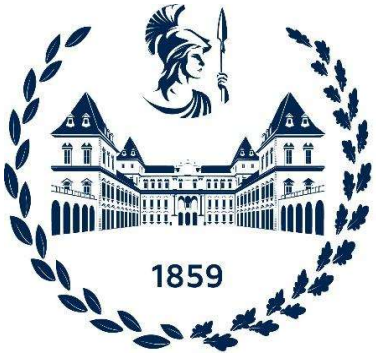

Politecnico di Torino

Master's Degree in Mechatronic

Engineering Control technologies for industry 4.0

Interaction between Prometheus 2.0 rover devices. Hardware/software assembly and communication

Supervisors:

Prof. Ing. Massimo Violante

Prof. Ing. Jacopo Sini

Company reference:

Ing. Joseph Toma

Candidate:

Amir Rustemi

Academic Year 2024/2025

Index

List of Figures

Abstract

The Prometheus 2.0 rover represents Axist's solution for offshore measurements and constitutes a significant step forward in the development of mobile robotic platforms dedicated to inspection and metrological applications. Compared to its predecessor, the system introduces a more advanced hardware/software architecture able to improve motion control, measurement processes, and operator feedback. A key case study is the monitoring of flange ovalization inside monopiles, a critical task in offshore wind energy structures, where traditional measurement methods prove impractical.

The rover's architecture is based on distributed microcontrollers, Ethernet communication, and a dedicated workstation for supervision and data storage. Within this configuration, particular attention was given not only to the design of the subsystems, such as the azimuth and elevation servos coupled with the Ovality Box and the power management unit, but also to the complete study, design, and validation of all hardware connections and cabling, carried out during the development process to guarantee electrical robustness, signal integrity, and operational safety.

A central aspect of the control system is the implementation of a Kalman filter, employed to estimate wheel velocity and compensate for noise in sensor data. This recursive approach, lightweight yet robust, allows the rover to refine its motion feedback, ensuring consistency between commanded movements and measured displacements. Together with structured PPM (Pulse Position Modulation) signal parsing and carefully designed communication packets, the filter strengthens the reliability of the entire feedback loop, which is essential to provide the operator with real-time awareness of the rover's state.

Finally, the thesis outlines possible directions for future development. Among these, attention is given to the centralization of control on industrial-grade platforms and to the adoption of advanced perception systems. Although not yet implemented, these improvements delineate a trajectory for further increasing the rover's autonomy, robustness, and long-term applicability in complex industrial environments.

Chapter 1 – Introduction

1.1 Metrology: Evolution and Role in Industry

Metrology is the science concerned with measurement, its validity, repeatability, and traceability. In scientific and industrial fields, measurement is never an end in itself: it is the foundation for process control, product quality, and operational safety. Over time, the need to standardize measurement criteria led to the development of the International System of Units (SI), which today serves as the global reference for all physical quantities. This system allows us to speak a common language worldwide and underpins any form of technological or commercial exchange. From classical metrology, focused on calibration and standard references, we have moved to dynamic metrology, integrated into production processes and capable of delivering real-time, continuous data.

1.2 From analog measurement to digital data

With the advent of industrial automation, metrology became an integral part of production. The transition from analog instruments to digital sensors revolutionized how data are collected, transmitted, and analyzed. Modern measurement systems do more than detect a value: they interpret it, compare it to reference thresholds, and generate useful signals to adjust the process. In this context, metrology has become a strategic tool to improve efficiency, reduce waste, and ensure repeatability.

Furthermore, with the introduction of ISO standards and quality management systems (e.g., ISO 9001), metrology has become even more crucial: it not only measures, it certifies. Companies must demonstrate control over their measurement processes, ensuring that instruments are calibrated, verified, and traceable.

1.3 Metrology in industry and mechatronic systems

In industry, every decision relies on data, and every reliable data point stems from a well-executed measurement. From dimensional control of machined parts to energy consumption monitoring, metrology is everywhere.

A measurement error can trigger a cascade of inefficiencies: out-of-tolerance parts, additional costs, safety risks. For this reason, metrology is recognized as a core discipline

in advanced manufacturing systems, especially in automated or high-precision environments. Quality standards not only require process control, but also documentation of measurements and associated uncertainties. This means metrology is not only about "what" is measured, but also about "how" and "why".

In a mechatronic system, metrology is the meeting point between the physical and digital worlds. Each sensor is a window to reality: it captures physical quantities, translates them into electrical signals, and makes them available for digital processing. The ability to react in real time, adjust parameters, or optimize system response depends on the accuracy and consistency of measurements. Without solid metrology, control loses its meaning. The integration of smart measuring devices, digital communications, and advanced control logic has made metrology an active component in the design of complex systems. In this context, measurement is no longer an isolated operation, but a distributed and connected function.

Modern metrology is more than a technical discipline, it is a transversal competence that supports quality, reliability, and safety in every productive and technological field. Its applications go beyond measurement itself, extending into certification, diagnostics, and intelligent data management. In a constantly evolving industrial context, dominated by automation, digitalization, and traceability requirements, metrology provides assurance: every measured value, if properly managed, becomes a reliable piece of information on which to build control and innovation

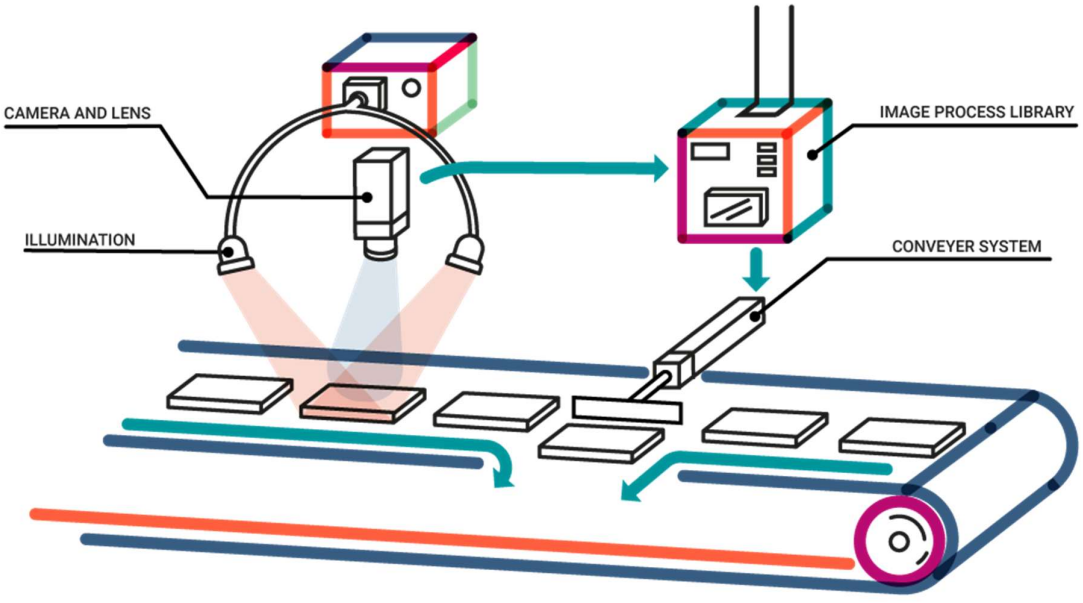

***Figure 1**: Example of line production with metrology systems*

1.4 Axist s.r.l.

Axist is an engineering company specializing in the design of innovative dimensional measurement solutions and the provision of dimensional testing and reverse engineering services using portable instrumentation.

It was founded in 2001 in partnership with the French group F.I.T - ESIC, a leader in topographic photogrammetry. Initially, it served clients in the automotive, mold, and foundry industries.

A few years later, it began working in the aeronautical sector, becoming a supplier to Alenia and Agusta Helicopters (now companies of the Leonardo Group) and to a large portion of the company's top-tier subcontractors, also offering consulting and services for all major projects. Since that time, a team of Axist technicians has been working permanently at the Grottaglie site, also handling measurements of flight parts and analyzing the collected data.

In the same years, Axist also became the Italian distributor of Spatial Analyzer, a software recognized as a benchmark in the aeronautical market.

Since 2009, the company has expanded its presence in the energy, research, oil & gas, and steel markets, opening offices in the United Arab Emirates.

The company has also established itself to conduct significant research projects internally or in collaboration with external partners and institutions: in particular, in association with the Cetma research center in Brindisi, it has developed new calibration systems for machine tools and augmented reality systems.

Subsequently, starting in 2016, the company further expanded, adding topography, photogrammetry, and aerial photogrammetric survey services, thanks to specialized engineers, cutting-edge software, and methodologies.

Shortly thereafter, the R&D department was strengthened to develop robotic measurement systems: the robot became a true measuring machine, replacing traditional control gauges for in-line production.

In 2019, the e-commerce project (www.MetrologyStore.com) was also born, aimed at the online sale of tools necessary for measuring instruments. Finally, with a view to providing an increasingly integrated service, Axist Robotics was founded in 2020. This innovative and cutting-edge business unit combines the world of dimensional testing with that of in-

line production. Using common anthropomorphic robots, the Robotics division provides a turnkey product that offers customers a variety of benefits in a single solution.

The company now has multiple locations throughout Italy and beyond, allowing it to reduce the time and costs of on-site customer service. Furthermore, as already mentioned, it operates in various industrial settings across different sectors, enabling it to offer new technologies to customers.

The company operate in a various set of sectors, like:

- automotive;
- aeronautics;
- space;
- nuclear;
- naval;
- railway;
- oil and gas;

Axist maintains a high performance also thanks to the advanced instruments that provide to its technicians, below are reported some of them:

- Robotic measuring arm (Fig.2), that allows to have a portable measuring system with 3D technology

***Figure 2**: CAM2 Edge*

- Remote controlled aircraft systems (Fig.3)

***Figure 3**: Remote controlled drone*

- Total-stations (laser scanner)

***Figure 4**: Leica Total Station TDRA6000*

These are some examples of the innovative instrumentations Axist uses to guarantee high precision during the measurements.

Axist Robotics is a company branch established to meet automation needs: it pioneered the use of robots for dimensional measurements.

The robots used are derived from conventional ones, to which kinematic, elastic, and thermal compensation are applied to achieve excellent performance in terms of volumetric accuracy. In some specific cases, the robots also include the insertion of precision encoders on the main axes to translate the actual position of the end-effector. The robot integrates optical or contact measurement sensors and generates measurement reports similar to those obtained with conventional CMMs.

Axist Robotics has developed AVRIS, a dedicated environment for managing these families of robots, which represent the modern alternative to control gauges for meeting in-line measurement demands. They guarantee the same robustness as gauges but also have all the flexibility of robots, allowing for reprogramming measurements when switching between jobs.

Axist Robotics also designs and builds machines and systems to meet its customers' specific needs. Some examples of these include:

- Equipment for joining aircraft fuselages, using axis systems controlled and feedback-driven by laser trackers.
- Robots with ultrasonic probes for measuring the thickness of deep-drawn sheet metal.
- Magnetic field mapping machines (CERN customer)
- Dedicated measuring instruments (F4E customer for the ITER project)
- Large tripods for high-altitude use of laser trackers.

Chapter 2 – Prometheus 1.0

2.1 What is Prometheus

Prometheus is the name that Axist Robotics gave to its new project, officially finished and presented to the public in March 2024. This project is a Rover designed for a company that plants wind turbines, in the middle of the sea, where all types of measurements are not easy to take maintaining a certain accuracy. This remote-controlled robot is the answer to the client's questions and needs:

- How can we make measurements offshore?
- How can we do in a safety way?
- How can we guarantee the required accuracy?

In order to introduce the rover, we first need to explain what its purpose is. As said before the customer was a company in the offshore wind turbines industry, so all the measurements were taken in the sea, where the environment conditions were not the best. All the wind turbines are mainly composed by:

- wind turbines (Fig.5);
- monopiles (Fig.6).

With the term monopile is indicated the lower part of the entire structure, in particular that one that is planted in the seabed and emerges from the water where afterwords will be mounted the wind turbine.

***Figure 6**: offshore wind turbine*

***Figure 5**: monopile*

Since the monopile is violently hammered several times in order to be planted into the ground, it's necessary to check that the surface hasn't been damaged by these impacts. So the part that need to be inspected is the upper surface of the monopile, by checking if the hammer hit in the right way and in the right position. It is required an impressive accuracy in these measurements because a little defect on the surface, that can be a basic non planarity could lead to serious structural issues once the turbine is installed, given its size and weight.

On the Fig.7 could be seen how the surface, called flange, that must be inspected looks like.

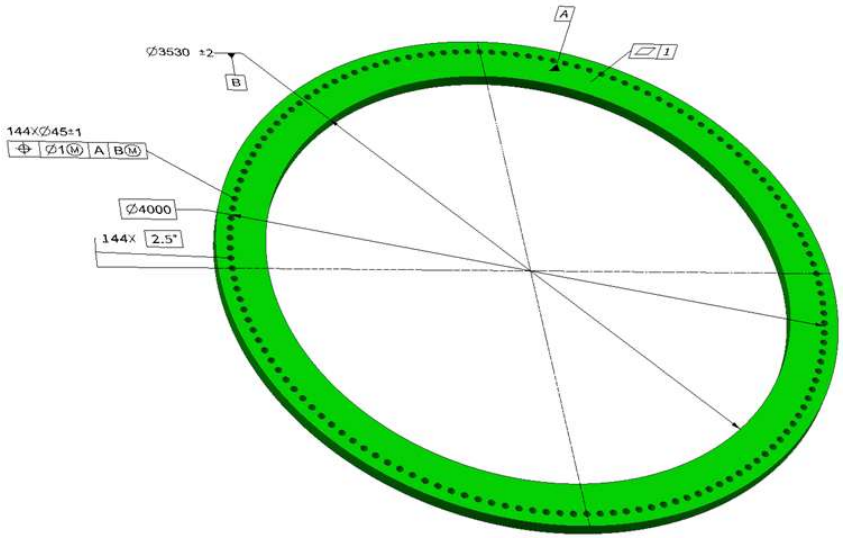

***Figure 7**: Flange*

To answer this questions, Axist Robotics designed Prometheus, that is basically a rover equipped with all metrology instruments needed to make all the required measurements, in automatic way. That means that the company has found a solution that allows to check and inspect the flange in the sea without needing to have someone in danger, on the monopile, but the technician could stay in safe on the ship and remote control the rover taking all the needed data to analyse the flange.

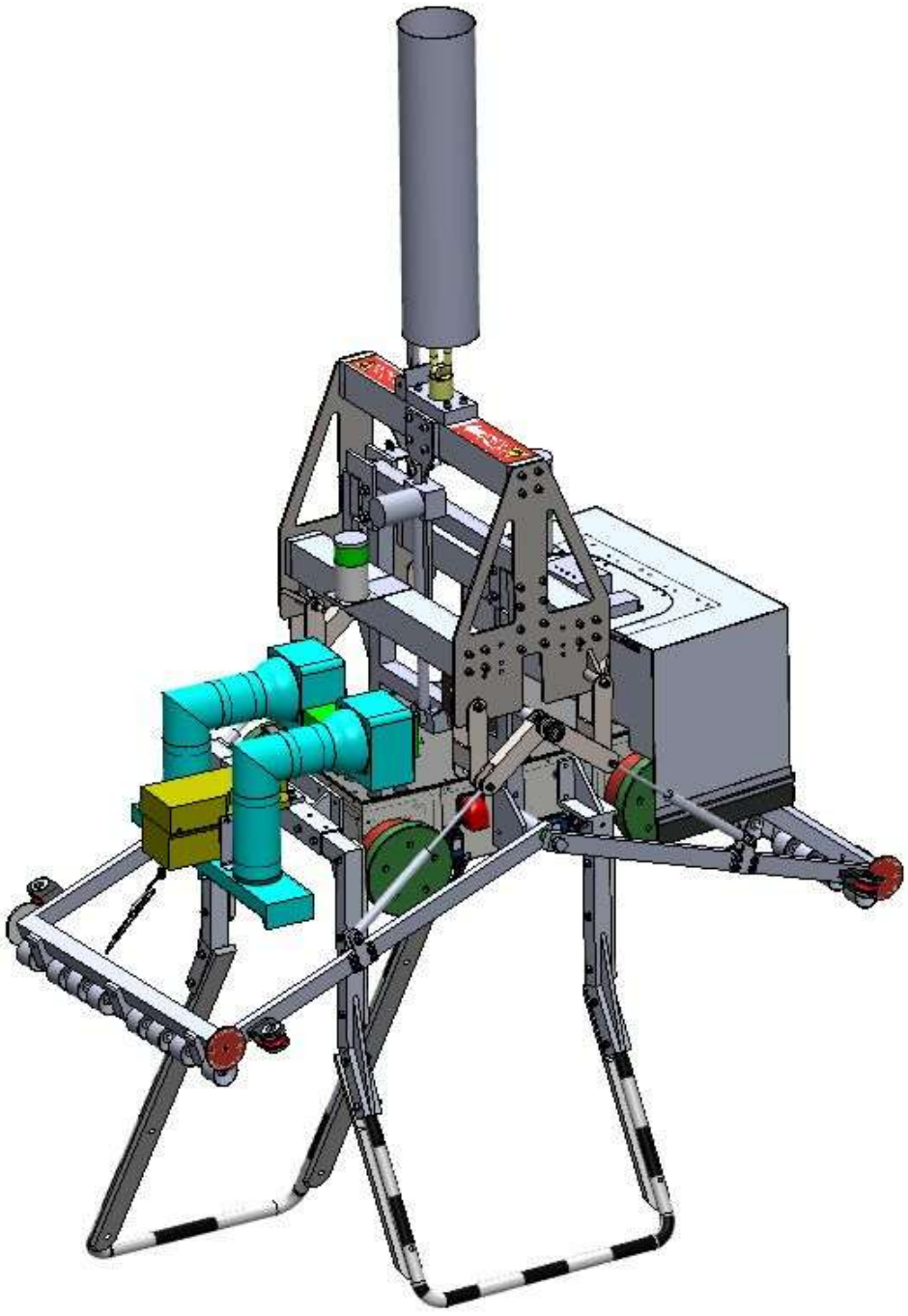

***Figure 8**: Rover cad model*

2.2 Hardware architecture

The architecture of the rover is very complex, but we can try to divide all the components into macro groups as follows:

- power supply;
- metrology;
- movement;
- signal transmission;
- fans and wiper.

2.2.1 Power supply

The source of electricity is provided by three batteries, that for simplicity and above all for the possibility of finding spare parts in every place are common batteries used for trucks. All of them have the same specifications, in particular have a tension equal to 12 V and an electric capacity of 28 Ah. One of these is dedicated just to supply cleaning and wiping components and landing system basically, since these elements are mounted outside the rover, this battery is labelled as the external one. With regard to the other two, they are connected each other in series to provide 24 V and be able to supply all the other parts of Prometheus that are not covered by the external battery.

As said before, inside the rover there are too many elements and obviously is very difficult to choose all of them to be powered by 12 V or 24 V so were used some DC-DC converter that allows to supply all devices in the correct way, in particular were used:

- DC-DC converter 12 V to 9 V;
- DC-DC converter 12 V to 8.4 V;
- DC-DC converter 24 V to 12 V;
- DC-DC converter 24 V to 9 V;
- DC-DC converter 24 V to 5 V;

A lot of elements are not directly powered by the batteries, but they are connected to each other with specific connection for data transmission that also allow power flow, i.e. usb connections.

2.2.2 Metrology components

Prometheus's aim is to make specific and high accurate measurements, so it is provided of devices able to take this type of analysis, all of these components are called metrology devices.

In order to be able to take all the required measurements the rover is equipped with those main metrology components:

- two digital output inclinometers BWS2000;
- three lasers 3D GOCATOR 2640;
- two cameras.

The inclinometers are useful because thanks their use it is possible to have some feedback about the plane in which the rover is riding, first of all they give the angle of inclination in case of slope. Meanwhile, the second application is for stability, in fact using these devices it is possible to define if the machine is physically stable or if it is in presence of something that interferes with the equilibrium state, i.e. too much wind, it is realizable by checking the angle of inclination, if it is not constnat it means that for sure the rover is not completely stopped or is in presence of bad weather conditions.

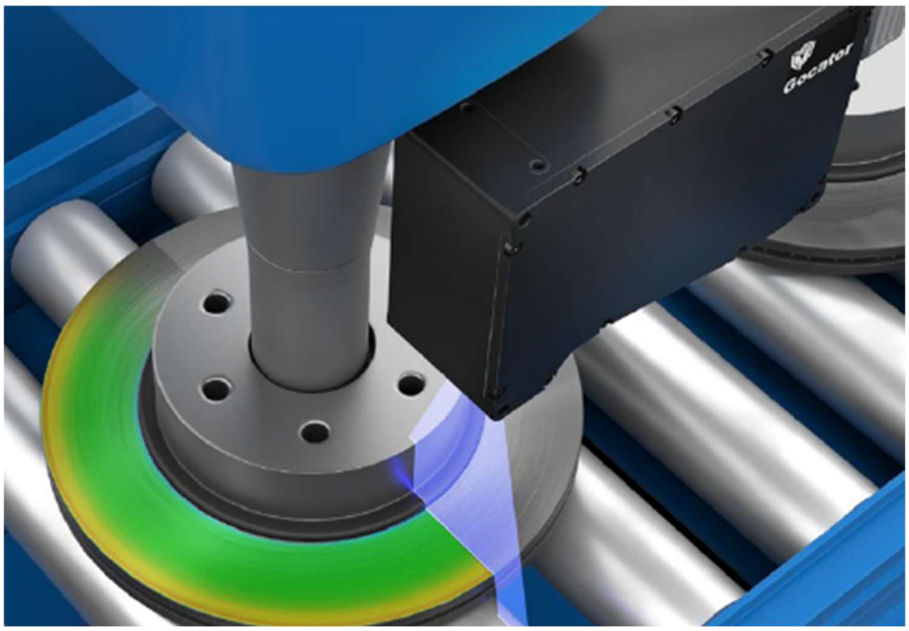

***Figure 9**: Example of Gocator scan laser*

The most important parts of the metrology system are the Gocator (Fig. 9) lasers because they are those that actually make the effective measurement required to certificate the condition of the flange. Series laser profilers use a 4K+ imager to generate high-resolution

profile and surface data for measurement of microscopic features such as defect detection.

The machine is also equipped with two high resolution cameras, but in the first design they were used just in a passive way, i.e. to have the possibility to check what happened in case of problems.

2.2.3 Movement

The rover was designed to make measurements on flanges that have circular geometry, so basically the machine has to move in a fixed trajectory. The smartest way to solve this request is to design Prometheus as a train, without the capability of turning. The flange can be seen as rails, so the wheels are as rigid as those on trains, as shown in figure 10.

***Figure 10**: Zoom on wheels*

The platform is equipped with four wheels, organized in two pairs: one at the front and one at the rear. Each pair of wheels is connected and moves in sync, meaning that the front wheels rotate together and so do the rear ones. This setup helps ensure smooth and balanced motion. Due to this configuration, to ensure the movement, are needed two different electrical motors that supply both pairs of wheels, making sure that the direction of all wheels must be equal at each moment.

2.2.4 Fans and wiper

In the rover, given the environment where it must work, is endowed of a windshield wiper that permit to clean from dust, water or any other possible dirt that can fake the measurement results present on the flange. It has also a fans system that ensure air recycling inside the cover to avoid overheated and water penetration too.

2.3 Hardware communication

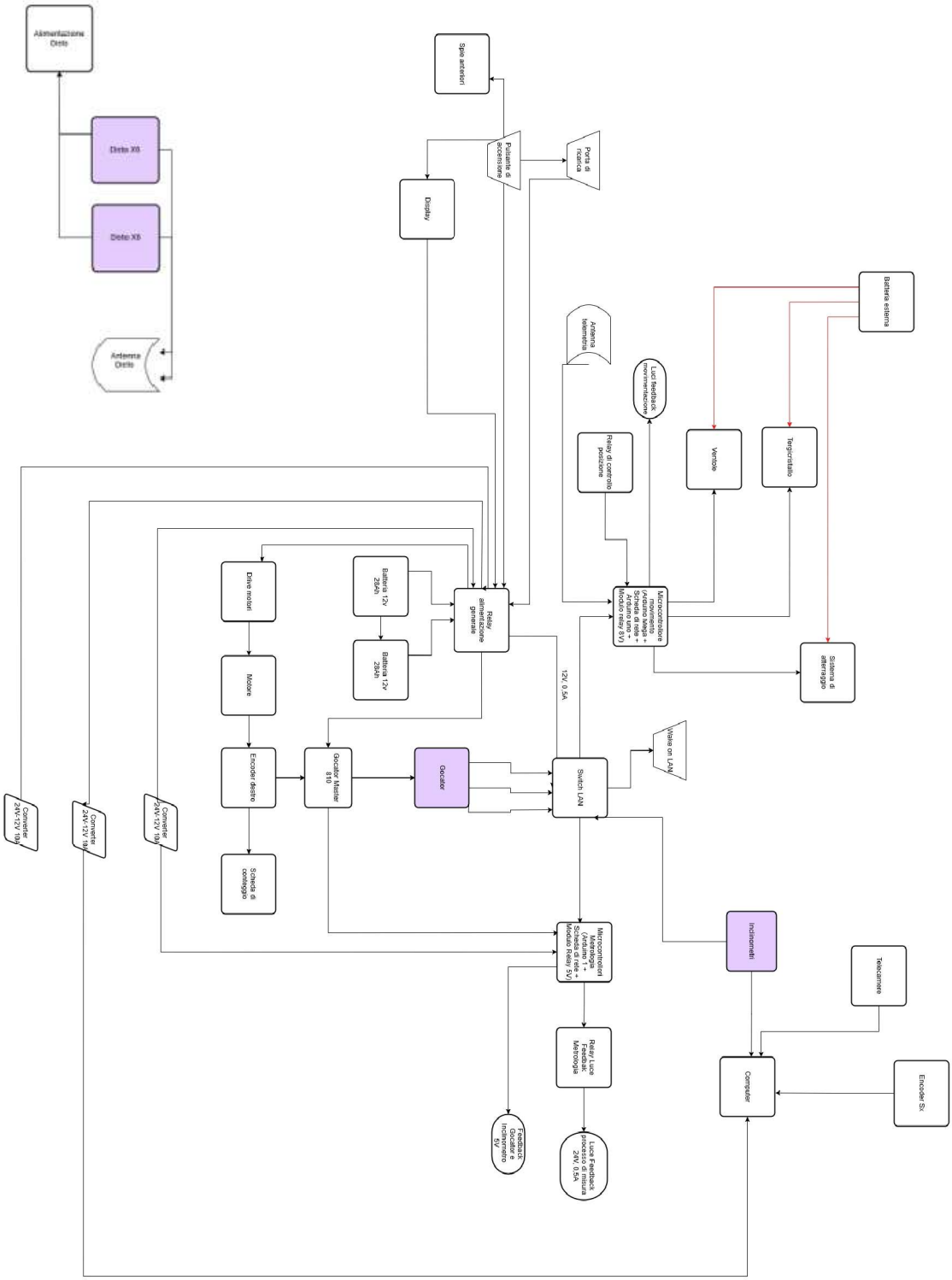

***Figure 11**:Functional scheme Rover 1.0*

The current control system consists of two main components:

- ARDUINO MOVEMENT:

A microcontroller that activates a motor to drive a wheel. It does not receive encoder feedback. Instead, it drives the motor by setting a digital output high for a fixed duration (MOVE TIME), then stops by setting it low for a fixed duration (WAIT TIME). This cycle of motion and pause repeats continuously unless externally interrupted.

- PC ROVER:

A PC-based process that monitors the advancement of the wheel using an encoder. It samples encoder data with a delay of up to 100 ms and uses it to determine when ARDUINO MOVEMENT is moving (MOVE) or stationary (WAIT).

Before entering into advanced control strategies or integration details, it is essential to understand the fundamental motion routine that governs the system during its initial operation. This basic sequence illustrates how communication and coordination occur between the controller, the motor driver, and the feedback system during a typical motion cycle.

The process begins when an external control unit, radio control, transmits a signal to initiate the cycle. This trigger indicates to the motion module that it must activate a predefined sequence. From this point, the control logic transitions into the active phase. The motor is powered on for a precise and calibrated time frame, defined by software with the variable MOVE_TIME. During this time interval, current flows through the motors, generating the torque required to move the wheels. Once the work time reaches the given threshold, the controller stops the current flow, forcing the motor to shut down temporarily. The motors will be off for a determined time entered inside the variable WAIT_TIME. This pattern formed by different phases that alternate, motor on then off, continues in a loop. The simplicity of this mechanism allows predictable and repeatable movements, particularly useful during test cycles or routine calibration phases. Parallel to this operation, a supervisory unit (referred to as PC ROVER) passively monitors the evolution of the movement. It does so by reading position data coming from the encoder, as is possible to see in FIG.13 installed on the drive system. A rotary encoder is an electromechanical device that converts the angular position or motion of a rotating shaft into digital or analogue signals. When the motor begins to rotate, the rotary encoder's shaft, which is mechanically coupled to it, starts rotating. This rotational motion is transferred to an internal disc, which is mounted directly to the encoder shaft. The disc is either optically or magnetically patterned, depending on the type of encoder. A light

source, typically an LED, shines through the disc. As the disc rotates, the interaction between the disc pattern and the sensor generates varying signals. The structure of the encoder is reported below in the Figure 14.

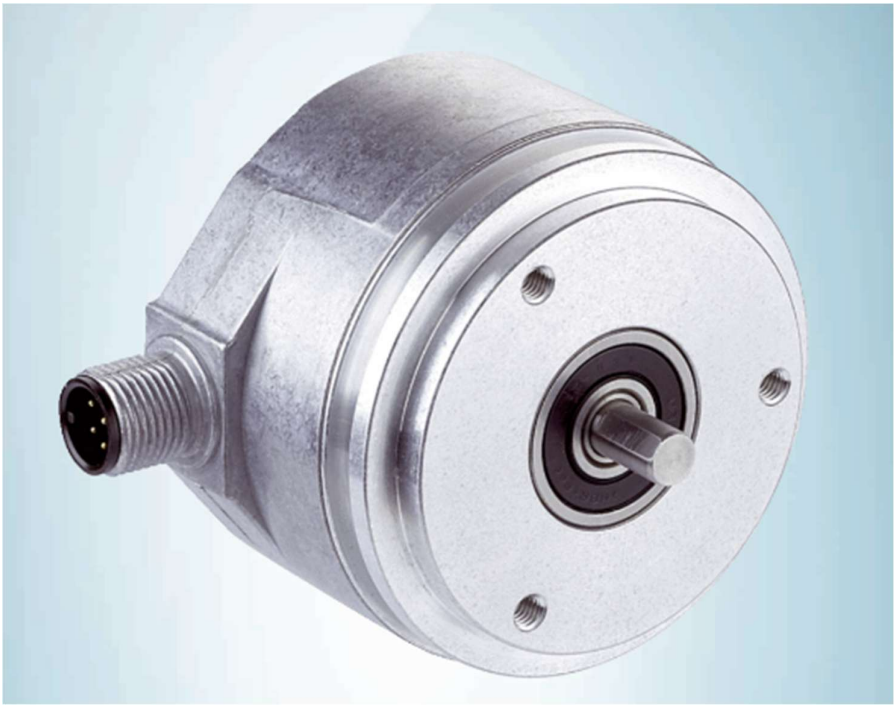

***Figure 12**: Encoder DFS60-A SICK*

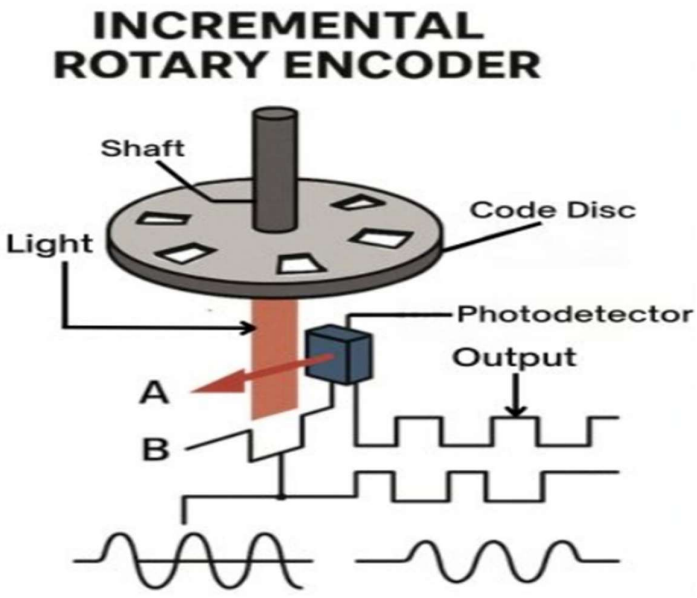

***Figure 13**: Encoder functionality*

Because the encoder produces a pulse train proportional to the angular displacement of the wheel, it can be used to reconstruct the kinematics of the system with high precision. One key aspect of this monitoring phase is that processing tasks are only performed during the motor's inactive window (WAIT_TIME). This ensures that no interference affects the motion behavior and that CPU resources are optimally allocated. Timing analysis, error detection, and basic telemetry evaluation are all conducted in these short pauses. The interferences on the stabilizing time are kept under observation thanks the use of inclinometers.

***Figure 14**: Inclinometer*

Instead of relying on internal timers or synchronous clocks, the PC firmware infers the transition between motion and stillness by analyzing variations in encoder output. When pulses stop, it assumes the motor is in its waiting phase. Conversely, the resumption of pulses indicates active movement. This strategy simplifies synchronization without needing constant handshakes between the units.

The described approach, though minimal, reflects a forced engineering choice in order to respect budget and delivery time of the project. This rhythm allows simple hardware to perform complex tasks without concurrency.

Moreover, by using encoder feedback as the basis for inference, rather than synchronization, the architecture remains flexible and decoupled. This has advantages in terms of fault tolerance: even if communication from the encoder is delayed or noisy, the PC firmware can still reconstruct motion trends with sufficient confidence.

This initial configuration, although basic, establishes a foundation upon which more sophisticated layers of control can be built. It also represents a valuable test case for verifying the correct behavior of each subsystem before enabling more autonomous features.

2.3.1 Current limitation

While the initial cycle structure offers simplicity and quite robustness, it also presents several limitations that should be taken into account when evaluating system scalability and performance. One notable constraint is the absence of a feedback loop in the motion controller. The module responsible for motor activation (ARDUINO MOVEMENT) operates independently of any real-time positional data. This means that once the motor is activated, there is no mechanism in place to stop or adjust its behavior based on the actual displacement of the wheel. It functions blindly, relying solely on timing parameters rather than physical outcomes. Another important aspect is the use of fixed timing parameters. Both MOVE_TIME and WAIT_TIME are static values. While this simplifies implementation, it also introduces variability in performance. Factors like mechanical friction, varying surface conditions, issues on motors or changes in load can all affect the actual distance covered during a given MOVE_TIME. Likewise, the WAIT_TIME is intentionally set longer than strictly necessary, just to ensure that the processing tasks on PC ROVER can complete reliably. This results in suboptimal efficiency and reduced throughput. Additionally, the system lacks any form of adaptive control. The PC ROVER, despite having access to real-time position information via the encoder, is not capable of actively influencing the behavior of ARDUINO MOVEMENT. It cannot interrupt the movement, shorten the cycle, or make adjustments based on anomalies or specific operational requirements. This passive monitoring limits the overall intelligence of the system. These limitations do not prevent basic functionality, but they do impose restrictions on precision, responsiveness, and adaptability aspects that may become more critical in future phases of development.

Chapter 3 - Prometheus 2.0

3.1 Necessary changes

Due to the limitations of the first configuration the following main changes were needed:

- As intended in the second chapter the first improvement to do was to convert the rover from a time-based architecture to an event-based architecture. In order to reach this goal, it was necessary to implement a bidirectional communication between ARDUINO MOVEMENT and PC ROVER since in the first project the controller was able to send information to the computer but it couldn't receive.
- Another important change needed was the implementation of a system for measuring the internal diameter of the flange, so as to be able to detect possible ovalizations. This system is provided by its own controller (Arduino uno) and is called *ovality box.*
- Due to the new changes it was also necessary to modify the supply system since the previous scheme it was no longer suitable for the new configuration.

It was also needed to redefine all the hardware communications due to the improvements but also because Prometheus 2.0 was projected and mounted starting from zero in order to have two different and functional rovers.

3.2 Arduino Movement

As previously mentioned, the communication between the movement's controller and the PC ROVER had to be revised. In this subchapter will be described the new configuration.

The microcontroller chosen was Arduino Mega 2560, shown in FIG 16, this decision was not made arbitrarily, but after evaluating various aspects such as compatibility, ease of development, and flexibility in firmware management. One of the main advantages of the Arduino Mega is its versatility. It supports a wide range of communication protocols like SPI, I^2C, UART, PWM, which makes it suitable for handling multiple peripherals without additional hardware. This flexibility becomes particularly useful in systems that evolve over time, where adding or replacing components should not require major redesigns.

Another key factor is the abundance of input/output pins. With 54 digital I/O and 16 analog inputs, the board offers enough channels to connect sensors, encoders, relays, and

debugging lines all at once. This reduces the need for multiplexers or additional I/O expanders, simplifying the wiring and minimizing delays or signal interference.

To establish a reliable connection with the remote processing unit, the Arduino Mega has been coupled with an Ethernet Shield. The rationale behind this addition is primarily based on communication stability and scalability. Wired Ethernet, unlike wireless protocols, offers predictable transmission times and significantly reduces the risk of interference – a key requirement when monitoring motion cycles and issuing commands with strict timing constraints.

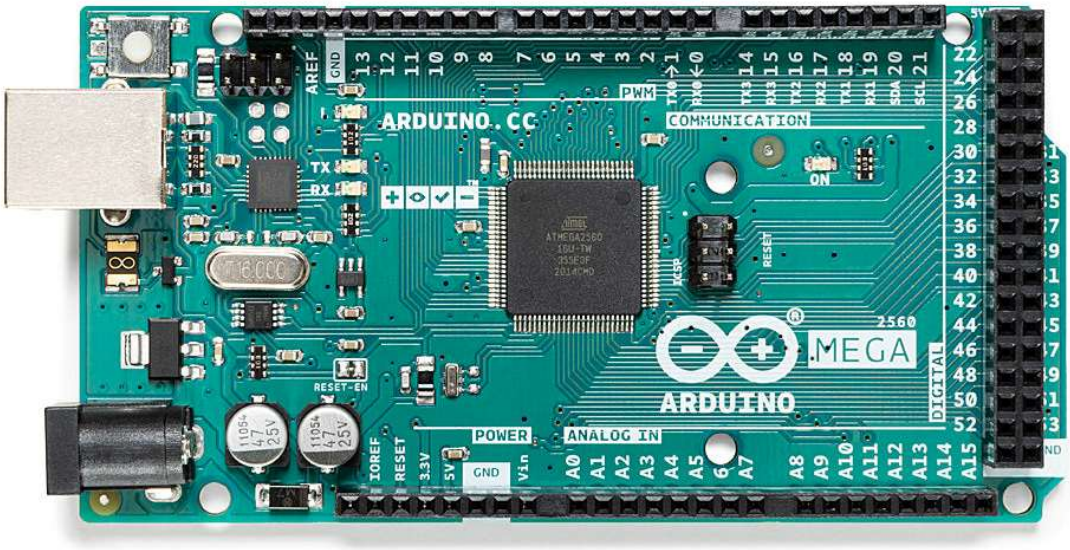

***Figure 15**: Arduino Mega Board*

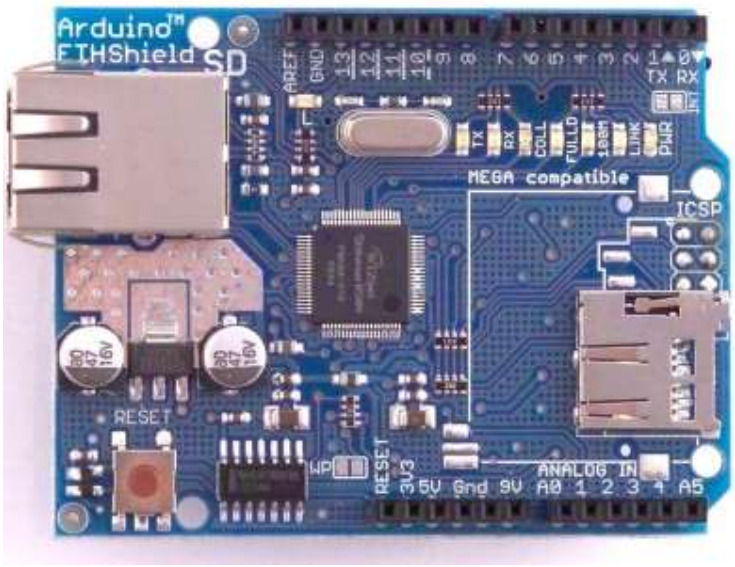

***Figure 16**: Ethernet shield for Arduino Mega Board*

This combination allowed to have a microcontroller capable to connect to Internet and construct a solid communication between it and other devices of the environment, first of all with the computer of Prometheus 2.0. The Ethernet shield is basically a cover that adds an ethernet port and extend all the pins that are present in Arduino boards, thus not limiting the use of the latter. This cover is connected with wire to a switch LAN in which is also connected the PC ROVER and a Wi-Fi antenna that gives the internet connection. The hardware connection of the complete system will be described later.

3.2.1 Arduino Movement Logic

The original architecture followed a fixed-cycle logic:

- The microcontroller activated the motor for a constant duration (MOVE_TIME), then paused for WAIT_TIME.
- The PC observed motion via encoder feedback but could not intervene or alter motion timing.

This arrangement proved effective for basic cycles, but suffered from several key limitations:

- No feedback loop: movements were blind and fixed in time, independent of actual wheel displacement.
- Inflexible WAIT/MOVE durations: static timing did not account for real-world variability such as friction, terrain, or load.

- No error correction or command acknowledgment: the system could not respond dynamically.

To overcome these constraints, the system was redesigned to support bi-directional UDP communication, incorporating predictive timing adjustment and active synchronization.

Scope of the New Protocol

The new design aims to:

- Enable PC ROVER to actively control each motion cycle.
- Introduce a Kalman filter to estimate velocity and correct MOVE duration adaptively.
- Maintain a robust fail-safe behavior when the link is lost.

The communication channel is based on UDP datagrams, which may occasionally be delayed or dropped, but offer low-latency and minimal overhead.

Key features:

- Motion commands sent from PC to microcontroller include:
    - MOVE_NOW trigger
    - MOVE duration override (OVR)
    - Initialization and configuration parameters
- The microcontroller sends back ACKs with:
    - Command echo
    - Current state (MOVE / WAIT / IDLE)
    - Cycle synchronization flags

System Behavior in Brief

- During normal operation:
    - PC estimates current wheel speed after each MOVE.
    - It computes the required time to reach the next step (e.g., 200 mm) and sends it as an OVR.
    - Arduino executes the MOVE using the received OVR or defaults to preset MOVE_TIME.
    - The cycle repeats with each step corrected based on encoder feedback.
- In case of communication failure:

- Arduino reverts to fallback behavior (MOVE_TIME + WAIT_TIME).
- This guarantees basic functionality even without updates.

3.2.3 Kalman Filter

To improve the control of the rover, more precisely to correct at each iteration the duration of the moving time, in order to take in account any possible change on the weight, on the slope of the flange, the battery charge level or any other problem that could raise or decrease the distance covered in the initial fixed time of move, we implemented a Kalman One Step Filter. In control theory, Kalman filtering also known as linear quadratic estimation is an algorithm that uses a series of measurements observed over time, including statistical noise and other inaccuracies, to produce estimates of unknown variables that tend to be more accurate than those based on a single measurement, by estimating a joint probability distribution over the variables for each time-step. The filtering technique adopted in the control protocol corresponds to a Dynamic Kalman one-step predictor and filter, specialized to a one-dimensional case. Unlike general multi-state formulations used in complex systems, this implementation estimates only a single hidden state, in our case the average velocity of the wheel.

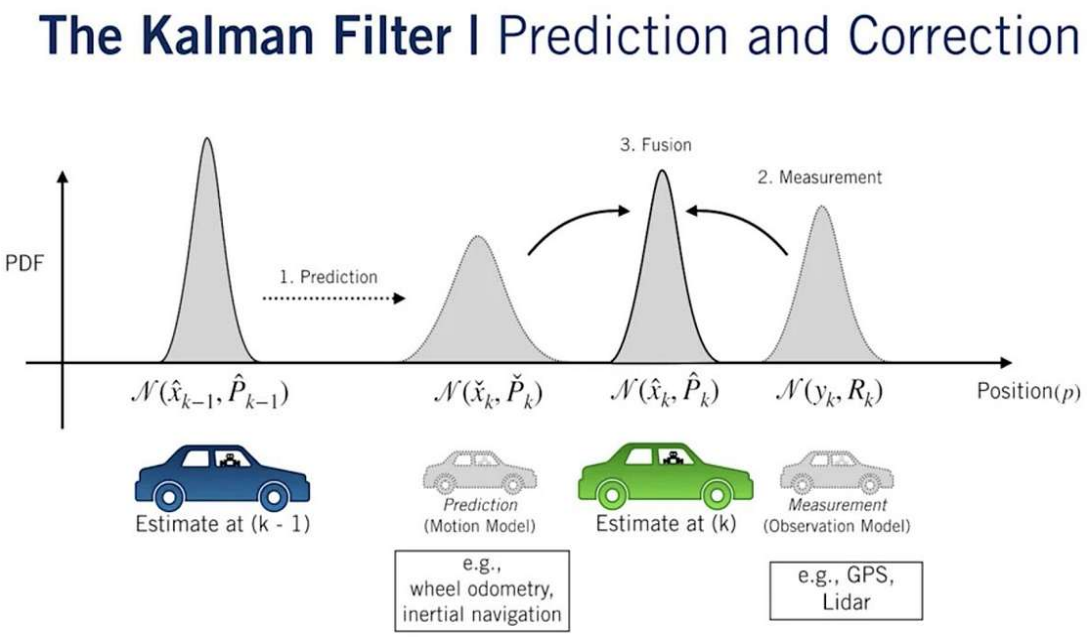

***Figure 17**:Dynamic Kalman filter/predictor*

The mathematical explanation is reported below.

Given the state equation:

$$x_{k+1} = A * x_k + B * u_k + v_1 \qquad \text{where} \qquad v_1 \sim N(0, Q)$$

Given the measurement equation:

$$y_k = C * x_k + v_2 \qquad \text{where} \qquad v_2 \sim N(0, R)$$

Explanations of the variables:

x_k → is the state vector at time step k

u_k → is the control input

y_k → is the measured output

A, B, C → are the system matrices

Q, R → process and measurement noise covariance matrices

We have also the recursive equations as described below.

Prediction step:

$$\hat{x}_{k|k-1} = A * \hat{x}_{k-1|k-1} + B * u_{k-1} \qquad \text{Predicted state estimate}$$

$$P_{k|k-1} = A * P_{k-1|k-1} * A' + Q \qquad \text{Predicted error covariance}$$

Correction step:

$$\tilde{y}_k = y_k - C * \hat{x}_{k|k-1} \qquad \text{Innovation residual}$$

$$S_k = C * P_{k|k-1} * C' + R \qquad \text{Innovation covariance}$$

$$K_k = P_{k|k-1} * C' * S_k^{-1} \qquad \text{Optimal Kalman gain}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k r_k \qquad \text{Corrected state estimate}$$

$$P_{k|k} = (I - K_k * C) * P_{k|k-1} \qquad \text{Corrected estimate covariance}$$

These are all the formula describing the Dynamic Kalman one step filter and predictor, but in our case, we had only one state to correct, it mean that the mathematical formulation is highly simplified as follows.

State and measurement equations:

$$v_{k+1} = v_k + w_k \qquad \text{where} \qquad w_k \sim N(0, Q)$$

$$z_k = \frac{d_k}{t_k} + v_k + n_k \qquad \text{where} \qquad n_k \sim N(0, R)$$

$$z_k := \frac{d_k}{t_k}$$

Explanations of the variables:

d_k → Represent the distance measured by encoder in *mm* for a move k

t_k → Represent the move duration used in *ms*

z_k → Represent the observed velocity sample in $\frac{mm}{ms}$

v_k → Represent the estimated velocity at step k in $\frac{mm}{ms}$

The process and measurement noise covariance matrices Q, R were set as follows.

- Q = $(0.0003\ \frac{mm}{ms})^2$→ as explained before it is the process noise covariance matrix, but in our case is a scalar number, due to the number of hidden states. This parameter represents the uncertainty of the dynamic model v_{k+1}. A small value like 0.0003 was chosen, because consider the velocity of the rover almost constant, admitting only small variation from a step to another one due to friction, small irregularities in the ground, load oscillations or low battery level. In other words is possible to say that in this type of configuration the filter 'trusts' the previous estimate more.
- R = $(\frac{encoder\ resolution}{t_k})^2 + 0.0001$→ It is the measure noise covariance matrix, like Q also R is a scalar number in this case. In other words, it is the uncertainty related to the speed sample calculated by the encoder. The term $\frac{encoder\ resolution}{t_k}$, represents the minimum error with which you can estimate the mean displacement in an interval. If t_k has a long value the relative error decreases, contrariwise if the error has a high value, it means that t_k is small. It was added the term 0.0001 because it avoids that in presence of high values of t_k the filter does not consider the measurement as perfect.

To briefly resume the consideration above, it is possible to say that:

- For small values of Q → the velocity is considered almost constant
- The value of R depends on the encoder resolution and on the duration of the move phase → the longer the duration, the more reliable the measurement is considered

This is a sort of compromise because:

- If Q was too small → the filter would become unstable and overreact to every measurement
- If R was too small → the filter would place too much trust in the encoder, risking oscillations due to noise.

Below are reported the recursive updates of the case:

Prediction→ $v_{pred} = v_{est}$

$P_{pred} = P_{est} + Q$

Kalman gain → $K = \frac{P_{pred}}{P_{pred}+R}$

Correction→ $v_{est} = v_{pred} + K * (z - v_{pred})$

$P_{est} = (1 - K) * P_{pred}$

This scalar version retains the general filter logic, but with reduced complexity, suitable for real-time processing on embedded systems.

In order to avoid any other possible error or fail in the Kalman filter a threshold was chosen and used following this logic.

if $| z - v_{pred} |$ > threshold → ignore the update.

Example with 2 steps of Kalman filter

Below is reported an example that explains the functionality of the Kalman filter implemented in our system.

Since the external controller activates the cycle all the initial parameters are set:

- Target step → 200 mm
- Initial override time → 1100 ms (empirical estimate)
- Move time → 1100 ms

Here the PC ROVER sends all the udp packets to Arduino Movement and the microcontroller executes the commands, after completing them it returns results to the computer.

- Encoder feedback → 190 mm
- Observed velocity → $\frac{encoder\ feedback}{move\ time} = \frac{190}{1100} \cong 0.173\ {}^{mm}/_{ms}$

In this moment the Kalman filter has all the requires data to correct the error.

To reach 400 mm → $\frac{(400-190)}{0.173} \cong 1214\ ms$

After 1.214 seconds, the new move time imposed by PC ROVER based on the Kalman computations and sent to the microcontroller, it returns the following results.

- Encoder feedback → 403 mm, that means that in this step have been travelled 403-190 = 213 ms
- Observed velocity → $\frac{encoder\ feedback}{move\ time} = \frac{213}{1100} \cong 0.175\ {}^{mm}/_{ms}$

So, the Kalman computes again the correct move time and the loop continues till the end of the mission.

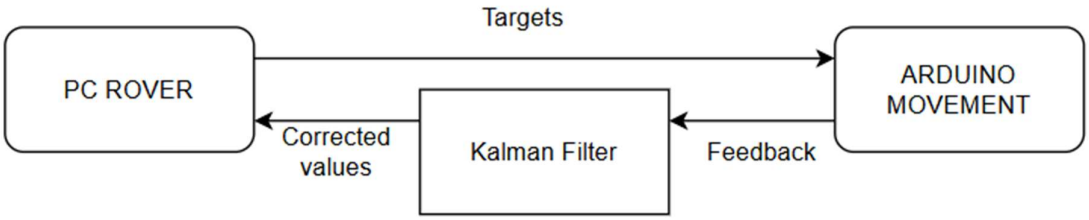

***Figure 18**: Control logic with Kalman Filter*

3.2.4 Electrical motors and drive

The motion of Prometheus 2.0 is guaranteed thanks to two electrical motors, each was dedicated to a single pair of wheels, one for the front axle and one for the rear axle. The picture below shows the features of each motor.

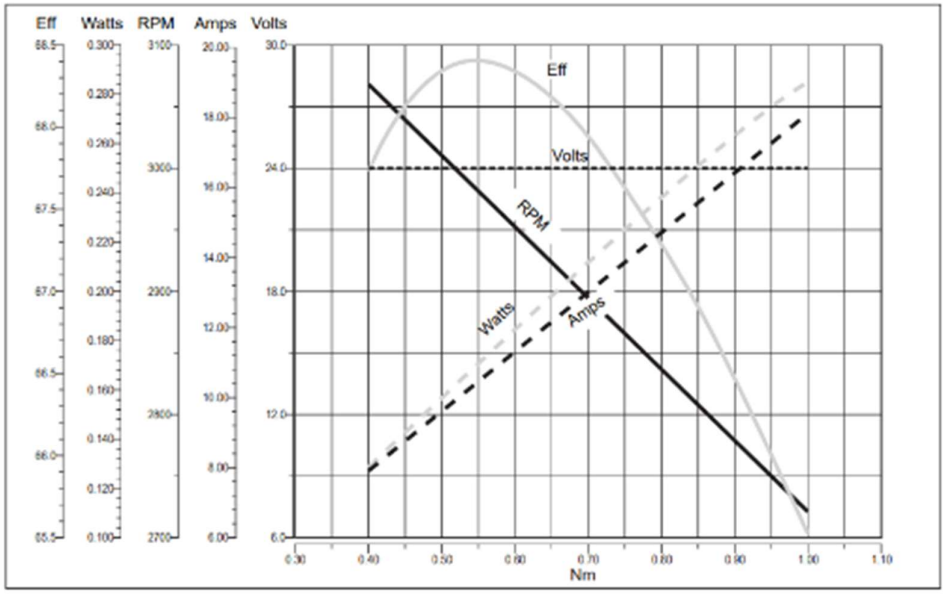

***Figure 19**: Electrical motor performance*

Power	Power supply	Current flow	Velocity	Max Torque
250 W	24 V	15 A	3000 rpm	0.8 Nm

Is easy to imagine that a velocity of 3000 rpm is too much higher than the one needed for a rover developed for metrology usage, so it was necessary to implement a reductor capable of reducing the number of revolutions.

The reductor has a reduction factor i=10 with a maximum value of exiting torque Mn=21 Nm. Considering a dynamic efficiency μ_d equal to 84%, the following calculation allows us to understand how the behavior of the combination of the two elements.

Max speed → $V_{out} = \frac{V_{in}}{i} = \frac{3000}{10} = 300\ rpm$

Max angular speed → $\omega_{out} = 2 * \pi * \frac{V_{out}}{60} \cong 31.416\ {}^{rad}/_{s}$

Max exiting torque → $T_{out} = T_{mot} * i * \mu_d = 0.8 * 10 * 0.84 = 6.72\ Nm$

Max exiting power → $P_{out} = P_{mot} * \mu_d = 250 * 0.84 = 210\ W$

Since the maximum exiting torque of the system is 6.72 Nm that is much smaller than the maximum torque admitted by the reducer 21 Nm, the motor and the reducer were compatible each other.

It was not possible to connect directly the electrical motors to Arduino, because the current drawn and the voltage required are far greater than the driving capabilities of the

***Figure 21**: Motors coupled with reducers*

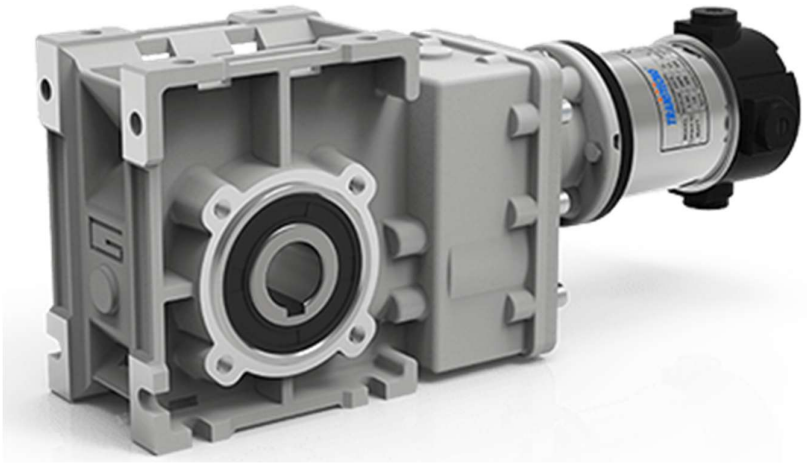

***Figure 20**: Reducer motor*

microcontroller's digital outputs. It was necessary to interpose a dedicated drive capable of performing the following functions:

- Power adjustment: the drive receives a low-power control signal and translates it into a modulated power supply suitable for the motor.
- Controlling the direction of travel: allows to reverse the direction of rotation, forward and reverse, without having to mechanically intervene on the connections.
- Velocity regulation: thanks to PWM modulation, the drive regulates the average voltage applied to the motor, and therefore its speed.
- Protection: It integrates current limiters, overtemperature detection and alarm management, ensuring protection for both the motor and the electronic system.

***Figure 22**: Drive motor PLN-40, by trastecno*

This particular drive could be supplied with a voltage inside the range +10 V dc to +30 Vdc, with a flow current of 44 A. Basically it has the power input, voltage and ground, 2 different outputs to be connected to electrical motors, both pairs with positive and negative voltage. The most interesting and tricky part was the link between PLN-40 and Arduino movement. The first problem was that the drive could not be directly driven by Arduino through the PWM signal but only accepted an input voltage between 0 and 10 V dc. The solution of this issue was to interpose between the two components a third one, or rather digital voltage signal converter capable to translate a PWM signal into voltage 0-10 V dc as required by the drive.

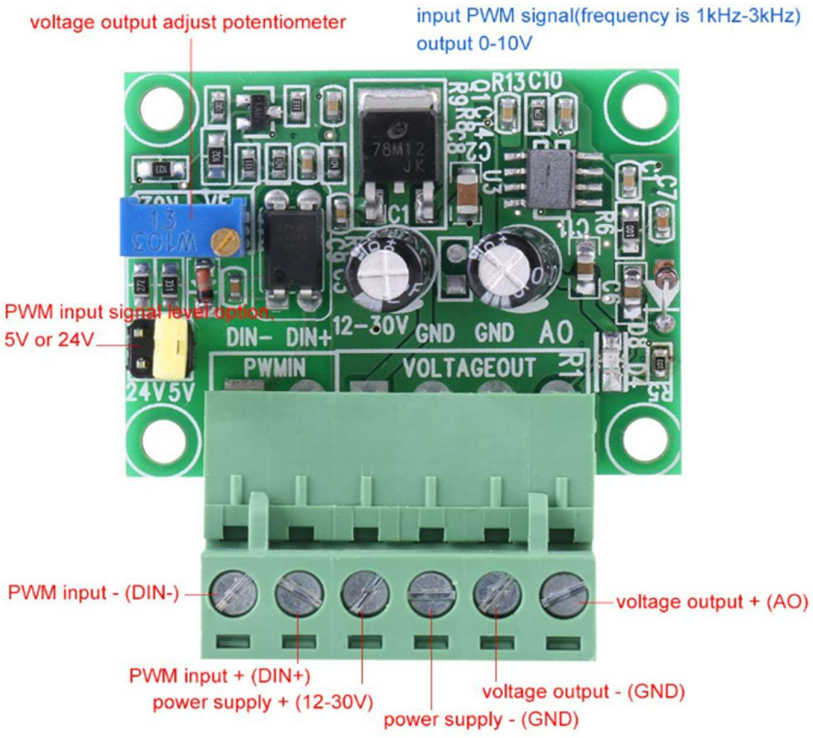

***Figure 23**: PWM converter module*

PWM, or pulse width modulation, refers to signals that vary their logic level over time, going from logic 0 to logic 1 for the duration of a period (duty cycle). The ratio between the time the logic level is at 1 and 0 is called the duty cycle. A 50% duty cycle means that the time the voltage is at 1 is half the time the signal is at zero. The signal can be modulated so that the ratio can vary from 0% to 100%. At a 0% ratio, the output will be at zero volts, while at a 100% ratio, the output will be at logic 1 and the next cycle begins when the previous cycle ends.

As shown in the FIG 23 the module needs, starting to the left, in the first input the ground associated to the PWM signal that has to be connected to the second entry, in the third and into the fourth respectively has to be connected to the alimentation of the module (+12/30 V dc). Finally, at the end are present the output connection, in which is possible to find the translated signal in voltage. So it is possible to control the velocity of the motors using a PWM signal that is proportionally converted in voltage. After ensuring the compatibility of the components, it is necessary to create the connections for the direction of motion and for reading and resetting the alarms. The drive presents 2 different connectors, called CN2 and CN3. Here is shown the second connector with the specifications given by the producer.

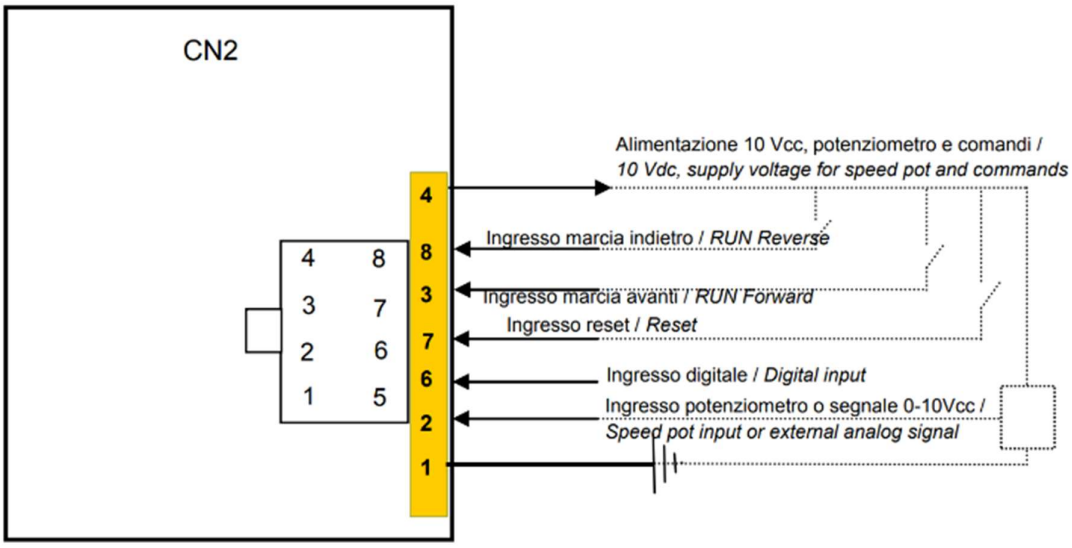

***Figure 24**: Connector 2 of PLN-40*

As shown in Figure 24, the converted PWM signal coming from the module previously described, enters the drive through the pin 2.2 with the ground in the pin 2.1.

The drive, thanks to the pin 2.4 is able to guarantee a power supply of 10 V dc, so it is possible to exploit this output to power the direction of the motion but also for the reset command. Since only one source of 10 V is available and its need to supply three different pins, 2.3, 2.8 and 2.7. It is mandatory to insert a component that allows voltage to reach all pins, for this reason we use a relay module with 4 channels, supplied with 5 V dc directly by the output power present on Arduino mega.

Here is how the pins were set up:

- Pin2.2 → Voltage coming from PWM module
- Pin2.1 → Ground coming from PWM module
- Pin2.3 → Run forward
- Pin2.8 → Run backward
- Pin2.7 → Reset alarms
- Pin2.4 → Voltage source directly from CN2

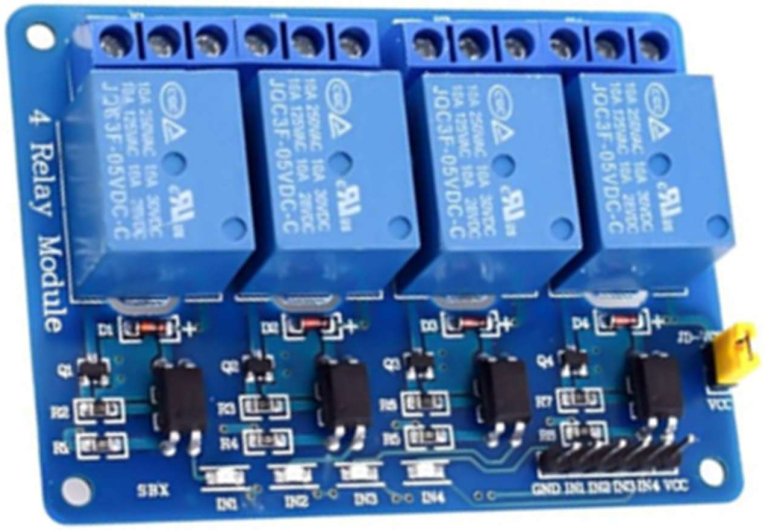

***Figure 25**: Relay module with 4 channels*

The relay module in Fifure 25 has this configuration:

- Vcc → Voltage supply coming from Arduino
- Gnd → Ground of Arduino
- IN 1 → Digital input coming from Arduino, which commands the first channel
- IN 2 → Digital input coming from Arduino, which commands the second channel
- IN 3 → Digital input coming from Arduino, which commands the third channel
- IN 4 → Digital input coming from Arduino, which commands the fourth channel

Then each channel has:

- COM → Called 'common', is the central terminal of the relay that acts as a movable switch. It is the moving pole of the electrical switch controlled by the relay. It is the terminal to which the current or voltage you wish to control is sent. Its function is to physically move between two fixed contacts to complete or interrupt a circuit.
- NO → Called 'normally open', is a terminal that, when inactive, is electrically isolated from the common terminal (COM). Between COM and NO there is an open circuit, which has a theoretically infinite resistance and does not allow the flow of current. When the relay coil is energized, the generated magnetic field activates an electromagnet. This mechanically forces the moving contact (COM) to move and physically connect the circuit to the NO terminal. This action closes the circuit between COM and NO, allowing current to flow.
- NC → Called 'normally close', literally works the opposite of NO.

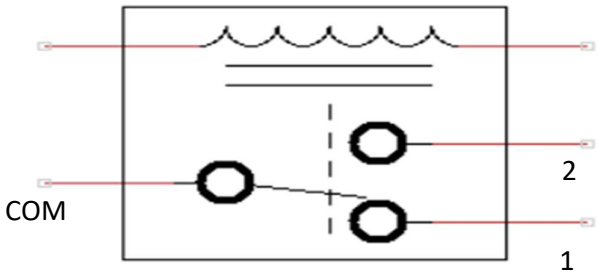

***Figure 26**: Relay functionallity*

Based on the FIG 26, if at the end of the output 1 is linked something and COM does not have voltage, the relay is NC otherwise if COM does not have voltage and the component is linked to the output 2 the relay is NO.

We used all the channels of the relay module as normally opened, in particular the voltage of pin 2.4 is connected to the COM of all channels used in this relay. As already mentioned, this component was introduced to allow us to run forward, backward and for resetting alarms so we use just three channels. The first one is connected to the pin 2.3, dedicated to the forward motion, the second one is connected to the pin 2.8 for the backward and the last one is linked to the pin 2.7 for resetting alarms. Before showing the complete scheme of this system, we have to introduce CN3 and its functionality.

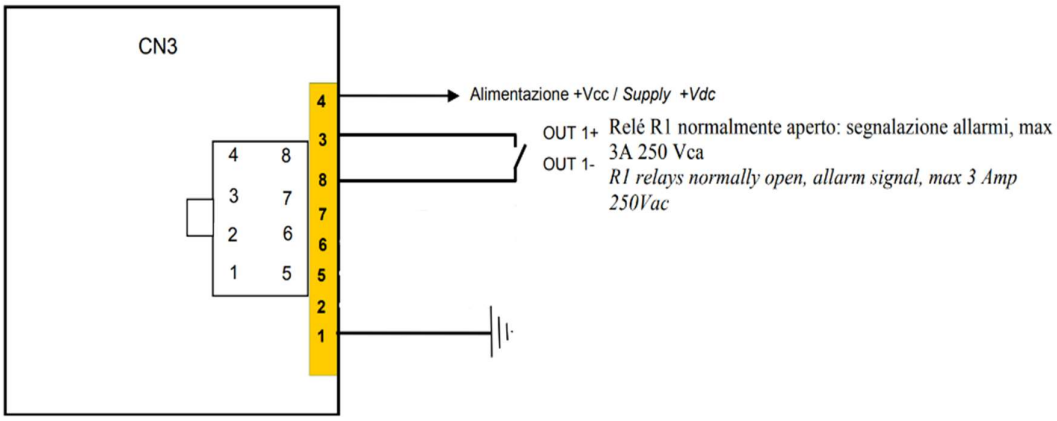

***Figure 27**: CN3 of PLN40*

As shown in the Figure 28, the connector 3 allows to detect errors in the drive. The mechanism of the connector is based on the theory of relay, if the circuit between pins 3.8 and 3.3 is closed, it means that the drive is in error. To implement this useful option it was necessary connect the pin 3.8 to a voltage source and the pin 3.3 to a digital pin of

Arduino, so in presence of an error the connector close the internal relay and the voltage is free to reach the digital pin, and by firmware is easy to red when the pin is high or low. A voltage divider was used to take under control the amount of current flow reaching the Arduino's pin to avoid malfunction. Considering that the first resistor R1 has 2.2 KΩ and R2 has 10 KΩ.

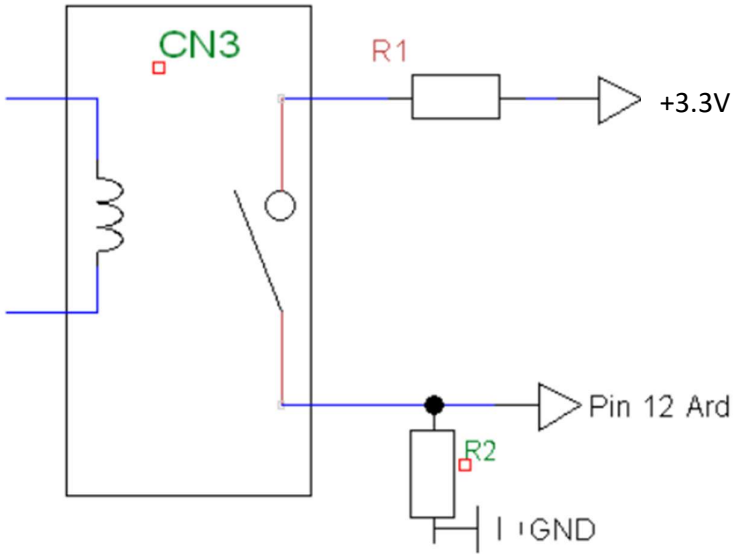

***Figure 28**: Connection of CN3 with voltage divider*

$$V_{pin\ 12} = \frac{R_2}{R_1 + R_2} * V_{in} = \frac{10}{2.2 + 10} * 3.3 = 2.7\ V$$

$$I_{pin\ 12} = \frac{3.3}{2200} = 1.5\ mA$$

The maximum value of current that could affect the digital pin is 1.5 mA, we are sure that it could not damage the microcontroller.

After having correctly connected the two drive connectors, we are able to detect and also to reset an alarm. As seen previously, thanks to the configuration used for CN3 if the system goes in error the relay between the pin 3.3 and 3.8 closes, a small current flow reaches the Arduino's pin we have decided to and we have the possibility to detect it, in other words we are able to check when there is an error. When the drive is in fault, for example if it receives both forward and backward commands together, it stop working and, in that moment, using our firmware we send the command to close the relay of the

third channel, that is linked to the reset pin 2.7. Below are reported the microcontroller pins used for the motion:

- Arduino pin 4 → move backward, CN2 pin 2.8
- Arduino pin 5 → reset command, CN2 pin 2.7
- Arduino pin 8 → move forward, CN2 pin 2.3
- Arduino pin 10 → alarm tracker, CN3 pin 3.3

The pins and their tasks were chosen based on the ease of physically making the connection and obviously based on which ones were still free.

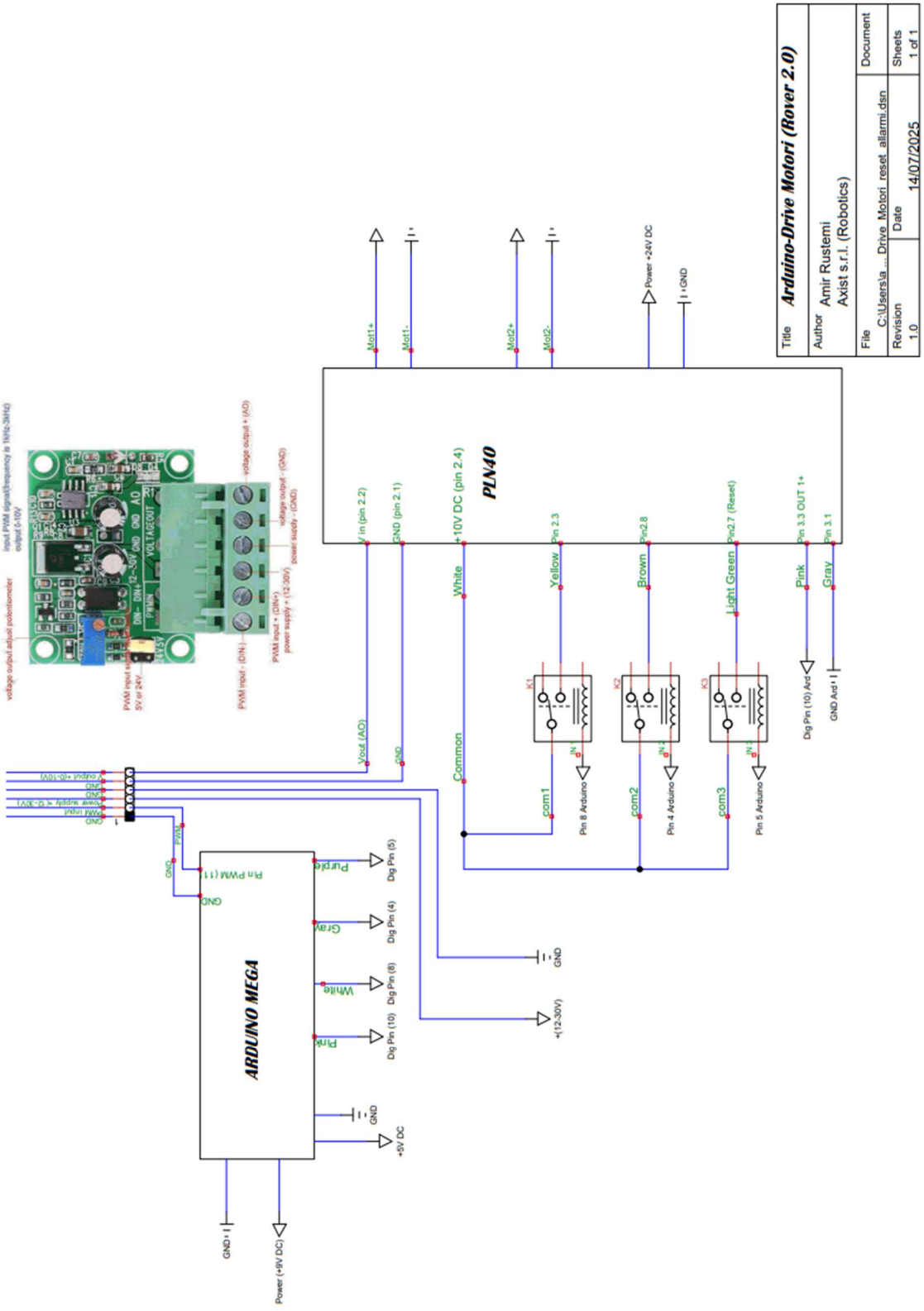

***Figure 29**: Arduino Movement and motor's drive scheme*

3.2.4 Radio controller

As said before, who takes the decision is the so-called PC ROVER, but who gives physically the command to Arduino Movement is an external controller. The FlySky FS-i6X radio control system, paired with the FS-iA10B receiver, was chosen for remote control of the system. This solution proved to be ideal for the project thanks to the large number of available channels (up to ten), which allows for the management of not only the main movement commands but also additional functions and auxiliary actuators. The availability of such a large number of channels allowed for the design of a modular control architecture, without the need for expansions or higher-level radio systems.

Another strong point of the FS-iA10B receiver is the possibility to select between two signal output modes: traditional PWM or serialized PPM.

PWM (Pulse Width Modulation), in this configuration, each channel generates a separate signal, with pulses varying between 1 and 2 ms and a period of approximately 20 ms. This is a very common and universally compatible mode, but it has the disadvantage of requiring a wire for each channel. In the case of ten channels, this would mean ten separate connections to the microcontroller, increasing wiring complexity.

PPM (Pulse Position Modulation): In this case, all channels are transmitted on a single wire as a train of consecutive pulses. Each pulse represents the position of a channel, while a longer interval acts as a synchronization signal and indicates the start of a new frame. In this way, the microcontroller receives a single signal and reconstructs the values of all channels by measuring the durations of the pulses in sequence. A complete frame lasts about 20–22 ms, so the channel refresh rate remains approximately 50 Hz, similar to traditional PWM.

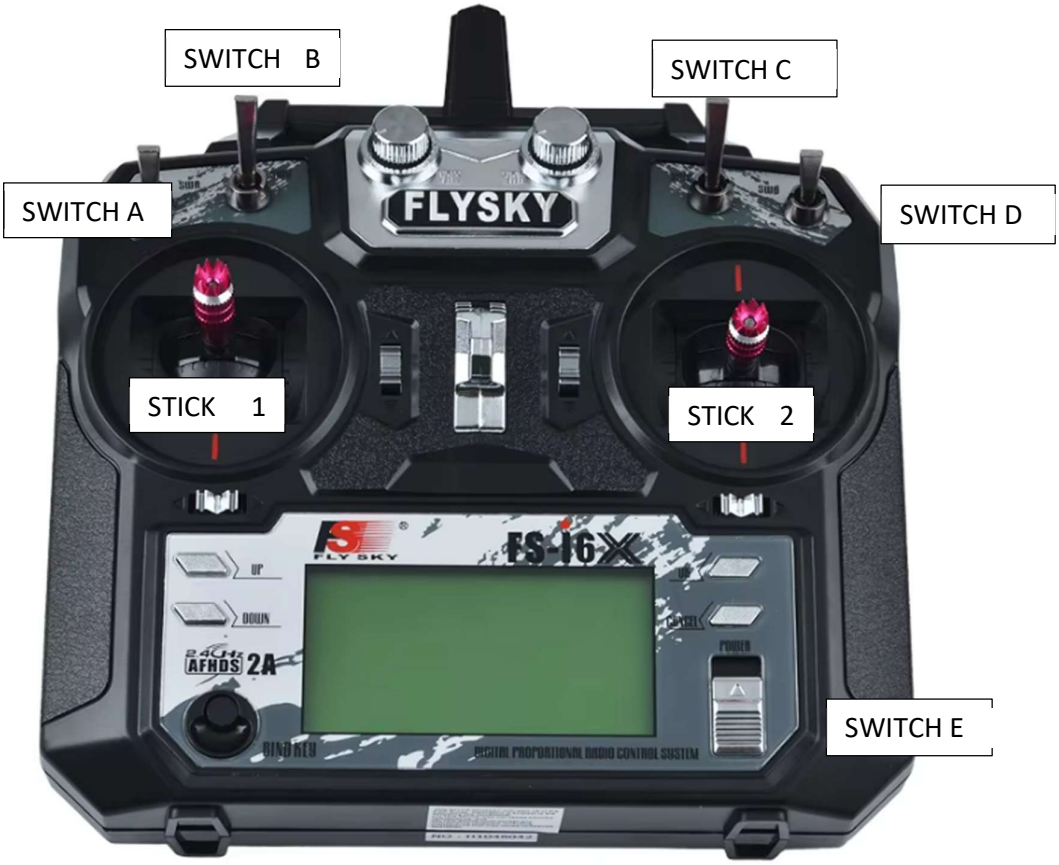

***Figure 30**: Radio controller*

STICK 1	**Up/Down**: Handling in automatic configuration	**To Up**. Enabled movement for automatic measurement. **To Down (*Default*)**. Automatic measurement is interrupted.
	Left/Right: Landing kinematics handling in Landing and Measurement/Shipment configurations	**To Left**. Kinematics are moved towards the Measurement/Shipment configuration. **To Right**. Kinematics are moved towards the Landing configuration. **Center (*Default*)**. Kinematics locks movement.
STICK 2	**Up/Down**: Manual handling for repositioning Rover.	**To Up**. Forward movement. **To Down**. Backward movement. Allows resetting number of steps automatically taken.

		Center (*Default*). Rover stopped.
	Left/Right: Not used	**Center** (*Default*)
SWITCH A	Enablement for use of the remote control	**To Up (*Default*)**. Remote control, despite being ON, is unusable. No Switch or Stick has any effect. **To Down**. Remote control is usable. All Sticks and Switches are enabled.
SWITCH B	Switch from automatic configuration to manual	**To Up (*Default*)**. The remote control is in manual configuration. Allows enabling manual movement forward and backward (Stick 2 Up/Down) and changing landing system configuration (Stick 1 Left/Right). Automatic movement is disabled, Stick 1 Up/Down has no effect. **To Down**. Remote control is in automatic configuration. Allows disabling manual movement. Stick 1 Left/Right and Stick 2 Up/Down have no effect. Stick 1 Up/Down is enabled, as described above.
SWITCH C	Enable/disable reset alarms	**To Up** (*Default*) . Reset command is disabled. **To Down.** Reset command is activated, during reset any other command does not have effect.
SWITCH D	Enable/disable cleaning cycle during automatic measurement phase	**To Up (*Default*)**. Cleaning cycle is engaged and active. **To Down**. Cleaning cycle is disengaged, bypassed during automatic handling procedure.
SWITCH E	Radio control activation	-

In the label is possible to see what type of command every switch and stick do, is necessary to say that in the stick 1 was implemented the command regarding the landing system

that actually is not used. This is useful, because the rover was designed taking into account that in the future it could be inserted.

The first step was to define a unique mapping between the PPM channels, coming from the FS-iA10B receiver, and the high-level commands used by the rover. The channels were associated with sticks and switches as follows:

- Channel 2 → stick 2
- Channel 3 → stick 1, vertical move
- Channel 4 → stick 1, horizontal move
- Channel 5 → switch A
- Channel 6 → switch B
- Channel 7 → switch C
- Channel 8 → switch D

Each channel is represented by the duration of a pulse in µs, but working with these values it is very hard and is subject to noise. Using this code:

```
for (byte channel = 1; channel <= MAX_CHANNELS; ++channel)
{
  int16_t value = static_cast<int16_t>(ppmReader1.rawChannelValue(channel)) - PULSE_OFFSET;
  if (value != values[channel - 1])
    changed = true;
  values[channel - 1] = value > 0 ? value : 0;
}
```

Is possible to take the raw values of the PPM channels and convert them into integers/booleans commands that the rest of the firmware can easily use. Basically the PPMReader library returns the duration of the impulse, for each channel, then a fixed offset is subtracted to align the values to the chosen operating range. The normalized value is saved in an integer array values[]. If the result is negative, it is forced to zero, eliminating any invalid readings.

The values are then inserted into this structure:

```
typedef struct {
   bool radio_enable;
   int  scan, move, mode, land, wipe, reset, misc;
} PPMCommands;
```

To move from the microseconds of PPM to reliable commands, thresholds consistent with the radio's behavior have been defined:

CMD_VALID_TRESHOLD_MIN = 500 µs

CMD_VALID_TRESHOLD_MAX = 1700 µs

It means that values under 500 µs and over 1700 µs are discarded because considered glitch values or affected by noise. A range of movement for stick was defined:

MOVE_CMD_MIN = 600 µs

MOVE_CMD_MAX = 1600 µs

CENTER_VALUE = (MOVE_CMD_MIN + MOVE_CMD_MAX)/2 = 1100 µs

To avoid noise or any other signal flickering a dead band around the zero position (center value) was inserted:

MOVE_CMD_ZERO_TOLERANCE = 0.1(1600-600) = 100 µs*

It means that like the values discarded before, also the signals inside the range between 1000 µs and 1200 µs are not taken in account or better are rounded to 1100 µs and considered as center value. When the RC receiver sends a PPM pulse, its duration is never perfect. Even if you hold the stick steady at the center, the values fluctuate slightly, for example, a theoretical 1100 µs command might be read as 1096, 1104, 1098, 1102. These small variations are jitter, noise or natural instability in the signal. If the firmware took every variation literally, the rover would see constant changes even without moving the stick, the motor would jitter, and the switches would click randomly. To manage this jitter, it introduces tolerances, that is, margins within which a change is ignored because it is considered noise, not a real command.

For continuous commands that is to say for the two sticks:

MOVE_CMD_JITTER_TOLERANCE = 50 µs

This means that variations smaller than 50 µs are considered noise. Only when the variation exceeds this value or when the change remains stable for a certain time is the new value accepted. Thus, the movement is fluid, without vibrations around zero.

For discrete commands (switches):

SWITCH_CMD_JITTER_TOLERANCE = 200 µs

Here the tolerance is higher, 200 µs, because the switches have only two positions (ON/OFF) and must change only if the jump is sharp and stable. In practice, as long as a pulse remains in an intermediate range or changes only slightly, the parser ignores it. Only when it significantly exceeds the ON/OFF threshold is it recorded.

Practical Example

Stick move

If it oscillates between 1000 µs and 1200 µs i.e., within the zero tolerance of ±100 µs, the code interprets it as zero movement. In this range, small oscillations due to jitter or mechanical inaccuracy of the stick are ignored.

Only when the value exceeds 1200 µs (upwards) or drops below 1000 µs (downwards), the code recognizes it as real motion, translating it into a proportional value between −100 and +100.

Switch reset

If the value fluctuates between 1490 and 1510 µs due to jitter, ±10 µs, the parser does not switch state immediately.

Only if the signal remains stable >1500 µs, with 200 µs margin and required stability time, the command is recorded as ON.

Failsafe behavior

In RC design, one of the first problems we tackle is failsafe behavior. What should the vehicle do when the radio link goes down, whether from interference, a dead battery, or someone flipping the kill switch? Letting it blindly obey its last command or, heaven forbid, act on garbage data from the receiver's noise floor is how things get broken. That's why our parser has a built-in failsafe. The moment it loses the *radio_enable* signal, it kills all the commands and defaults everything back to a safe, neutral state, without exceptions.

So, here's how our failsafe works. If the radio link drops and the *radio_enable* signal goes low, the parser doesn't freeze or panic. Instead, it executes a hard reset to a known safe state.

Here is exactly what happens.

Movement is killed: The move command zeros out. The vehicle stops dead in its tracks.

Landing sequence is aborted: Any active land command is immediately halted (land = STOP). We don't want it trying to land without a link.

All non-essential functions are shut down: scan, wipe are all turned OFF.

We fall back to manual control: The mode is switched to MANUAL. It's the simplest, most predictable state.

We lock out the reset: The reset function is disabled (reset = OFF) to prevent any accidental triggers during a vulnerable period.

The whole point is to avoid leaving the rover in some weird, limbo state. Instead, it defaults to a stable, predictable, and safe condition. This makes the vehicle's behavior completely deterministic, even when things go wrong. The system must always default to a secure condition until the operator is back in the loop and gives the all-clear.

3.3 Ovality box

The monopile is undoubtedly one of the most critical components in offshore structures, both from a structural and functional standpoint. In particular, the flange, the element designed to interface with the tower or other accessories, must maintain impeccable geometric precision. After all, if it were to become ovalized, the consequences would be severe: difficulties during assembly, misalignment, and not least, dangerous stress concentrations that would compromise the reliability of the bolted connection. It is for precisely these reasons that checking for ovalization is an indispensable verification during inspection and maintenance phases.

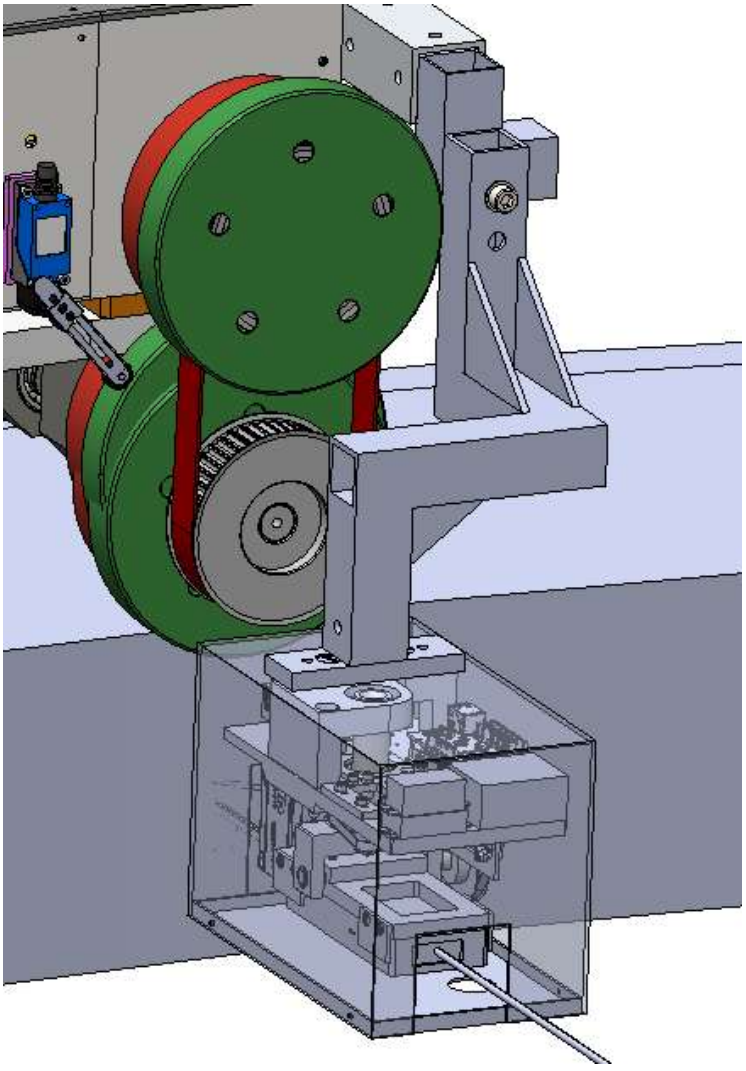

***Figure 31**: Ovality box mounted on the rover*

To address this need, we developed a system capable of performing the measurement directly in the field, mounted onboard the rover that operates inside the monopile. A

fundamental requirement was to ensure the system could be easily mounted and dismounted, while still guaranteeing impeccable stability during surveys. The location we identified for installation is on the lateral side, on the flange's internal surface, where the rover can operate on the area of interest with sufficient ease.

Another key challenge we had to account for was protecting the measurement system itself. Since the device is mounted on the side of the rover and has to operate within the tight confines of the monopile, we must always maintain a certain safety margin between the vehicle and the structure. This buffer zone is essential to prevent the sensor from crashing against the monopile's internal walls or even the rover itself during positioning or movement, a scenario that would almost certainly damage it. This is precisely why the tool cannot be designed to span the flange's entire diameter. Such a solution would be too bulky and would place the instrument in an overly exposed position, putting it at constant risk of impacts. As a result, the idea of taking direct measurements of opposite diameters proved unworkable. To ensure the integrity and reliability of our tool, we were forced to find an alternative solution: one that is less invasive, more compact, and, most importantly, safer, in keeping with our geometric and operational constraints.

To overcome the limitations of the lateral mounting position and the need to maintain a safety distance from the rover, we designed a sensorized box that acts as a reference unit. Inside this box, we integrated two independent measurement instruments, oriented in opposite directions:

The first faces outward from the rover, toward the opposite side of the flange, and measures the distance from the box to the farthest edge (which we can think of as the "external diameter").

The second faces inward, toward the rover itself, and measures the distance from the box to the nearest flange surface.

Geometrically, you can imagine the box as having two main directional vectors: one "outgoing" (external) and one "incoming" (internal). Each corresponds to a measurement axis that provides the distance to a different side of the flange. By combining these two pieces of information, we can obtain an indirect but reliable measure of ovalization. Instead of trying to cover the entire diameter with a single instrument, which would be impractical due to size and safety issues, we used two partial distances acquired in opposite directions. The difference and variation in these distances, collected as the rover moves along the inner circumference, allow us to accurately reconstruct flange deformations and thus estimate its ovalization.

Inside the ovality box are present two different instruments, both of them are specialized in what were our needs.

Leica DISTO X6

- This laser distance meter is our long-distance specialist tool for measuring the distance from the box to the opposite side of the flange. It's rugged, precise, and built to perform in challenging environments.
- Measures up to 250 meters with an accuracy of ±1 mm.
- Shock-resistant, IP65 rated, and features a touch screen that's easy to use even in tough conditions.
- Starts up in less than a second and can store hundreds of measurements.

In practice, it handles the "external diameter," where reliable medium to long-distance measurements are critical.

Micro-Epsilon optoNCDT 1900

- This sensor is a masterpiece of precision. It specializes in capturing ultra-high-resolution measurements at close range.
- Repeatability <0.1 µm: It can detect even the tiniest variations.
- Extremely high speed: Takes dynamic measurements at up to 10,000 times per second**.**
- Immunity to ambient light**:** It isn't fooled by challenging lighting conditions inside the monopile.
- Its job is to measure the distance from the box to the nearest flange surface, providing the data needed to reconstruct ovalization with extreme detail.

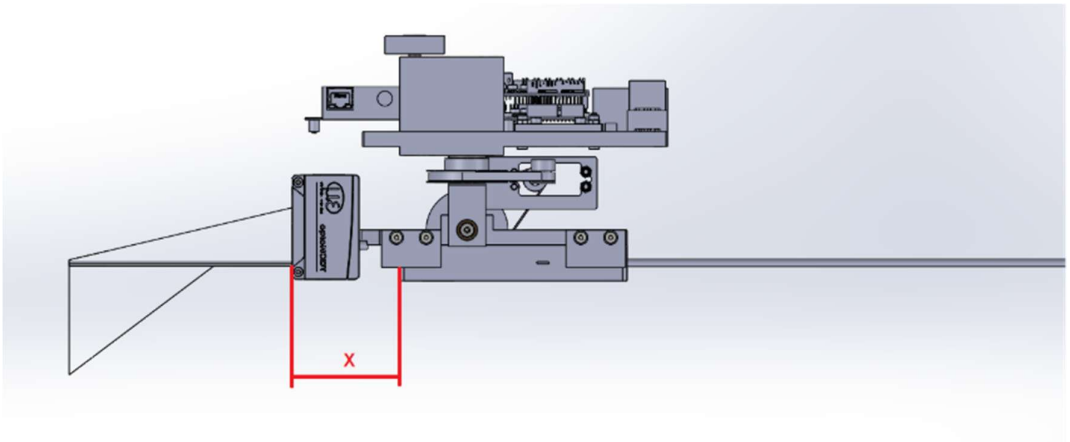

***Figure 32**: Lasers in ovality box view*

Together, these two instruments give us complementary measurements: one looks far, the other looks close, but both with absolute precision. It is the ideal combination for reliable inspection. The calculation is simple, it is enough to know the distance between the two lasers, x, and the length of the leica DISTO and the measurement of the diameter is given.

As shown in FIG 35 the ovality box is gifted of two servo controls thus is possible to make adjustment on elevation axis and on azimuth. To make that movement was implemented another microcontroller, based on Arduino uno, located inside the box whose task is only to control the ovality tool.

3.4 Arduino feedback

We have already saw how the movement and related part were designed, studying the corresponding microcontroller, and the ovality box was already presented. So far two microcontrollers have already been exposed but a third microcontroller has been inserted inside the rover called Arduino feedback.

As you can easily guess from the name the main task of this microcontroller is to give a visual feedback of the telemetry status. The rover 2.0 was equipped with a light turret, with four different colours each one describing a different situation.

Colour of light turret	Arduino feedback pin	Steady on	Blinking
White	4	/	PC ROVER turned on
Blue	5	/	Initialization completed
Green	6	/	Measurement in progress
Red	7	/	Fault
All	/	Rover turned on with PC ROVER turned off	/

This Arduino take also under control the status of the batteries, giving a feedback of the voltage value. To be able to detect the battery status using the microcontroller it was necessary to add a module compatible with it, so we used INA260.

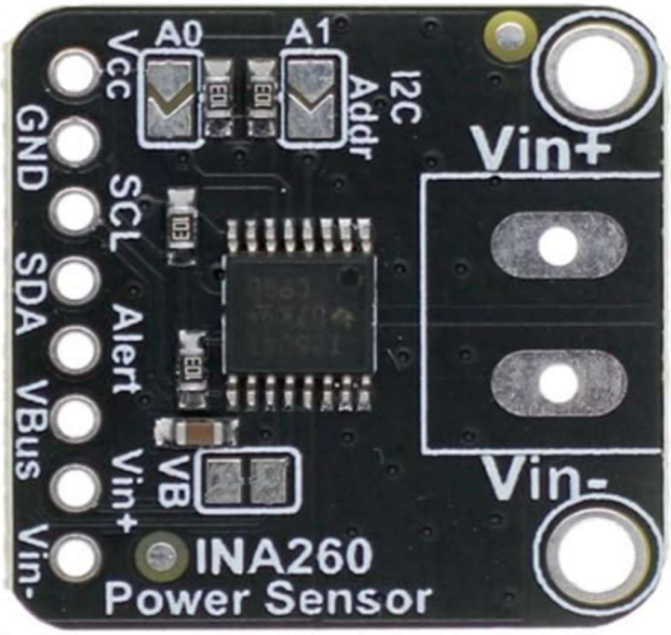

Figure 33: Module INA260

To monitor the power supply and consumption of our Ovality Box, we chose a small but powerful component: the INA260 module. It's a smart sensor from Texas Instruments (made easy to use by Adafruit) that measures voltage, current, and power in real time, all in one, with no extra circuitry required.

It has everything built-in:

- A tiny precision resistor (shunt) that lets it "sense" the current flow.
- A clean amplifier and a high-resolution analog-to-digital converter (16-bit).
- An I²C interface to talk directly to the control board, like an Arduino.

In practice, it's plug-and-play: connect it with two wires, and it returns ready-to-use digital values.

It is able to measure:

- Current**:** Up to 15 Amps, with milliampere accuracy.
- Voltage**:** Up to 36 Volts, perfect for battery and rover power systems.

- Power**:** It calculates this itself by multiplying voltage and current, so the microcontroller doesn't have to do heavy math.
- It's compact and simple: no external components, no complicated wiring.
- It's accurate: less than 1% error thanks to factory calibration.
- It's robust: handles voltage spikes and current surges.
- It's flexible: works for both battery-powered and fixed power systems.

In the Ovality Box, the INA260 is our energy watchdog:

- It monitors the battery and warns us if it's running low.
- It keeps an eye on motor consumption: if they draw too much current (due to friction or a jam), we stop them before they break.
- It tells us how much power we're using, so we can estimate how long the rover can keep operating.
- It's part of the safety system: if something goes wrong, it cuts off the power.

In short, with the INA260, we have complete and reliable energy control with minimal effort and maximum results.

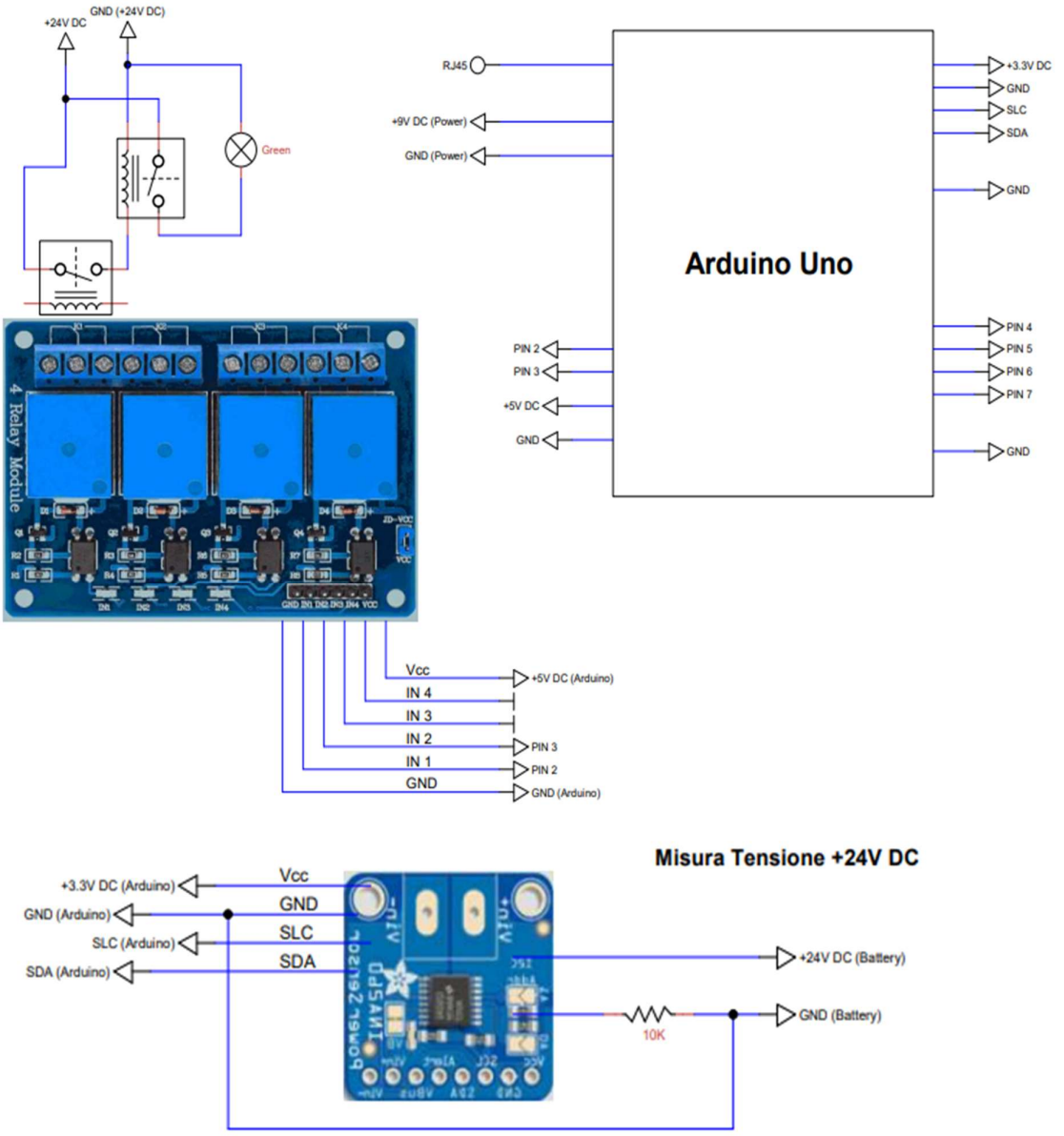

***Figure 34**: Arduino feedback scheme*

The calculations of the battery level is computed by firmware using this code:

```
if (currentMillis - previousMillis >= intervalMillis) {

previousMillis = currentMillis;

misuraTensioneBatteria();

}
```

The implementation is based on the principle of a timed state machine, utilizing the millis() function to manage timing in a non-blocking manner. The intervalMillis variable defines the sampling period, 1000ms = 1s. On each iteration of the main loop (loop()), the algorithm checks whether the time elapsed since the last execution exceeds the predefined sampling period. If the condition is met, the measurement function is called, and the reference timestamp is updated. This approach ensures periodic and isochronous data updates while simultaneously maintaining system responsiveness, as it avoids using the blocking delay()function and allows the microcontroller to execute other operations during the waiting intervals.

tensione = ina260.readBusVoltage() / 1000;

The ina260.readBusVoltage() function returns the bus voltage value read by the sensor, expressed in milliVolts (mV) as an integer. To convert this value to the standard unit of measurement (Volts), it is scaled by dividing it by 1000. This operation not only performs a unit conversion but also transforms the data type from integer to floating-point, ensuring the precision of the measured value. The result of the conversion is stored in the tensione (voltage) variable.

For example, a read value of 24700 mV, divided by 1000, yields the value in Volts: 24.7 V

3.5 Power supply

This entire system requires a big amount of power, to guarantee enough tension the all supply scheme was revisited. In the second chapter was explained that the first rover was powered by three batteries, each one disbursing 12 V dc, this configuration could no longer guarantee the correct functioning of Prometheus 2.0, so a fourth new battery was added.
In the first design the batteries were divided in this way:

- Internal batteries → two were connected in series with a total value of 24 V
- External battery → the remaining one was dedicated to supply just some auxiliary devices

Among all the component used for the second rover the one that required a higher supply voltage did not exceed 24 V.

The new batteries used are all four equal, in particular they provide 12 V with 28 Ah each one. Given the specific power requirements of the devices used, it didn't make sense to connect them all in series, thus having a total voltage of 48 V which would have required

an unnecessary addition of some DC-DC converters to perform 24 V as the higher value requested. The configuration needed to balance two critical demands: operational independence and safety during recharging.

Two key requirements guided the design:

- Recharging without removing the batteries→ the rover is designed to operate in environments where frequently swapping out batteries would be impractical and inefficient. To address this, the system was configured to allow the batteries to be recharged directly onboard, within their housing. This approach streamlines operations, minimizes downtime, and reduces the risks associated with repeatedly handling heavy and sensitive components.
- Preventing charging during operation→ concurrently, safety protocols demanded a strict rule, the system must prevent any scenario where the batteries are simultaneously powering the rover and being charged by an external source. This overlapping state is potentially hazardous, as it could lead to overheating, uncontrolled electrical currents, or accelerated degradation of the battery cells. To mitigate this risk, the architecture incorporates electrical interlocks and logical controls that detect the rover's operational status and automatically disable charging whenever the motors or other subsystems are active.

Functionally, these twin requirements mean the power management system must perform a selective switching operation.

When the rover is active, whether moving or taking measurements, the circuit completely isolates the external charging line from the battery pack.

Conversely, when the rover is idle and placed in a maintenance state, the circuit permits a secure connection to the charger, initiating a safe recharging process that does not interfere with the onboard electronics.

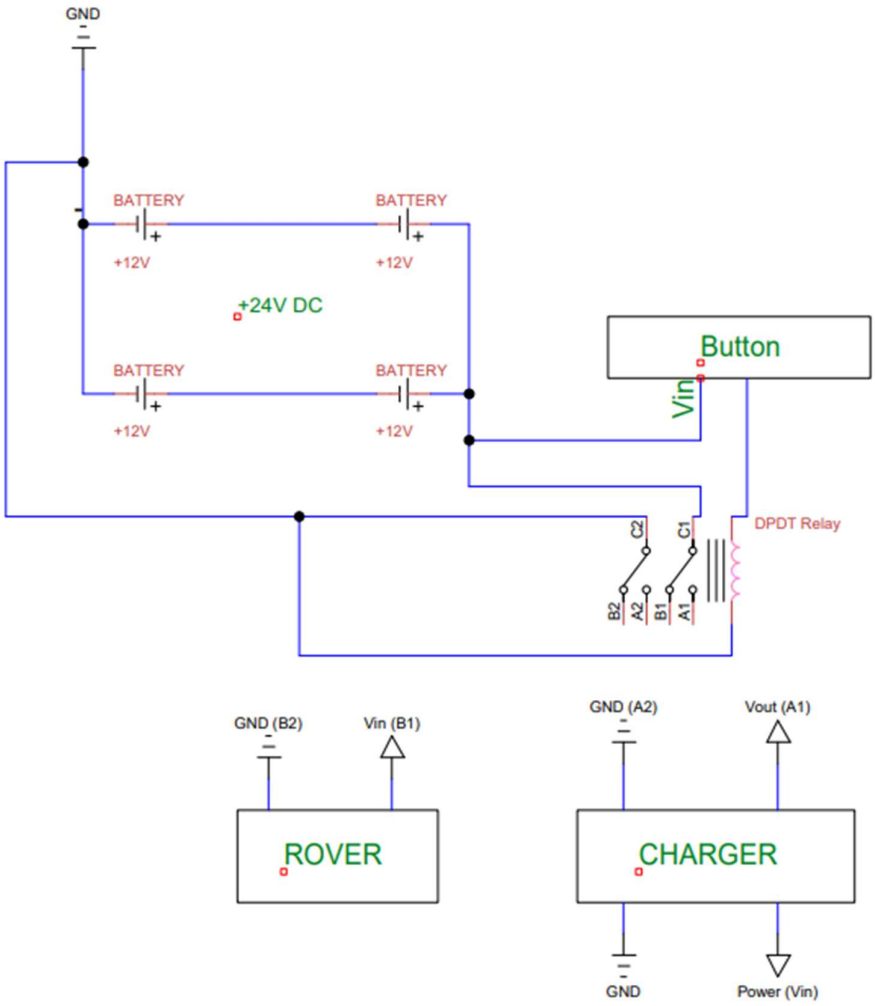

***Figure 35**: Power system scheme*

To provide the rover with the necessary power, a battery pack was created that combines the advantages of series and parallel connections. Here's how it was built:

- First two 12 V batteries were connected in series, creating a first group providing 24 V.
- The same process was repeated with the other two batteries, forming a second identical 24 V group.
- Finally, these two 24 V groups were connected together in parallel.

The result is a single power source that maintains the required 24 V voltage but doubles the overall capacity. This way, the rover does not only receive the appropriate voltage to operate but also benefits from significantly extended runtime thanks to the increased energy reserve.

3.6 Workstation and HMI

Feedback is a crucial element for the rover's operation. It is the mechanism that provides a constant and updated return on the status of all its subsystems and working conditions. In an application like this, it is not enough to send commands and assume they will be executed, the operator must be able to verify in real-time what is happening to react promptly to any unforeseen events or anomalies.

Feedback essentially serves two key functions:

- On one hand, it is a form of active control, as it allows for monitoring the rover's actual response to the commands given.
- On the other hand, it is a fundamental safety tool that prevents operating "in the dark" and allows for halting or modifying manoeuvres as soon as a dangerous or unexpected condition arises.

For this system to be truly effective, however, an adequate user interface is essential. It must be designed to present all necessary information clearly and in an organized manner. It's not just about displaying numerical data or graphs, but about creating an environment that gives the operator an immediate overview while also allowing quick access to technical details when needed. To meet this need, the system architecture includes, in addition to the microcontrollers and the onboard computer (the PC-ROVER), an external processor: the workstation. The latter plays a central role, acting as the collection, processing, and storage point for all data coming from the rover. Dedicated software installed on the workstation maintains active communication with the vehicle, receives feedback packets, processes them, and presents them through graphical interfaces, tables, or chronological logs. This way, every acquired parameter, be it electrical data, position, sensor status, or a measurement, is not only displayed in real-time but also saved systematically to allow for subsequent analysis.

The workstation, therefore, is not just an additional computer. It is the genuine supervision and control hub of the entire system, it is here that the operator makes decisions, it is here that data is recorded and stored, and it is here that the human-machine interface is realized. Without this infrastructure, feedback would merely be a flow of signals difficult to decipher, with the workstation, however, it becomes usable information, capable of guiding the operator through every phase of the rover's use.

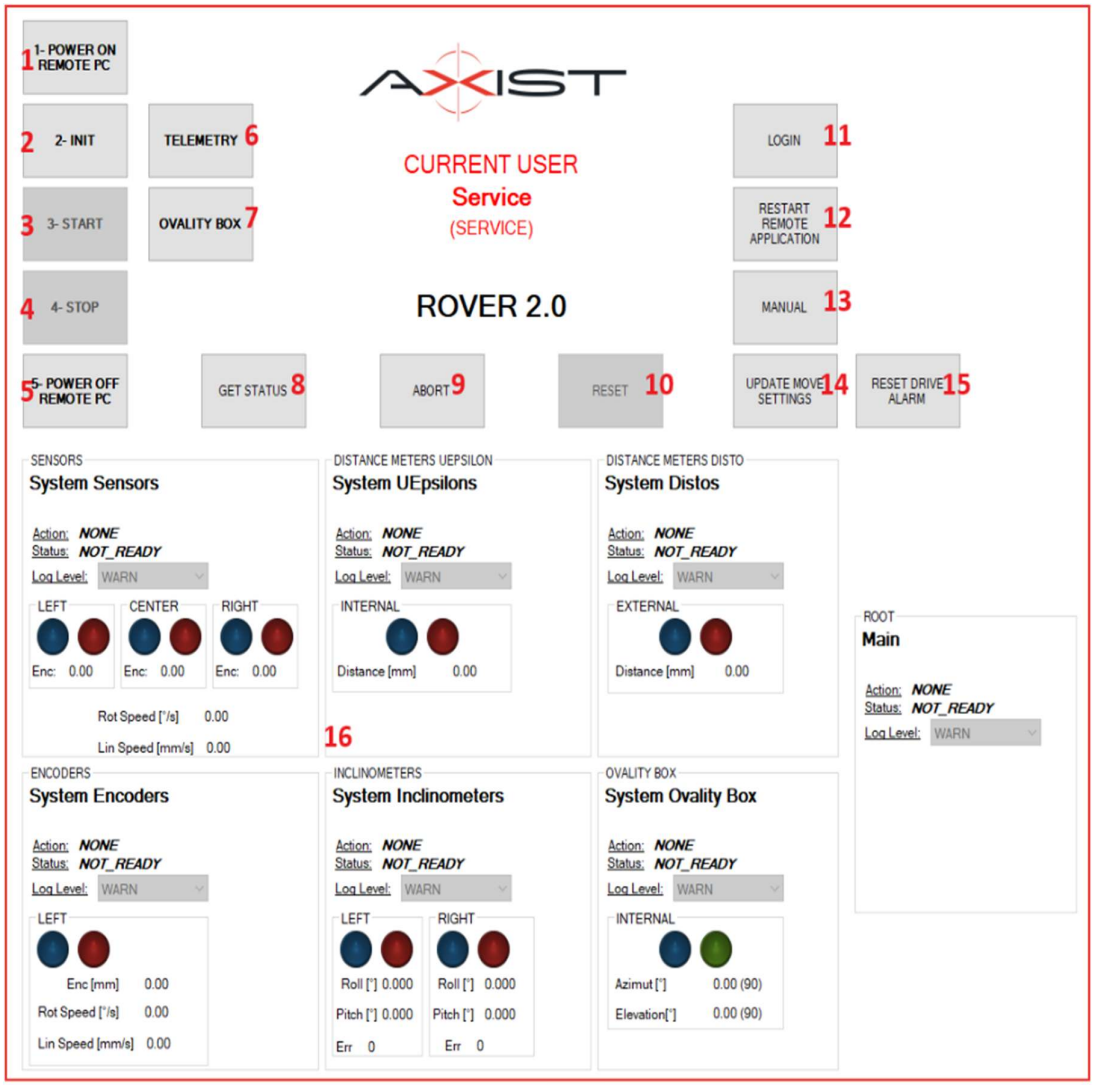

***Figure 36**: HMI*

The Home tab contains the main commands to be executed on the remote PC.

1. POWER ON REMOTE PC: Turns on the remote PC ROVER and start remotely the application
2. INIT: Connects in remote the application to sensors and prepare them and all the system for the acquisition
3. START: Starts acquisition and measurement on remote
4. STOP: Stops any previously started acquisition
5. POWER OFF REMOTE PC: Shuts down remotely the PC ROVER

6. TELEMETRY: Opens the Telemetry window that allows to connect and see the camera and the measurement behavior
7. OVALITY BOX: Opens the Ovality Box control window
8. GET STATUS: Checks communication and gets current remote PC ROVER status
9. ABORT: Aborts current operation and tries to fix any pending errors on remote
10. RESET: Forces reset on remote device
11. LOGIN: Updates Current User
12. RESTART REMOTE APPLICATION: Restarts remote application on PC ROVER
13. MANUAL: Opens User Manual
14. UPDATE MOVE SETTINGS: Update Move settings from configuration in Arduino movement
15. RESET DRIVE ALARM: Resets any alarm in motor's drive, just like the command give using the radioconroller
16. CURRENT STATUS: panel displays state of system components, using 'get status' to refresh it.

These are the main buttons on the HMI and their functionality but is well to know that with administrator login is possible to change important parameters inside Arduino movement.

To record a dataset the following steps must be completed:

1. DETACH POWER SUPPLY: detach the charger from the rover's body and cover up the port.
2. POWER THE ROVER: Power the system from the emergency button.

Wait for the white LED of the metrology feedback to be only one blinking.

3. POWER THE DISTO: While the ovality box is upside down press the power button on the Disto. Make sure that all the cables (power, ground and two LAN cables) are connected and not loose.
4. PUT DISTO ON MEASURE MODE: press again on the power button on the Disto to put it in measure mode.

WARNING → if the steps 3 and 4 are not completed correctly INIT will always fail!

5. POWER ON REMOTE PC: From the HOME Tab click TURN ON REMOTE PC.

The remote PC ROVER should automatically turn ON together with the Rover. However, always push the 1-POWER ON REMOTE PC button for the sake of robustness of the procedure.

***Figure 37**: Power on remote pc*

6. 2-INIT

***Figure 38**: INIT button*

7. 3-START:

***Figure 39**: START measurement*

8. LOWER OVALITY BOX: Make sure to lower the Ovality Box before proceeding.
9. TRANSPORT THE ROVER ON THE FLANGE: Position the rover on the flange.
10. ADJUST POSITION: Move the Rover to adjust the position on the flange. If no issues are present the sling can be detached from the crane hook and the measurement procedures can start. Always check the visual feedbacks to know about the status of the rover.
11. ADJUST OVALITY BOX: Using the OVALITY BOX (RoverFlangeAnalyzer>Home) adjust the direction of the ovality box. Make sure that the laser points to the opposite cardinal point marker and halfway between the top and bottom surfaces of the flange.

Save the configuration with SET DEFAULT to memorize the correct azimuth and elevation datas.

To save this configuration press save and exit otherwise the new default position will not be saved.

To check this close and open the ovality box menu again to see if the correct azimuth and elevation are still memorized.

12. PERFORM MOVEMENT: Move the Rover on the flange to acquire all the data needed using the radio transmitter. The measure is carried on automatically without the need for manual intervention.
13. STOP AUTOMATIC MOTION: once the rover returns to the initial cardinal point stop the automatic motion of the rover with the radio controller.
14. STOP MEASUREMENT: From the HOME Tab click 4-STOP to stop the measurement of the Rover.

***Figure 40**: Stop measurement*

15. POWER OFF REMOTE PC: From the Home Tab click 5-POWER OFF REMOTE PC. Once the PC is off all the lights of the metrology feedback should be on and blinking.

***Figure 41**: Power off remote PC ROVER*

17. DISABLE POWER: Press the emergency button to disable power

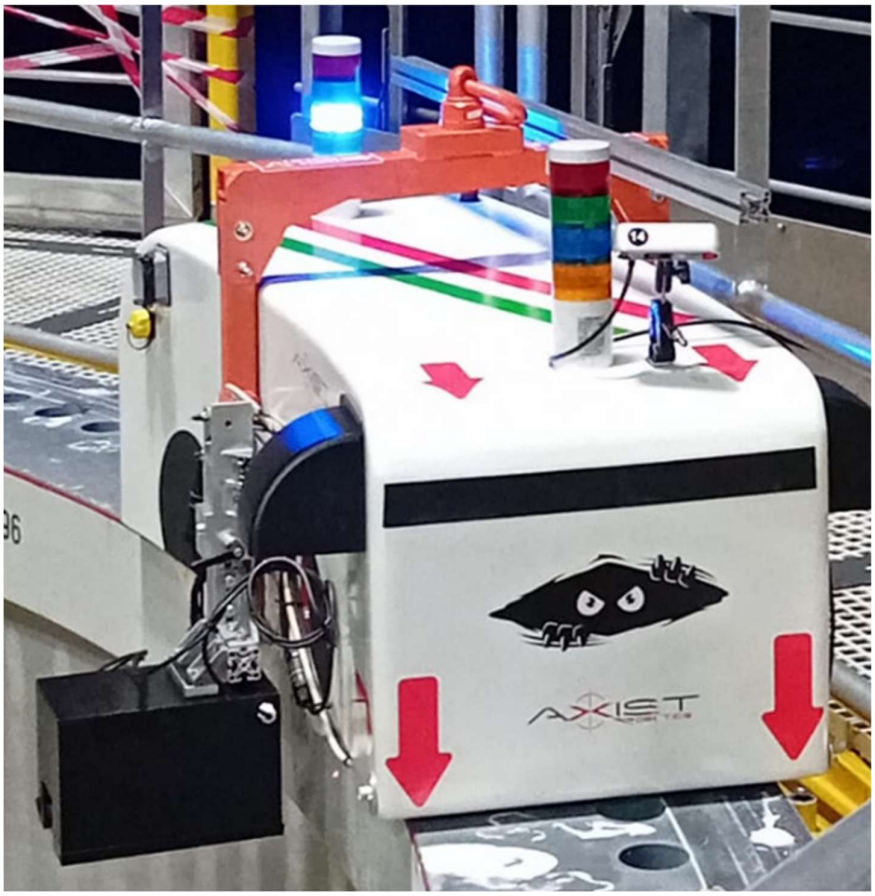

***Figure 42**: Rover 2.0 on working*

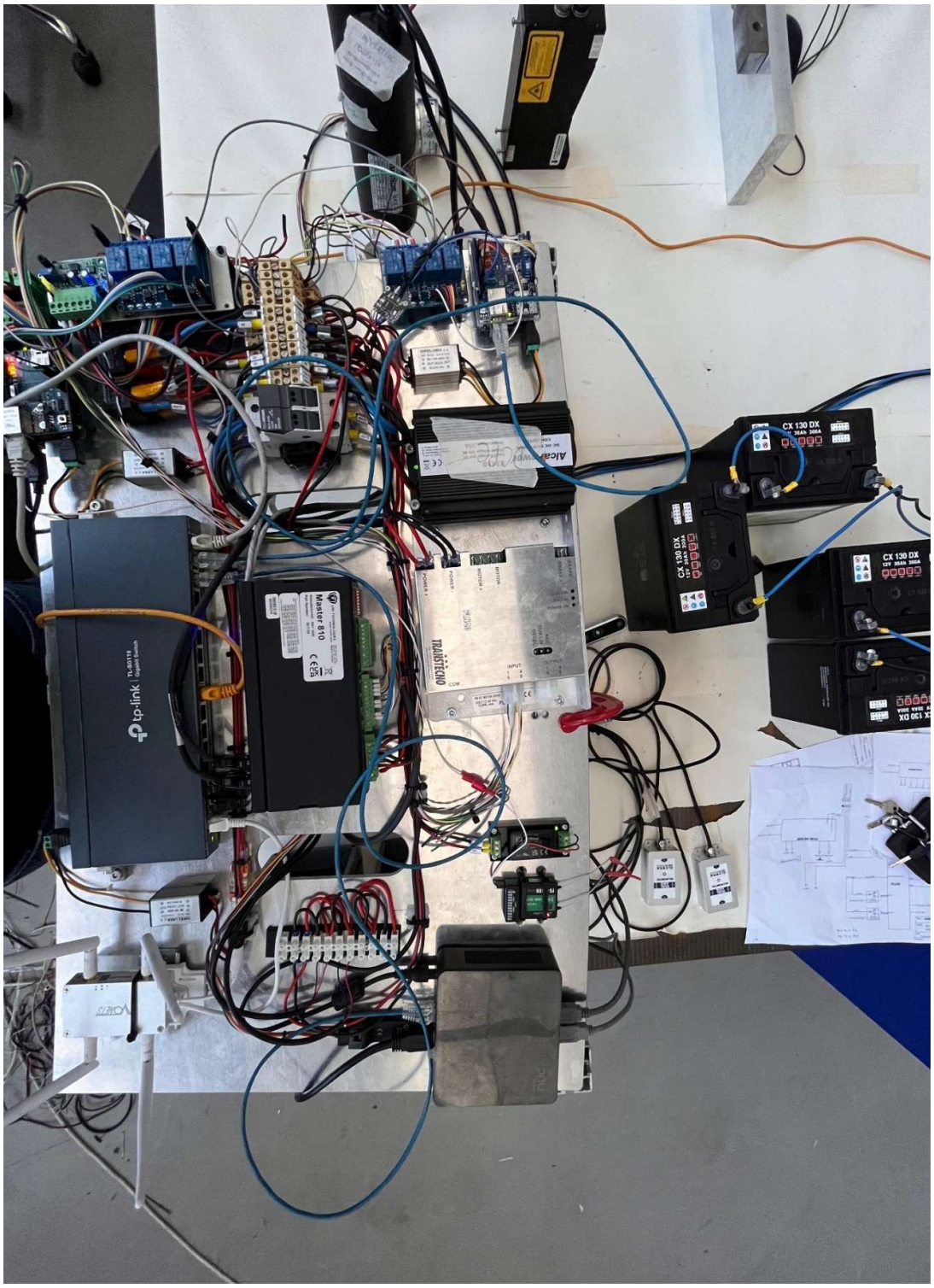

***Figure 43**: Rover 2.0 in development*

Chapter 4- Future improvements

The new rover already represents a clear step forward from the previous generation, featuring improved feedback functions, pan/tilt positioning for the Ovality Box, and the acquisition of measurements and energy telemetry.

However, the distributed architecture based on multiple microcontrollers (Arduino + shields + dedicated firmware) presents inherent limitations in terms of:

- Maintainability → Multiple firmware versions need to be updated and verified for each component, making system maintenance more complex and vulnerable to errors.
- Scalability → Adding I/O and functions requires extra wiring and additional code, making future development cumbersome and inflexible.
- Synchronization → Coordinating timing, filters, and motion across different nodes is highly complex within a distributed and decoupled architecture.
- Industrial Robustness → The use of non-standard protocols, "maker"-style connectors, and limited Electromagnetic Compatibility (EMC) makes the system less reliable for demanding industrial environments.

To consolidate control, synchronization, and diagnostics, the possible solution is to migrate to a single, unified industrial platform: a Beckhoff PLC with TwinCAT 3 and an EtherCAT (Ethernet for Control Automation Technology) backbone. This approach centralizes all command, motion, and monitoring logic onto one highly integrated and robust industrial system.

A PLC, Programmable Logic Controller, is an industrial computing platform designed for three core purposes:

1. Executing deterministic and cyclic logic tasks (real-time operations).
2. Reliably managing distributed I/O with built-in diagnostics.
3. Ensuring electrical/EMC robustness and simplified maintenance.

Beckhoff's architecture merges an Embedded PC (running Windows for logging and services) with a real-time PLC runtime (TwinCAT 3). This runtime governs all cyclic tasks, motion control, I/O, and communications. The EtherCAT backbone updates thousands of I/O points within millisecond timeframes, utilizing Distributed Clocks for precise, sub-millisecond synchronization across the entire system.

Key advantages:

- Unified Software Project → A single, integrated TwinCAT project for the entire machine's logic, motion, and operator interface.
- Integrated Diagnostics → Powerful, readily available troubleshooting tools like real-time oscilloscopes, system logs, module status monitoring, and centralized alarm management.
- Unified Web-Based HMI → A centralized human-machine interface (HMI) accessible via a standard web browser on any device, without dedicated client software.
- Comprehensive I/O Portfolio → A huge selection of native I/O terminals for every need, including digital, analog, serial, motion, and functional safety (Safety over EtherCAT).

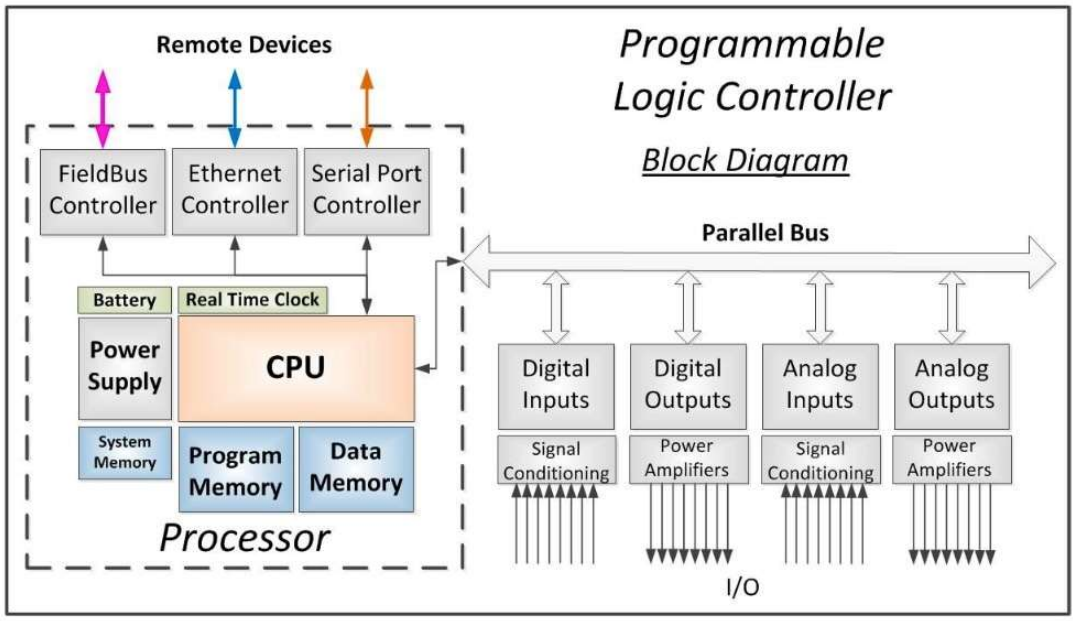

***Figure 44**: PLC architecture*

4.1 PLC architecture target

The system's core will be a Beckhoff Embedded PC running the TwinCAT 3 Runtime. This combines a Windows OS for services and interfaces with a deterministic real-time PLC environment for machine control. All peripheral devices will be connected via the EtherCAT bus, serving as a high-speed communication backbone. I/O modules can be installed directly on the machine.

Every signal currently handled by microcontrollers, from radio control PPM signals to digital inputs, relays, and motors, can be integrated using specialized EtherCAT I/O modules:

- Standard digital modules for ON/OFF commands, buttons, and status signals;
- High-speed digital modules for complex signals such as PPM or encoders;
- Analog modules for precise voltage and current measurements.

Motion control can be handled by Beckhoff motion terminals or external EtherCAT drives, integrated directly into the PLC cycle. This centralizes the control logic, improving axis coordination and enabling advanced motion profiles, such as acceleration ramps, torque limits, and interlocks.

The system remains compatible with the existing UDP protocol using TwinCAT's communication libraries.

The operator interface becomes simpler and centralized: a single web-based HMI, accessible from any browser without additional installations, replaces previous fragmented solutions.

Required Analysis and Preliminary Studies

Migrating to a PLC architecture requires careful planning:

- Mapping of all existing nodes (microcontrollers, sensors, actuators) and their translation to EtherCAT modules;
- Evaluation of necessary interfaces: some sensors may require dedicated gateways;
- Assessment of the computational power needed to ensure real-time performance;
- Cost-benefit analysis: although the initial investment is higher than microcontroller-based solutions, the advantages in reliability, reduced maintenance, and scalability make the PLC solution advantageous in the medium to long term.

One of the most important aspects of migrating to a Beckhoff PLC involves communications. The current architecture relies on UDP (User Datagram Prodocol) packets exchanged between microcontrollers and the workstation. This logic can be maintained without difficulty using TwinCAT 3, which provides native libraries for handling

UDP and TCP (Transmission Control Protocol) sockets. This ensures backward compatibility with already developed systems, avoiding the need to completely rewrite the PC-side software.

From the operator's perspective, all this data can be delivered through a web-based HMI. This means it is no longer necessary to install proprietary software on individual machines: a browser is sufficient to access dashboards, graphs, and logs, both locally and remotely.

A crucial node in the transition to a Beckhoff PLC involves the integration of metrological sensors. Some devices already use interfaces compatible with the industrial world, while others originate in more laboratory contexts and require specific adaptations.

For example:

- The Micro-Epsilon optoNCDT 1900 sensor can be connected via high-resolution analog output, RS422 serial, or Ethernet. In both cases, Beckhoff offers compatible terminals that allow direct acquisition, with the ability to synchronize sampling with the PLC's control cycles.
- The Leica DISTO X6, on the other hand, is designed primarily for manual applications and often uses Bluetooth connections. To integrate it into an industrial setting, it is preferable to equip it with a wired interface (serial or Ethernet) or, if necessary, use a dedicated gateway to translate the protocol into one manageable by the PLC.
- Sensors using protocols not typical of the industrial sector, such as I^2C, will need to be replaced with equivalents featuring industrial analog or digital interfaces.

Safe energy management is one of the most critical aspects of the rover's architecture. In particular, it is essential to ensure that batteries can be recharged without being removed from the vehicle, while simultaneously preventing charging while the rover is operational to avoid risks of overheating or damage to components.

With the transition to a Beckhoff PLC, these aspects can be managed directly by the control logic:

- The PLC can control the contactors that allow or inhibit charging based on the operational status of the rover;
- It can constantly monitor voltage, current, and power consumption via analog modules or industrial sensors, replacing devices like the previously used INA260;

- It can generate alarms and activate fail-safe strategies, such as shutting down non-critical loads or automatically disconnecting the charger under abnormal conditions.

Furthermore, with the integration of TwinSAFE modules, the safety level can be extended to functional aspects: emergency motor shutdown, management of stop buttons, interlocks, and redundancy. This way, safety is no longer just an external constraint on the electronics but becomes an integral part of the system's logic.

4.2 Vision system

One of the most promising future upgrades for the rover is the integration of a vision system based on the existing onboard camera. Currently, the camera serves as a passive tool, providing a video stream for the operator to monitor the environment and operations. However, this same resource could be enhanced through computer vision techniques and artificial intelligence models, transforming it into an active sensor.

The concept involves using algorithms capable of automatically analysing captured images, extracting useful information, and returning it to the control system as structured data. Convolutional Neural Networks (CNNs) are primarily used for this purpose, representing the de facto standard for vision applications. A CNN is designed to simulate the biological visual system: initial layers detect simple features like edges and lines, while deeper layers combine these elements into increasingly complex shapes, eventually recognizing full objects or scenes.

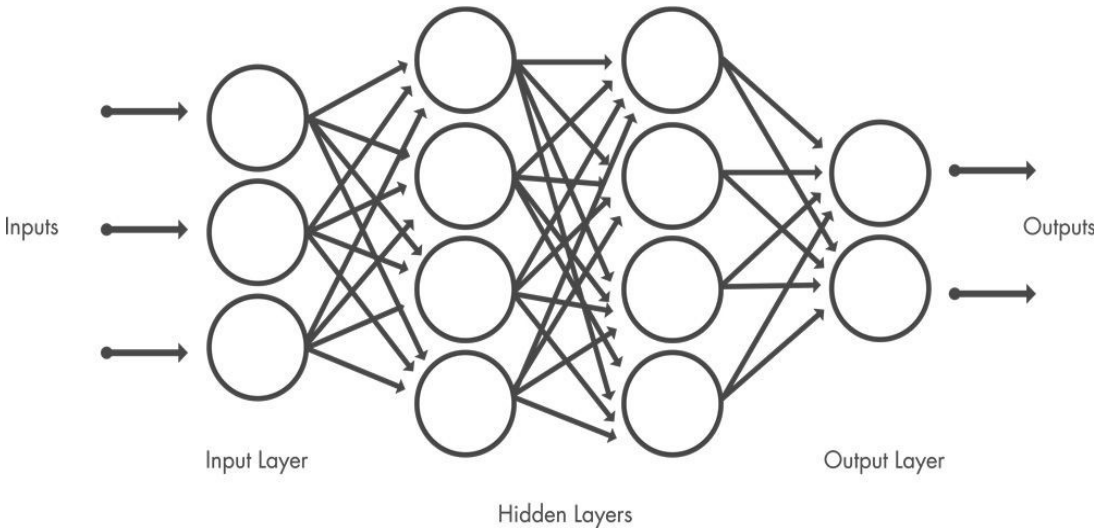

***Figure 45**: CNNs architecture*

Such a vision system would enable several currently absent functionalities:

- Object recognition and classification: The neural network could accurately identify environmental elements (obstacles, flanges, bolts, areas of interest), reporting them to the operator or using them as input for automation.
- Anomaly detection: By training the network on reference "correct" images, it would be possible to flag deformations, wear, or defects in structural components.
- Spatial localization: Object detection techniques (e.g., YOLO, Faster R-CNN) would allow not only detecting presence but also the exact position of objects within the image.
- Assisted navigation: Combining vision with depth estimation algorithms (stereo camera or depth from motion), the rover could avoid obstacles or follow paths autonomously.

From a practical standpoint, integrating neural network-based vision would require:

- Dedicated computing hardware, such as embedded GPUs or edge devices (e.g., NVIDIA Jetson), capable of running neural networks in real time without burdening the PLC.
- Training datasets, consisting of images collected under real working conditions, necessary to "teach" the model to recognize specific objects and situations.
- Processing pipeline, from image capture and data normalization to inference via the neural network and sending results (labels, coordinates, probabilities) to the rover's control system.

An interesting aspect is that these techniques do not merely provide an "electronic eye" but transform the rover into a system capable of interpreting scenes. The operator would no longer receive only a raw video stream to interpret manually, but pre-filtered and structured data, such as:

- "Obstacle detected 1.2 m on the left side";
- "Flange identified, ovality outside standard range";
- "Clear area for movement".

This could drastically reduce reliance on direct human intervention, increasing operational autonomy and reliability. Simultaneously, computer vision could become a tool for documentation and traceability, storing images and processed results as objective inspection reports.

The ultimate value of this integration is not merely automation, but the creation of a virtuous cycle of improvement. Each mission generates new data, which can be used to further refine and retrain the neural networks, making them increasingly accurate and robust over time. The rover transitions from a simple measurement tool into a true partner, a intelligent diagnostic platform that seamlessly integrates perception, analysis, and documentation into a single, continuous workflow, forever enhancing how we explore and interact with complex environments.

Chapter 5 - Conclusions

The development of the Prometheus 2.0 rover, as presented in this thesis, marks more than just an incremental improvement, it constitutes a fundamental redesign of the platform, with significant advances in architecture, functionality, and overall reliability. By adopting a distributed network of microcontrollers linked through Ethernet communication and managed by a dedicated supervision workstation, we have achieved tight coordination among complex subsystems including motion, metrology, and power management. This integration has been crucial in enabling sophisticated behaviors and precise control.

A key focus throughout this work has been the implementation of robust feedback mechanisms. Through careful signal parsing, structured communication protocols, and the application of a Kalman filter for motion estimation, the system now delivers greater accuracy and resilience during operation. These elements work together to create a rover that not only performs tasks but does so with a level of dependability essential for real-world applications.

Another critical aspect, often overlooked but vital to real-world performance, was the attention given to hardware connections and cabling. By prioritizing electrical robustness and signal integrity, we ensured that the rover operates safely and consistently, even in challenging environments such as offshore wind farms. The result is a compact, yet highly capable platform capable of collecting reliable metrological data during demanding inspection missions.

Beyond these technical contributions, this project underscores the value of a multidisciplinary approach. It bridges metrology, robotics, and control theory, turning conceptual designs into a fully functional system ready for industrial use. This transition from experimental prototype to practical tool represents one of the most important outcomes of the research.

Looking ahead, several promising directions emerge. Centralizing control on industrial-grade PLCs and incorporating AI-driven vision systems based on neural networks could further elevate the platform's capabilities. While these enhancements lie beyond the current scope, they point toward a clear trajectory: evolving Prometheus from a semi-automatic assistant into an intelligent, autonomous inspection agent.

In summary, Prometheus 2.0 embodies a substantial step forward in the development of inspection robotics. By uniting robust hardware, reliable software, and advanced

measurement technologies, it demonstrates how mobile robotic platforms can become indispensable partners in industrial settings. This work not only provides a solid foundation for future development but also serves as a springboard toward the next generation of intelligent inspection systems.

Bibliography

1. Axist S.p.A. Collaudi dimensionali e metrologia industriale a Rivoli (TO). Disponibile su: https://www.axist.it/

2. LMI Technologies. *Datasheet: Gocator 2600 Series – 3D Smart Laser Line Profilers*. Disponibile su: https://ftp.stemmer-imaging.com/webdavs/docmanager/166909-DATASHEET_Gocator_2600.pdf

3. BWSensing. BWS2000 Series Technical Manual. Disponibile su: https://www.bwsensing.com/upload/userfile/BWS2000_TM.pdf

4. Prof. Michele Taragna. Kalman filter theory. Estimation, filtering, and system identification, Politecnico di Torino, (A.A. 2024-2026)

5. Transtecno. I EC IP66 – DC Electric Motors – 0521A. Disponibile su: https://transtecno.com/wp-content/uploads/2015/01/I_EC_IP66-DC-electric-motors_210513_0521.pdf

6. Transtecno. 260id-Transtecno-G-CM_CMP-Wormgearmotors_0317-50_Hz.pdf (catalogo Worm Gear Motors CM / CMP). Disponibile su: https://transtecno.com/wp-content/uploads/2017/10/260id-Transtecno-G-CM_CMP-Wormgearmotors_0317-50-Hz.pdf

7. Transtecno. PLN20 – PLN40 Installation and Maintenance Instructions (Manuale PLN20/PLN40). Disponibile su: https://www.transtecno.com/wp-content/uploads/2023/05/PLN_Installation-and-Maintenance-instructions_1222-1.pdf

8. LC-LM358-PWM2V. *PWM to Voltage Converter Module (0-10 V)*. Disponibile su: https://makerhero.com/img/files/download/3PSC3-manual.pdf

9. ELEGOO. *Modulo Relay 4 canali DC 5 V con optoisolamento.* Datasheet / manuale tecnico. Disponibile su: https://www.elegoo.com/blogs/arduino-projects/elegoo-dc-5v-relay-module-tutorial

10. Adafruit. *Adafruit INA260 Current + Voltage + Power Sensor Breakout – Datasheet / Manual*. Disponibile su: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina260-current-voltage-power-sensor-breakout.pdf