immediately stop.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10

- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: to increase the reliability it is used other 2 redundant components with a majority comparator. The failure occurrence is equal for each logic and the comparator is take a look as a component without any possible failure.
- *Action Priority after suggested prevention/detection action:* L

The analysis of the second failure is:

- *Failure mode*: the logic sends incorrect value to the motor drivers and to the main CPU .
- *Failure cause*: electrical failure .
- *Failure effect*: the drivers receives incorrect command values that produce incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: to increase the reliability it is used other 2 redundant components with a majority comparator. The failure occurrence is equal for each logic and the comparator is take a look as a component without any possible failure.
- *Action Priority after suggested prevention/detection action:* L

7.5.4 Arm controller

The arm controller controls the arm with both inverse kinematics and joint-by-joint approach. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the logic doesn't send any value to the motor drivers and to the main CPU.
- *Failure cause*: electronic failure.

- *Failure effect*: the drivers doesn't receive any values from the controller and the arm immediately stop.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: check the feedback values stored in the main CPU. If it is not recent the rover goes in a stable state (rover stop).
- *Action Priority after suggested prevention/detection action:* L

The analysis of the second failure is:

- *Failure mode*: the logic sends incorrect value to the motor drivers and to the main CPU.
- *Failure cause*: electrical failure.
- *Failure effect*: the drivers receives incorrect command values that produce incorrect behaviour of the arm.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action can be possible because the feedback given by the sensors can not be compared with the command value given by the controller because there is some delay due to actuators.

7.6 (Logic system) General logic

In this section the components analyse are:

- Main CPU
- BMS
- Ethernet switch

Following the analysis for each component:

7.6.1 Main CPU

The main CPU controls and interacts with all the subsystem logic of the rover.The failure analysis for this component is:

- *Failure mode*: the logic sends any value to the subsystem logic.
- *Failure cause*: electrical failure.
- *Failure effect*: the rover is not controlled and can have incorrect behaviour.
- *Severity*: 10
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: H
- *Suggested prevention/detection action*: each subsystem logic check if the command sends by the main CPU is recent or not. If it is not stop the subsystem in a stable state.
- *Action Priority after suggested prevention/detection action:* L

7.6.2 BMS

The BMS is the device that manages and guarantees the safety of the batteries power supply. It is between the batteries and the different subsystem logic, included main CPU and motor drivers. The failure can be different. The analysis of the first failure is:

- *Failure mode*: BMS logic fails.
- *Failure cause*: electronic failure.

- *Failure effect*: the batteries are in short circuit and consequently they worms and catch on fire.
- *Severity*: 10
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: even if the rover is in a safety state because it is powered off, the short circuit in the BMS logic can catch on fire the batteries.
- *Action Priority*: H
- *Suggested prevention/detection action*: install another BMS in parallel with the previous to avoid the short circuit of the batteries.
- *Action Priority after suggested prevention/detection action:* L

The analysis of the second failure is:

- *Failure mode*: the BMS electrical part fails and returns in a safety state.
- *Failure cause*: electrical failure.
- *Failure effect*: the BMS logic detects the failure and turns off the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: no other action is required.

7.6.3 Ethernet switch

The Ethernet switch connects all the devices that use the Ethernet line. The failure analysis for this component is:

- *Failure mode*: the switch loses its capability of broken.
- *Failure cause*: electrical failure.

- *Failure effect*: the devices connected to the switch can be isolated producing incorrect and unpredictable behaviour of the rover.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: each device connected to the Ethernet switch check if the data, given by other devices on the line, is recent or not. If it is not, the logic changes its state in a safety configuration.
- *Action Priority after suggested prevention/detection action:* L

7.7 (Logic system) Rover position sensor

In this section, it is decided to analyse only the Inertia Measurement Unit, IMU, because the Global Position System, GPS, is used only in the american competition.

7.7.1 Inertia Measurements Unit (manual drive)

The IMU sensor measures the acceleration, both linear and angular, of the rover's centre of mass. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the sensor doesn't send any value to the main CPU.
- *Failure cause*: electronic or mechanical failure.
- *Failure effect*: the user is penalized because loses a feedback of the rover's dynamic.
- *Severity*: 5
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: check if the feedback data, stored in the main CPU, is recent or not. If it is not recent it is sent an error.

- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

The analysis of the second failure is:

- *Failure mode*: the sensor sends incorrect value to the main CPU.
- *Failure cause*: electrical or mechanical failure.
- *Failure effect*: the user is penalized because loses a feedback of the rover's dynamic.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: check the correctness of the feedback value, comparing it with the result of the inverse dynamic model. The inputs of the model are the feedback angular velocity and steering angle for each wheel and the feedback angles of the rocker-bogie system.
- *Action Priority after suggested prevention/detection action:* M

7.7.2 Inertia Measurement Unit (autonomous drive)

The IMU sensor measures the acceleration, both linear and angular, of the rover's centre of mass. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the sensor doesn't send any value to the main CPU.
- *Failure cause*: electronic or mechanical failure.
- *Failure effect*: incorrect behaviour of the rover because the autonomous drive hasn't any feedback of the rover's dynamics.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10

- *Current detection action*: check if the feedback data, stored in the main CPU, is recent or not. If it is not recent it is sent an error and skip the mode in the manual drive.
- *Action Priority*: L
- *Suggested prevention/detection action*: no other action is required.

The analysis of the second failure is:

- *Failure mode*: the sensor sends the incorrect value to the main CPU.
- *Failure cause*: electrical or mechanical failure.
- *Failure effect*: incorrect behaviour to the rover.
- *Severity*: 10
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: H
- *Suggested prevention/detection action*: check the correctness of the feedback value, comparing it with the result of the inverse dynamic model. The inputs of the model are the feedback angular velocity and steering angle for each wheel and the feedback angles of the rocker-bogie system. If the values are uncorrected send and error and split the mode to manual drive.
- *Action Priority after suggested prevention/detection action:* M

7.8 (Logic system) Transmission system

The transmission system connects the rover to the base station. The components analysed in this section are:

- Radio transmitter
- Radio receiver
- Transmission logic

Following the analysis for each component:

7.8.1 Radio transmitter

The radio transmitter transmits data from the rover (main CPU) with the base station. The failure analysis for this component is:

- *Failure mode*: break of the hardware.
- *Failure cause*: electrical failure.
- *Failure effect*: the rover is isolated from the base station. It can not send any feedback.
- *Severity*: 10
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: H
- *Suggested prevention/detection action*: to check the connection, both the rover and the base station send an acknowledge if they receive a message from the others. If the acknowledge is not sent by the base station the rover stops or returns to the base station using the same trajectory travelled before. Also, it is possible to try to turn on a new connection.
- *Action Priority after suggested prevention/detection action:* L

7.8.2 Radio receiver

The radio receiver receives data from the base station to the rover (main CPU). The failure analysis for this component is:

- *Failure mode*: break of the hardware.
- *Failure cause*: electrical failure.
- *Failure effect*: the rover is isolated from the base station. It can not receive any command.
- *Severity*: 10
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.

- *Action Priority*: H
- *Suggested prevention/detection action*: in order to check the connection, both the rover and the base station send an acknowledge if they receive a message from the others. If the acknowledge is not sent by the base station the rover stops or returns to the base station using the same trajectory travelled before. Also, it is possible to try to turn on a new connection.
- *Action Priority after suggested prevention/detection action:* L

7.8.3 Transmission logic

The transmission logic is connected both to the transmitter and the receiver. It controls all the transmission from and to the base station. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the logic can not receive and transmit any data.
- *Failure cause*: leaving of the transmission range.
- *Failure effect*: the rover is isolated from the base station.
- *Severity*: 10
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: H
- *Suggested prevention/detection action*: in order to check the connection both the rover and the base station send an acknowledge if they receive a message from the other. If the acknowledge is not sent by the base station the rover stops or returns to the base station using the same trajectory travelled before. Also, it is possible to try to turn on a new connection.
- *Action Priority after suggested prevention/detection action:* L

The analysis of the second failure is:

- *Failure mode*: the logic doesn't send and receive any data.
- *Failure cause*: electrical failure.

- *Failure effect*: the rover is isolated from the base station.
- *Severity*: 10
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: H
- *Suggested prevention/detection action*: in order to check the connection, both the rover and the base station send an acknowledge if they receive a message from the other. If the acknowledge is not sent by the base station the rover stops or returns to the base station using the same trajectory travelled before. Also it is possible to try to turn on a new connection.
- *Action Priority after suggested prevention/detection action:* L

7.9 (Logic system) Cameras system

In this section the components analysed are:

- Stereoscopic camera (manual drive)
- Stereoscopic camera (autonomous drive)
- Tool camera

Following the analysis for each component:

7.9.1 Stereoscopic camera (manual drive)

The stereoscopic camera generates video feedback of the rover's motion. It is usable by the user as video feedback or to create a point cloud of the environment usable by the autonomous navigation. The failure analysis for this component is:

- *Failure mode*: the camera doesn't send any data of the environment.
- *Failure cause*: electrical failure or direct hits.
- *Failure effect*: the user loses the video feedback and the points cloud.
- *Severity*: 7
- *Occurency*: 2

- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: the only detection action can be done by the user. It can drive the rover without the video feedback of the stereoscopic camera using only the other telemetry feedbacks.
- *Action Priority after suggested prevention/detection action:* M

7.9.2 Stereoscopic camera (autonomous drive)

The stereoscopic camera generates video feedback of the rover's motion. It is usable by the user as video feedback or to create a point cloud of the environment usable by the autonomous navigation. The failure can be different.
The analysis of the first failure is:

- *Failure mode*: the camera doesn't send any data of the environment.
- *Failure cause*: electronic failure or direct hit.
- *Failure effect*: the autonomous driver loses the points cloud.
- *Severity*: 8
- *Occurency*: 2
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: the autonomous navigation algorithm checks if the points cloud is unstable or not updated. If it is not, the rover goes in a stable configuration (rover stop) and it takes the control to the user.
- *Action Priority after suggested prevention/detection action:* L

The analysis of the second failure is:

- *Failure mode*: the camera sends an incorrect representation of the environment.
- *Failure cause*: electrical failure or direct hit.

- *Failure effect*: the autonomous navigation, due to incorrect feedbacks produces incorrect behaviour of the rover.
- *Severity*: 8
- *Occurency*: 4
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: H
- *Suggested prevention/detection action*: the autonomous navigation algorithm checks if the points cloud is unstable or not updated. If it isn't the rover goes in a stable configuration (rover stop) and it takes control to the user. Also, it is possible to integrate a proximity sensors that can be used as an alternative to the points cloud.
- *Action Priority after suggested prevention/detection action:* M

7.9.3 Tool camera

The tool camera generates the video feedback of the end effector. The failure analysis for this component is:

- *Failure mode*: the camera doesn't send any data to the main CPU.
- *Failure cause*: electrical failure or direct hits.
- *Failure effect*: the user loses the video feedback of the end effector.
- *Severity*: 7
- *Occurency*: 3
- *Detection index*: 10
- *Current detection action*: no action.
- *Action Priority*: M
- *Suggested prevention/detection action*: any detection action is improved because the user can uses the tool with the other telemetry feedback as stereoscopic camera and proximity sensors of the Tool Center Point.

8 Recap of the analysis

The following tables recap the analysis for all the components. The initials used means:

- **S**: Severity
- **O**: Occurrency
- **AP**: Action Priority
- **AP a.d.a.**: Action Priority after detection action

8.1 (Mobility system) Wheels system

Component	**Failure mode**	**S**	**O**	**AP**	**AP a.d.a.**
Brushless motor	coils' wires burn	8	2	M	L
Clutch	electrical part fail	7	2	M	L
Gear reduction	gear fail	8	2	L	L
Grouser	grouser slip on the terrain	6	10	M	M
Grouser	break connection with external skin	8	2	L	L
Main grouser	break connection with external skin	8	2	M	L
External skin	plastic deformation of the steel	4	2	L	L
External/internal cylinder	permanent deformation of the steel	7	3	M	L
Wheel's hub	break of polymeric structure	8	2	M	L
Slewing ring	break of polymeric structure	8	2	M	L
Motor encoder	incorrect signal output	5	4	L	L
Motor encoder	no signal output	8	2	L	L
Motor Hall sensors	incorrect signal output	5	4	L	L
Motor Hall sensors	no signal output	8	2	L	L
Gear reduction encoder	incorrect signal output	5	4	L	L
Gear reduction encoder	no signal output	8	2	L	L

8.2 (Mobility system) Steering system

Component	**Failure mode**	**S**	**O**	**AP**	**AP a.d.a.**
Bracket	permanent deformed	7	3	M	M
Bracket	very permanent deformed	8	2	M	M
Stepper motor	coils' wires burn	8	2	M	L
Gear reduction (stepper motor)	gear fail	8	2	L	L
Motor encoder	incorrect signal output	5	4	L	L
Motor encoder	no signal output	8	2	L	L
Reduction shaft encoder	incorrect signal output	5	4	L	L
Reduction shaft encoder	no signal output	8	2	L	L
Steering support on Rocker-Bogie	break of polymeric structure	8	2	M	M

8.3 (Mobility system) Rocker-Bogie system

Component	**Failure mode**	**S**	**O**	**AP**	**AP a.d.a.**
Torsion bar encoder	incorrect signal output	5	4	M	M
Torsion bar encoder	no signal output	8	2	M	M
Bogie joint encoder	incorrect signal output	5	4	M	M
Bogie joint encoder	no signal output	8	2	M	M
Slewing ring of hub rocker chassis	break of polymeric structure	8	2	M	M
Hub rocker chassis	break or deformation of steel	7	2	M	M
Rocker arm	break or deformation of steel	8	2	M	M
Torsion bar	break of the structure	8	2	M	M
Slewing ring torsion bar	break of the structure	8	2	M	M

8.4 Arm system

Component	**Failure mode**	**S**	**O**	**AP**	**AP a.d.a.**
Slewing ring base	failure of the steel structure	8	2	M	M
Gearwheel base	break or round teeth	8	2	M	M
Support plate of stepper motor	fail the steel support	8	2	M	M
Master/slave pinion gears	break of the gears	8	2	M	M
Arm structure	permanent steel deformation	7	3	M	M
Hub elbow	break of the component	8	2	M	M
Forearm	permanent steel deformation	7	3	M	M
Wrist hub	break of polymeric structure	8	2	M	M
Wrist center part	break of polymeric structure	7	2	M	M
Wrist-tool interface	break of polymeric structure	8	2	M	M
Base stepper motor	coils' wires burn	7	2	M	L
Base stepper motor reduction	fail of the gears	7	2	M	M
Base stepper motor encoder	incorrect signal output	5	4	M	M
Base stepper motor encoder	no signal output	8	2	M	M
Shoulder stepper motor	coils' wires burn	7	2	M	L
Shoulder stepper motor reduction	fail of the gears	7	2	M	M
Shoulder stepper motor encoder	incorrect signal output	5	4	M	M

Shoulder stepper motor encoder	no signal output	8	2	M	M
Elbow stepper motor	coils' wires burn	7	2	M	L
Elbow stepper gear reduction	fail of the gears	7	2	M	M
Elbow stepper motor encoder	incorrect signal output	5	4	M	M
Elbow stepper motor encoder	no signal output	8	2	M	M
First dof of wrist servomotor	coils' wires burn	6	2	L	L
First dof of wrist servomotor	gear reduction fail	6	3	L	L
First dof of wrist servomotor	MCU/encoder fail	6	2	L	L
Second dof of wrist servomotor	coils' wires burn	6	2	L	L
Second dof of wrist servomotor	gear reduction fail	6	3	L	L
Second dof of wrist servomotor	MCU/encoder fail	6	2	L	L
Third dof of wrist servomotor	coils' wires burn	6	2	L	L
Third dof of wrist servomotor	gear reduction fail	6	3	L	L
Third dof of wrist servomotor	MCU/encoder fail	6	2	L	L
Pliers' lead screw support	break polymeric structure	8	3	M	M
Pliers' structure	break polymeric structure	8	3	M	M
Grabber's brush motor	coils' wires burn	8	3	L	L
Grabber's lead screw support	break polymeric structure	8	3	M	M
Grabber's structure	break polymeric structure	8	3	M	M

8.5 (Logic system) Mobility/arm logic

Component	Failure mode	S	O	AP	AP a.d.a.
Driver encoder logic	incorrect signal output	7	2	M	L
Driver encoder logic	no signal output	8	2	M	L
Driver wheel's motor	incorrect signal output	7	2	M	M
Driver wheel's motor	no signal output	8	2	M	L
Mobility controller	no signal output	8	2	M	L
Mobility controller	incorrect signal output	8	2	M	L
Arm controller	no signal output	8	2	M	L
Arm controller	incorrect signal output	8	2	M	M

8.6 (Logic system) Mobility/arm logic

Component	Failure mode	S	O	AP	AP a.d.a.
Main CPU	no signal output	10	2	H	L
BMS	logic fail	10	2	H	H
BMS	electrical part fail	8	2	M	M
Ethernet switch	failure logic	8	2	M	L

8.7 (Logic system) Mobility/arm logic

Component	Failure mode	S	O	AP	AP a.d.a.
IMU(manual drive)	no signal output	5	2	L	L
IMU(manual drive)	incorrect signal output	8	2	M	M
IMU(autonomous drive)	no signal output	8	2	M	L
IMU(autonomous drive)	incorrect signal output	10	2	H	M

8.8 (Logic system) Transmission system

Component	**Failure mode**	**S**	**O**	**AP**	**AP a.d.a.**
Radio transmitter	hardware breaks	10	2	H	L
Radio receiver	hardware breaks	10	2	H	L
Transmission logic	no transmit/receive data(out of range)	10	2	H	L
Transmission logic	no transmit/receive data(logic failure)	10	2	H	L

8.9 (Logic system) Transmission system

Component	**Failure mode**	**S**	**O**	**AP**	**AP a.d.a.**
Stereoscopic camera(manual drive)	no signal output	7	2	M	M
Stereoscopic camera(autonomous drive)	no signal output	8	2	M	L
Stereoscopic camera(autonomous drive)	incorrect signal output	8	4	H	M
Tool camera	no signal output	7	3	M	M

9 Conclusion and results

In this section are shown the results of the FMEA analysis.
The aim of this analysis is to reduce the Action Priority of more danger failures to L and the others to M. Action Priority equal to H must be avoided.
In this case, most of the electronic failures have Action Priority reduced to L or M, only BMS logical failure has Action Priority equal to H because it is not possible to open the circuit.
This type of failure is very rare then no more action is improved to reduce the Action Priority.
All the mechanical failures have Action Priority equal to M because any detection or mitigation action can be applied to them. It is assumed that were well designed and mechanical failure analysis was previously applied to them.

Part V
Mobility controller

10 Introduction

The mobility controller is the logic part that interacts between the main logic and the drivers of the steering and wheel motor. It calculates the correct steering angles and angular velocities of the wheels given as input the desired velocity of the rover's centre of mass in polar coordinates, the feedback of the steering and wheel motors and of the rocker-bogie system.
Following the recap of the input/output of the controller.
Controller input:

- **Sigma_deg:** it is the desired angle of the CM velocity module respect longitudinal axis of the rover. It can span between $-5°$ and $+5°$, to avoid the steering on one of the central wheel that can produce bigger stress on its.
- **Vtan:** it is the desired tangential velocity normalized of the CM respect the motion direction of the rover, depending on the steering angle of the wheels. It spans between -1 and +1 to have positive and negative velocity independently to the angle *Sigma_deg*.
- **Error:** it is a flag, calculated by the main CPU or the failure detection software, that represent a critical condition of the mobility system or upper-level logic. It is equal to 1 if something it is wrong otherwise it is equal to 0 and it means that all the components work in a proper manner.
- **URC/ERC:** it is a flag that customizes the maximum velocity reachable by the rover. If this flag is equal to 1 means that the URC mode is active (maximum velocity equal to $1m/s$) otherwise if it is equal to 0 the desired mode is ERC (maximum velocity equal to $0.5m/s$). This flag can be changed during normal work.
- **Parking_flag:** it is the flag that selects the parking mode. It is equal to 1 the parking mode is enabled.
- **Enable_correction:** it is a flag that enables the correction given by the mobility controller to the calculated wheels angular velocity to adapt them from a 2 dimension model to a 3 dimension model. If the flag is equal to 1 the correction is enabled.
- **Omega_feed_RPM:** it is the wheels motor feedback angular velocity given by the encoders of the wheels (array of 6 elements). It is used

by the controller to know the actual condition of the motors and if all of them work in a good way.

- **Alpha_feed_rad:** it is the steering motors feedback encoders of the wheels (array of 6 elements). It is used by the controller to know the actual steering condition and if all the motors work in a good way.
- **beta_feed_encoder_rad:** it is the feedback angle of the left rocker respect nominal situation (positive counterclockwise, left side view).
- **rho1_feed_encoder_rad:** it is the feedback angle of the left bogie respect nominal situation (positive counterclockwise, left side view).
- **rho_feed_encoder_rad:** it is the feedback angle of the right bogie respect nominal situation (positive counterclockwise, left side view).

Controller output:

- **Omega_wheel_sat_RPM:** it is the command sends by the controller to the motors drivers (array of 6 elements). Those velocities are saturated to have time by time the correct ratio between them to avoid the slipping of one or more of them. The wheel with the bigger speed needs more time to reach the needed velocity respect the slowest, but to have correct dynamic the settling time must be the same.
- **Alpha_rad:** it is the command sends by the controller to the steering motors drivers (array of 6 elements). Similarly to the other output, the final steering angles are reached with a stair shape.
- **Flag_steer:** it is a flag that represents if the steering system is busy or not. If it is equal to 1 means that the steering system is active or the rover is in a transition state (from *Running mode* to *Parking mode* or vice versa).

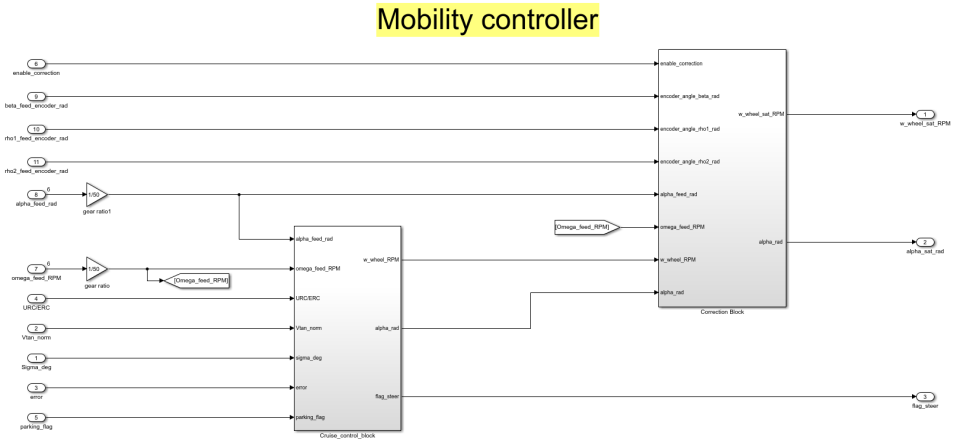


Figure 15: Mobility controller Simulink scheme

11 Cruise control Block

The Cruise control block is the subsystem that given the command inputs (*Vtan*,*Sigma_deg*,*Error*,*Parking_flag* and *URC/ERC*) and the feedback (*omega_feed_RPM* and *alpha_feed_rad*) decides which mode is activated and the needed angular velocity and steering angle for each wheels.

Important note: *both the feedbacks are related to the wheels and not to the motors, then before enter in the Cruse control block are divided by the gear reduction ratio.*

This block is divided in 2 others subsections:

- **Decision_logic_v3**.
- **Dynamic_block**.

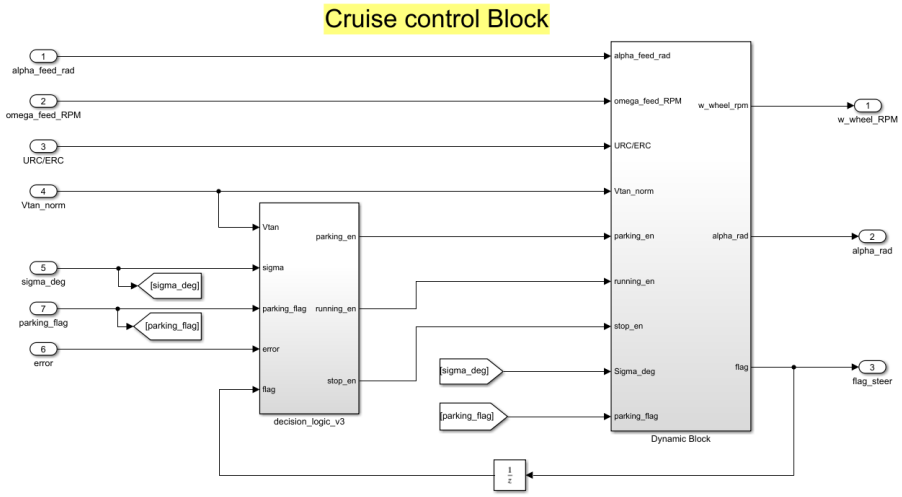


Figure 16: Cruise control Simulink scheme

11.1 Decision logic block

This subsection decides which mode is activated taking into account the following inputs:

- *Vtan*: tangential velocity of the CM.
- *Error*: error flag given by the main CPU or fail detection function.
- *Parking_flag*: enable flag of the parking mode.
- *Busy_flag*: flag that takes into account if the rover can or not change its mode and if it is in a transient condition.
- *Delta_sigma*: it is the difference between the current *Sigma_deg* value an the previous. This value is an absolute value.

The output of this section are:

- *Parking_en*: enable of parking mode.
- *Running_en*: enable of running mode.
- *Stop_en*: enable of stop mode.

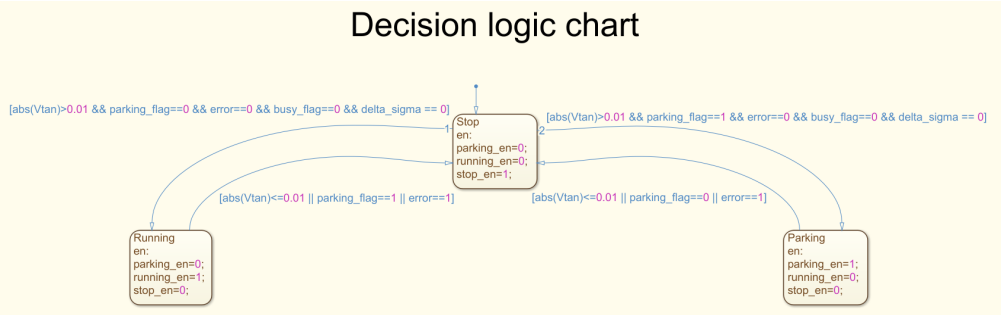


Figure 17: Decision logic Stateflow scheme

This component is a states machine with 3 states:

1. *Stop*: it is the initial state that sets as 1 the *Stop_en* and 0 the others. The rover remains stucked.

2. *Running*: it is the state that sets to 1 the *Running_en* and 0 the others. The rover is driven with the instantaneous centre of rotation outside from the rover.

3. *Parking*: it is the state that sets to 1 the *Parking_en* and 0 the others. The rover turns around itself.

The transition between the states are:

- From *Stop* to *Running*:
  - absolute value normalized of *Vtan* must be greater respect 0.01. It means that the velocity must be greater than 1% of the maximum velocity. In ERC configuration 1% is $5mm/s$, whereas for URC is $10mm/s$.
  - *parking_flag* must be equal to 0.
  - *busy_flag* must be equal to 0.
  - *error* must be equal to 0.
  - *delta_sigma* must be equal to 0.

  All the cases are connected by the logic operator *and*.

- From *Running* to *Stop*:
  - absolute value normalized of *Vtan* must be lower respect 0.01.

- *parking_flag* must be equal to 1.
- *error* must be equal to 1.

All the cases are connected by the logic operator *or*.

- From *Stop* to *Parking*:
  - absolute value normalized of *Vtan* must be greater respect 0.01.
  - *parking_flag* must be equal to 1.
  - *error* must be equal to 0.
  - *busy_flag* must be equal to 0.
  - *delta_sigma* must be equal to 0.

  All the cases are connected by the logic operator *and*.

- From *Parking* to *Stop*:
  - absolute value normalized of *Vtan* must be lower respect 0.01.
  - *parking_flag* must be equal to 0.
  - *error* must be equal to 1.

  All the cases are connected by the logic operator *or*.

This component is crucial for the safety of the rover because in case of problems on the upper-level logic or of the mobility system the rover goes immediately to *Stop* mode. Also during the transition between *Running* to *Parking* or vice versa the rover goes to *Stop* mode in order to set correctly the steer and avoid damage to the steering and wheels system.

11.2 Dynamic block

The dynamic control calculates the correct angular velocity and steering angle for each wheel related to the mode that is activated.
The inputs of this section are:

- *alpha_feed_rad*: wheels feedback steering angles (array of 6 elements).
- *omega_feed_RPM*: wheels feedback angular velocities (array of 6 elements).
- *URC/ERC*: flag that select the the maximum velocity of the rover.
- *Vtan*: tangential velocity normalized of the CM.
- *parking_en*: enable of the subsection *Parking_mode*.
- *running_en*: enable of the subsection *Running_mode*.

- *Stop_en*: enable of the subsection *Stop_mode*.
- *Sigma_deg*: angle of the center of mass velocity respect longitudinal axis of the rover.
- *Parking_flag*: flag that activates the signal of parking (input of the controller given by the webapp).

The outputs of this section are:

- *w_wheel_RPM*: angular velocity of the wheels calculated by the Akermann model (array of 6 elements).
- *alpha_rad*: steering angle of the wheels calculated by the Ackermann model (array of 6 elements).
- *busy_flag*: it is the flag that represents the transient condition. If *busy_flag* is equal to 1 it means that the transition condition is activated.

  The *Dynamic block* is divided in 3 parts, one for each mode:

  – Parking_mode.
  – Running_mode.
  – Stop_mode.

  Those 3 parts are an "enable subsystem" block, then time by time only one of them is activate thanks to the *Decision_logic* block.

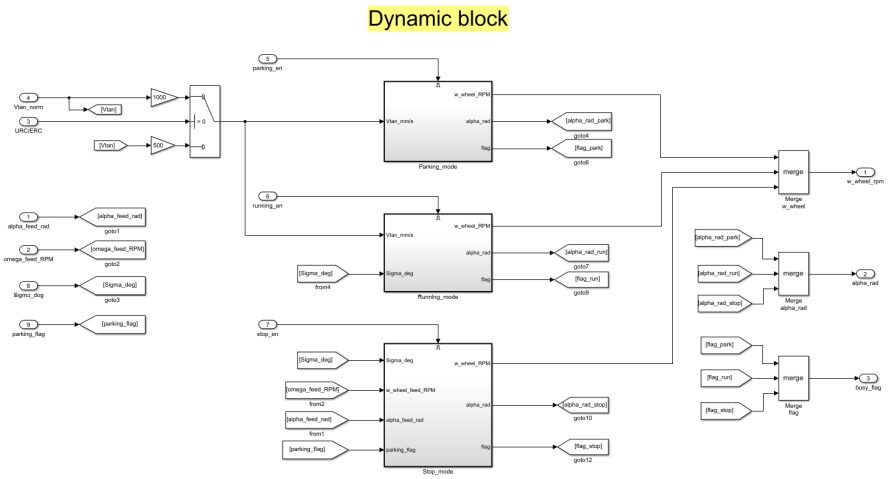


Figure 18: Dynamic block Simulink scheme

11.2.1 Parking_mode

The parking mode is activated when the rover's centre of rotation is inside of it, in a more simple way, when the central wheels have not the same direction of motion.
After some simulation and testing, it is decided to collapse the working range only on the turning around itself because the behaviour has not a significant reduction and also the stress acting on both the steering system and the wheels is the minimum. In order to have minimum stress, all the wheels must have a velocity different from zero and the steering angle is less than 65◦ respect the longitudinal axis of the rover.
Taking into account the previous concept, the input command *sigma_deg* is a constant value equal to 90◦ and the tangential velocity *Vtan* can spans between the minimum and maximum value. This range is selected by the command flag *URC/ERC*. If the flag is 1 means that the needed maximum velocity available by the rover is between $-1000mm/s$ and $+1000mm/s$ otherwise the range is between $-500mm/s$ and $+500mm/s$.
To avoid the tip-over of the rover when it is in parking mode, the tangential velocity is divided by $8 * 10^{17}$ because this condition is near the singularity of the Ackermann model*[*]*.
The *busy_flag* is always 0 because parking mode does not cover transient condition (parking mode is activated only if the wheels have the desired steering angles).

[*] The singularity of the Ackermann model is present because in "Velocity_Block" there is a division by $cos(Sigma\_rad)$. In this case, is 0, then the velocity of the wheels is near to infinite because the CM has a distance from the centre of the Ackermann model. Thanks to this gain the velocity of the wheels is in the correct range of values.

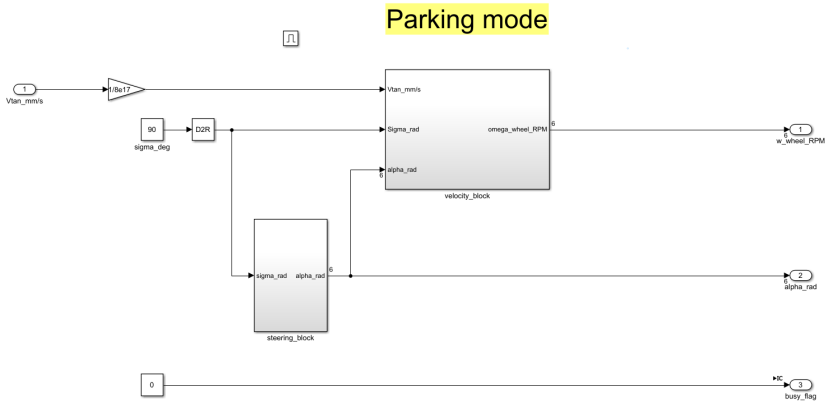


Figure 19: Parking mode block Simulink scheme

The component *velocity_block* and *steering_block* are showed later.

11.2.2 Running_mode

The running mode is activated when the rover's centre of rotation is outside of it. In this condition it is possible to change the direction of the rover without stop it, but only if the command input must be inside the operating range of this mode.
The range of the input tangential velocity $Vtan$ is equal to the parking mode and depends on the flag *URC/ERC*.
The range of the second input, *sigma_deg*, must be between $-5\circ$ and $+5\circ$. This value is decided to have the minimum distance of the centre of rotation around $400mm$ from the middle of the central wheel.
The input $Vtan$ is tuning coherently with the Sigma_rad value to have a velocity that decreases inverse proportionally with the Sigma_rad value: $Vtan = Vtan * (1 - |sin(Sigma\_rad * 5)|)$. This tuning is done to have a deceleration of the rover when it steers to avoid the over tip of it.

In this mode, the output *busy_flag* is always equal to 0 because it does not cover the transient condition.

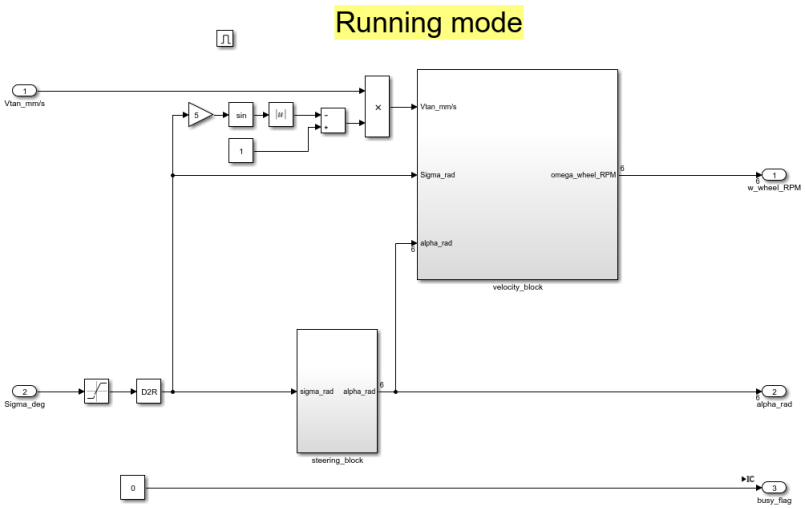


Figure 20: Running mode block Simulink scheme

The component *velocity_block* and *steering_block* are showed later.

11.2.3 Stop_mode

The Stop mode is the stable state of the rover. In this condition, the wheels can steer both with angles of parking and running mode without delay due to deceleration because the rover is stopped.
The first block called *sigma_block* decides if the input sigma value is inside the range of running or parking mode, then saturates it and transmit this value to the *steering_block*.
The last block called *decision_logic_stop* selects the correct steering angles and feedback busy_flag.
The component *velocity_block*, *steering_block* and *decision_logic_stop* are showed later.

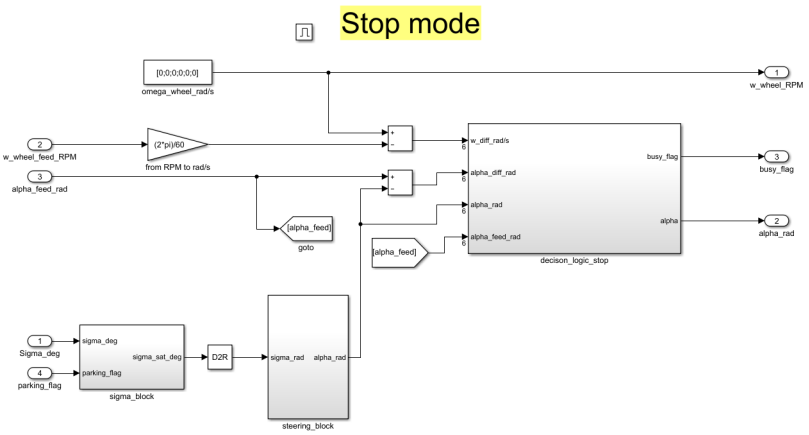


Figure 21: Stop mode block Simulink scheme

11.2.4 Decision_logic_stop

This component is used inside the *Stop_mode* subsystem and it selects the correct values of the steering angle, *alpha*, and *busy_flag*.

Its inputs are:

- *w_diff_rad_s*: it is the difference between the feedback wheels angular velocity and the desired that, in this case, it is equal to 0 for each wheel (array of 6 elements).
- *alpha_diff*: it is the difference between the feedback steering angles and the desired ones (array of 6 elements).
- *alpha_rad*: it is the desired steering angles (array of 6 elements).
- *alpha_feed*: it is the feedback steering angles (array of 6 elements).

The outputs are:

- *alpha*: it is the steering angle for each wheel coherently with the actual state (array of 6 elements).
- *busy_flag*: it is the flag that takes a look if the rover is in a transient condition or not. If it is equal to 1 means that the rover is busy otherwise is equal to 0.

This block is a 4 states machine in which the states are:

- *begin*: it is the initial state of the chart. It is activated only when the controller starts to have a safety state. In this state the rover does nothing because *busy_state* is equal to 1 and *alpha* = *alpha_feed*.
- *rover_stop*: it is the neutral state (stable state) of the machine. In this state, the rover is stopped and it is able to steer the wheels. Also thanks to the *busy_flag* equal to 0 it is possible to change mode but during the permanency *alpha* = *alpha_feed*.
- *rover_steer*: in this state the rover is steering then the *busy_flag* is equal to 1 and *alpha* = *alpha_rad*. Thanks to this state is possible to split the mode from running to parking and vice versa.
- *rover_deceleration*: in this state the rover decelerate to have wheels velocity equal to 0 and it can not steer because *busy_flag* equal to 1 then *alpha* = *alpha_feed*. Thanks to this state is possible to split the mode from running to parking and vice versa.

The transition between the states are:

- From *begin* to *rover_stop*:
  the maximum of the absolute of *w_diff_rad_s* must be lower or equal to 0.1 and the maximum of the absolute of *alpha_diff* must be lower or equal to 0.035.
- From *begin* to *rover_steer*:
  the maximum of the absolute of *w_diff_rad_s* must be lower to 0.1 and the maximum of the absolute of *alpha_diff* must be bigger to 0.035.
- From *rover_stop* to *rover_steer*:
  the maximum of the absolute of *w_diff_rad_s* must be lower respect 0.1 and the maximum of the absolute of *alpha_diff* must be bigger to 0.035.
- From *rover_steer* to *rover_stop*: the maximum of the absolute of *alpha_diff* must be lower or equal to 0.035.

- From *rover_stop* to *rover_deceleration*:
  the maximum of the absolute of *w_diff_rad_s* must be greater or equal to 0.1.

- From *rover_deceleration* to *rover_steer*:
  the maximum of the absolute of *w_diff_rad_s* must be lower to 0.1.

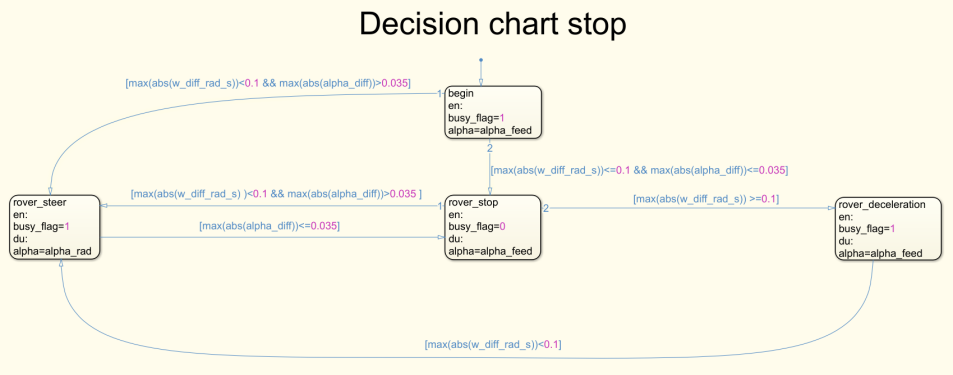


Figure 22: Decision chart stop Stateflow scheme

11.2.5 Steering_block

This subsystem is used inside all the 3 modes (Parking, Running and Stop) to calculate the correct steering angle of each wheel taking into account the polar angle of the CM velocity respect the longitudinal axis of the rover (Sigma_deg).

The steps in this block are:

1. Calculate curvature radius of the CM with the formula:
   $r = f/sin(Sigma\_rad)$. [1]

2. Saturate the calculated curvature radius to avoid infinite value, it can produce overflow, when *Sigma_rad* is equal to 0.

3. Calculate the steering angle for each wheel:

   - Wheel1: $alpha[1] = atan(-d/(r * cos(Sigma|_r ad) + a/2)$
   - Wheel2: $alpha[2] = atan(-d/(r * cos(Sigma\_rad) - a/2)$
   - Wheel3: $alpha[3] = 0$
   - Wheel4: $alpha[4] = 0$

[1] Rocker-bogie dimensions (f,a,b,c,d,e,f) are explained in the section "Rocker-bogie system dimensions (usable by the Ackermann model)"

- Wheel5: $alpha[5] = atan(e/(r * cos(Sigma\_rad) + c/2)$
- Wheel6: $alpha[6] = atan(e/(r * cos(Sigma\_rad) - c/2)$.

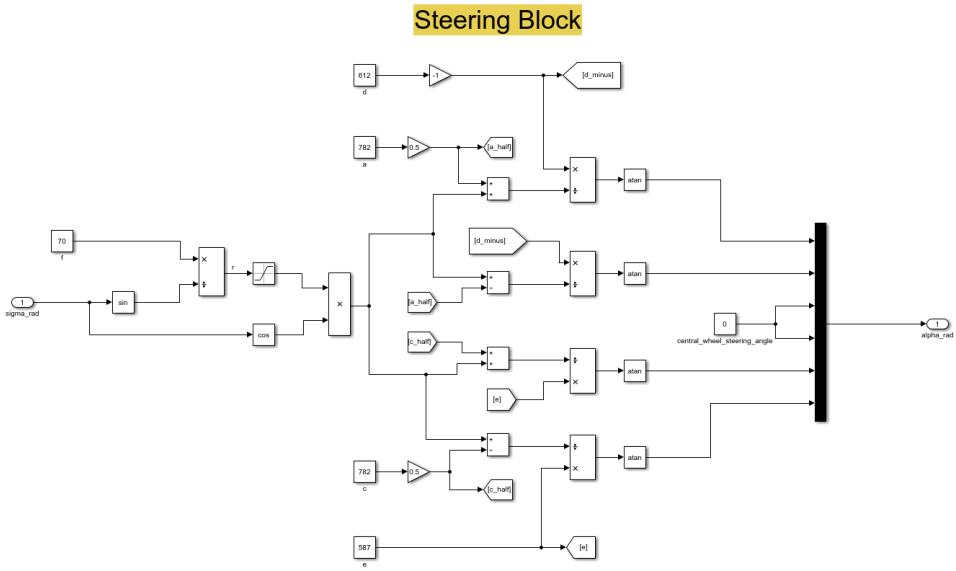


Figure 23: Steering_block Simulink scheme

11.2.6 Velocity_block

This subsystem is used in Parking and Running mode to calculate the correct velocity for each wheel.
The steps in this block are:

1. Calculate curvature radius of the CM with the formula:
   $r = f/sin(Sigma\_rad)$.

2. Saturate the curvature radius of the CM to avoid problem when
   $Sigma\_rad = 0$.

3. Calculate curvature radius for each wheel:

   - Wheel1: $cur_radius[1] = [r*cos(Sigma\_rad)+a/2]/[cos(alpha[1])]$
   - Wheel2: $cur_radius[2] = [r*cos(Sigma\_rad)-a/2]/[cos(alpha[2])]$
   - Wheel3: $cur_radius[3] = r * cos(Sigma\_rad) + b/2$
   - Wheel4: $cur_radius[4] = r * cos(Sigma\_rad) - b/2$
   - Wheel5: $cur_radius[5] = [r*cos(Sigma\_rad)+c/2]/[cos(alpha[5])]$
   - Wheel6: $cur_radius[6] = [r*cos(Sigma\_rad)-c/2]/[cos(alpha[6])]$.

4. Calculate angular velocity around center of rotation of the rover: $omega\_CR = Vtan * sign[cos(Sigma\_rad)]/[r * cos(Sigma\_rad)]$.

5. Calculate the velocity of each wheel: $v\_wheel[i] = omega\_CR * cur\_radius[i]$.

6. Calculate the angular velocity of each wheel: $w[i] = v\_wheel[i]/wheel\_radius$.

7. Convert the wheels angular velocity from rad/s to RPM: $omega\_wheel\_rad/s * 60/(2 * pi\_greco)$.

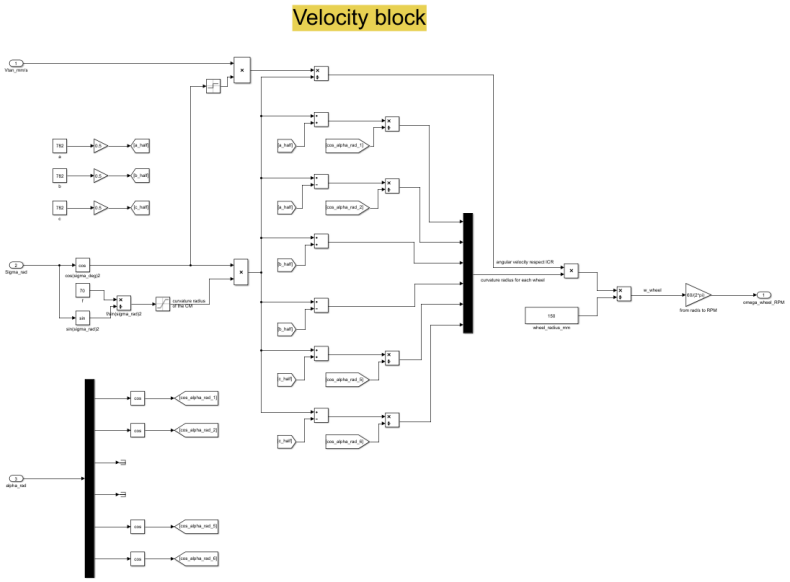


Figure 24: Velocity_block Simulink scheme

12 Correction Block

This system corrects and saturates the wheels angular velocity taking into account the dynamic of the rocker-bogie system. This correction is important because in *Cruise_control_block* the rover is modelled using the Ackermann model, like a car. This model is a 2-dimensional model, because doesn't take a look the suspension dynamic.
But the working terrain of the rover is off-road then to avoid the slipping of the wheels when it overpasses a hill or an obstacle it is needed a 3-dimensional model. In *Correction_block* both the problems are solved.

The inputs of this block are:

- *Enable_correction*: it is a flag sent by the user to activate or not the correction from 2D to 3D model (if it is 1 the correction is enabled).
- *encoder_angle_beta_rad*: it is the feedback of the torsion bar encoder.
- *encoder_angle_rho1_rad*: it is the feedback of the left bogie joint encoder.
- *encoder_angle_rho2_rad*: it is the feedback of the right bogie joint encoder.
- *alpha_feed_rad*: it is the feedback steering angle of the wheels (array of 6 elements).
- *w_wheel_RPM*: it the desired wheels angular velocity calculated by the 2D model (array of 6 elements).

The output of the block is the wheels angular velocity corrected in 3D model *omega_correction_RPM* (array of 6 elements).

This system is divided into 3 subsystem:

- *From_2D_to_3D_model*
- *ratio_block_omega*
- *ratio_block_alpha*.

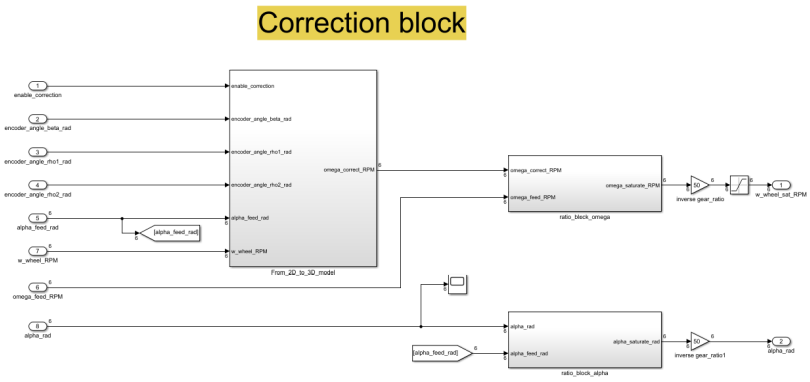


Figure 25: Correction_block Simulink scheme

12.1 From 2D to 3D Block

This block corrects the desired angular velocity of the wheels taking into account the variation of the Rocker-bogie position.
Its subsystem are:

- *Trigonometric model of the suspension sys*
- *Omega_correction_block*

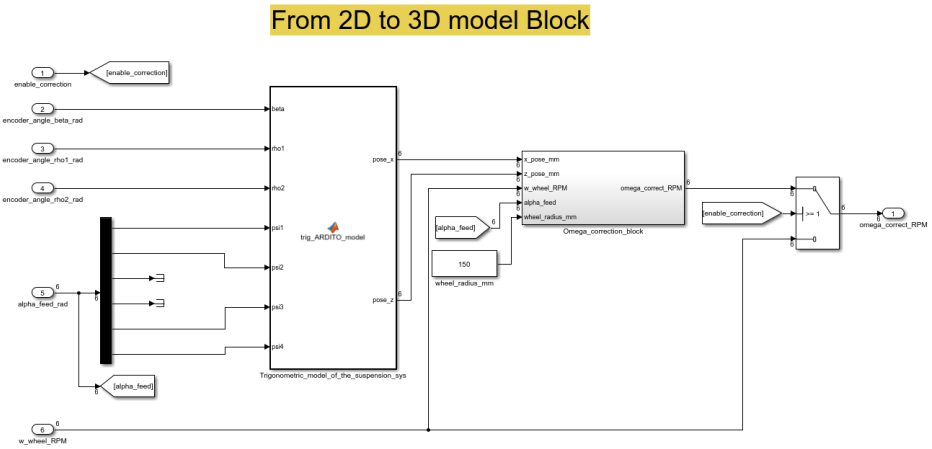


Figure 26: From_2D_to_3D_model_block Simulink scheme

12.1.1 Trigonometric model of the suspension sys

This function is based on the trigonometric model of the Rocker-bogie system explain more detailed in its section 14 "Rocker-bogie trigonometric model".
The inputs of the Matlab function are:

- *beta* (angle of the left rocker)
- *rho1* (angle of the left bogie)
- *rho2* (angle of the right bogie)
- *psi1* (steering angle of wheel1)
- *psi2* (steering angle of wheel2)
- *psi3* (steering angle of wheel5)
- *psi4* (steering angle of wheel6)

The output of the Matlab function are:

- *pose_x* (position along x respect CM of the wheels, array of 6 elements)
- *pose_z* (position along z respect CM of the wheels, array of 6 elements)

12.1.2 Omega correction block

This block calculates the correct angular velocity of each wheel to avoid the slipping of one or more of them caused by not flat terrain.
The inputs of this block are:

- *x_pose_mm*: position of the center of the wheels along x axis (array of 6 elements);
- *w_wheel_RPM*: desired angular velocity of the wheels in 2D (array of 6 elements);
- *wheel_radius_mm*: radius of the wheels;
- *alpha_feed_rad*: feedback steering angles (array of 6 elements);
- *z_pose_mm*: position of the center of the wheels along z axis (array of 6 elements).

The output of this block is the corrected wheels angular velocity *omega_correct_RPM* (array of 6 elements). The steps in this block are:

1. Corvert *w_wheel_RPM* from RPM to rad/s:
   $w\_wheel\_rad/s[i] = w\_wheel\_RPM[i] * (2 * \pi)/60$.
2. Derive the x and z position to have the velocities along the same axis (it is a discrete derivative):
   $V\_x\_model[i] = d/dt(x\_pose\_mm[i])$
   $V\_z\_model[i] = d/dt(z\_pose\_mm[i])$.
3. Calculate the components of the tangential velocity of the wheels along x and y axis:
   $V\_x[i] = (w\_wheel\_rad/s[i] * wheel\_radius\_mm) * cos(alpha\_feed\_rad[i])$
   $V\_y[i] = (w\_wheel\_rad/s[i] * wheel\_radius\_mm) * sin(alpha\_feed\_rad[i])$.
4. Subtract the velocity along x axis to longitudinal velocity for each wheels to find the real velocity along x axis of the wheels: $V\_x\_real[i] = V\_x[i] - V\_x\_model[i]$.

   ***Note**: in case of hill the wheel velocity given by the the previous model is negative, because using the fundamental triangle's theorem, the hypotenuse (the leg of the rocker for example) remain constant but the*

cathetus along z increase (because the wheel climbs the hill) then the cathetus along x decrease. In other words when the wheel 1, for example, climbs a hill the distance between wheel and rocker joint remain constant but change the component along the axis in a inverse proportional ratio.
We are interested to have the same velocity of the wheels along x, then if we subtract the x component of the velocity given by the trigonometric model of the rocker-bogie the result is the increasing of the angular velocity of the wheels that are climbing an obstacle and a reduction of it when the wheels are climbing down.

5. Calculate the corrected tangential velocity of the wheels:
   $V = sqrt(V\_x\_real^2 + V\_y^2 + V\_z\_model^2)$.

6. Calculate the correct angular velocity of the wheels and multiply them with the sign of the input angular velocity:
   $omega\_correct\_rad/s = (V * wheel\_radius\_mm) * sign(w\_wheel\_rad/s)$.

7. Transform the angular velocities of the wheels from rad/s to RPM:
   $omega\_correct\_RPM = omega\_correct\_rad/s * 60/(2 * \pi)$.

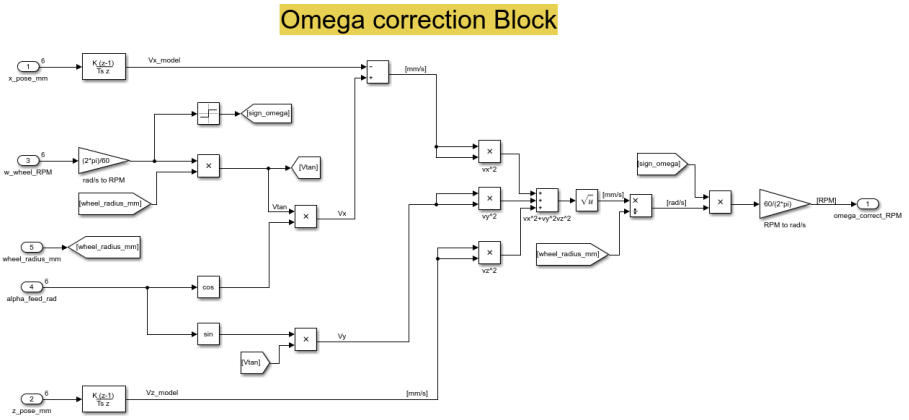


Figure 27: Omega_correction_block Simulink scheme

12.2 Ratio block omega

The previous block calculates the correct angular velocity for each wheel to avoid slipping of the wheels in off-road terrain. Those results can not be used by the motor driver controllers because they do not take a look the different dynamic between the wheels.
To have a correct behaviour of the rover all the wheels must have the same

settling time, in other words, both the faster and the lowest wheels must reach the corrected angular velocity at the same time.
In order to do that, this block maintains constant time by time the ratio between the wheels angular velocity.

The subsection in this system are:

- *ratio_function*: it calculates the saturated error of the Proportional-Integrative (PI) controller to have the same ratio time by time.
- *PI_block*: it calculates the reached wheels angular velocity with a PI controller with saturated tracking error.

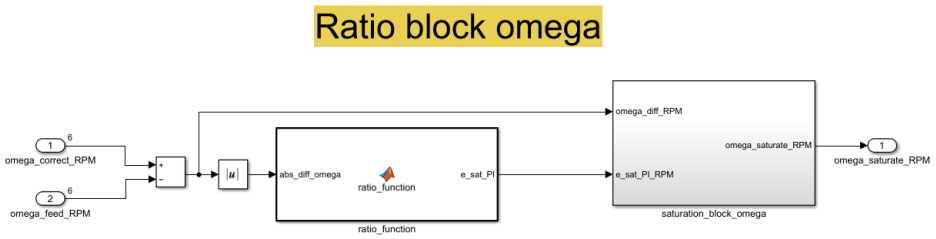


Figure 28: Ratio_block_omega Simulink scheme

12.2.1 Ratio function

This function calculates the tracking error saturated value for each wheel. The input of this function is the real tracking error between wheel calculated and feedback angular velocity.

The steps of this block are:

1. Find the maximum of the tracking errors module of the wheels.
2. Initialized the values e_sat_PI and *ratio* to an array of 6 elements equal to 0.
3. If the maximum funded in step 1. is greater than 0 for each wheel calculate the ratio between the module of the wheel angular velocities and the maximum calculated at point 1:
   $ratio[i] = abs\_diff\_omega[i]/max\_diff$. The saturated tracking error is:
   $e\_sat\_PI = e\_sat * ratio[i]$.

   Note: *it is possible to put a threshold different from 0 in order to avoid the previous computation if the velocity of the rover is too slow.*

```
function e_sat_PI  = ratio_function(abs_diff_omega)

max_diff=max(abs_diff_omega);
ratio=zeros(6,1);
e_sat_PI=zeros(6,1);
e_sat=3.5;
if(max_diff > 0)
    for k=1:6
        ratio(k)=abs_diff_omega(k)/max_diff;
        e_sat_PI(k)=e_sat*ratio(k);
    end
end

end
```

Figure 29: Ratio_function Matlab code

12.2.2 PI block

This block is composed by the PI controller of the wheels angular velocity.

The inputs of this block are:

- *omega_diff_RPM*: it is the difference between desired and feedback angular velocity of the wheels (array of 6 elements).
- *e_sat_PI_RPM*: it is the saturation of the PI controller of each wheel (array of 6 elements).

The output is the wheels angular velocity *omega_saturate_RPM* (array of 6 elements).
The single PI controller feels like not in a closed loop but it is closed before this system, at the beginning of *Ratio_function*.

The *Saturated_PI* block parameters are:

- Proportional = 0.5;
- Integrator = 4;
- Derivative = 0.

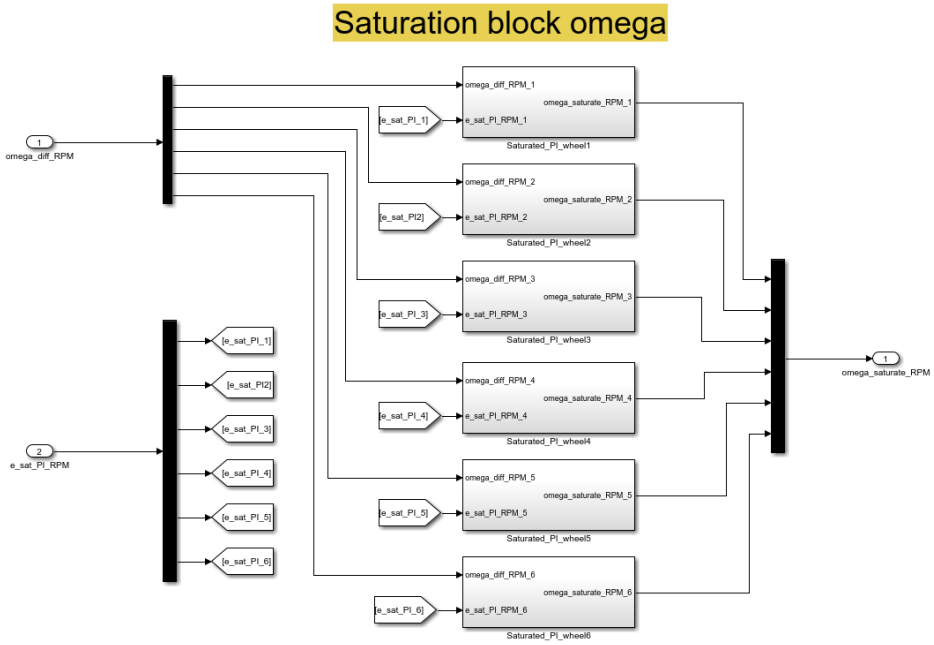


Figure 30: Saturation_block_omega Simulink scheme

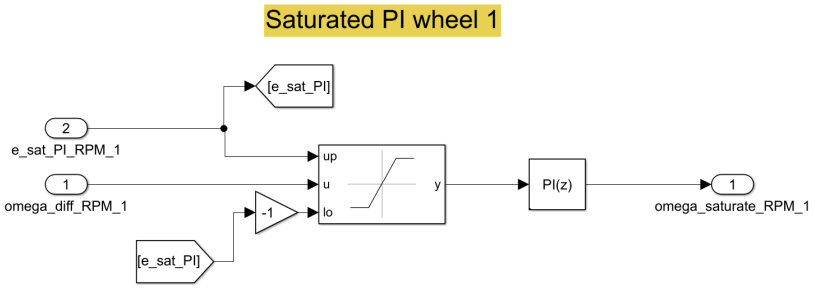


Figure 31: Saturation_PI_wheel1 Simulink scheme

12.3 Ratio block alpha

The functions of this block are the same as *Ratio_block_omega* but applied to the wheels steering angle.
In this case in *Ratio_function_alpha* the saturation *e_sat* is equal to 0.08.
The PI controller parameters are equal to the parameters for angular velocity controller.

13 Rocker-bogie trigonometric model

13.1 Reference frame and rocker-bogie angles

With this model is possible to know the position of the wheels centre along x and z-axis using the rotation angle of the rockers and bogies.
The reference frame axis positioned on CM are:

- x along longitudinal rover axis (positive from back to the front of the rover);
- y along transversal rover axis (positive coming out from the surface);
- z coherent with the right-hand rule (positive towards down).

The model is based on the rotation of each component of the rocker-bogie system respect the centre of mass of the rover.

Rocker-bogie angles are:

- β : left rocker angle
- $-\beta$: right rocker angle
- $\rho 1$: left bogie angle
- $\rho 2$: right bogie angle

All the angles are positive counter clockwise if they are viewed from the left side of the rover.

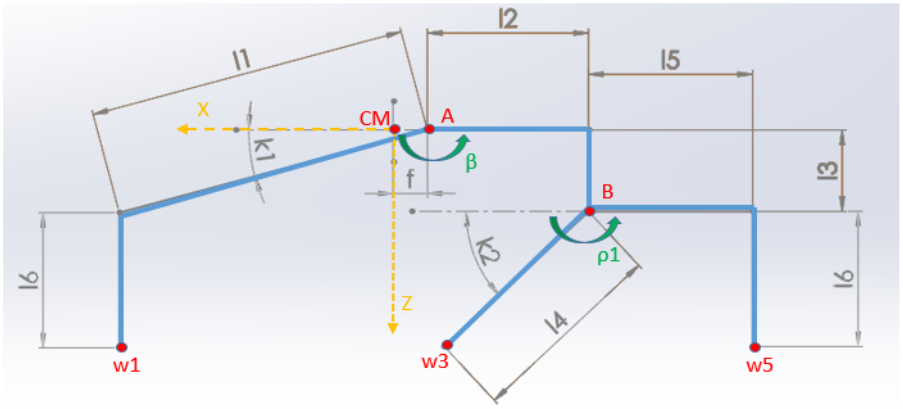


Figure 32: Left rocker-bogie, lateral view

13.2 Components homogeneous matrix

The rotations matrix for each component are:

- Left rocker (rotation around y axis):
$$R_{rockerL} = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix}$$

- Right rocker (rotation around y axis):
$$R_{rockerR} = \begin{bmatrix} cos(\beta) & 0 & -sin(\beta) \\ 0 & 1 & 0 \\ sin(\beta) & 0 & cos(\beta) \end{bmatrix}$$

- Left bogie (rotation around y axis):
$$R_{bogieL} = \begin{bmatrix} cos(\rho 1) & 0 & sin(\rho 1) \\ 0 & 1 & 0 \\ -sin(\rho 1) & 0 & cos(\rho 1) \end{bmatrix}$$

- Right bogie (rotation around y axis):
$$R_{bogieR} = \begin{bmatrix} cos(\rho 2) & 0 & sin(\rho 2) \\ 0 & 1 & 0 \\ -sin(\rho 2) & 0 & cos(\rho 2) \end{bmatrix}$$

- Wheel (rotation around z axis):
$$R_{wi} = \begin{bmatrix} cos(\phi_i) & -sin(\phi_i) & 0 \\ sin(\phi_i) & cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The homogeneous matrix for all the component are:

- Rocker left:
$$T_{RL}^{CM} = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) & -f \\ 0 & 1 & 0 & 0 \\ -sin(\beta) & 0 & cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rocker right:
$$T_{RR}^{CM} = \begin{bmatrix} cos(\beta) & 0 & -sin(\beta) & -f \\ 0 & 1 & 0 & 0 \\ sin(\beta) & 0 & cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Bogie left with respect to rocker left:

$$T_{BL}^{RL} = \begin{bmatrix} cos(\rho 1) & 0 & sin(\rho 1) & -l2 \\ 0 & 1 & 0 & 0 \\ -sin(\rho 1) & 0 & cos(\rho 1) & l3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Bogie right with respect to rocker right:

$$T_{BR}^{RR} = \begin{bmatrix} cos(\rho 2) & 0 & sin(\rho 2) & -l2 \\ 0 & 1 & 0 & 0 \\ -sin(\rho 2) & 0 & cos(\rho 2) & l3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Wheel 1 with respect to rocker left:

$$T_{w1}^{RL} = \begin{bmatrix} cos(\phi 1) & -sin(\phi 1) & 0 & l1 * cos(k1) \\ sin(\phi 1) & cos(\phi 1) & 0 & 0 \\ 0 & 0 & 1 & l6 - l1 * sin(k1) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Wheel 2 with respect to rocker right:

$$T_{w2}^{RR} = \begin{bmatrix} cos(\phi 2) & -sin(\phi 2) & 0 & l1 * cos(k1) \\ sin(\phi 2) & cos(\phi 2) & 0 & 0 \\ 0 & 0 & 1 & l6 - l1 * sin(k1) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Wheel 3 with respect to bogie left:

$$T_{w3}^{BL} = \begin{bmatrix} 1 & 0 & 0 & l4 * cos(k3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l4 * sin(k3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Wheel 4 with respect to bogie right:

$$T_{w4}^{BR} = \begin{bmatrix} 1 & 0 & 0 & l4 * cos(k3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l4 * sin(k3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Wheel 5 with respect to bogie left:
$$T_{w5}^{BL} = \begin{bmatrix} cos(\phi 3) & -sin(\phi 3) & 0 & -l5 \\ sin(\phi 3) & cos(\phi 3) & 0 & 0 \\ 0 & 0 & 1 & l6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Wheel 6 with respect to bogie right:
$$T_{w6}^{BR} = \begin{bmatrix} cos(\phi 4) & -sin(\phi 4) & 0 & -l5 \\ sin(\phi 4) & cos(\phi 4) & 0 & 0 \\ 0 & 0 & 1 & l6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally the homogeneous matrix of the wheels with respect to the CM are:
Note: following it's used 's' and 'c' as 'sin' and 'cos' in order to reduce the dimension of the matrices

- Wheel 1:
$T_{w1}^{CM} = T_{RL}^{CM} * T_{w1}^{RL}$
$$T_{w1}^{CM} = \begin{bmatrix} c(\beta) * c(\phi 1) & -c(\beta) * s(\phi 1) & s(\beta) & x1 \\ s(\phi 1) & c(\phi 1) & 0 & 0 \\ -c(\phi 1) * s(\beta) & s(\beta) * s(\phi 1) & c(\beta) & z1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
with:
$$x1 = -f + s(\beta) * (l6 + l1 * s(k1)) + l1 * c(\beta) * c(k1)$$
$$z1 = c(\beta) * (l6 + l1 * s(k1)) - l1 * c(k1) * s(\beta)$$

- Wheel 2:
$T_{w2}^{CM} = T_{RR}^{CM} * T_{w2}^{RR}$
$$T_{w2}^{CM} = \begin{bmatrix} c(\beta) * c(\phi 2) & -c(\beta) * s(\phi 2) & -s(\beta) & x2 \\ s(\phi 2) & c(\phi 2) & 0 & 0 \\ c(\phi 2) * s(\beta) & -s(\beta) * s(\phi 2) & c(\beta) & z2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
with:
$$x2 = -f - s(\beta) * (l6 + l1 * s(k1)) + l1 * c(\beta) * c(k1)$$
$$z2 = c(\beta) * (l6 + l1 * s(k1)) + l1 * c(k1) * s(\beta)$$

- Wheel 3:
  $T_{w3}^{CM} = T_{RL}^{CM} * T_{BL}^{RL} * T_{w3}^{BL}$

  $$T_{w3}^{CM} = \begin{bmatrix} p1 & 0 & p2 & x3 \\ 0 & 1 & 0 & 0 \\ -p2 & 0 & p1 & z3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  with:

  $p1 = c(\beta) * c(\rho1) - s(\beta) * s(\rho1)$
  $p2 = c(\beta) * s(\rho1) + s(\beta) * c(\rho1)$

  $x3 = -f - l2 * c(\beta) + l3 * s(\beta) - l4 * c(k3) * (s(\beta) * s(\rho1) - c(\beta) * c(\rho1)) +$

  $l4 * s(k3) * (c(\beta) * s(\rho1) + s(\beta) * c(\rho1))$
  $z3 = l3 * c(\beta) + l2 * s(\beta) - l4 * c(k3) * (c(\beta) * s(\rho1) + s(\beta) * c(\rho1)) -$

  $l4 * s(k3) * (s(\beta) * s(\rho1) - c(\beta) * c(\rho1))$

- Wheel 4:
  $T_{w4}^{CM} = T_{RR}^{CM} * T_{BR}^{RR} * T_{w4}^{BR}$

  $$T_{w4}^{CM} = \begin{bmatrix} p3 & 0 & p4 & x4 \\ 0 & 1 & 0 & 0 \\ -p4 & 0 & p3 & z4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  with:

  $p3 = c(\beta) * c(\rho2) + s(\beta) * s(\rho2)$
  $p4 = c(\beta) * s(\rho2) - s(\beta) * c(\rho2)$

  $x4 = -f - l2 * c(\beta) - l3 * s(\beta) + l4 * c(k3) * (s(\beta) * s(\rho2) + c(\beta) * c(\rho2)) +$

  $l4 * s(k3) * (c(\beta) * s(\rho2) - s(\beta) * c(\rho2))$
  $z4 = l3 * c(\beta) - l2 * s(\beta) - l4 * c(k3) * (c(\beta) * s(\rho2) - s(\beta) * c(\rho2)) +$

  $l4 * s(k3) * (s(\beta) * s(\rho2) + c(\beta) * c(\rho2))$

- Wheel 5:
  $T_{w5}^{CM} = T_{RL}^{CM} * T_{BL}^{RL} * T_{w5}^{BL}$

  $$T_{w5}^{CM} = \begin{bmatrix} -c(\phi3) * h1 & s(\phi3) * h1 & c(\beta) * s(\rho1) + s(\beta) * c(\rho1) & x5 \\ s(\phi3) & c(\phi3) & 0 & 0 \\ -c(\phi3) * h2 & s(\phi3) * h2 & c(\beta) * c(\rho1) - s(\beta) * s(\rho1) & z5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with:

$h1 = (s(\beta) * s(\rho1) - c(\beta) * c(\rho1))$
$h2 = (c(\beta) * s(\rho1) + s(\beta) * c(\rho1))$

$x5 = -f + l5 * (s(\beta) * s(\rho1) - c(\beta) * c(\rho1)) + l6 * (c(\beta) * s(\rho1) + s(\beta) * c(\rho1)) - l2 * c(\beta) + l3 * s(\beta)$
$z5 = l5 * (c(\beta) * s(\rho1) + s(\beta) * c(\rho1)) - l6 * (s(\beta) * s(\rho1) - c(\beta) * c(\rho1)) + l3 * c(\beta) + l2 * s(\beta)$

- Wheel 6:
  $T_{w6}^{CM} = T_{RR}^{CM} * T_{BR}^{RR} * T_{w6}^{BR}$

$$T_{w6}^{CM} = \begin{bmatrix} c(\phi4) * h3 & -s(\phi4) * h3 & c(\beta) * s(\rho2) - s(\beta) * c(\rho2) & x6 \\ s(\phi4) & c(\phi4) & 0 & 0 \\ -c(\phi4) * h4 & s(\phi4) * h4 & c(\beta) * c(\rho2) + s(\beta) * s(\rho2) & z6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

with:

$h3 = (s(\beta) * s(\rho2) + c(\beta) * c(\rho2))$
$h4 = (c(\beta) * s(\rho2) - s(\beta) * c(\rho2))$

$x6 = -f - l5 * (s(\beta) * s(\rho2) + c(\beta) * c(\rho2)) + l6 * (c(\beta) * s(\rho2) - s(\beta) * c(\rho2)) - l2 * c(\beta) - l3 * s(\beta)$
$z6 = l5 * (c(\beta) * s(\rho2) - s(\beta) * c(\rho2)) + l6 * (s(\beta) * s(\rho2) + c(\beta) * c(\rho2)) + l3 * c(\beta) - l2 * s(\beta)$

The pose of the wheels are the first three elements of the fourth column of their homogeneous matrices.
The model takes a look that a changing on the inputs angles, in particular the angles of the rocker-bogie joints, produces a changing both on the x and z position of the wheels.
The position on y axis doesn't change because the wheels are rigidly connected to the rover chassis.

13.3 Inverse trigonometric model

If we are interested to find the rocker-bogie's feedback angles (β,$\rho1$ and $\rho2$), knowing the wheels position, the formulas are:

- From wheel 1 find β:

$$s(\beta) = \frac{-(l1 * c(k1) * z1) + (x1 - f) * (l6 + l1 * s(k1)}{(l6 + l1 * s(k1))^2 + (l1 * c(k1))^2}$$

$$c(\beta) = \big[x1 + f - \big(\frac{-(l1 * c(k1) * z1) + (x1 - f) * (l6 + l1 * s(k1)}{(l6 + l1 * s(k1)^2 + (l1 * c(k1)^2}\big)\big] * \big(\frac{l6 + l1 * s(k1)}{l1 * c(k1)}\big)$$

- From wheel 3 find $\rho 1$:

$$c(\rho 1) = \frac{a + \big[z3 - l3 * c(\beta) - l2 * s(\beta)\big] * tg(k3 - \beta)}{l4 * \big[c(k3 - \beta) + s(k3 - \beta) * tg(k3 - \beta)\big]}$$

$$s(\rho 1) = \frac{a}{l4 * s(k3 - \beta)} - \big[\frac{a + \big[z3 - l3 * c(\beta) - l2 * s(\beta)\big] * tg(k3 - \beta)}{l4 * \big[c(k3 - \beta) + s(k3 - \beta) * tg(k3 - \beta)\big] * tg(k3 - \beta)}\big]$$

with: $a = x3 + f + l2 * c(\beta) - l3 * s(\beta)$

- From wheel 4 find $\rho 2$:

$$c(\rho 2) = \frac{b + \big[z4 - l3 * c(\beta) + l2 * s(\beta)\big] * tg(k3 + \beta)}{l4 * \big[c(k3 + \beta) + s(k3 + \beta) * tg(k3 + \beta)\big]}$$

$$s(\rho 2) = \frac{b}{l4 * s(k3 + \beta)} - \big[\frac{b + \big[z4 - l3 * c(\beta) + l2 * s(\beta)\big] * tg(k3 + \beta)}{l4 * \big[c(k3 + \beta) + s(k3 + \beta) * tg(k3 + \beta)\big] * tg(k3 + \beta)}\big]$$

with: $b = x4 + f + l2 * c(\beta) + l3 * s(\beta)$

13.4 Rocker-bogie measures and wheels position

From design the variables are:

- l1 = 554 mm;
- l2 = 280 mm;
- l3 = 141 mm;
- l4 = 343 mm;
- l5 = 285 mm;
- l6 = 234 mm;

- f = 70 mm;
- k1 = 15 degree;
- k2 = k3 = 43 degree.

The resultant wheels position are:

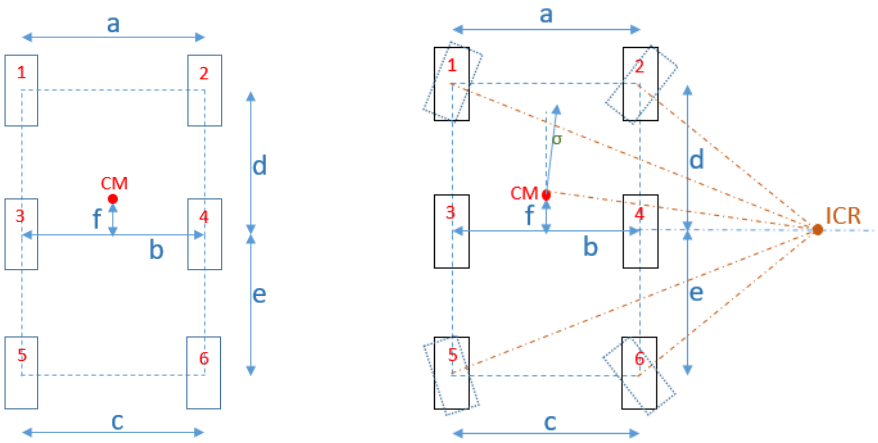
$$pose\_nominal = \begin{bmatrix} 465.12 & 465.12 & -99.14 & -99.14 & -635 & -635 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 377.38 & 377.38 & 374.92 & 374.92 & 375 & 375 \end{bmatrix}$$

These values are referred to the CM position. To have the current position on the z-axis respect nominal condition must subtract the previous components:
$pose\_z = pose\_z\_actual - pose\_nominal(3, :)$.

14 Rocker-bogie system dimensions (Ackermann model)

The rocker-bogie measurements used in *Velocity_block* and *Steering_block* are the following:

- $a = 782mm$. It is the distance between the front wheels.
- $b = 782mm$. It is the distance between the central wheels.
- $c = 782mm$. It is the distance between the rear wheels.
- $d = 612mm$. It is the distance along longitudinal axis between front and central wheels.
- $e = 587mm$. It is the distance along longitudinal axis between rear and central wheels.
- $f = 70mm$. It is the distance along longitudinal axis between central wheels axis and the CM.



In the previous figure, the picture on the right represents the rocker-bogie's interesting measures; instead, the figure on the left is the 2D dynamic model of the rover.

15 Results

In order to tune the correct saturation of the PID tracking error and PID gains it was used both Simulink and Coppeliasim environments.
The simulations on Simulink were very simple because they neglected all the dynamics delay. Instead, Coppeliasim simulation neglect only friction forces due to joints.
Unfortunately, the Model-Based V-shape was not closed because the rover was not built due to the Covid-19 lockdown.

Now, are shown the simulation of the Mobility controller in Simulink environment and on Coppeliasim with code-generation of its.

The command inputs in common between the two simulations are:

- *Vtan*: it is the module of the velocity of the Center of Mass in polar coordinate*[Figure 33]*.
- *Sigma_deg*: it is the angle of the velocity of the CM respect rover longitudinal axis in polar coordinate*[Figure 34]*.
- *Error flag*: it is the flag that represents if something is wrong on the rover. If it is equal to 1 the rover must be stopped*[Figure 35]*.
- *ERC_URC* flag: it is the flag that selects the maximum velocity. If the flag is equal to 1, URC is activated otherwise the default is ERC*[Figure*

36].

- *Parking_flag*: it is the flag that selects the parking mode. It is active when it is equal to 1*[Figure 37].*
- *Enable_correction*: it is the flag that enables the correction of the wheels angularity velocity using the feedback given by Rocker-bogie encoders*[Figure 38].*

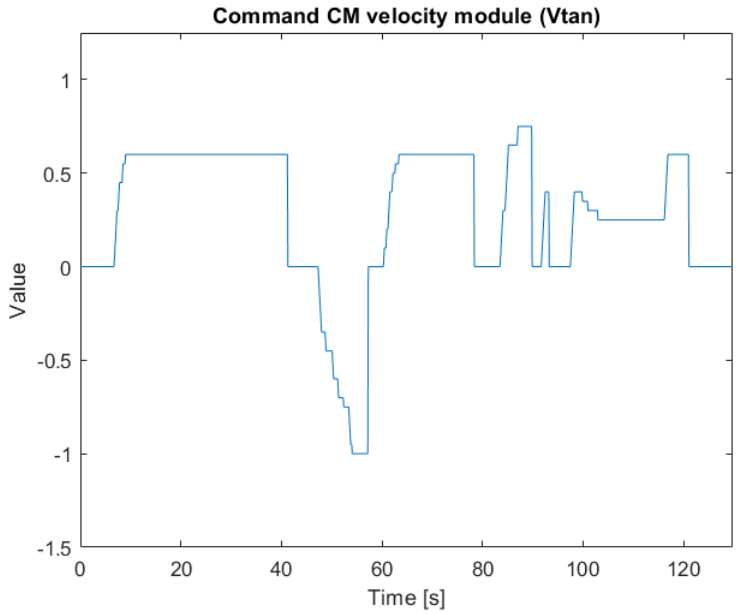


Figure 33: Plot of Vtan values

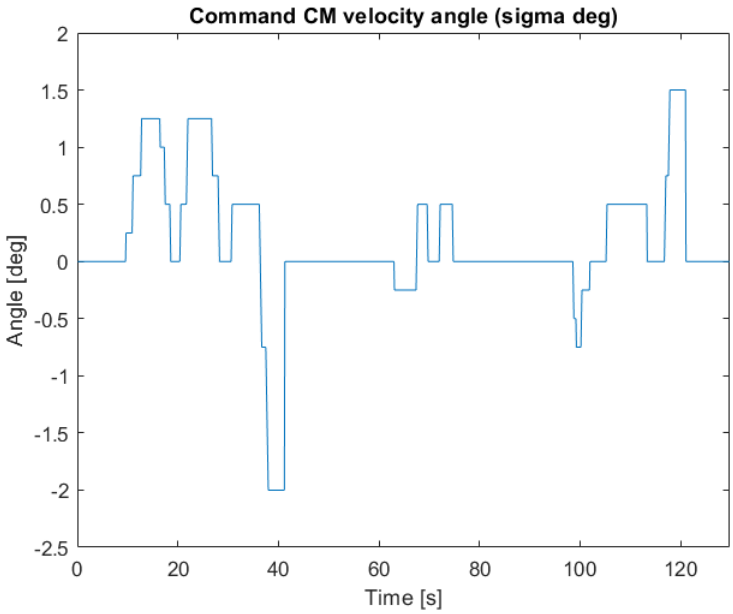


Figure 34: Plot of sigma_deg values

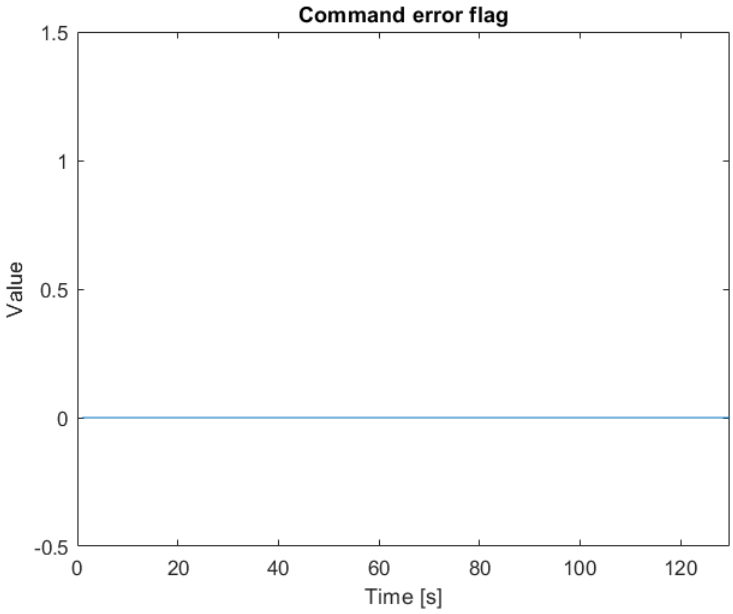


Figure 35: Plot of error flag

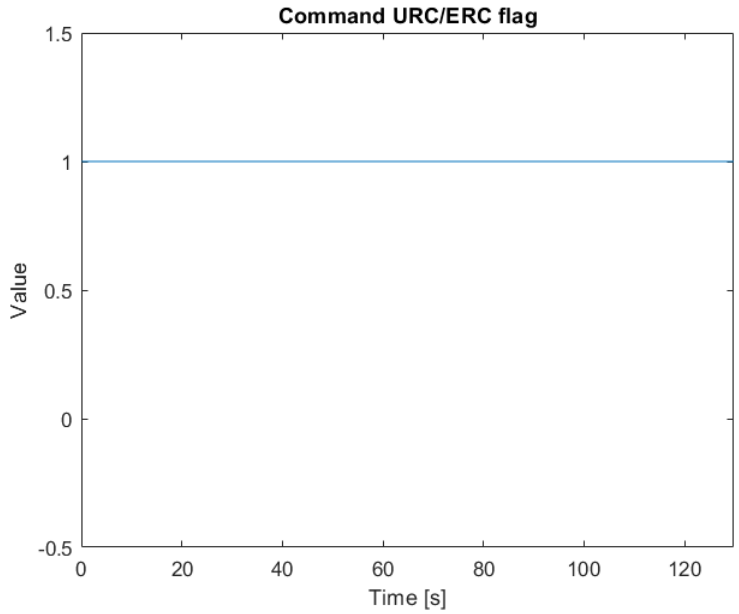


Figure 36: Plot of URC/ERC flag

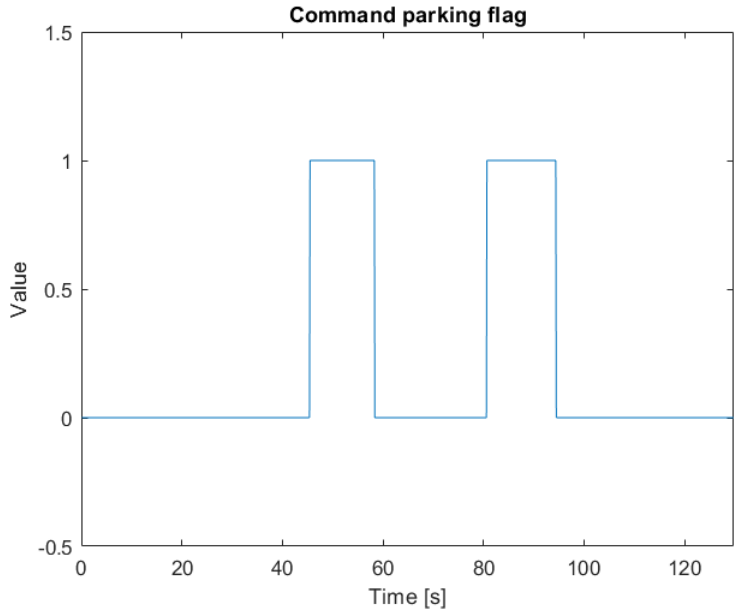


Figure 37: Plot of parking_flag

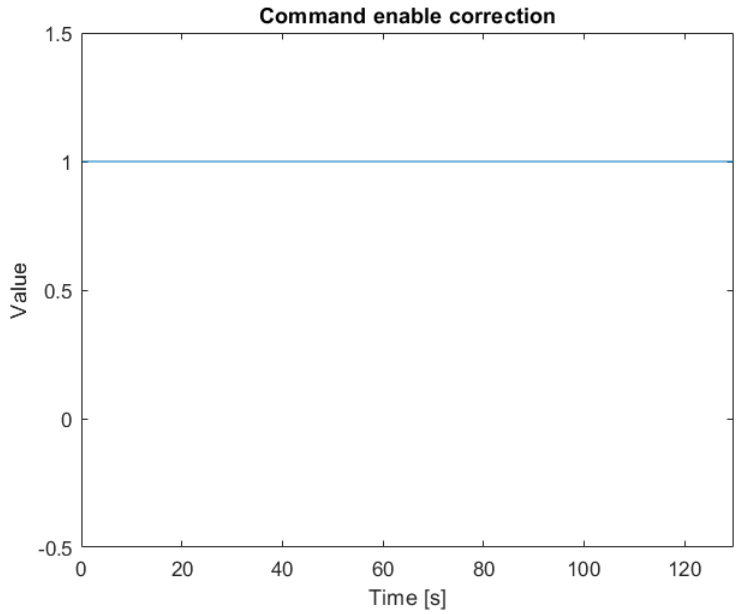


Figure 38: Plot of enable_correction flag

The feedbacks values for both the simulations are:

- *Omega_feed_RPM*: it is the motor feedback of each wheel.
- *Alpha_feed_rad*: it is the steering motor feedback of each wheel.
- *Beta_feed_rad*: it is feedback angle of left rocker*[Figure 39]*.
- *Rho1_feed_rad*: it is feedback angle of left bogie*[Figure 40]*.
- *Rho2_feed_rad*: it is feedback angle of right bogie*[Figure 41]*.

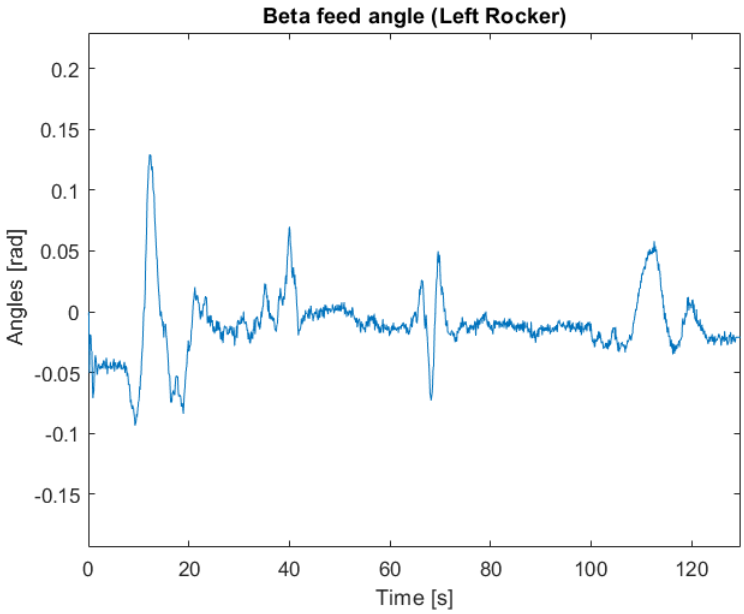


Figure 39: Plot of beta feedback angle

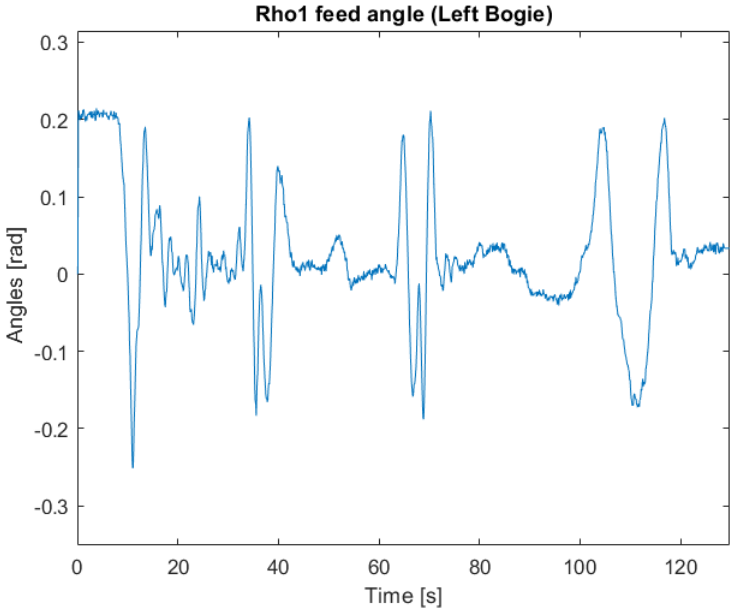


Figure 40: Plot of rho1 feedback angle

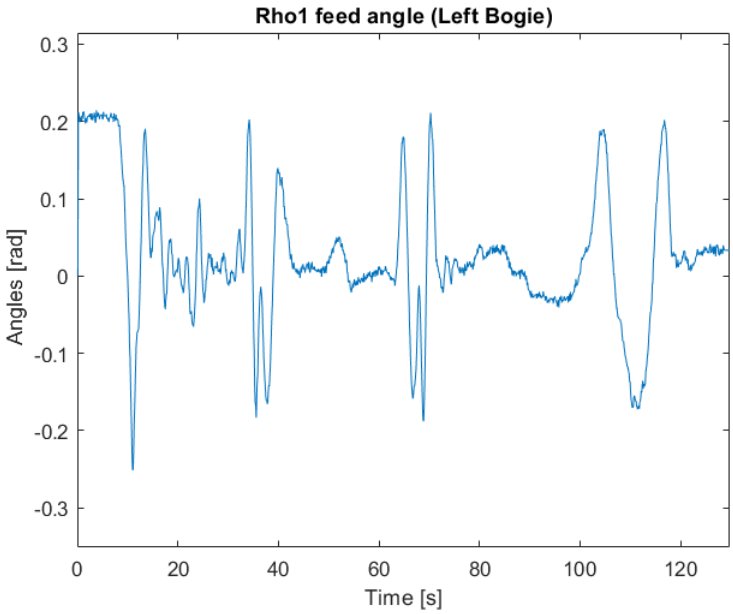


Figure 41: Plot of rho1 feedback angle

The feedbacks *Omega_feed_RPM* and *Alpha_feed_rad* are depending on the simulation environment.
For the Matlab-Simulink simulation, the feedbacks are equal to the outputs in delay of one sampling time (0.1 seconds) because there isn't any simulation model of the rover implemented in this environment.
On Coppeliasim environment, the feedbacks are corrupted by sensors noise and by mechanical dynamics, neglecting friction forces and motors dynamics.
The *Omega_feed_RPM* and *Alpha_feed_rad* feedbacks for Matlab-Simulink are showed in *Figure[42-43]* instead for Coppeliasim environment are showed in *Figure 44-45]*:

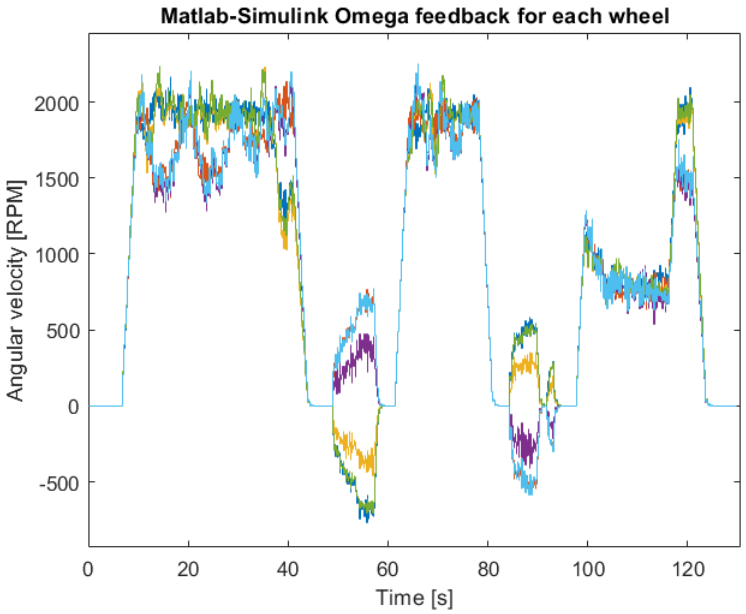


Figure 42: Omega feedbacks in Matlab-Simulink environment

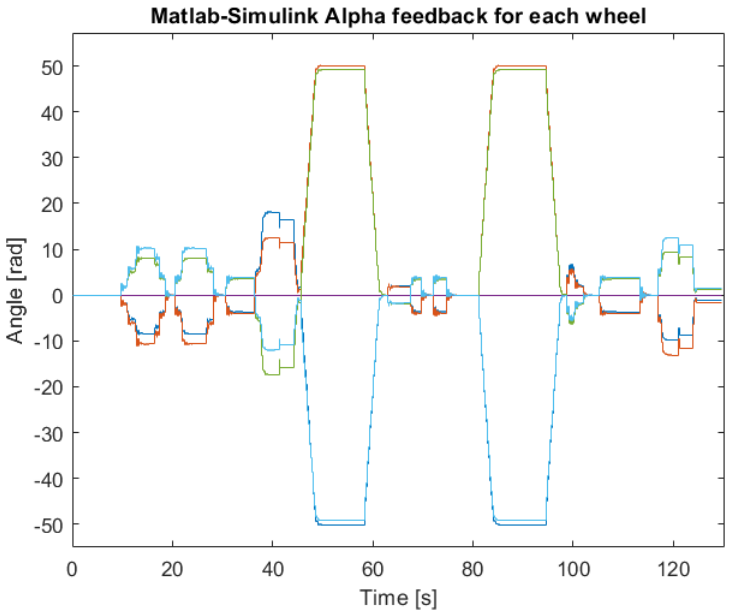


Figure 43: Alpha feedbacks in Matlab-Simulink environment

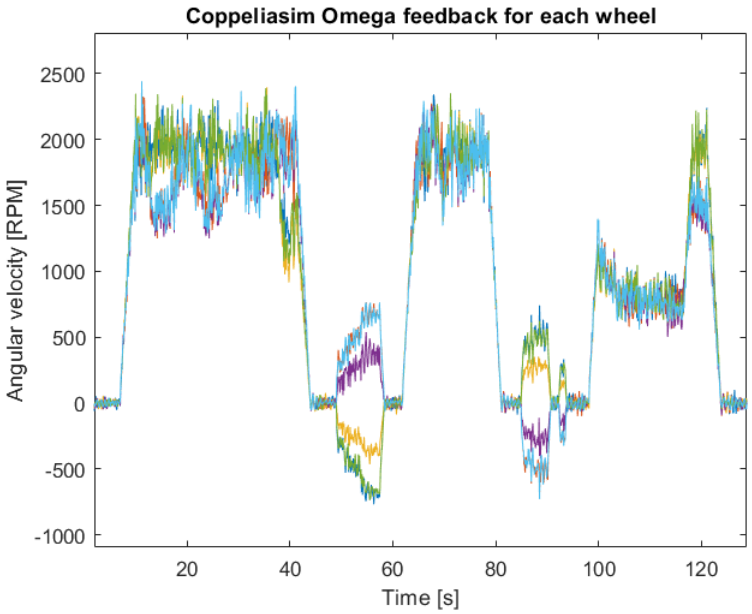


Figure 44: Omega feedbacks in Coppeliasim environment

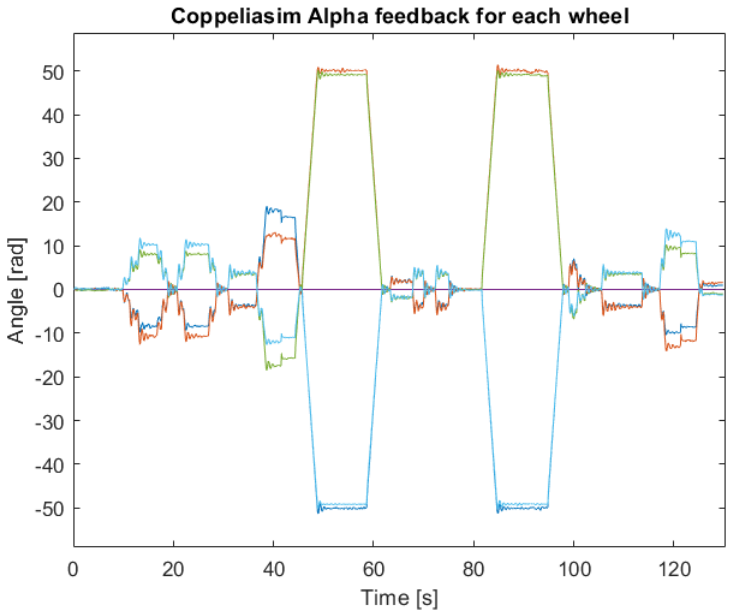


Figure 45: Alpha feedbacks in Coppeliasim environment

Feedbacks have the same behaviour in both the simulation environment that underlines the correctness setting of gain in the Mobility Controller. It is possible to notes that the main differences of Coppeliasim respect Simulink environment are the ripple at the ending of the transition part, due to a mechanical delay and the noise that affects the measurements.
In *Figure 45* are showed these differences in a focus in the plot that compares the angular velocity of wheel 1 both in Matlab-Simulink and Coppeliasim.

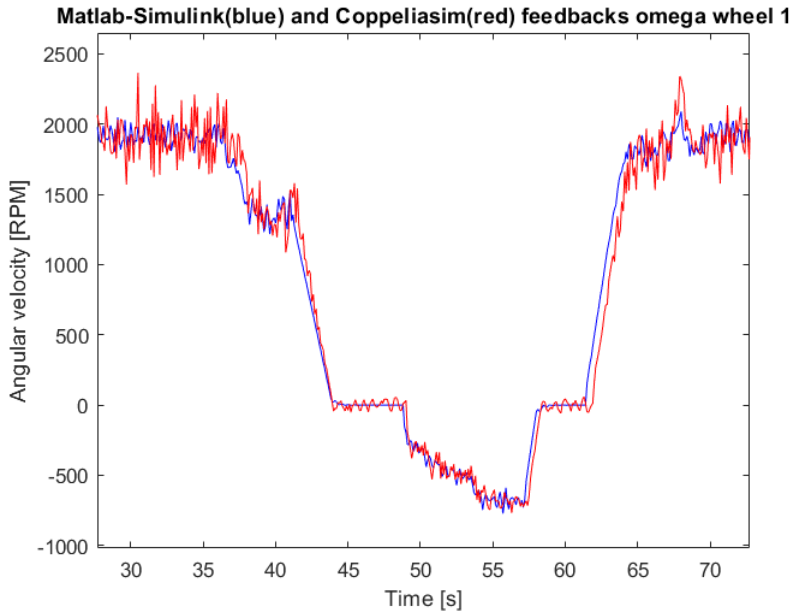


Figure 46: Comparison between Matlab-Simulink simulation (blue) and Coppeliasim (red)

The outputs of the controller in both the simulation environments are very similar to the feedbacks values because feedbacks are normally in delay respect the controller outputs.
In *Figure 47* is underlined these differences for the angular velocity of wheel 1.

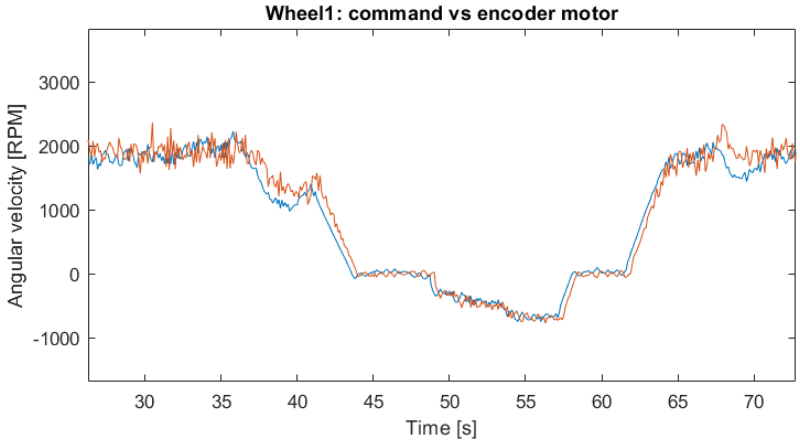


Figure 47: Comparison between command output (blue) and feedback (red) on Coppeliasim environment

In order to analyse the correctness between inputs and outputs, it is focused on the simulation on Coppeliasim because was more complete for what concern mechanical dynamics and delays.
The Mobility controller functionalities are:

1. Select the correct angular velocity and steering angles for each wheel.
2. Saturate the behaviours of both the angular velocity and steering angles for each wheel to avoid slipping of them.
3. Tuning the command output to have same settling time both for the angular velocity and steering angle for each wheel.

All of the 3 requirements are reached as is showed in *Figure 44-45*. Also, the controller stops the rover when *parking_flag* is activated as you can see in *Figure 37 and 45* in order to avoid an unpredictable behaviour of the rover.

Part VI
Failure Mode Effect and Diagnostic Analysis of the mobility system

16 Introduction

The Failure Mode Effect and Diagnostic Analysis (FMEDA) is a systematic analysis that can be applied to all components with a relevant effect on system behaviour.
The FMEDA takes the results of FMEA and acts the mitigation on the more dangerous failure.
The algorithm is divided into 2 parts:

- Diagnostic part: it detects the failure in a real-time condition. The diagnostic algorithm works in 2 different steps:
  - Flag generation: reads feedback values from sensors and indicates that fails a plausibility check.
  - Diagnosis generator: recognises patterns inside the flags vector hence the fault affected components and triggers the most appropriate mitigation algorithm.
- Mitigation part: it mitigates the effect of the failure on the system. If it is not possible it places the rover is a stable configuration (rover stop).

In this thesis, the FMEDA is applied only on the Mobility and Steering systems because they are the only systems connected to the Mobility Controller.
In the following subsection will be showed the algorithm's parts and how they work.

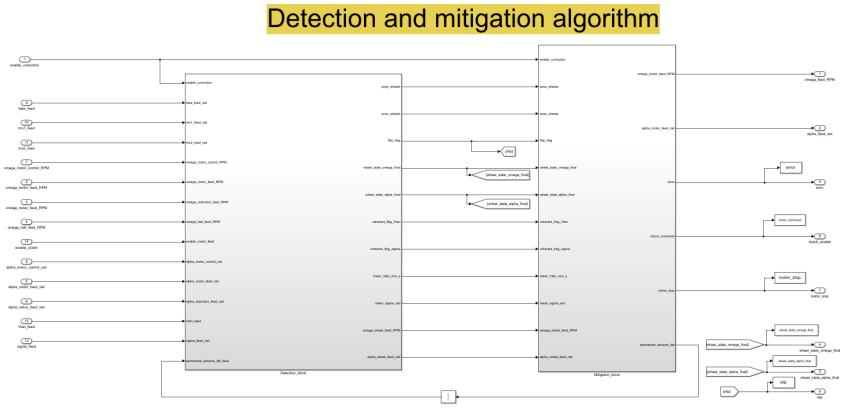


Figure 48: Overall FMEDA algorithm Simulink scheme

17 Detection Block

The algorithm uses feedbacks of encoders, hall sensors and IMU to detect possible failures. Then not all the failures analysed in section 8.1 and 8.2 (FMEA of Mobility System-Wheel system and Mobility System-Steering system) can be detected.

The single failure detected are:

- Traction motor hall sensors fail (**H-TM**).
- Traction motor encoder fail (**E-TM**).
- Traction reduction encoder fail (**E-TRG**).
- Traction motor fail (**TM**).
- Traction reduction fail (**TRG**).
- Traction clutch fail (**TC**).
- Steering motor or reduction fail (**SM** or **SRG**).
- Steering motor encoder fail (**E-SM**).
- Steering reduction encoder fail (**E-SRG**).

Also, double and some third failure (all wheel sensors fail) are detected. In order to reduce the complexity of the algorithm, it is decided to collapse some double failure to the most dangerous single failure that composed them. For

example, a double failure like Motor and hall sensors fail is collapsed to Motor fail because in this case hall sensors no needed any mitigation.

The inputs of the Detection part are:

1. *Enable_correction*: it is the flag that can enable the correction due to the position of Rocker-bogie system. It is the same flag as *Mobility Controller*.
2. *beta_feed_rad*: it is the feedback of the left rocker angle.
3. *rho1_feed_rad*: it is the feedback of the left bogie angle.
4. *rho2_feed_rad*: it is the feedback of the right bogie angle.
5. *omega_motor_control_RPM*: it is the command for each wheel calculated by *Mobility controller* (array of 6 elements).
6. *omega_motor_feed_RPM*: it is the motor encoder feedback for each wheel (array of 6 elements).
7. *omega_reduction_feed_RPM*: it is the reduction encoder feedback for each wheel (array of 6 elements).
8. *omega_hall_feed_RPM*: it is the hall feedback for each wheel (array of 6 elements).
9. *enable_clutch_feed*: it is the feedback of the output *enable_clutch* of *Mitigation_block* (array of 6 elements).
10. *alpha_motor_control_rad*: it is the command for each wheel, also central wheels, calculated by the *Mobility controller* (array of 6 elements).
11. *alpha_motor_feed_rad*: it is the feedback of the motor steering angle for each wheel (array of 6 elements).
12. *alpha_reduction_feed_rad*: it is the feedback of the reduction steering angle for each wheel (array of 6 elements).
13. *Vtan_feed*: it is the velocity of the Center of Mass feedback given by Inertia Measurements Unit (IMU) in polar coordinate.
14. *sigma_feed*: it is the feedback angle of the velocity of the Center of Mass given by IMU in polar coordinate.
15. *permanent_sensors_fail_feed*: it is the feedback of the output flag *permanent_sensors_fail* of *Mitigation_block* that represents if 2 or more sensors in the wheel fail (array of 6 elements).

The outputs of the block are:

1. *posx_wheels*: it is the position of the wheels along longitudinal rover axis respect to CM calculated by the trigonometric model of Rocker-bogie.

2. *posz_wheels*: it is the position of the wheels along normal rover axis respect to CM calculated by the trigonometric model of Rocker-bogie.

3. *Slip_flag*: it is the flag that represents if the rover dynamic is different or not respect to the desired one.

4. *wheel_state_omega_final*: it represents the state of failure of the wheels, each number corresponds to a failure (array of 6 elements).

5. *wheel_state_alpha_final*: it represents the state of failure of the steers, each number corresponds to a failure (array of 6 elements).

6. *coherent_flag_Vtan*: it is the flag that represents if the dynamics of the wheels are coherent each others or not, taking into account the tangential velocity Vtan. When it is equal to 1 means that the dynamics are different.

7. *coherent_flag_sigma*: it is the flag that represents if the dynamics of the wheels are coherent each others or not, taking into account the polar angle of the tangential velocity. When it is equal to 1 means that the dynamics are different.

8. *mean_Vtan_mm_s*: it is the mean of the calculated tangential velocity of the rover taking into account the velocity of the wheels.

9. *mean_sigma_rad*: it is the mean of the calculated angle of the tangential velocity of the rover taking into account the wheels steering angle.

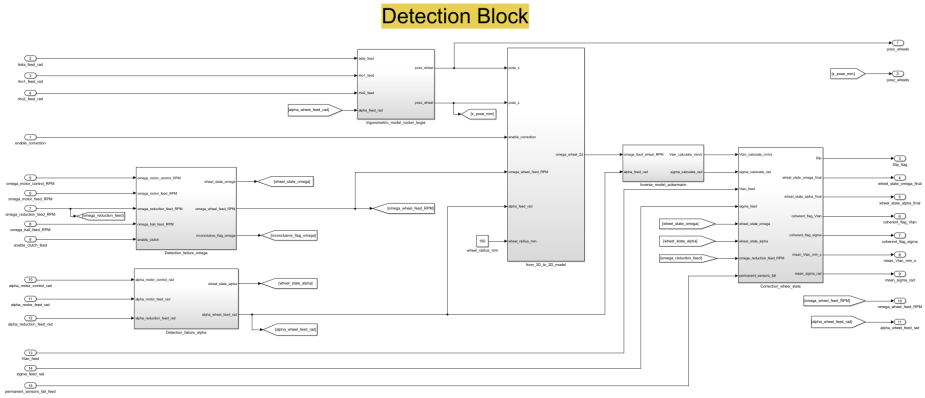


Figure 49: Detection Block Simulink scheme

17.1 Detection failure omega Block

This block diagnostics failures analysing the feedbacks of the wheels.
The inputs on this block are:

- *Omega_motor_control_RPM*: it is the command signal given by the controller to the wheels drivers (array of 6 elements).
- *Omega_motor_feed_RPM*: it is the feedback given by the encoders sensors of the wheels motor (array of 6 elements).
- *Omega_reduction_feed_RPM*: it is the feedback given by the encoders sensors of the wheels reduction (array of 6 elements).
- *Omega_hall_feed_RPM*: it is the feedback given by the hall sensors of the wheels motor (array of 6 elements).
- *Enable_clutch_feed*: it is the feedbacks of the commands that open the clutch for each wheel (array of 6 elements).

The outputs of this block are:

- *Wheel_state_omega*: it is the current state of the wheel. Each state means different failure (array of 6 elements).
- *Omega_wheel_feed_RPM*: it is the feedback of the wheels selected by the diagnostic block.
- *Inconclusive_omega_flag*: it is the flag that corresponds to an inconclusive sequence of checking flags. If it is equal to 1 means that there is an inconclusive sequence (array of 6 elements).

The diagnostic is done independently for each wheel. Those blocks are inside *Detection_failure_omega_wheel* and are called *Detection_wheel(i)* with i between 1 to 6.

17.2 (Detection failure omega) Detection wheel(i)

The fist operations in this block are the generation of six flags. This flags are important because they compose the sequence that characterises possible failures.
All the flags are calculated by the same function, called *generate_flag_fcn*, repeated 6 times, one for each flags.
The input of this function are:

- *diff*: it is the difference between the other 2 inputs.
- u: it is the first input value.
- z: it is the second input value.

The steps in this function are:

- if u is equal to 0.
  - if z is equal to 0, the output y is equal to 0.
  - otherwise, threshold equal to 2 RPM
    - if the absolute value of *diff* is greater or equal to the threshold, the output y is equal to 1.
    - otherwise, the output y is equal to 0.
- otherwise
  - if z equal to 0, the threshold is equal to 2 RPM
    - if the absolute value of *diff* is greater or equal to the threshold, the output y is equal to 1.
    - otherwise, the output is equal to 0.
  - otherwise, the threshold is equal to 0.2
    - if the absolute value of *diff* divided by u is greater or equal to the threshold, the output y is equal to 1.
    - otherwise, the output is equal to 0.

The flags must be in the following order:

1. *Flag_1*:
   it is the result of the comparison between *Omega_motor_control_RPM* and *Omega_motor_feed_RPM*. Both the values are multiplied by the gear ratio (1/50).

2. *Flag_2*:
   it is the result of the comparison between *Omega_motor_control_RPM* and *Omega_hall_feed_RPM*. Both the values are multiplied by the gear ratio (1/50).

3. *Flag_3*:
   it is the result of the comparison between *Omega_hall_feed_RPM* and *Omega_motor_feed_RPM*. Both the values are multiplied by the gear ratio (1/50).

4. *Flag_4*:
   it is the result of the comparison between *Omega_motor_feed_RPM* (multiplied by gear ratio 1/50) and *Omega_reduction_feed_RPM*.

5. *Flag_5*:
   it is the result of the comparison between *Omega_reduction_feed_RPM* and *Omega_hall_feed_RPM* (multiplied by gear ratio 1/50).

6. *Flag_6*:
   it is the result of the comparison between *Omega_motor_control_RPM* (multiplied by gear ratio 1/50) and *Omega_reduction_feed_RPM*.

After it is selected the best feedback for each wheel using the function *comparator_feed*. This function receives as input the 3 angular velocity feedbacks (motor encoder multiplied by the gear ratio, hall sensors multiplied by gear ratio and reduction encoder) and their differences respect the needed angular velocity of the motor and reduction. The output of this function is the angular velocity feedback of the reduction encoder if it is different from 0 otherwise the feedback with the minimum difference respect the control value.

The output of this block is the same as *Detection failure omega* but related only for a wheel.

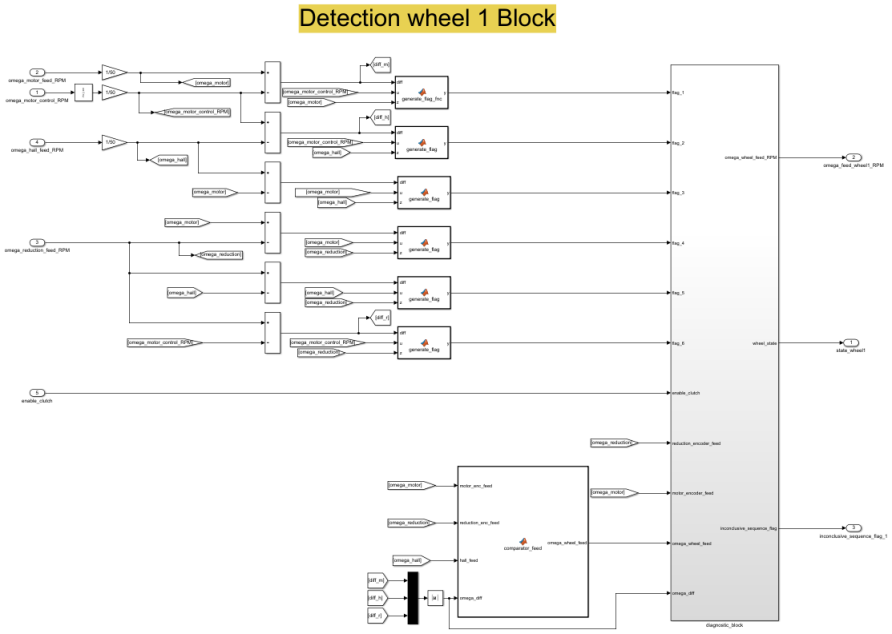


Figure 50: Detection wheel 1 Block Simulink scheme

```
function y = generate_flag_fnc(diff,u,z)
    if(u == 0)
        if(z == 0)
            y = 0;
        else
            threshold = 2;
            if (abs(diff) >= threshold)
                y = 1;
            else
                y = 0;
            end
        end
    else
        if(z == 0)
            threshold = 2;
            if (abs(diff) >= threshold)
                y = 1;
            else
                y = 0;
            end
        else
        threshold = 0.2;
        if(abs(diff/u) >= threshold)
            y = 1;
        else
            y = 0;
        end
        end
    end
end
```

```
function omega_wheel_feed = comparator_feed(motor_enc_feed,...
    reduction_enc_feed,hall_feed,omega_diff)

min_omega = min(omega_diff);
if(reduction_enc_feed == 0)
    if(omega_diff(1) == min_omega)
        omega_wheel_feed = motor_enc_feed;
    else
        if(omega_diff(3) == min_omega)
        omega_wheel_feed = reduction_enc_feed;
        else
            omega_wheel_feed = hall_feed;
        end
    end
else
    omega_wheel_feed = reduction_enc_feed;
end
end
```

Figure 51: On the left function *generate_flag_fnc*, on the right function *comparator_feed*.

17.2.1 (Detection wheel(i)) Diagnostic Block

This block is composed by 2 parts:

- *Diagnostic_function_omega*:
  this function generates the 11 enable flags (one for each the failure) and the inconclusive sequence flag, given the inputs: *enable_clutch*, *Omega_motor_feed_RPM*, *Omega_reduction_feed_RPM* and the differences of all the feedbacks respect *Omega_motor_control_RPM* calculated in *Detection_wheel(i)*.

- Failures_block:
  this block selects the correct failure acting at the moment and it sets the correct wheel state that represents the failure detected.

This two blocks are shown following.

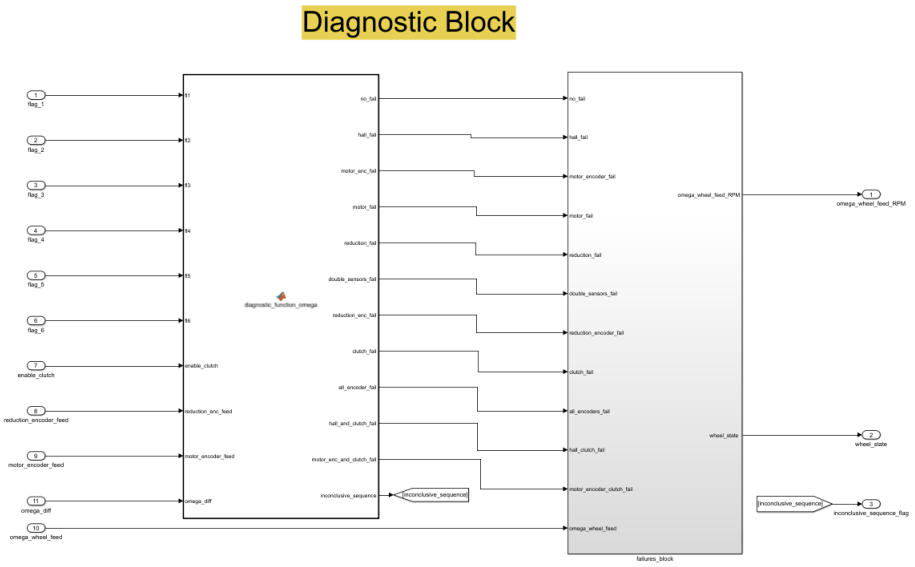


Figure 52: Diagnostic Block Simulink scheme

Diagnostic_function_omega: The *Diagnostic_function* generates the enable flags for each failure. Thanks to this function, only a failure can be activated time by time. The list of failures detected by this function is the following:

- *No failure*: the sequence of flags for this condition is $[0, 0, 0, 0, 0, 0]$.
- **E-TM** or *Traction motor encoder fail*: the sequence of flags for this condition is $[1, 0, 1, 1, 0, 0]$.
- **E-TRG** or *Traction reduction gear encoder fail*: the sequence of flags for this condition is $[0, 0, 0, 1, 1, 1]$, *Omega_reduction_feed_RPM* must be equal to 0 RPM and *enable_clutch* must be equal to 1.
- **H-TM** or *Traction motor hall sensors fail*: the sequence of flags for this condition is $[0, 1, 1, 0, 1, 0]$.
- **TM** or *Traction motor fail*: the sequence of flags for this condition are:
  - $[1, 1, 0, 0, 0, 1]$ and *enable_clutch* flag equal to 1.
  - $[1, 1, 0, 0, 0, 1]$, *enable_clutch* flag equal to 0 and *reduction_enc_feed* less than 2 RPM.
  - $[1, 1, 0, 1, 1, 1]$, *enable_clutch* equal to 1 and *motor_encoder_feed* less than 100 RPM.

- **TRG** or *Traction reduction gear fail*: the sequence of flags for this condition are:
  - $[1, 1, 0, 0, 0, 1]$, *enable_clutch* flag equal to 0 and *reduction_enc_feed* less than 2 RPM.
  - $[0, 0, 0, 1, 1, 1]$ and *reduction_enc_feed* less than 5 RPM.
- **TC** or *Traction Clutch fail*: the sequence of flags for this condition are:
  - $[0, 0, 0, 1, 1, 1]$ and *Omega_reduction_feed_RPM* greater than 5 RPM.
  - $[1, 1, 0, 1, 1, 1]$ with *enable_clutch* equal to 0 and *Omega_reduction_feed_RPM* different from 0 RPM.
- *Double sensor fail*:the sequence of flags for this condition are:
  - $[1, 1, 1, 1, 1, 0]$.
  - $[1, 1, 1, 1, 1, 1]$ and the minimum of the difference between the feedbacks and motor command is less than 2 RPM.
  - $[0, 1, 1, 1, 1, 1]$ and *Omega_reduction_feed_RPM* equal to 0 RPM.
  - $[1, 0, 1, 0, 1, 1]$ and *Omega_reduction_feed_RPM* equal to 0 RPM.

  In this failure are collapsed 3 different failure type: **H-TM + E+TM** or *Hall sensors + Motor encoder fail*, **H-TM + E-TRG** or *Hall sensors + Reduction gear encoder fail* and **E-TM + E-TRG** or*Motor encoder fail + Reduction gear encoder fail*.
- **E-TM + TM** or *Traction motor encoder + Traction motor fail*: the sequence of flags for this condition are:
  - $[1, 1, 1, 1, 0, 1]$ and *enable_clutch* equal to 1.
  - $[1, 1, 1, 1, 0, 1]$, *enable_clutch* equal to 0 and *reduction_enc_feed* greater or equal to 2 RPM.
- **E-TM + TRG** or *Traction motor encoder + Traction reduction gear fail*: the sequence of flags for this condition are:
  - $[1, 1, 1, 1, 0, 1]$, *enable_clutch* equal to 0 and *reduction_enc_feed* less than 2 RPM.
  - $[1, 0, 1, 1, 1, 1]$ and *reduction_enc_feed* less than 5 RPM.
- **H-TM + TM** or *Hall sensors + Traction motor fail*: the sequence of flags for this condition are:
  - $[1, 1, 1, 0, 1, 1]$ and *enable_clutch* equal to 1.
  - $[1, 0, 1, 1, 1, 1]$ and *reduction_enc_feed* greater or equal than 5 RPM.

- **H-TM + TRG** or *Hall sensors + Traction reduction gear fail*: the sequence of flags for this condition are:
  - $[1, 1, 1, 0, 1, 1]$ and *enable_clutch* equal to 0.
  - $[0, 1, 1, 1, 1, 1]$ and *reduction_enc_feed* less than 5 RPM.
- **E-TRG + TM** or *Traction reduction gear encoder + Traction motor fail*: the sequence of flags for this condition is $[1, 1, 0, 1, 1, 1]$, *enable_clutch* equal to 1 and *Omega_motor_feed_RPM* greater than 100 RPM.
- **E-TRG + TRG** or *Traction reduction gear encoder + Traction reduction gear fail*: the sequence of flags for this condition are:
  - $[1, 1, 0, 1, 1, 1]$, *enable_clutch* equal to 0 and *Omega_motor_feed_RPM* equal to 0 RPM.
  - $[1, 0, 0, 1, 1, 1]$ and *reduction_enc_feed* less than 5 RPM.
  - $[0, 0, 0, 1, 1, 1]$, *reduction_enc_feed* equal to 0 RPM and *clutch_enable* equal to 0.
- **E-TM + TC** or *Traction motor encoder and clutch fail*: the sequence of flags for this condition is $[1, 0, 1, 1, 1, 1]$ and *Omega_reduction_feed_RPM* greater or equal than 5 RPM.
- **H-TM + TC** or *Hall sensors and clutch fail*: the sequence of flags for this condition is $[0, 1, 1, 1, 1, 1]$ and *Omega_reduction_feed_RPM* greater or equal than 5 RPM.
- **H-TM + E-TM + E-TRG** or *All sensors fail*: the sequence of flags for this condition is $[1, 1, 1, 1, 1, 1]$ and the minimum of the difference between the feedbacks and motor command is equal or bigger than 2 RPM

In order to reduce the number of enable flags of the failures *Enable subsystem*, *Motor encoder + Motor fail*, *Reduction gear encoder + Motor fail* and *Hall sensors + Motor fail* are collapsed into *Motor fail*. Same reduction is used for double failure with Reduction gear as failure.
The final outputs of this functions are:

- *no_fail* flag.
- *hall_fail* flag.
- *motor_enc_fail* flag.
- *motor_fail* flag.

- *reduction_fail* flag.
- *double_sensors_fail* flag.
- *reduction_enc_fail* flag.
- *clutch_fail* flag.
- *all_encoders_fail* flag.
- *hall_and_clutch_fail* flag.
- *motor_enc_and_clutch_fail* flag.
- *inconclusive_sequence* flag.

Failures_block: This block selects the correct angular velocity feedback of the wheel and the wheel state related the current failure.

The following list cover all the *enable subsystem* Simulink model in this block:

- *No fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 0.
- *Hall fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 1.
- *Motor encoder fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 1.
- *Reduction gear encoder fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 2.
- *Motor fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 3.
- *Reduction gear fail state*:

  - omega_wheel_feed = omega_feed.
  - wheel_state = 4.

- *Clutch fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 5.

- *Double sensors fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 6.

- *Hall sensors + Clutch fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 7.

- *Motor encoder + Clutch fail state*:
  - omega_wheel_feed = omega_feed.
  - wheel_state = 7.

- *All sensors fail state*:
  - omega_wheel_feed = 800000.
  - wheel_state = 8.

17.3 Detection failure alpha Block

This block diagnostics failures of the steering system detecting the angles feedbacks of the steering system.
The inputs of this block are:

- *Alpha_motor_control_rad*: it is the steering command calculated by the *Mobility controller* sent to the steering motor drivers (array of 6 elements).

- *Alpha_motor_feed_rad*: it is the motors feedback angles given by the motor encoders (array of 6 elements).

- *Alpha_reduction_feed_rad*: it is the reductions feedback angles given by the reduction encoders (array of 6 elements).

The outputs of these blocks are:

- *Alpha_wheel_feed_rad*: it is the angles feedback of the steering reductions (array of 6 elements).

- *Wheel_state_alpha*: it is the state of the steering system of the wheels that represent the type of failure (array of 6 elements).

The diagnostic are done independently for each steering wheel. Those blocks are inside *Detection_failure_alpha_wheel* and are called *Detection_alpha_wheel(i)* with i between 1 to 6.

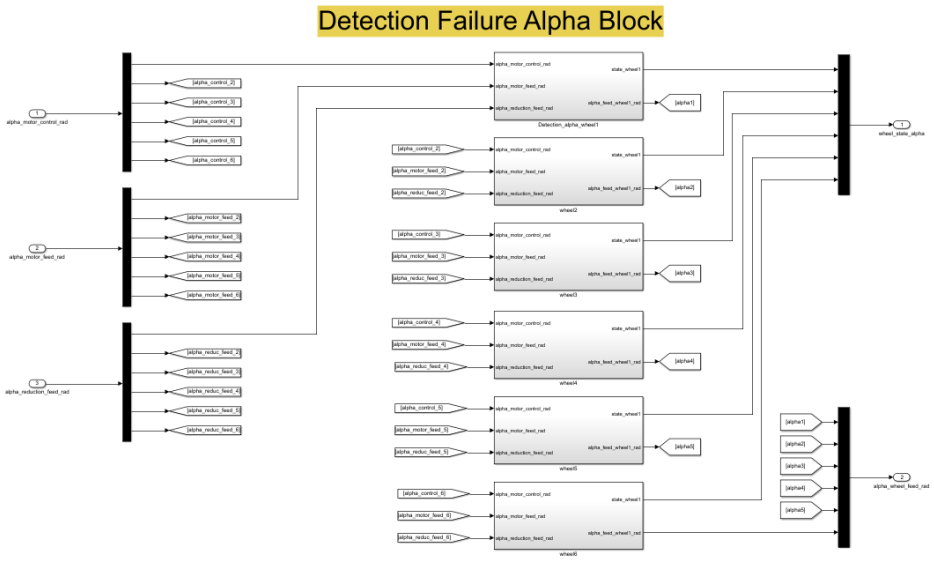


Figure 53: Detection failure alpha Block Simulink scheme

17.4 (Detection failure alpha Block) Detection alpha wheel(i)

This block diagnostics for each steering system its possible failure. The inputs and outputs of this block are the same as *Detection_failure_alpha_block* but related only to a wheel.
The first step in this block is the calculation of the differences between the command motor steering angle and the feedbacks of motor and reduction encoders.
The results of this step are 3 flags that compose a sequence used to detect the type of failure.
The 3 flags are:

1. *Flag_1*: it is the result of the comparison between the difference (in absolute value) between *Alpha_motor_command_rad* and *Alpha_motor_feed_rad* and a threshold *th_alpha_motor* equal to 2 rad.

2. *Flag_2*: it is the result of the comparison between the difference (in absolute value) between *Alpha_motor_feed_rad*, multiply by gear ratio

(1/50), and *Alpha_reduction_feed_rad* and a threshold *th_alpha_reduction* equal to 0.04 rad.

3. *Flag_3*: it is the result of the comparison between the difference (in absolute value) between *Alpha_motor_command_rad*, multiply by gear ratio (1/50), and *Alpha_motor_feed_rad* and a threshold *th_alpha_reduction* equal to 0.04 rad.

The list of possible failure detected by *diagnostic_function_alpha* are:

- *No failure*: the sequence of flags for this condition is $[0, 0, 0]$.
- **E-SM** or *Steering motor encoder failure*: the sequence of flags for this condition is $[1, 1, 0]$.
- **E-SRG** or *Steering reduction gear encoder failure*: the sequence of flags for this condition is $[0, 1, 1]$.
- *Critical failure*: the sequence of flags for this condition is $[1, 0, 1]$. In this condition are collapsed the following failure:
  - **SM** or *Steering motor failure*
  - **SRG** or *Steering reduction gear failure*
  - **E-SM + SM** or *Steering motor encoder + Steering motor failure*
  - **E-SRG + SM** or *Steering reduction gear encoder + Steering motor failure*
  - **E-SM + SRG** or *Steering motor encoder + Steering reduction gear failure*
  - **E-SRG + SRG** or*Steering reduction gear encoder + Steering reduction gear failure*
- **E-SM + E-SRG** or *Steering motor encoder + Steering reduction gear encoder*: the sequence of flags for this condition is $[1, 1, 1]$.

The corresponding *wheel_state_alpha* for each failure are:

- *No failure*: *wheel_state_alpha* equal to 0.
- *Steering motor failure*: *wheel_state_alpha* equal to 1.
- *Steering reduction gear failure*: *wheel_state_alpha* equal to 2.
- *Critical failure*: *wheel_state_alpha* equal to 3.
- *Steering motor encoder + Steering reduction gear encoder*: *wheel_state_alpha* equal to 4.

If any of the previous configuration are present the *Inconclusive_sequence_alpha* is equal to 1.
The feedback output *alpha_wheel_feed_rad* is equal to the more closed to the control command between the 2 feedbacks encoders.

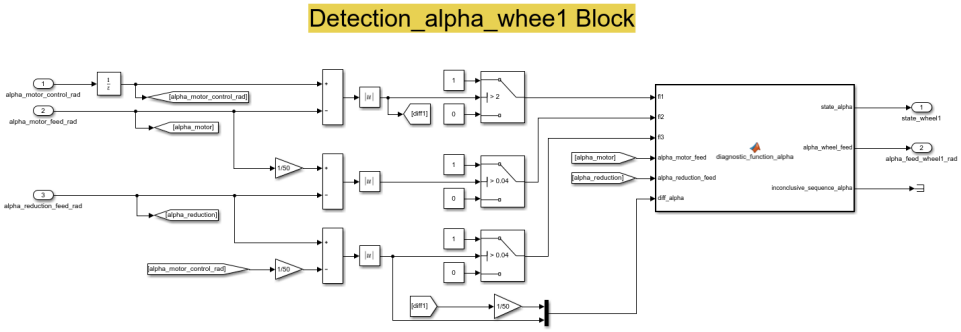


Figure 54: Detection_alpha_wheel_1 Block Simulink scheme

17.5 Trigonometric model rocker-bogie Block

It is the same model used in the Mobility Controller and it is described in part 5 section 13.1.1.

17.6 From 3D to 2D model Block

The function of this block is to cancel the correction applied to the wheels angular velocity to avoid slipping of them when they climb obstacles or hills. In other words, it is needed to return to the Ackermann model from a 3D model.
In order to do this, it is needed the position of the wheels respect the Center of Mass, previously calculated by *Trigonometric_model_rocker_bogie* block.

The inputs of this block are:

- *pose_x*: it is the position of the wheels respect CM to along the x-axis (longitudinal axis) of the rover (array of 6 elements).
- *pose_z*: it is the position of the wheels respect to CM along the z-axis (normal axis) of the rover(array of 6 elements).
- *Omega_wheel_feed_RPM*: it is the feedback angular velocities of the wheels (array of 6 elements).
- *wheel_radius_mm*: it is the wheels radius dimension. It is constant and equal to $150mm$.
- *alpha_feed_rad*: it is the feedback steering angles of the wheels (array of 6 elements).
- *enable_correction*: it is the flag command, send by the user, that enable the correction given by the rocker-bogie model.

The output of this block is the 2D feedback angular velocity for each wheel *omega_wheel_2D*.
The steps in this block are:

1. Derive the module of linear velocity along x and z axis from *pose_x* and *pose_z*: $Vx\_model = d/dt(pose\_x)$ and $Vz\_model = d/dt(pose\_z)$.

2. Calculate the tangential linear velocities of the wheels from angular velocities and wheel radius:
   $linear\_vel[i] = omega\_wheel\_feed\_RPM[i] * wheel\_radius$.

3. Find the angle between the module of the linear velocity and the linear velocity along z, given by rocker-bogie trigonometric model, for each wheel: $\theta[i] = acos(Vz\_model[i]/linar\_vel[i])$

4. Calculate linear velocity along x and y axis for each wheel:
   $Vx\_wheel[i] = linear\_vel[i] * cos(alpha\_feed\_rad[i]) * sin(\theta[i])$
   $Vy\_wheel[i] = linear\_vel[i] * sin(alpha\_feed\_rad[i]) * sin(\theta[i])$

5. Calculate the 2D module of the linear velocity for each wheel:
   $V\_2D\_linear[i] = sqrt((Vx\_model[i] + Vx\_wheel[i])^2 + (Vy\_wheel[i])^2)$

6. Calculate the angular velocities of the wheels:
   $omega\_wheel\_2D[i] = V\_2D\_linear[i]/wheel\_radius\_mm$.

7. Check if the *enable_correction* is active or not. If it is active the final angular velocity is equal to *omega_wheel_2D* otherwise is equal to *omega_wheel_feed_RPM* because it is already the angular velocity calculated by Ackermann model.

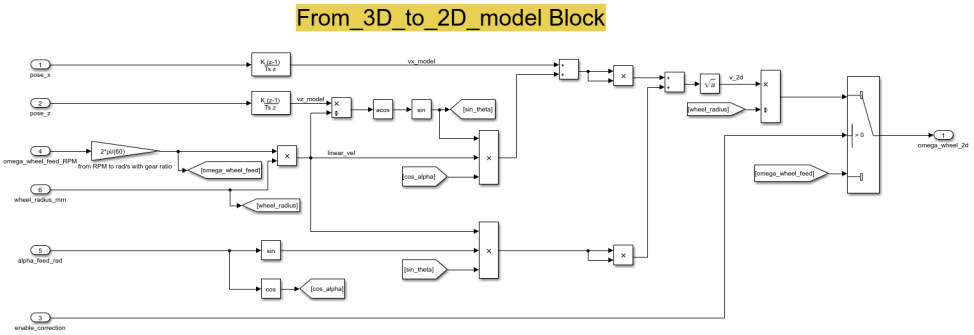


Figure 55: From_3D_to_2D_model Block Simulink scheme

17.7 Inverse Ackermann model Block

This block is the inverse of Ackermann model and calculates from *omega_wheel_2D* and *alpha_feed_rad* the corresponding *Vtan* and *sigma* for each wheel.

The inputs of this block are:

- *omega_wheel_2D*: it is the feedback 2D angular velocity of the wheels, calculated by *From_2D_to_3D_model* block (array of 6 elements).
- *alpha_feed_rad*: it is the feedback steering angle of the wheels (array of 6 elements.

The outputs of this block are:

- *Vtan_calculate_mm/s*: it is the velocity module of the Center of Mass calculated from the feedbacks of each wheel (array of 6 elements.
- *sigma_calculate_rad*: it is the angle in the polar coordinate of the module of CM velocity calculated from the feedbacks of each wheel (array of 6 elements.

This block is composed of 2 subsystem:

- *Steering inverse model*
- *Velocity inverse model*

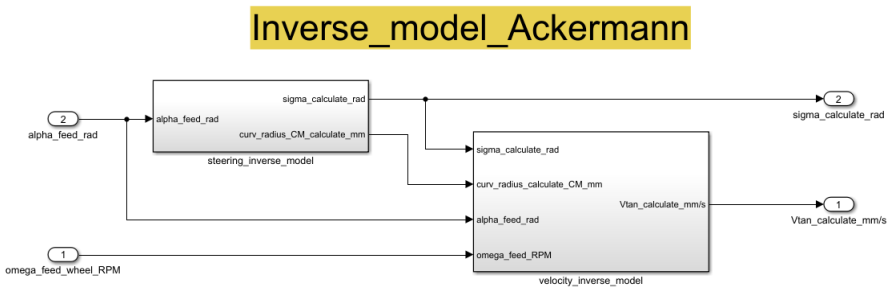


Figure 56: Inverse_model_Ackermann Block Simulink scheme

17.7.1 Steering inverse model

This block calculates, for each wheel, the angle of the module of the CM velocity in polar coordinate and CM curvature radius, given the steering angles of them.
The procedure is the following:

1. Calculate the CM angle for steering wheels (neglecting central wheels):
   $sigma[1] = atan(f/((-d/tan(alpha\_feed\_rad[1]) - a/2)))$
   $sigma[2] = atan(f/((-d/tan(alpha\_feed\_rad[2]) + a/2)))$
   $sigma[5] = atan(f/((e/tan(alpha\_feed\_rad[5]) - c/2)))$
   $sigma[6] = atan(f/((e/tan(alpha\_feed\_rad[6]) + c/2)))$

2. Check if the module of *sigma[i]* with i = 1,2,5,6 is less respect $0.09rad$ (about 5.2 deg). If it is, means that $sigma\_calculated\_rad[i] = sigma$ otherwise $sigma\_calculated\_rad[i] = abs(sigma)$.

   Notes: *This is an important step because the inverse of Ackermann model has a singularity near sigma =* $1.5rad$*. In this case the result is not unique then can be equal to 1.5 or -1.5 rad (because they have same steering configuration of the wheels). This problem appears because in the used model the inputs Vtan and sigma_deg are not correlated, than can be changed separately. Normally in polar coordinate it is not possible to do this, but have steering and velocity not related is more human friendly.*

3. Calculate the *sigma_calculated_rad* angle for central wheels:
   $sigma\_calculated\_rad[3] = (sigma\_calculated\_rad[1]+sigma\_calculated\_rad[5])/2$
   $sigma\_calculated\_rad[4] = (sigma\_calculated\_rad[2]+sigma\_calculated\_rad[6])/2$

4. Calculate the curvature radius of the CM respect to Instantaneous Center of Rotation (ICR) for each wheel:
   $curv\_rad\_CM\_calculate\_mm[i] = f/sin(sigma\_calculate\_rad[i])$

17.7.2 Velocity inverse model

This block calculates the module of the CM velocity given:

- *sigma_calculate_rad*: angle of CM velocity module respect longitudinal rover axis (array of 6 elements).
- *alpha_feed_rad*: steering angles of the wheels (array of 6 elements).
- *curv_rad_CM_calculate_mm*: calculated CM curvature radius for each wheel (array of 6 elements).
- *omega_wheel_2D_mm/s*: angular velocity of the wheels in 2D model (array of 6 elements).

The procedure is the following:

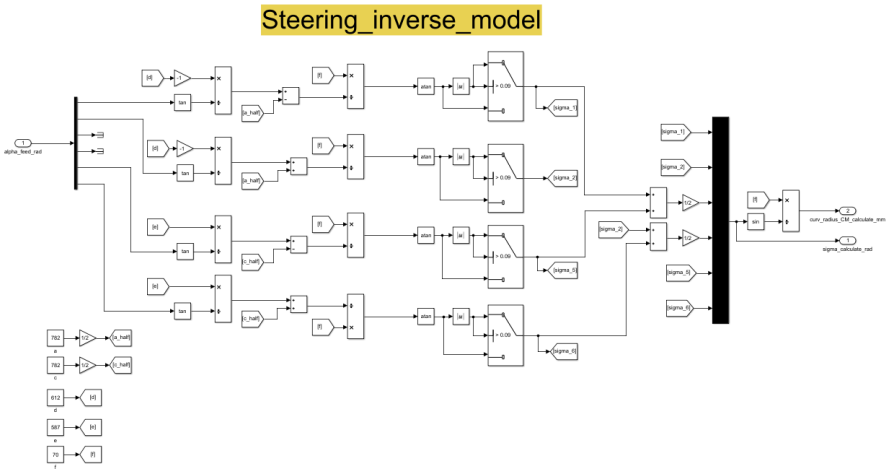


Figure 57: Steering_inverse_model Block Simulink scheme

1. Calculate the curvature radius for each wheel:
   $curv\_radius\_wheel[1] = (curv\_rad\_CM\_calculate\_mm[1]*cos(sigma\_calculate\_rad[1])+ a/2)/cos(alpha\_feed\_rad[1])$
   $curv\_radius\_wheel[2] = (curv\_rad\_CM\_calculate\_mm[2]*cos(sigma\_calculate\_rad[2])- a/2)/cos(alpha\_feed\_rad[2])$
   $curv\_radius\_wheel[3] = (curv\_rad\_CM\_calculate\_mm[3]*cos(sigma\_calculate\_rad[3])+ b/2)$
   $curv\_radius\_wheel[4] = (curv\_rad\_CM\_calculate\_mm[4]*cos(sigma\_calculate\_rad[4])- b/2)$
   $curv\_radius\_wheel[5] = (curv\_rad\_CM\_calculate\_mm[5]*cos(sigma\_calculate\_rad[5])+ c/2)/cos(alpha\_feed\_rad[5])$
   $curv\_radius\_wheel[6] = (curv\_rad\_CM\_calculate\_mm[6]*cos(sigma\_calculate\_rad[6])- c/2)/cos(alpha\_feed\_rad[6])$

2. Calculate the module of the CM velocity for each wheel:
   $Vtan\_calculate\_mm/s[i] = (omega\_feed\_2D\_rad/s[i]*wheel\_radisu\_mm* curv\_rad\_CM\_calculate\_mm[i]*cos(sigma\_calculate\_rad[i]))/(curv\_radius\_wheel[i]* sign(cos(sigma\_calculate\_rad[i])))$

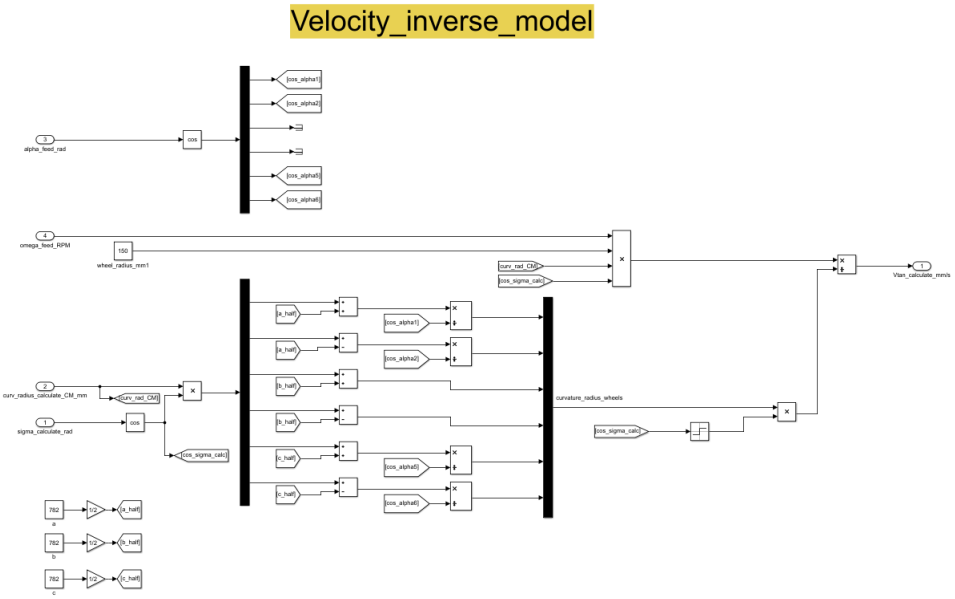


Figure 58: Velocity_inverse_model Block Simulink scheme

17.8 Correction wheel state Block

This block calculates the correct *omega wheel states* and the 2 coherent flags between the wheels taking into account the feedbacks behaviour of the rover. The inputs of these blocks are:

- *Vtan_calculate_mm*: it is the calculated module of the Center of Mass velocity for each wheel (array of 6 elements).
- *Sigma_culculate_mm*: it is the calculated angle of the module of the CM velocity respect longitudinal rover axis (array of 6 elements).
- *Vtan_feed*: it is the module of CM velocity given by Inertia Measurement Unit sensor.
- *Sigma_feed*: it is the angle of the module of CM velocity respect longitudinal rover axis given by the IMU sensor.
- *wheel_state_omega*: it is the state of failure for each wheel (array of 6 elements).
- *wheel_state_alpha*: it is the state of failure for each steering system (array of 6 elements).
- *omega_reduction_feed_RPM*: it is the feedback angular velocity given by reduction encoder for each wheel (array of 6 elements).

- *permanent_sensor_fail*: it is the flag that underlines if there are two or more sensors fail in a wheel (array of 6 elements).

The outputs of this block are:

- *Slip*: it is a flag that underlines if the rover has different behaviour respect to wheels configuration.
- *wheel_state_omega_final*: it is the final failure state of each wheel (array of 6 elements).
- *wheel_state_alpha_final*: it is the final failure state of the steering systems. In this case are equal to the input *wheel_state_alpha* (array of 6 elements).
- *Coherent_flag_Vtan*: it is the flag that underlines if all the wheels have correct velocities or not. If the velocities are not coherent between each other the flag is equal to 1.
- *Coherent_flag_sigma*: it is the flag that underlines if all the wheels have correct configurations or not. If the configurations are not coherent each other the flag is equal to 1.
- *mean_Vtan_mm_s*: it is the mean of the calculated CM velocity module of all the wheels.
- *mean_sigma_rad*: it is the mean of the calculated angle of CM velocity module respect longitudinal rover axis of all the wheels.

This block is composed of 3 blocks:

- *Correction_omega_state*
- *Coherence_block*
- *Slip_block*

Also in are present 2 external functions of the previous blocks that calculate the mean of *Vtan_calculate_mm/s* and *sigma_calculate_rad*.
The function applied on *sigma_calculate_rad* is a simple Simulink mean function, instead the function applied on *Vtan_calculate_mm/s* uses the calculated values only if the wheel state is less or equal to 4.

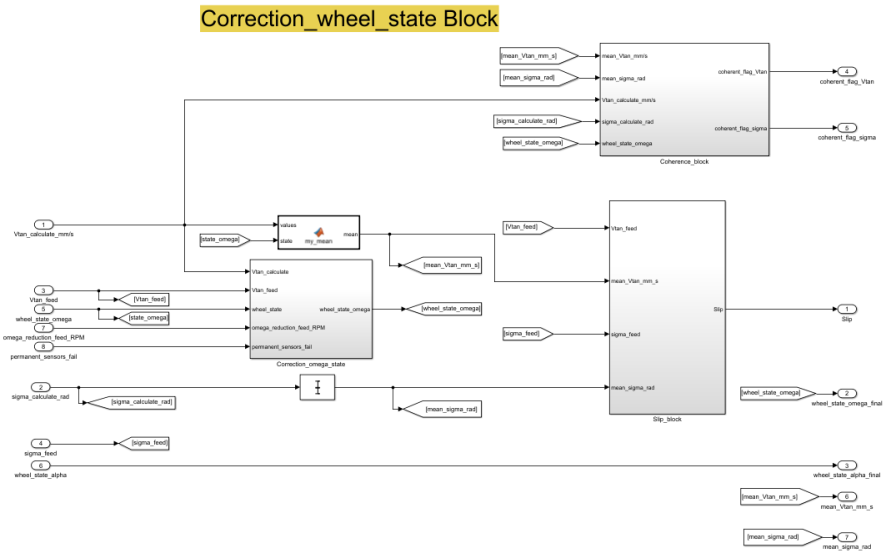


Figure 59: correction_wheel_state Block Simulink scheme

17.8.1 Correction omega state Block

This block corrects the omega failure state for each wheel taking into account the feedback dynamic given by IMU and permanent failure of more than 2 sensors.
The inputs of this block are:

- *Vtan_calculate*: it is the calculated CM velocity module for each wheel (array of 6 elements).
- *Vtan_feed*: it is the feedback of the CM rover velocity module given by IMU.
- *wheel_state*: it is the omega failure state for each wheel (array of 6 elements).
- *permanent_sensors_fail*: it is the flag that underlines if there are two or more sensors fail for each wheel (array of 6 elements).

The output of this block is the corrected omega failure state for each wheel. The correction logic for each wheel is the following.

Correction_wheel_[i]: The inputs of this block are:

- *diff*: it is the difference between *Vtan_calculate[i]* and *Vtan_feed*.

- *wheel_state_omega*: it is the omega failure state of the wheel.
- *perm_sensors_fail*: it is the flag that underlines if there are 2 or more sensors fail in the wheel.
- *abs_omega_reduction*: it is the absolute value of the reduction encoder feedback.

The output of this block is the final omega wheel state *wheel_state*.

The logic step in this block are the following:

- if the *wheel_state_omega* is equal to 5 or 7;
  - if *diff* is less or equal to 0.02
    - if *abs_omega_reduction* is less than 2, then *wheel_state[i]* is equal to 4.
    - otherwise *wheel_state[i]* is equal to 5.
  - otherwise
    - if *wheel_state_omega* equal to 5, then *wheel_state* is equal to 2.
    - otherwise *wheel_state* is equal to 6.
- otherwise
  - if *wheel_state_omega* equal to 3 or 4 and *perm_sensors_fail* equal to 1, then *wheel_state* is equal to 8.
  - otherwise *wheel_state* is equal to *wheel_state_omega*.

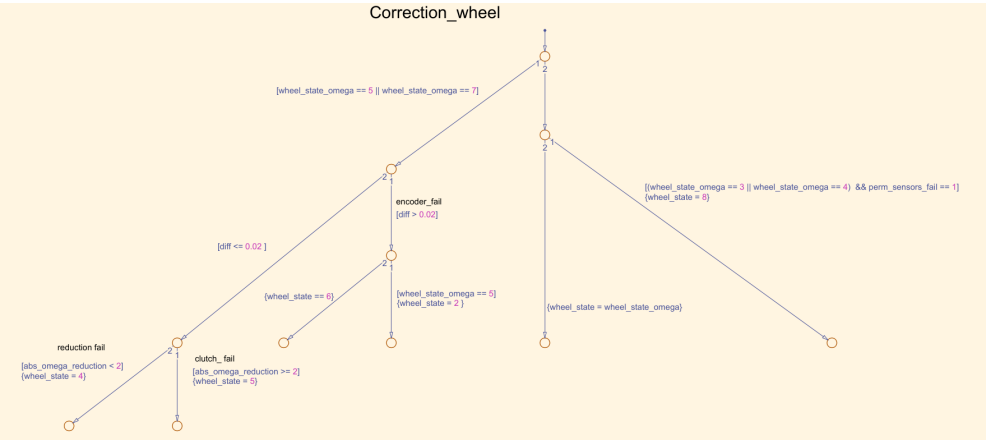


Figure 60: correction_wheel [i] Stateflow chart

17.8.2 Coherent block

This block checks if all the wheels dynamics are coherent or not. The coherences between all the wheels is very important because it can be used during the transient condition to detect possible failure, otherwise, the detection works only during steady-state.
The inputs of this block are:

- *mean_Vtan_mm_s*: it is the mean values of the calculated CM velocity module using steering angle and angular velocity for each wheel.
- *mean_sigma_rad*: it is the mean values of the calculated angle of CM velocity module respect longitudinal rover axis using steering angle and angular velocity for each wheel.
- *Vtan_calculate_mm_s*: it is the calculated CM velocity module using steering angle and angular velocity for each wheel (array of 6 elements).
- *sigma_calculate_rad*: it is the calculated angle of CM velocity module respect longitudinal rover axis using steering angle and angular velocity for each wheel (array of 6 elements.
- *wheel_state_omega*: it is the final wheel failure state (array of 6 elements.

The outputs of this block are:

- *coherent_flag_Vtan*: it is the flag that underlines if all the wheels angular velocities are coherent or not. If it is equal to 0 means that all the wheels are coherent.
- *coherent_flag_sigma*: it is the flag that underlines if all the wheels steering angle are coherent or not. If it is equal to 0 means that all the wheels steering angles are coherent.

The procedures in this blocks are:

1. Calculate, for each wheel, the absolute value of differences between *mean_Vtan_mm/s* and *Vtan_calculate_mm_s*:
   $delta\_Vtan[i] = abs(mean\_Vtan\_mm/s - Vtan\_calculate\_mm\_s[i])$.
2. Calculate, for each wheel, the final *delta_Vtan* taking into account the wheels failure state:
   - If *wheel_state_omega* less than 3: $delta\_Vtan\_final[i] = 0$.
   - else: $delta\_Vtan\_final[i] = delta\_Vtan[i]$.

3. Calculate for each wheel the absolute value of the differences between *mean_sigma_rad* and *sigma_calculate_rad*:
   $delta\_sigma[i] = abs(mean\_sigma\_rad - sigma\_calculate\_rad[i])$.

4. Calculate the maximum value of *delta_Vtan* and compare it with a threshold equal to 135 mm/s. If it is bigger or equal to the threshold the *coherent_flag_Vtan* is equal to 1, otherwise is equal to 0.

5. Calculate the maximum value of *delta_sigma*, if *mean_sigma_rad* is greater than 0.09 rad multiply it to 1/4 .Then compare the results with a threshold equal to 0.01 rad. If the it is bigger or equal to the threshold the *mean_Vtan_mm/s* is greater or equal to 10 mm/s the *coherent_flag_sigma* is equal to 1, otherwise is equal to 0.

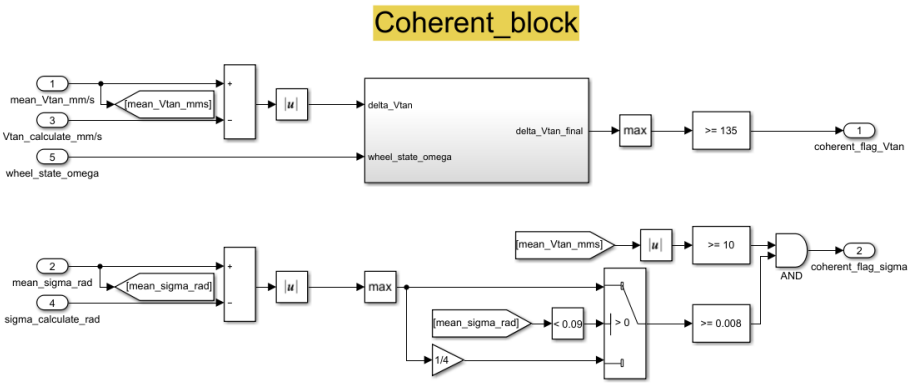


Figure 61: Coherent Block Simulink scheme

17.8.3 Slip block

This block checks if the rover has a different dynamics respect the calculated behaviour by wheels velocity and steering angles.
The inputs of this controller are:

- *Vtan_feed*: it is the feedback of the CM velocity module given by IMU.
- *mean_Vtan_mm_s*: it is the mean of the calculated CM velocity module given by wheels angular velocity.
- *sigma_feed*: it is the feedback angle of CM velocity module given by IMU.
- *mean_sigma_rad*: it is the mean of the calculated angle of CM velocity module given by wheels steering angle.

The output of this block is the *Slip* flag. The flag is equal to 1 if the absolute value of difference between *Vtan_feed* and *mean_Vtan_mm_s* is bigger than 150 mm/s or the absolute value of difference between *sigma_feed* and *mean_sigma_rad* is bigger than 0.08 rad.

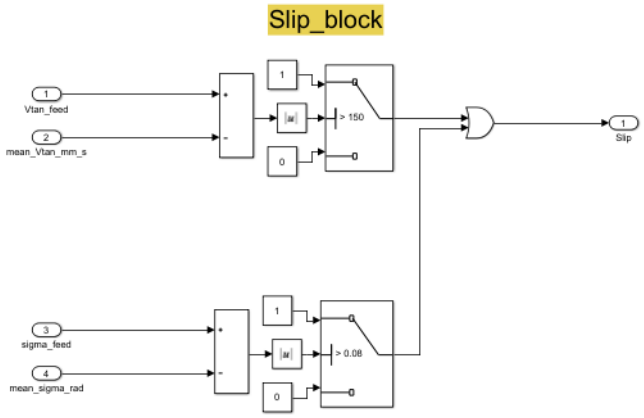


Figure 62: Slip Block Simulink scheme

18 Mitigation Block

The Mitigation block mitigates the selected failure for each wheel to reduce the effect of them an the behaviour of the rover.
The inputs of this block are:

- *enable_correction*: it is the command flag that enables the wheels angular velocity correction given by Rocker-bogie trigonometric model.
- *posx_wheels*: it is the position of the wheels along longitudinal rover axis respect CM calculated by Rocker-bogie trigonometric model (array of 6 elements).
- *posz_wheels*: it is the position of the wheels along normal rover axis respect CM calculated by Rocker-bogie trigonometric model (array of 6 elements).
- *Slip_flag*: it is the flag that underlined if the rover is slipping or not respects theoretical dynamic calculated by current wheels configuration.
- *wheel_state_omega_final*: it is the failure state of each wheel (array of 6 elements).

- *wheel_state_alpha_final*: it is the failure state of the steering system for each wheel (array of 6 elements).
- *coherent_flag_Vtan*: it is the flag that underlines if all the wheels angular velocity are coherent or not between each others. If it is equal to 0 means that the wheels are coherent.
- *coherent_flag_sigma*: it is the flag that underlines if all the wheels steering angle are coherent or not between each others. If it is equal to 0 means that the wheels are coherent.
- *mean_Vtan_mm_s*: it is the mean value of the calculated CM velocity module of the wheels.
- *mean_sigma_rad*: it is the mean values of the calculated angle of CM velocity module of the wheels.
- *omega_wheel_feed_RPM*: it is the feedback value of the angular velocity of each wheel (array of 6 elements).
- *alpha_wheel_feed_rad*: it is the feedback value of the steering angle of each wheel (array of 6 elements).

The outputs of this block are:

- *omega_motor_feed_RPM*: it is the final feedback angular velocity of the wheels motor (array of 6 elements).
- *alpha_motor_feed_rad*: it is the final feedback motor steering angle of the wheels (array of 6 elements).
- *error*: it is the flag underlines if there are some critical failures inside the rover system. If it is equal to 1 means that something is braked critically.
- *clutch_command*: it is the flag command that activates or not the clutch for each wheel (array of 6 elements). If it is equal to 1 means that the clutch is activated, otherwise motor and reduction are not connected (freewheel).
- *motor_stop*: it is the flag command that deactivates the motor for each wheel (array of 6 elements). It is correlated with *enable_clutch*. If the flag is equal to 0 means that the motor can work normally, otherwise it must be stopped (no torque).
- *permanent_sensors_fail*: it is the flag that underlines if there are 2 or more sensors fail for each wheel (array of 6 elements). When the flag is equal to 1 means that there are 2 or more sensors failed.

This block is divided into 2 subsystems:

- *Mitigation_sys*: this block decides the correct mitigation for each failure type.
- *Feedback_block*: this block selects the correct feedbacks sent to the Mobility controller.

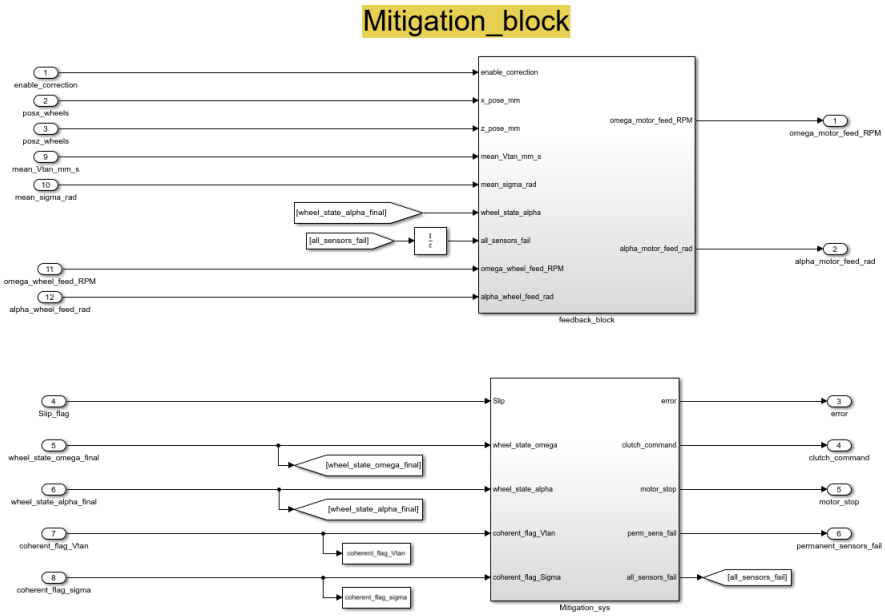


Figure 63: Mitigation Block Simulink scheme

18.1 (Mitigation block) Mitigation wheel [i]

This block is inside *Mitigation_block*. It is decided to not explain it because its only function is to produce the vector of 6 elements of each output of *Mitigation_wheel[i]* and connect the error flag of each wheel with an *Or* logic. The inputs of this block are:

- *Slip*: it is the flag that underlined if the rover is slipping or not respect to theoretical dynamic calculated by current wheels configuration.
- *wheel_state_omega[i]*: it is the failure state for wheel [i].
- *wheel_state_alpha[i]*: it is the failure state of steering system for wheel [i].
- *coherent_flag_Vtan*: it is the flag that underlines if the velocity of wheels are coherent with each other or not.

- *coherent_flag_sigma*: it is the flag that underlines if the velocity of wheels are coherent with each other or not.
- *clutch_command_feed[i]*: it is the feedback flag command that enables the clutch [i]. When it is equal to 1 means that the clutch is activated.
- *motor_stop_feed[i]*: it is the feedback flag command that stops the motor [i]. When it is equal to 1 means that the motor must be stopped (no torque).
- *error_feed*: it is the feedback flag that underlines if there is a critical failure on the wheels or on the steering systems.
- *counter_omega_feed*: it is a value usable in order to detect if a wheel failure is permanent or not.
- *counter_alpha_feed*: it is a value usable in order to detect if a steering system failure is permanent or not.

The outputs of these blocks are:

- *error*: it is the flag that underlines if there is a critical failure in the wheel [i] or in its steering system.
- *clutch_command[i]*: it is the flag command that enables the clutch [i]. When it is equal to 1 means that the clutch is activated.
- *motor_stop[i]*: it is the flag command that stops the motor [i]. When it is equal to 1 means that the motor must be stopped (no torque).
- *counter_omega*: it is a value usable to detect if a wheel failure is permanent or not.
- *counter_alpha*: it is a value usable to detect if a steering system failure is permanent or not.
- *permanent_sensors_fail*: it the flag that underlines if there are 2 or more sensors fail in the wheel. When the flag is equal to 1 means that there are 2 or more sensors failed.
- *all_sensors_fail*: it is the flag that underlines if all the sensors fail or not in a wheel. When it is equal to 1 means that all sensors fail and there is not any coherent feedback for that wheel.

In this block, there are 2 switch-case, one for the wheel failure and the other for the steering system.
The first switch-case receive value between 0 and 8, without value 7, one for each type of failure.
The operations inside each enable subsystem are:

- Value 0:
  it means that no fail are acting in the wheel. The corresponding outputs are:
  - *error* = 0.
  - *clutch_command* = *clutch_command_feed*.
  - *motor_stop* = *motor_stop_feed*.
  - *counter_omega* = 0.
  - *permanent_sensors_fail* = 0.
  - *all_sensors_fail* = 0.
- Value 1:
  it means that motor encoder or hall sensors fails in the wheel. The corresponding outputs are:
  - *error* = 0.
  - *clutch_command* = *clutch_command_feed*.
  - *motor_stop* = *motor_stop_feed*.
  - *counter_omega* = 0.
  - *permanent_sensors_fail* = 0.
  - *all_sensors_fail* = 0.
- Value 2:
  it means that wheel reduction encoder fails. The corresponding outputs are:
  - *error* = 0.
  - *clutch_command* = *clutch_command_feed*.
  - *motor_stop* = *motor_stop_feed*;
  - *counter_omega* = 0.
  - *permanent_sensors_fail* = 0.
  - *all_sensors_fail* = 0.
- Value 3:
  it means that wheel's motor fails. The corresponding outputs are:
  - *error* = 0.
  - *clutch_command* = 0 if *counter_omega* greater or equal to 5; otherwise *clutch_command* = 1. A logic AND relates the output flag with the feedback.
    The machine state in this logic is:

current clutch command negative logic	**clutch command feed**	**clutch command output**
0	0	0
1	0	0
0	1	0
1	1	1

Thanks to this logic, when the *clutch_command* is sets to 0 (clutch open), the output remains always 0, independent from the calculated one.

- *motor_stop* = 1 if *counter_omega* greater or equal to 5; otherwise*motor_stop* = 0. A logic OR relates the calculated flag with its feedback.
  The machine state in this logic is:

current motor stop	**motor stop feed**	**motor stop output**
0	0	0
1	0	1
0	1	1
1	1	1

Thanks to this logic, when the *motor_stop* is sets to 1 (motor off without any torque), the output remains always 1, independent from the calculated one.

- *counter_omega* = *counter_omega* + 1 if *wheel_state_past* equal to 3 and *coherent_flag_Vtan* equal to 1; otherwise *counter_omega* = 0.
- *permanent_sensors_fail* = 0.
- *all_sensors_fail* = 0.

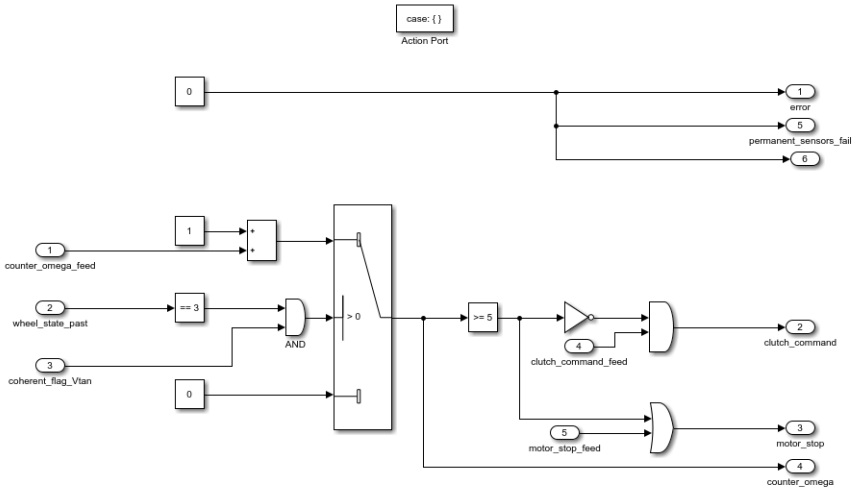


Figure 64: Motor fail switch case Simulink scheme

- Value 4:
  it means that wheel's reduction fails. The corresponding outputs are:
  - *error* = 1 if *counter_omega* greater or equal to 5; otherwise *error* = 0.
  - *clutch_command* = 0.
  - *motor_stop* = 1.
  - *counter_omega* = *counter_omega* + 1 if *wheel_state_past* equal to 4 and *coherent_flag_Vtan* is equal to 1; otherwise *counter_omega* = 0.
  - *permanent_sensors_fail* = 0.
  - *all_sensors_fail* = 0.
- Value 5:
  it means that wheel clutch fails. The corresponding outputs are:
  - *error* = 0.
  - *clutch_command* = 0 if *counter_omega* greater or equal to 10 otherwise *clutch_command* = 1.
  - *motor_stop* = 1 if *counter_omega* greater or equal to 10; otherwise *motor_stop* = 0.
  - *counter_omega* = *counter_omega* + 1 if *wheel_state_past* equal to 5; otherwise *counter_omega* = 0.

- *permanent_sensors_fail* = 0.
- *all_sensors_fail* = 0.

- Value 6:
  it means that two sensors fail in the wheel. The corresponding outputs are:
  - *error* = 0.
  - *clutch_command* = 1.
  - *motor_stop* = 0.
  - *counter_omega* = *counter_omega* + 1 if *wheel_state_past* equal to 6; otherwise *counter_omega* = 0.
  - *permanent_sensors_fail* = 1 if *counter_omega* greater or equal to 5; otherwise *permanent_sensors_fail* = 0.
  - *all_sensors_fail* = 0.

- Value 8:
  it means that all sensors fail in the wheel. The corresponding outputs are:
  - *error* = 1 if *counter_omega* greater or equal to 8; otherwise *error* = 0.
  - *clutch_command* = 0 if *counter* greater or equal to 3; otherwise *clutch_command* = 1. A logic AND relates the output flag with the feedback.
    The machine state in this logic is:

current clutch command negative logic	**clutch command feed**	**clutch command output**
0	0	0
1	0	0
0	1	0
1	1	1

    Thanks to this logic, when the *clutch_command* is sets to 0 (clutch open), the output remains always 0, independent from the calculated one.
  - *motor_stop* = 1 if *counter* greater or equal to 3; otherwise *motor_stop* = 0. A logic OR relates the calculated flag with its feedback.

The machine state in this logic is:

current motor stop	**motor stop feed**	**motor stop output**
0	0	0
1	0	1
0	1	1
1	1	1

Thanks to this logic, when the *motor_stop* is sets to 1 (motor off without any torque), the output remains always 1, independent from the calculated one.

- *counter* is equal to *counter_feed* + 1 if *wheel_state_past* is equal to 8, otherwise *counter* is equal to 0.
- *counter_omega* = *counter_omega* + 1 if *wheel_state_past* equal to 6 and *slip* equal to 1; otherwise *counter_omega* = 0.
- *permanent_sensors_fail* = 1.
- *al_sensors_fail* = 1.

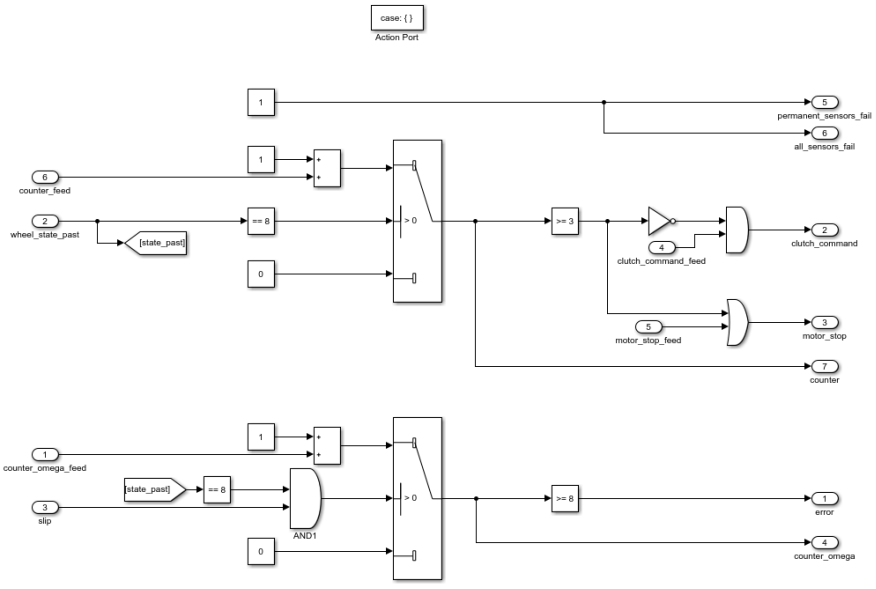


Figure 65: All sensors fail switch case Simulink scheme

- Default:
  it is the default case, it is used if a critical error is persistence. The corresponding outputs are:

- *error* = 1.
- *clutch_command* = 1.
- *motor_stop* = 0 .
- *counter_omega* = 0.
- *permanent_sensors_fail* = 0.
- *all_sensors_fail* = 0.

The second switch-case is dedicated to the steering system failure. It receives value between 0 and 4, one for each failure type.
The operations inside each enable subsystem are:

- Value 0:
  it means that no failure is present in the steering system. The corresponding outputs are:
  - *error* = 0.
  - *counter_alpha* = 0.
- Value 1:
  it means that steering motor encoder fails. The corresponding outputs are:
  - *error* = 0.
  - *counter_alpha* = 0.
- Value 2:
  it means that steering reduction encoder fails. The corresponding outputs are:
  - *error* = 0.
  - *counter_alpha* = 0.
- Value 3:
  it means that a critical failure is present in the steering system. The critical failure can be the failure of the motor or reduction and a failure of both the encoders. The corresponding outputs are:
  - *error* = 1 if *counter_alpha* is greater or equal to 5; otherwise *error* = 0.
  - *counter_alpha* = *counter_alpha* + 1 if *wheel_state_past* equal to 3 and *coherent_flag_sigma* equal to 1; otherwise *counter_alpha* = 0.
- Value 4:
  it means that both encoders fails in the steering system. The corresponding outputs are:

- *error* = 1 if *counter_alpha* is greater or equal to 5; otherwise *error* = 0.
- *counter_alpha* = *counter_alpha* + 1 if *wheel_state_past* equal to 4, otherwise *counter_alpha* = 0.

- Default:
  it is the default case, it is used if a critical error is persistence. The corresponding outputs are:
  - *error* = 1.
  - *counter_alpha* = 0.

Before both the switch case it is present a function that maps persistence critical failure, when error is equal to 1, at default case (*wheel_state* = -1 and *alpha_state* = -1), neglecting the current failure state.

18.2 Feedback block

This block calculates correct feedbacks sent to the Mobility Controller.
The inputs of this block are:

- *enable_correction*: it is the command flag that enables the correction on the wheel angular velocity due to Rocker-bogie trigonometric model.
- *x_pose_mm*: it is the position of the wheels along longitudinal rover axis respect to CM calculated by Rocker-bogie trigonometric model (array of 6 elements).
- *z_pose_mm*: it is the position of the wheels along normal rover axis respect to CM calculated by Rocker-bogie trigonometric model (array of 6 elements).
- *mean_Vtan_mm_s*: it is the mean values of the calculated CM velocity module using steering angle and angular velocity for each wheel.
- *mean_sigma_rad*: it is the mean values of the calculated angle of CM velocity module using steering angle and angular velocity for each wheel.
- *wheel_state_omega*: it is the state of failure for each wheel (array of 6 elements).
- *wheel_state_alpha*: it is the state of failure of the steering system for each wheel (array of 6 elements).

- *omega_wheel_feed_RPM*: it is the feedback value of the angular velocity of each wheel (array of 6 elements).
- *alpha_wheel_feed_rad*: it is the feedback value of the steering angle of each wheel (array of 6 elements).
- *all_sensors_fail*: it is the flag that underlined if all the sensors are failed for each wheel (array of 6 elements).

The outputs of this block are:

- *omega_motor_feed_RPM*: it is the final motor feedbacks of the wheels sent to the Mobility controller (array of 6 elements).
- *alpha_motor_feed_rad*: it is the final motor feedbacks of the steering system sent to the Mobility controller (array of 6 elements).

In this block there are 2 subsystems:

- *Ackermann_+_correction*: this block calculates for each wheel its angular velocity using as input the mean value of *Vtan* and *sigma* previously calculated. Also, it is applied the correction due to the trigonometric model of Rocker-bogie if *enable_correction* flag is activated.
- *feed_function*: this function select the correct angular velocity for each wheel taking into account the current wheel and steering failure state.

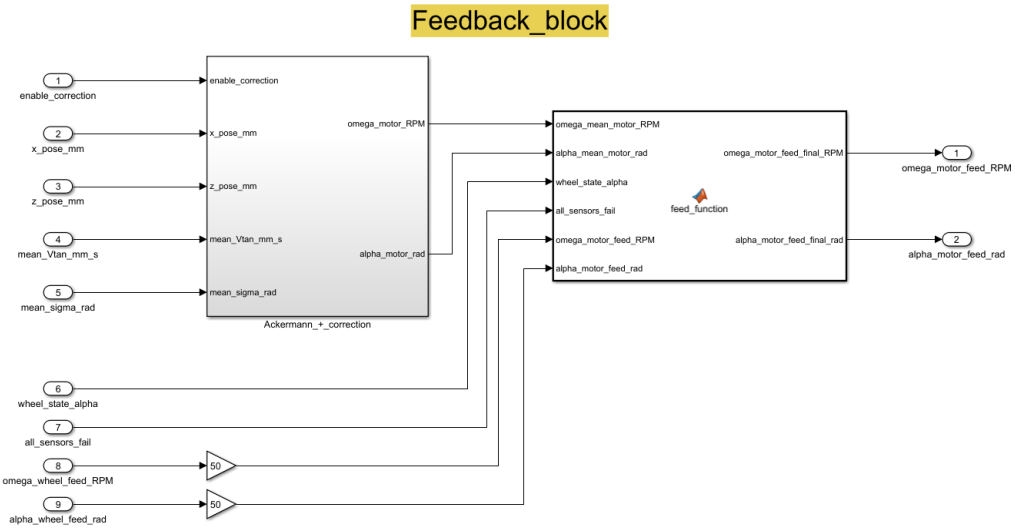


Figure 66: Feeback Block Simulink scheme

18.2.1 Ackermann + correction Block

This block calculates the feedback angular velocity for each wheel using Ackermann model and correction due to Rocker-bogie trigonometric model. The subsystem that composes this block are:

- *Steering_block*: it is the same block of Mobility Controller. Its description is in section 11.2.5.
- *Velocity_block*: it is the same block of Mobility Controller. Its description is in section 11.2.6.
- *Correction_block*: inside this subsystem there is *Omega_correction_block* that is the same block of Mobility controller. Its description is in section 12.1.2.

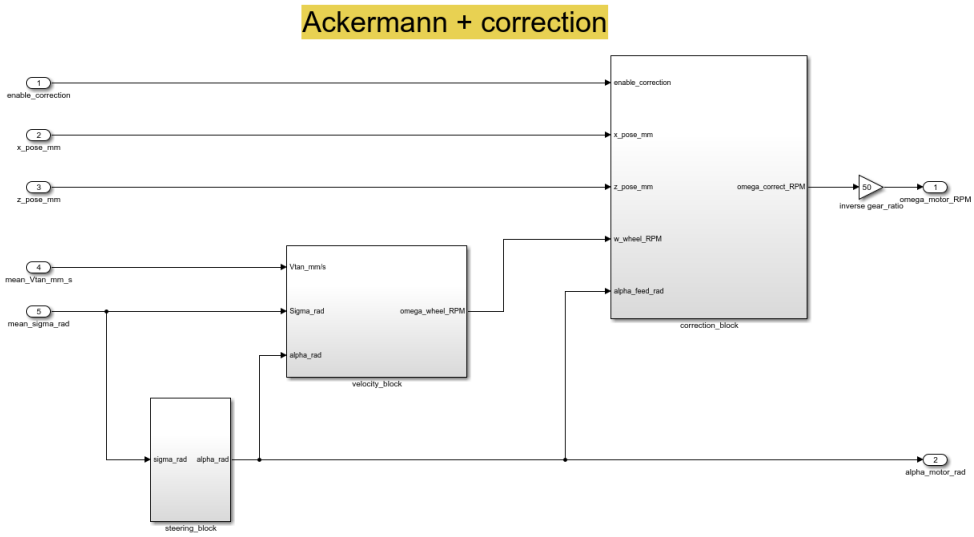


Figure 67: Ackermann + Correction Block Simulink scheme

18.2.2 Feed function

This function selects the correct angular velocity for each wheel taking into account the current wheel and steering failure state.
The inputs of this function are:

- *omega_mean_motor_RPM*: it is the angular velocity of the wheels calculated by *Ackermann_+_correction* using as input the mean value of *Vtan* and *sigma* (array of 6 elements).
- *omega_state*: it is the wheel failure state (array of 6 elements).

- *Omega_motor_feed_RPM*: it is the feedback of the wheels multiply by gear ratio (array of 6 elements).
- *all_sensors_fail*: it is the flag that underlines if all the sensors of the wheel are fail or not (array of 6 elements).
- *alpha_motor_feed_rad*: it is the feedback of the wheels steering angle multiply by gear ratio (array of 6 elements).
- *alpha_mean_motor_rad*: it is the steering angle of the wheels calculated by *Ackermann_+_correction* using as input the mean value of *Vtan* and *sigma* (array of 6 elements).

The output are the feedbacks *omega_motor_feed_final_RPM* (array of 6 elements) and *alpha_motor_feed_final_rad* that are sent to the Mobility Controller.

The operations in this function, for each wheel, are two:

- Check if $all\_sensors\_fail[i]$ is equal or not to 1.
  If it is, $omega\_motor\_feed\_final\_RPM[i] = omega\_mean\_motor\_RPM[i]$,
  otherwise $omega\_motor\_feed\_final\_RPM[i] = omega\_motor\_feed\_RPM[i]$.
- Check if $wheel\_state\_alpha[i]$ is equal or not to 4.
  If it is, $alpha\_motor\_feed\_final\_rad[i] = alpha\_mean\_motor\_rad[i]$,
  otherwise $alpha\_motor\_feed\_final\_rad[i] = alpha\_motor\_feed\_rad[i]$

19 Results

In order to have a complete analysis of the algorithm's dynamic, it is decided to spit the tests in 2 part.
The first part of tests is done on Simulink, analysing the feedback given by Coppeliasim after the simulation. This type of test was fundamental to check the correctness of the detection algorithm.
The second part of tests is done on Coppeliasim with the integration of the algorithm in the simulation thanks to code-generation. Those tests were useful to analyse the work of the mitigation algorithm.
Following are shown the results of the test.

19.1 Test results on Simulink

The results of the tests for wheel failure are the following:

Injected failure	**Theoretical detection time (seconds)**	**Real detection time delay(seconds)**
H-TM	0.1	0.1
E-TM	0.1	0.2
E-TRG	0.1	0.1
TM	0.1	0.1
TRG	0.6	0.7
TC	0.1	4.8
H-TM + TM	0.1 H-TM + 0.1 TM	0.1 H-TM + 0.1 TM
H-TM + TRG	0.1 H-TM + 0.6 TRG	0.1 H-TM + 0.9 TRG
H-TM + TC	0.1 H-TM + 0.1 TC	0.3 H-TM + 1.6 TC
H-TM + E-TRG	0.1 H-TM + 0.1 E-TRG	0.1 H-TM + 0.1 E-TRG
H-TM +E-TM	0.1 H-TM + 0.1 E-TM	0.1 H-TM + 0.1 E-TM
E-TM +E-TRG	0.1 E-TM + 0.1 E-TRG	0.1 E-TM + 0.1 E-TRG
E-TM + TM	0.1 E-TM + 0.1 TM	0.1 E-TM + 0.1 TM
E-TM + TC	0.1 E-TM + 0.1 TC	0.2 E-TM + no detected TC
E-TM + TRG	0.1 E-TM + 0.6 TRG	0.1 E-TM + 0.7 TRG
E-TRG + TM	0.1 E-TRG + 0.1 TM	0.1 E-TRG + 0.1 TM
E-TRG +TRG	0.1 E-TRG + 0.6 TRG	0.1 E-TRG + 0.6 TRG
E-TRG + TC	0.1 E-TRG + 0.1 TC	0.1 E-TRG + No detected TC
TM + TC	0.1 TM + 0.1 TC	0.1 TM + 0.1 TC
TC +TRG	0.1 TC + 0.6 TRG	0.1 TC + 0.8 TRG
H-TM + E-TM +E-TRG	0.1 H-TM + 0.1 E-TM + 0.1 E-TRG	0.1 H-TM + 0.1 E-TRG + 0.1 E-TRG

Most of the failure have theoretical delay about one sampling time (0.1 seconds), only the reduction fail needed more time to be detected. Unfortunately, the failure of the clutch was very difficult to detect because the rover dynamic changes too little when the failure is present.

For what concerns the failure of the steering system the time delay are very large, caused by too little variation of the dynamic of the rover. In fact, the variations of the steering angles are very low respect to wheels velocity. The rover steering angle is selected by the user then its dynamic is very low, instead, the velocity of the wheels is selected by the user but also changed by the controller algorithm to avoid slipping.
The results of the test are the following:

Injected Failure	**Theoretical detection time delay(seconds)**	**Real detection time (seconds)**	**Delta angle [deg]**
SM or SRG	0.1	5.7	2.3
E-SM	0.1	2.6	7.7 (with maximum steer velocity)
E-SRG	0.1	5.6	5.9 (with maximum steer velocity)
E-SM + E-SRG	0.1 E-SM + 0.1 E-SRG	2.6 E-SM + 2.1 E-SRG	5.8 E-SM (with maximum steer velocity)+ 2.3 E-SRG
E-SM+ SM	0.1 E-SM + 0.1 SM	2.7 E-SM + 1.4 SM	6.9 E-SM (with maximum steer velocity) + 3 SM
E-SRG + SM	0.1 E-SRG + 0.1 SM	2.2 E-SRG + 3.2 SM	4.6 E-SRG + 5.5 SM (with maximum steer velocity)

The maximum delta steering angle in the table represent the maximum variation of the steer respect desired position. At maximum steering velocity the delta is around 6.5 degree, but the failure is detected quite immediately. The real detection time is quite bigger because the user doesn't change immediately the steering angles and if the steering angle remains constant, the failure can not be detected.

19.2 Test result on Coppeliasim

The results of the tests for wheel failure are the following:

Injected failure	**Theoretical detection time (seconds)**	**Real detection time delay (seconds)**	**Expected mitigation delay (seconds)**	**Real mitigation delay (seconds)**
H-TM	0.1	0.1	none	none
E-TM	0.1	0.1	none	none
E-TRG	0.1	0.1	none	none
TM	0.1	0.1	0.5	0.5
TRG	0.6	0.6	0.5	0.5
TC	...	...	...	...
H-TM + TM	0.1 H-TM + 0.1 TM	0.1 H-TM + 0.1 TM	none H-TM + 0.5 TM	none H-TM + 1.2 TM
H-TM + TRG	0.1 H-TM + 0.5 TRG	0.1 H-TM + 0.6 TRG	none H-TM + 0.5 TRG	none H-TM + 0.5 TRG
H-TM + TC	...	...	...	...
H-TM + E-TRG	0.1 H-TM + 0.1 E-TRG	0.1 H-TM + 0.1 E-TRG	none H-TM + none E-TRG	none H-TM + none E-TRG
H-TM +E-TM	0.1 H-TM + 0.1 E-TM	0.1 H-TM + 0.2 E-TM	none H-TM + none E-TM	none H-TM + none E-TM
E-TM +E-TRG	0.1 E-TM + 0.1 E-TRG	0.1 E-TM + 0.1 E-TRG	none E-TM + none E-TRG	none E-TM + none E-TRG
E-TM + TM	0.1 E-TM + 0.1 TM	0.1 E-TM + 0.1 TM	none E-TM + 0.5 TM	none E-TM + 0.5 TM
E-TM + TC	...	...	...	...
E-TM + TRG	0.1 E-TM + 0.6 TRG	0.1 E-TM + 0.6 TRG	none E-TM + 0.5 TRG	none E-TM + 0.5 TRG
E-TRG + TM	0.1 E-TRG + 0.1 TM	0.1 E-TRG + 0.1 TM	none E-TRG + 0.5 TM	none E-TRG + 0.6 TM
E-TRG + TRG	0.1 E-TRG + 0.6 TRG	0.1 E-TRG + 0.6 TRG	none E-TRG + 0.5 TRG	none E-TRG + 0.5 TRG
E-TRG + TC	...	...	...	...
TM + TC	0.1 TM + 0.1 TC	0.1 TM + none TC	0.5 TM + 0.5 TC	0.5 TM + none TC

TC +TRG	...	...	...	...
H-TM + E-TM +E-TRG	0.1 H-TM + 0.1 E-TM + 0.1 E-TRG	0.1 H-TM + 0.1 E-TRG + 0.1 E-TRG	none H-TM + none E-TM + 0.3 E-TRG	none H-TM + none E-TM + 0.3 E-TRG

The tests in real time condition validate the previous results on Simulink. All the test with Clutch are not performed because , as it is previously underlined, the failure of this component is not detected and consequently mitigated.

The results of the steering failures tests are the following:

Injected Failure	**Theoretical detection time delay (seconds)**	**Real detection time (seconds)**	**Delta angle [deg]**	**Expected mitigation delay (seconds)**	**Real mitigation delay (seconds)**
SM or SRG	0.1	1	3.4	0.5	1.2
E-SM	0.1	3.4	7.3 (with maximum steer velocity)	none	none
E-SRG	0.1	3.8	8.4 (with maximum steer velocity)	none	none
E-SM + E-SRG	0.1 E-SM + 0.1 E-SRG	3.3 E-SM + 0.9 E-SRG	negligible because was in already in transient	none E-SM + 0.5 E-SRG	none E-SM + 1.3 E-SRG
E-SM+ SM	0.1 E-SM + 0.1 SM	5.7 E-SM + 1.9 SM	6.4 (with maximum steer velocity)	none E-SM + 0.5 SM	none E-SM + 0.8 SM
E-SRG + SM	0.1 E-SRG + 0.1 SM	3.9 E-SRG + 0.9 SM	4.6	none E-SRG + 0.5 SM	none E-SRG + 0.5 SM

The results are in according to the tests on Simulink. The big delta angles in some failures do not produce a visible uncorrected behaviour of the

rover and consequently too higher stress on the steering system.

Part VII
Thesis Results

The thesis aim is the improvement of the driveability of the rover using a Mobility controller and to increase its dependability in a mission-critical scenario.
Both Mobility controller and FMEDA algorithms are built using a Model-Based approach.

For what concern the driveability the Controller sets the correct steering angles and angular velocities of all the wheels taking into account the Ackermann model and Rocker-bogie dynamic.

The Diagnostic algorithm recognizes all the failure types except the failure of the clutch. During the simulation, it was underlined that if a wheel's clutch fails, the rover dynamic doesn't change. This condition can be present because it was decided that the friction coefficient between the wheel's thread and terrain are about 0.3. The friction coefficient is the minimum value that avoids lateral slipping during nominal motion.
This assumption is done because it is preferred to analyse the worst condition to have better detection.
The clutch can be detected using also the feedback absorbed current by the motor added to the current checking. Without using the motor model it is possible, experimentally, to understand the absorbed current of the motor in the no-load condition in the used range of angular velocities. This values can be stored in a look-up table and compared with the actual absorbed current taking into account the current angular velocity. If the difference of them is less than a threshold means that the motor is in no-load and the clutch is broken.

The results of the simulation on Coppeliasim says that the Mitigation algorithm works properly. All the actuated mitigations are the most safety for each failure, but for all-wheel sensors failure, it is possible another approach. In the thesis, this failure is mitigated opening the clutch and turning off the wheel motor, also if the rover slip for a few time the rover stops because its dynamic can be unpredictable.
The other approach can be the estimating of the angular wheel velocity (already present in the current mitigation algorithm) and using this value as a feedback for the Mobility controller. In this case, the rover can avoid the losing of the traction in the fail wheel but it is needed a better estimator in order to reduce the estimation error.
Future work correlated with this thesis can be:

- Install both the algorithm in a rapid prototyping board in order to check the correctness of their work in Processor in the loop test.
- Install both the algorithm in the rover and check the correctness of their work.
- To increase rover automation build an arm controller based on Model-Based approach and its Diagnostic/Mitigation algorithm taking a look the possible failures researched in FMEA thesis section.

Part VIII
References

1. National Aeronautic and Space Administration (1975) *Apollo Program Summary Report* JSC-09423

2. J. N. James et al (1965) *Mariner IV Mission to Mars (Part 1)*, NASA, Technical Report No. 32-782

3. G.M. Martinez, C.N.Newman, A.De Vicente-Retortillo, E. Fischer, N. O. Renno, M. I. Richardson, A. G. Fairèn, M. Genzer, S. D. Guzewich, R. M. Haberle, A.-M.Harri, O. Kemppinen, M. T. Lemmon, M. D. Smith, M. de la Torre-Juàrez & A. R. Vasavada (2017) *The Modern Near-Surface Martian Climate: A Review of In-situ Meteorological Data from Viking to Curiosity*

4. Sylvain Piqueux, Jennifer Buz, Cristopher S.Edwards, Joshua L. Bandfield, Armin Kleinbohl, David M. Kass, Paul O. Hayne, The MCS, THEMIS Teams (2019) *Widespread Shallow Water Ice on Mars at High Latitudesand Midlatitudes*

5. R. S. Saunders, R. E. Arvidson, G. D. Badhwar, W. V. Boynton, P. R. Christensen, F. A. Cucinotta, W. C. Feldman, R. G. Gibbs, C. Kloss JR., M. R. Landano, R. A. Mase, G. W. Mcsmith, M. A. Meyer, I. G. Mitrofanov, G. D. Pace, J. J. Plaut, W. P. Sidney, D. A. Spencer, T. W. Thompson and C. J. Zeitlin (2003) *2001 Mars Odyssey Mission Summary*

6. European Space Agency (2004)*Mars Express: A European Mission to the Red Planet* SP-1240

7. European Space Agency (2009) *Mars Express: The Scientific Investigations* ESA SP-1291

8. M. D.(Dan) Johnston, James E. Graf, Richard W. Zurek, Howard J. Eisen, Benhan Jai, James K. Erickson (2015) *The Mars Reconnaissance Orbiter Mission: From Launch to the Primary Science Orbit*

9. Venkatesan Sundararajan (2013) *Mangalyaan - Overview and Technical Architecture of India's First Interplanetary Mission to Mars*

10. NASA OIG (2013) *Mars Atmosphere and Volatile Evolution (MAVEN) Project* (IG-13-009)

11. H. Renault, N. Sergent, M.Chevallier, N.Kutrowski, A. Bacchetta, D. Temperanza (2015) *ExoMars 2016, Orbiter modul bus a GND development update*

12. B. M. Shaughnessy (2004) *Development of the Thermal Design for the Beagle 2 Mars Lander*

13. Dwayne Brown, Guy Webster, Sara Hammond, Lori Stiles, Gary Napier, Isabelle Laplante (2008) *Phoenix Landing: Mission to the Martian Polar North*

14. Robert Bonitz, Lori Shiraishi, Matthew Robinson, Joseph Carsten, Richard Volpe, Ashitey Trebi-Ollennu (2009) *The Phoenix Mars Lander Robotic Arm*

15. Robert G. Bonitz, Lori Shiraishi, Mattew Robinson, Raymond E. Arvidson,P. C. Chu, J. J. Wilson, K. R. Davis, G. Paulsen, A. G. Kusack, Doug Archer, Peter Smith (2008) *NASA Mars 2007 Phoenix Lander Robotic Arm and Icy Soil Acquisition Devise* J. Geophys. Res., 113

16. A. Gülhan (2018) *COMARS+ Instrumentation Package of the Schiaparelli Lander of the ExoMArs 2016 Mission*

17. A. Gülhan, T. Thiele, F. Siebe, R. Kronen (2018) *Combined Intrumentation Package COMARS+ for the ExoMars Schiaparelli Lander* Space Science Review, 214

18. W. B. Banerdt et al (2013) *InSight: A Discovery Mission to Explore the Interior of Mars* 44th Lunar and Planetary Science Conference, 1915

19. A. Trebi-ollennu, W. Kim, K. Ali, O. Khan, C. Sorice, P. Bailey, J. Umland, R. Bonitz, C. Ciarleglio, J. Knight, N. Haddad, K. Klein, S. Nowak, D. Klein, N. Onufer, K. Glazebrook, B. Kobeissi, E. Baez, F. Sarkissian, J. Lin (2018) *In Sight MArs Lander Robotics Instrument Deployment System* Space Science Reviews, 214

20. R. Arvidson, J. Bell, P. Belutta, N. Cabrol, J. Catalano, J. Cohel, L. Crumpler, D. Des Marais, T. Estlin, W. Farrand, R. Gellert, J. Grant, R. Greenberger, E. Guinness, K. Henkenhoff, J. Herman, K. Iagnemma, J. Johnson, G. Klingelhoefer, A. Yen (2010) *Spirit Mars Rover Mission : Overview and selected results from the northern Home Plate Winter Haven to the site of Scamander creter* Journal of Geophysical Research, 115

21. R.E. Arvidson, J.W. Ashley, J.F. Bell, M. Chojnacki, J. Cohen, T.E. Economou, W.H. Farrand, R. Fergason, I. Fleischer, P. Geissler, R. Gellert, M.P. Golombek, J.P. Grotzinger, E.A. Guinness, R.M. Haberle, K.E. Herkenhoff, J.A. Herman, K.D. Iagnemma, B.L. Jolliff, J.R. Johnson, G. Klingelhöfer, A.H. Knoll, A.T. Knudson, R. Li, S.M.

McLennan, D.W. Mittlefehldt, R.V. Morris, T.J. Parker, M.S. Rice, C. Schröder, L.A. Soderblom, S.W. Squyres, R.J. Sullivan, M.J. Wolff (2011) *Opportunity Mars Rover mission: Overview and selected results from Purgatory ripple to traverses to Endeavour crater* Journal of Geophysical Research, Vol. 116

22. R. Welch, D. Limonadi, R. Manning (2013) *Systems Engineering the Curiosity Rover: A Retrospective* SoSE

23. J. Grotzinger (2013) *Analysis of Surface Materials by the Curiosity Mars Rover* Science (New York,N.Y.), 341, 1475

24. S. Kassel (1971) *Lunokhod-1 Soviet Lunar Surface Vehicle* ARPA order No.189-1

25. A.T. Basilevsky, A.M. Abdraklimov, J.W. Head, C.M. Pieters, Y.Wu, L.Xiao (2015) *Geologic characteristic of the Luna 17/Lunokhod1 and Chang'E-3/Yutu landing sites, Northwest Mare Imbrium of the Moon* Elsevier

26. MCTamarney L.,Hammon S. (1989). *Mars Rover Concept Development* Proceedings of Spie

27. Matthias Winter, Sergio Rubio, Richard Lancaster, Chris Barclay, Nuno Silva, Ben Nye and Leonardo Bora (2017) *Detailed Description of the High-Level Autonomy Functionalities Developed fot the Exomars Rover*

28. J. Trunins, A. Curley, B. Osborne (2011) *Design of Mars rover mobility system* 62nd International Astronautical Congress (IAC), 11

29. Y. Kim, W. Eom, J.H. Lee, E.S. Sim (2012) *Design of Mobility System for Ground Model of Planetary Exploration Rover* Journal of Astronomy and Space Sciences, 29, 413-422

30. S. Aswath et al (2015) *Design and Development of an Intelligent Rover for Mars Exploration*

31. Paul Meacham, Nuno Silva, Richard Lancaster (2016) *The Development of the Locomotion Performance Model (LPM) for the Exomars Rover Vehicle*

32. N. Patel, R. Slade, J. Clemmet (2010) *The ExoMars rover locomotion subsystem* Journal of Terramechanics, Vol. 47, Issue 4

33. Nuno Silva, Richard Lancaster, Jim Clemmet (2013) *Exomars Rover Vehicle Mobility Functional Architecture and Key Design Drivers*

34. M.Woodall, G. Baechler, I. Salehinia, N.A. Pohlman (2020) *Mars Rover Rocker-Bogie Comparative Differential Study* IJIRSET, Vol. 9, Issue 4

35. W. Youngming ,Y. Xiaoliu,T. Wencheng (2009). *Analysis of Obstacle-climbing Capability of Planetary Exploration Rover with Rocker-bogie Structure* Information Technology and Computer Science, International Conference on. 1, 329-332

36. D. Michel, K. McIsaac (2012). *New Rocker-Bogie and Terramechanics-Based Wheel/Soil Interaction Model for Planetary Rovers*

37. T.P. Setterfied , A. Ellery (2013). *Terrain Response Estimation Using an Instrumented Rocker-Bogie Mobility System* Robotics, IEEE Tractions on. 29, 172-188

38. R. Gonzalez, D. Apostolopoulos, K. Iagnemma (2019) *Improving rover mobility through traction control: simulating rovers on the Moon* Autonomous Robots, 43

39. O. Toupet, J. Biesiadecki, A. Rankin, A. Steffy , G. Meirion-Griffith, D. Levine, M. Schadegg, M. Maimone (2018) *Traction Control Design and Integration Onboard the Mars Science Laboratory Curiosity Rover*

40. D. Kim, H. Hong, H. Kim, J.Kim (2012) *Optimal design and kinetic analysis of stair-climbing mobile robot with rocker-bogie mechanism* Mechanism and Machine Theory, 50, 90-108

41. H. Zhou, H.H. Ju (2010) *Research on the Method of Torque Distribution of Six-Wheel Rocker-Bogie Lunar Rover* ICMTMA, 3, 203-206

42. T.A. Neilson, J.A. Donaldson (2014) *Curiosity's Fault Tollerant Wakeup and Shoutdown Design* CSER

43. Automotive Industry Action Group (AIAG) and Verband Automobilindustrie (VDA) (2019) *AIAG & VDA FMEA Handbook, ISBN-13: 9781605343679*

44. Sharma Kapil, Srivastava Shobhit (2018) *Failure Mode and Effect Analysis (FMEA) Implementation: A literature Review*

45. R. Wang, Y. Feng, H. Yang (2019) *Construction project risk evaluation based on FMEA* IOP Conference Series, Earth and Environmental Science

46. E. Kulcsàr, T. Csiszér, J. Anonyl (2020) *Pairwise comparison based Failure Mode and Effects Analysis (FMEA)* MethodsX

47. K.L. Lu, Y.Y. Chen, L.R. Huang (2018) *FMEDA-Based Fault Injection and Data Analysis in Compliance with ISO-26262*

48. M. Catelani, L. Ciani, V. Luongo (2010) *The FMEDA approach to improve the safety assessment according to the IEC61508* Microelectronics Reliability, 50, 1230-1235

49. E.Bagalini, J.Sini, M.S. Reorda, M. Violante, H. Klimesch, P. Sarson (2017) *An automatic approach to perform the verification of hardware designs according to the ISO26262 functional safety standard* 18th IEEE LATS

50. J. Sini, M. D'Auria, M. Violante (2020) IEEE LATS