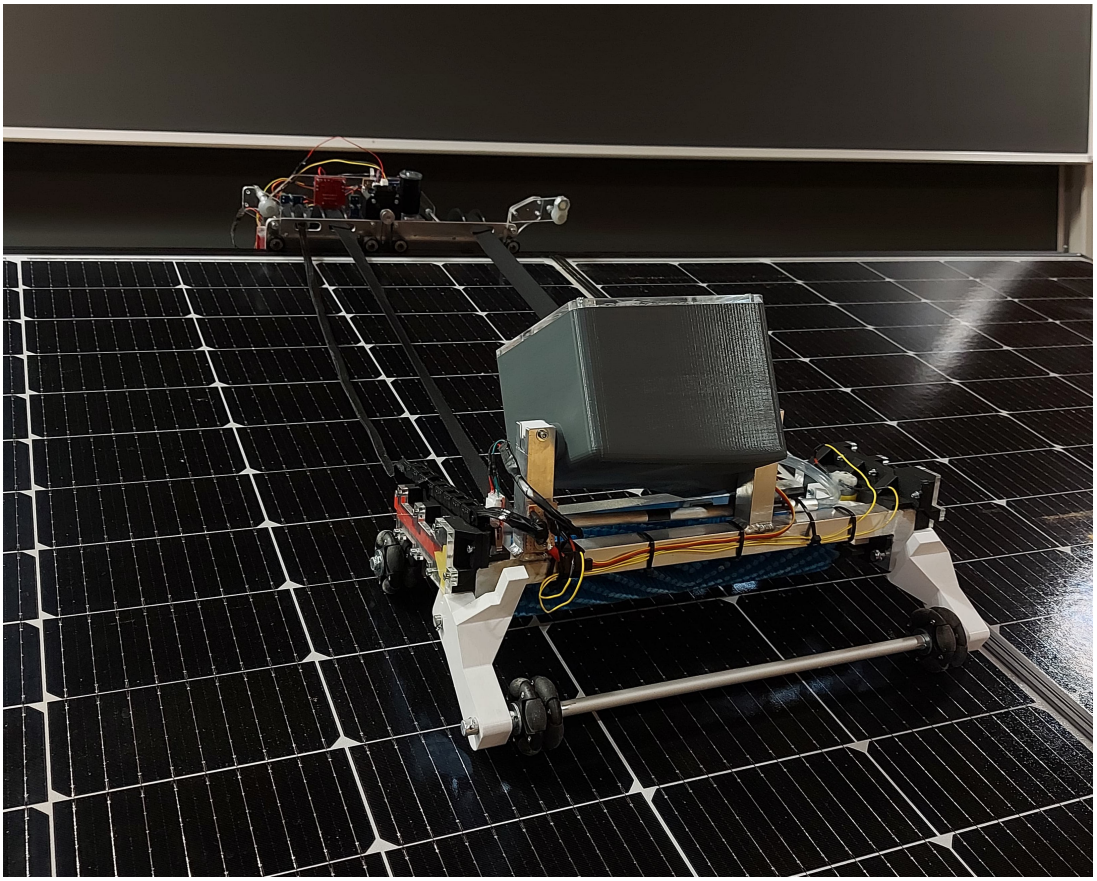
Product Design & Systems Engineering - Final Report

Autonomous solar panel cleaner - So'Clean



Group 1
Boesinger Garance
Cirillo Matteo
Depauw Blaise
Elmaleh Daniel
Huser Jérémie
Mignot Antonin

Teaching Assistant
Stathaki Chrysoula

Professors
Bellouard Yves
Charbon Edoardo

Swiss Federal Institute of Technology

January 8, 2024

Executive summary

Project Title: Autonomous Solar Panel Cleaner - So'Clean

- **Mission:** To provide a practical and efficient solution for maintaining energy efficiency in solar installations.
- **Vision:** To become a leading solution in solar panel maintenance.
- **Company Profile:** Developed at EPFL, integrating multidisciplinary expertise in engineering and design.

Purpose of the Product : The So'Clean robot is an autonomous solar panel cleaning system designed to enhance energy efficiency in photovoltaic (PV) installations, targeting small to medium-sized PV installations like domestic plants. It offers an efficient cleaning solution, operating autonomously and compatible with both new and existing installations.

- **Unique Proposition:** Autonomous cleaning system compatible with new and existing PV plants.
- **Competitive Advantage:** Efficient cleaning, affordability, cost-efficiency, easy to use and innovative design elements.
- **Development Status:** Prototype developed, constituting proof of concept for the design and demonstrating effective cleaning.

Technical Solutions : The final design of So'Clean involves a robotic structure moving a cleaning tool across solar panel surfaces, with a water-based scrubbing system. The 'cleaning curtain' concept features a carriage on a guide rail above the panels and a robot with omnidirectional wheels.

Market Positioning and Intellectual Property So'Clean targets individual households and small companies owning PV installations. It differentiates itself as an affordable and efficient solution. The project conducted a prior art search and identified potential for patenting the bistable mechanism and trademarking the product name and visual identity.

Financial & Market Analysis :

- **Market Opportunity:** Growing market segment of small to medium-sized solar plant owners.
- **Target Market:** Expanded from individual households to small companies in Switzerland/Europe.
- **Competitors:** Professional services or industrial-grade systems.
- **Pricing and Competition Analysis:** Estimated production cost of 200-300 CHF, selling price around 800 CHF. Installation costs estimated at 100-200 CHF. Competitive against manual and non-autonomous cleaners.
- **Cost-Efficiency Analysis:** In Ecublens, Switzerland, a 40 m^2 solar panel installation could save about 160 CHF annually with So'Clean, totaling 1600 CHF over 10 years. Locations like California may offer higher savings due to more sun and dust.

Team Effort and Work Structure : The project was developed by a diverse team of students at EPFL, using a mix of 3D printing, laser cutting, TIG welding, and soldering for the prototype. The team's approach to project management evolved from initial brainstorming to role-specific tasks within a horizontal hierarchy. Roles were distributed across various domains including mechanical design, intellectual property study, electronics, software architecture, and communication.

Concluding Remarks and Future Steps A SWOT analysis identified strengths in uniqueness and autonomy, with opportunities in market growth and government incentives. Despite challenges, the functional prototype demonstrates the capability in integrating mechanical, electrical, and software aspects. As a startup, So'Clean would need initial funding and additional personnel, specifically a manufacturing engineer, a marketing expert, and an accountant.

Contents

1 Product concept

1.1 General description

Context : With today's growing concerns for durability and sustainability, the need for efficient ways of producing green energy becomes apparent. One of them is solar energy, exploited through photovoltaic (PV) installations. While markets in this field are fast-growing around the world, a major problem arises when owning PV panels. Soiling, such as dust, pollen or bird feces naturally builds up on the panels and cause energy losses. Indeed, the shading of the PV cells affects the output of the entire installation, and maximum efficiency is only ensured by keeping panels clean. The goal of our product was hence to address these shortfalls by developing a system that could efficiently clean solar panels.

Features : Once installed, the product is entirely autonomous in its operation on the PV installation. It is designed to withstand the harsh outdoor environment and the cleaning is automatically carried out during the optimal time window for minimal energy use and best cleaning efficiency. With minimal water use, the system rinses, scrubs and dries solar panels and is able to efficiently prevent the build up of the soilants that most severely affect their energy output. These include pollen and dust cementation, animal feces, and small objects such as dead leaves and twigs. Note that snow is ignored as it has been shown to rapidly melt on PV panels and not cause major issues, according to Mr. B. Cuperly from Romande Energie. Lastly, the product has been designed to be particularly versatile so that it can not only be installed together with a new PV plant, but is also compatible to add onto an existing installation.

Specific use : The product is installed on a PV plant, connected to a power and water supply, setup up for the specific solar panels topology, and it then autonomously operates when needed. It is designed for use on small to medium sized PV installations such as a typical domestic plant on the inclined roof of a house. This choice is justified in the following.

1.2 Brief market analysis

Potential customers : Over their lifetime, any solar installation gets dirty, and accordingly, any owner is a potential customer. While cleaning solutions for large industrial solar farms already exist, most PV installations today, are owned by individual households or small companies that can not afford such sophisticated cleaners. These small to medium sized installations tremendously increases in number over the years, and this justifies the choice to target this specific kind of PV installations for the development of our product.
Potential customers are not only the owners of targeted PV plants but also the solar panels installation companies. *So'clean* is a dual-market organisation (both business to business and business to customer), targeting future solar installations, as well as currently existing ones.

Competitors : For targeted customers, existing solutions nowadays sum up to namely: calling a professional, climbing on your roof yourself, or buying a costly industrial grade cleaning system. All the options are expensive and unpractical or just outright dangerous. There is currently no efficient and affordable option for small to medium sized plants.

1.3 Unique selling points of your product

In light of the previous points, the product is envisioned to be completely autonomous, efficient, affordable, and compatible with both new and existing PV plants. It stands out as the first-of-its-kind autonomous solar panel cleaning robot, adapted for small to medium sized solar plants. It is a cost-effective analogue to industrial grade cleaning robots, that ensures consistent cleanliness of the solar panels. With no comparable solution currently available, the product is an investment that enables customer to save money by consistently keeping their solar panels clean. Our product democratizes maintenance of solar installations, making it the best choice for those seeking optimal performance and ease of use.

2 Technical solution

2.1 Technical and functional requirements

In light of the previously defined product concept, requirements are now set in order to guide the ideation phase. They reflect the stakeholder's needs and what we want to offer.
Notice: the product is referred to as *the solar panel cleaner* (SPC), and the terms *shall*, *should* are used for essential features (■) and nice to have features (□), respectively.

ID	Functional Requirements Description	Priority
F0	SPC shall be a system whose purpose is to clean soiling on panels of photovoltaics plants autonomously.	■
F1	During operation, the cleaning action of the SPC shall reach at least 95% of the solar panels surface after cleaning.	■
F2	SPC shall be adapted for plants up to 20kW, comprising panels close to standard size (1 x 1.7 m).	■
F3	SPC shall be compatible with new installation but also for its addition to existing plants.	■
F4	SPC and its cleaning methods shall not damage the panels' surface.	■
F6	SPC shall not obstruct solar panels except during the cleaning process.	■
F7	SPC shall be optimized to clean typical soilings on panels in Switzerland or similar regions. Snow is not considered as a soiling.	■
	Technical Requirements Description	
T1	SPC shall be able to clean a surface inclined from 15° up to 45°.	■
T2	SPC shall be able to operate at temperatures between 4°C and 35°C.	■
T3	SPC should be able to clean at a speed of at least 1 m^2/min.	□
T4	SPC should not use more than 1 l of water per 10 m^2 of cleaning.	□
T5	SPC prototype shall have a production cost of maximum 500 CHF.	■
T6	SPC prototype shall be constructed before December 15, 2023, and presented by December 22.	■
T7	SPC prototype shall be produced only once.	■
T8	SPC prototype should be constructible in EPFL workshops such as SPOT or SKIL.	□
T9	SPC prototype should weigh less than 5kg.	□
T10	SPC shall be reprogrammable via USB to allow software updates and parameter adjustments.	■
T11	SPC shall be designed to be easily repairable, each component can be changed without altering any other.	■
T12	SPC shall not use any detergent.	■
T13	SPC should be designed to minimize CO_2 emissions throughout its life cycle, including manufacturing, use, and disposal.	□
T14	SPC shall be handled without the risk of injury.	■
T15	SPC shall not fall from the PV installation during operation.	■

Table 1: Functional and technical requirements

The requirements in table 1 above were taken into account for the development of the prototype. Table 2 below shows additional requirements for the future final product. They are however only considered optional for the prototype.

ID	Optional Functional Requirements	Priority
O1	The total cost of acquiring and operating SPC shall be offset by energy efficiency gains within a period of 5 years, (costing less than 1000 CHF).	■
O2	SPC in its idle state, shall be able to withstand its environment. Be it all weather conditions and seasonal variations encountered in Switzerland including rain, snow or wind.	■
O3	SPC shall be a fully autonomous system that does not require any human control.	■
	Optional Technical Requirements	
O4	SPC shall withstand outdoor temperatures between -30°C and 50°C.	■
O5	SPC should have a maintenance interval of at least 3 years.	□
O6	SPC shall withstand low shocks up to 2g.	■
O7	SPC should emit a maximum of 65dB.	□
O8	SPC shall have a lifespan of at least 15 years.	■
O9	SPC should be designed to enable the recycling of 90% of its components at the end of its lifespan.	□
O10	SPC should have light indicators for signaling eventual malfunctions.	□

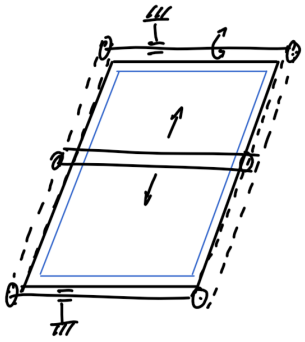
Table 2: Technical Requirements Table

2.2 Design phase: solutions explored

Early Considerations : To begin, and in light of the previous requirements, the problem at hand is first summarized to identify the main challenges of the system. The product to be developed is a solar panel cleaner that typically sits on a domestic PV plant on an inclined roofing. In the early ideation phase of the project, a plethora of solutions were explored. This ranging from the simple cleaning with sprinklers, mechanical vibrations or even electrostatic forces, to wilder solutions involving flying drones. The most promising concepts however, often shared common traits, that can be summed up by a few considerations for the general architecture of the product:

- **Robotic structure :** In light of requirement F3, the product has to be a standalone system, that can be added onto existing solar panels. This, combined with requirement F6, points quite naturally towards the use of a robotic structure for the SPC. To this end, the system's two main components would be **a kinematic structure** that moves a **cleaning tool** over the surface of the panels. In other words, the product would be a robot whose end-effector is some sort of cleaning equipment.
- **Cleaning action :** The main aspect of the product is the cleaning of the solar panels surface. It is thus essential to start by defining the cleaning equipment that will be used by the system. An easy and obvious choice is a water based scrubbing system which would consist of a wet brush, scrubbing the panel surface, and followed by a drying squeegee (see right of fig 5). While other solutions have been considered, testings and simulations have shown this first option to be the easiest and most efficient to implement. Further details are shown in appendix D.5.
- **End effector integration :** One last point to keep in mind is the connecting of the cleaning tool, at the end of the robotic structure, to both electrical and cleaning supplies. For instance, how to provide electrical power and water to the cleaning system.

With the above in mind, many solutions were explored and the most promising ones are presented here. These were later evaluated through a decision matrix to justify the final choice.



Panel squeegee is a mechanism inspired by the windshield wipers of a car. The system is mounted on the frame of a single solar panel and its cleaning tool that spans the panels width, is dragged up and down the surface. It is actuated *by-wire*, with motorized axles on top and bottom of the solar panel.
While very simple, this concept has to be installed on each panel of a PV plant. This means installing the system on existing plants is difficult as panels are not necessarily spaced out enough.

Figure 1: Panel Squeegee

Cleaning kart is a mobile cleaning robot that autonomously drives around on the panels surface while cleaning. This concept could be promising on a relatively flat roof, but with incline, it needs a safety tether. While very adaptable to many PV plant topologies, the projected complexity of the concept seems too big.

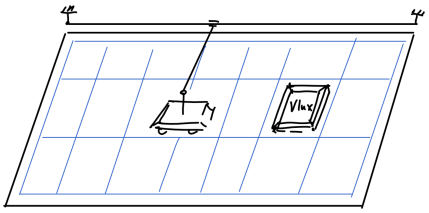


Figure 2: Cleaning kart

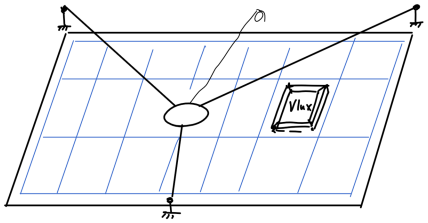


Figure 3: Spider cleaner

Spider cleaner is a concept inspired by cable suspended camera systems present sports stadiums. The cleaning tool is suspended by three motorized winches which can position it over the panel surface.
The concept works on any roof inclination but there were concerns regarding the end-effector integration. Supplying the tool with electrical power and cleaning supplies seems unpractical.

Cleaning Curtain is similar to the panel squeegee, but adapted to work on many panels. It consists of a cleaning tool that rests on the panels surface and that is suspended by straps to an overhead carriage. The latter is mounted on a guide rail and can move laterally in addition to being able to winch the cleaner up and down. The system is fairly simple in comparison to other options but does need an inclined roof to operate.

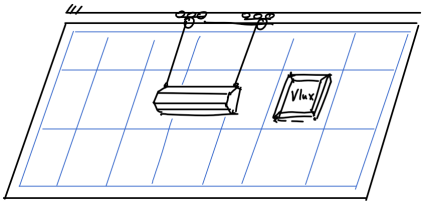


Figure 4: Cleaning curtain

Decision matrix : The previous concepts were evaluated in a decision matrix, shown in 3. Considering the fairly limited development time for the project, simplicity [S] was the main criterion for the selection, with a weight double the other. The other big selection criteria is efficiency [E], which here is the concept's ability to satisfy basic requirements. Additional selection criteria are the projected maintenance [M], cost [C], installation [I], and the originality [O] of the system.

Criteria	**E**	**S**	**M**	**C**	**I**	**O**	**Total**
Weight	2	2	1	1	1	1	
Cleaning curtain	0	0	0	0	0	0	**0**
Panel squeegee	-1	1	-1	-1	-2	-1	**-5**
Cleaning cart	-1	-2	-2	-2	0	1	**-9**
Spider cleaner	1	-1	0	0	-1	0	**-1**

Table 3: Decision matrix, supporting the choice of the cleaning curtain

2.3 Made technical choices

2.3.1 Selected design

As shown in the previous decision matrix, the cleaning curtain concept is adopted and figure 5 is a render of our first prototype, where the main components of the system are detailed:

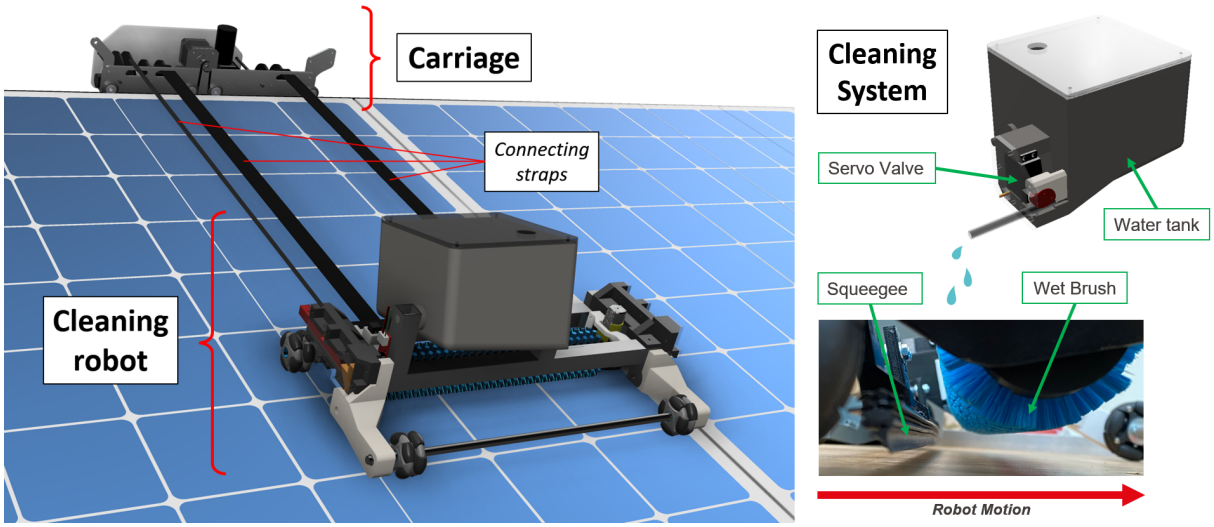


Figure 5: Overview of So'Clean system (left) and detailed cleaning system of the robot (right)

Kinematics : The carriage is sitting on a guide rail on the top row of the panels, and the cleaning robot rests on the panels surface, sitting on four 2 DOF (degrees of freedom) omnidirectional wheels. The two are linked together both mechanically and electrically by three connecting straps. Those can in turn be winched by the carriage thus raising and lowering the robot. As for lateral movements, the carriage can translate itself on its rail and drags the robot along. This works by having a toothed belt on the rail, which is coupled to a stepper motor on the carriage. Its rotation thus pulls on the belt, effectively translating the carriage on the rail (see fig.6).

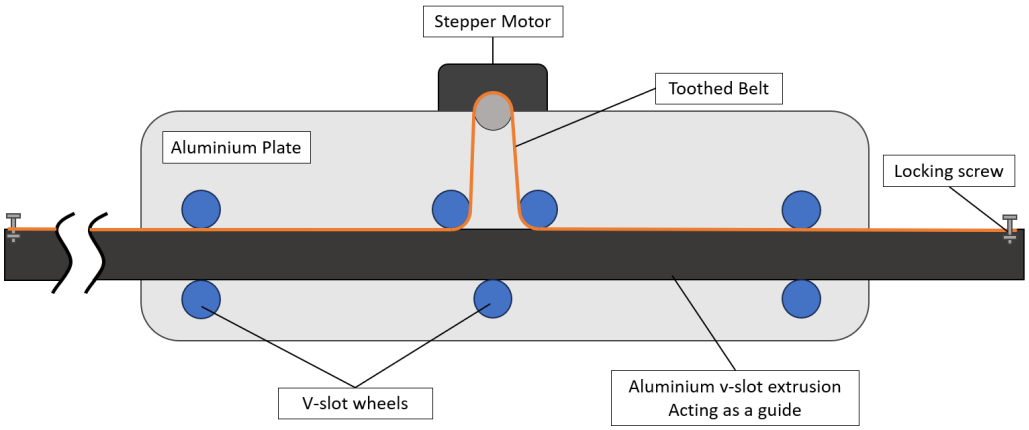


Figure 6: Carriage on its guide-rail system, that is based on v-slot profiles

Cleaning system : The equipment is shown on the right of figure 5. The robot is designed to clean while advancing and the cleaning system is accordingly made of a wet rotating brush, followed by a squeegee. The water is dispensed on the brush through a valve on a water tank. Notice that to regulate the pressure applied on the surface by the brush, it is mounted on spring loaded suspensions, as shown in appendix D.6.

Idling : On the top of a side of the solar panels is an idling station. This is a place where the system goes to for stand-by so as not to rest directly on the PV plant's surface in-between uses, and hinder efficiency. The idle station for the final product is also meant to shelter the system and is also equipped with an automatic tap that fills up the robot's water tank.

Bistable mechanism : The need for this feature will become apparent later, but briefly, it is a mechanism that allows the cleaning robot to lift its cleaning equipment off of the panel surface. The mechanism can be triggered (turned ON) when the robot is lowered all the way to the bottom edge of the panels, and it can be reset when the robot is raised all the way back up against the carriage. As stated before, this will be further detailed later on.

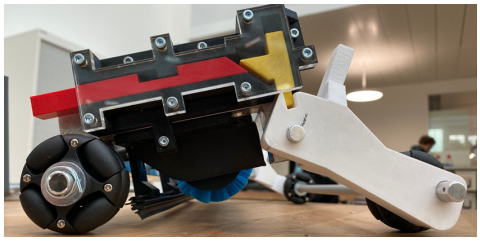


Figure 7: Triggered bistable mechanism

2.3.2 Concept of operation (ConOps)

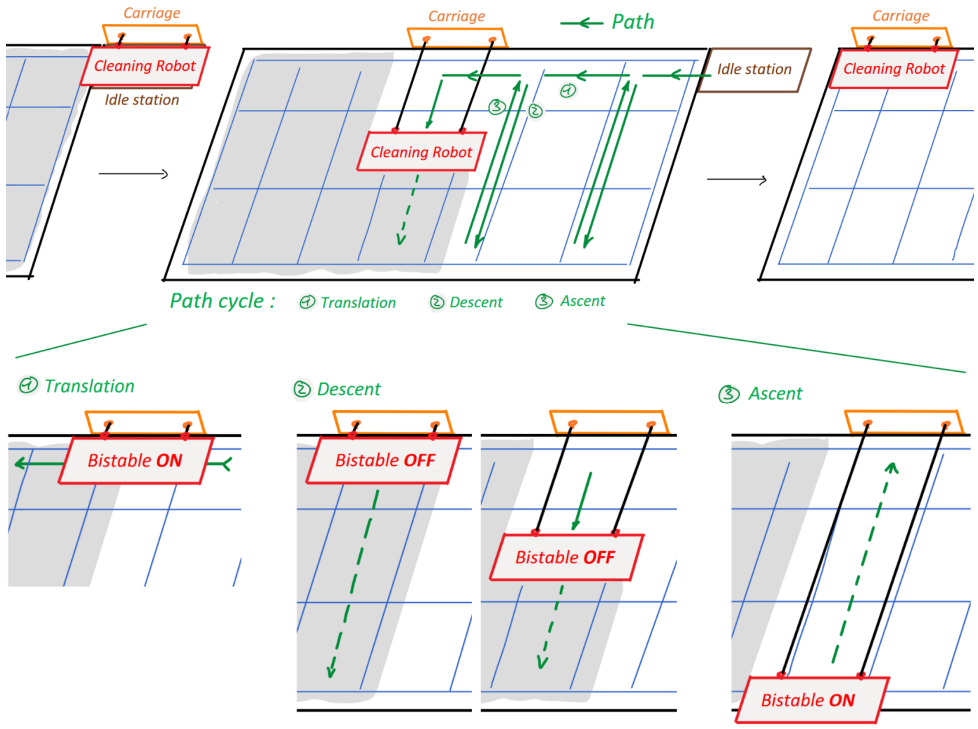


Figure 8: So'Clean Concept of Operations (ConOps)

While simple, the product's architecture imposes the cleaning robot to be raised close to the carriage for precise lateral movements. The ConOps in fig. 8 shows the entailed cleaning path of the robot is in green. The cleaning is broken down into cycles, each composed of three stages: **one cycle** = ① + ② + ③. Each stage is shown in the bottom of fig. 8 and detailed here:

① *Translation* : The carriage drags the robot over the next surface patch to be cleaned

② *Descent* : This is the stage where the cleaning is performed by the robot while being lowered by the carriage.

③ *Ascent* : Having cleaned during the descent, the robot is brought back up to the carriage, for the translation of the next cleaning cycle.

Such cycles are repeated until the whole surface of the plant is cleaned. At which point, the robot simply returns to its idling station for stand-by, until it is needed for another cleaning.

The same fig. 8 highlights the usage of the bistable mechanism during each cleaning stage. In the ascent ③ stage of each cycle, the robot is raised back to the carriage over the same surface patch that has just been cleaned. In order to avoid soiling the freshly cleaned surface, the robot trips the bistable mechanism while at the bottom of the panels, thus lifting its equipment off of the panels' surface for the ascent. The cleaning system is then lowered back onto the panels by resetting the bistable system, at the start of ②, before the start of the next cleaning descent.

2.3.3 Key elements in the design related to the unique selling points

An important aspect is the versatility of the product, and the way it can be easily installed on existing PV plants. Indeed, as further detailed in appendix D.7, setting up the robot only requires bolting the carriage's guide rail to the metal frame of the top row of solar panels. The system is then simply mounted on it, and after bolting the idling station to the side of the plant, and plugging a power and water supply, the system is operational.
This nicely transitions into the autonomy of the system, which is guaranteed by its package of sensors. With limit switches at both ends of the carriage's guide rail and on the bistable mechanism, the system can autonomously clean the plant without external input. Some further optimization for the final product could be done in the sensing of the environmental conditions. Indeed, in light of the requirements, while the product is meant to withstand harsh condition in its idling station, it is only meant to clean the panels in certain conditions. The product would thus need some sort of weather station to only schedule cleanings when there is no significant wind for instance.
Speaking of the idling station, the prototype did not implement the tap that refills the robot's water tank, this needs to be added onto the final product, as well as a general improvement of the station as to enable it to shelter the robot during standby. On the prototype, a simple aluminium plate was used for demonstration, on the final product on the other hand, some sort of box for storing the cleaner should be used as idling station.

2.3.4 Analysis & modelling of key elements and how they could be optimized

Kinematic model : The simplified kinematic graph of the system is shown on 9, where the carriage can translate along the rail on the panels with 1DOF. The robot is then simply attached to the carriage, which constrains only 1DOF. Lastly, the robot end lies on the panels' surface with 3DOF. The application of the Grübler-Kutzbach criterion yields 3 degrees of mobility, in contrast to the expected 2DOF of the system:

$$M = \sum_{i=0}^{k} f_i - 6(k - n + 1) = 3, \qquad DOF = 2 \qquad (1)$$

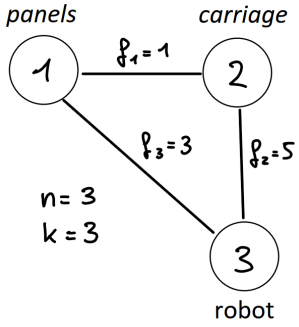


Figure 9: Kinematic graph

This is expected and can be explained by how the cleaning robot is unconstrained laterally on the surface of the panels. This is not a problem for the system as gravity along suffices to pull the robot downwards such that it sits straight under the carriage.

As stated above, this is only a simplified model to highlight the main degrees of freedom of the chain. More details could be provided on how each joint is implemented through various different parts, but no further insight is gained in that fashion.

Cleaning time : A rough model of the system was established in order to estimate the characteristics of critical components. Assuming the robots goes at a vertical speed $v = 0.1ms^{-1}$ and is $w = 0.3m$ wide, it is possible to calculate the cleaned surface per second $\Omega = v \cdot w = 0.03m^2s^{-1}$. With this, the time required to clean an installation of surface S is S/Ω. But as the robot needs to come back up after each cleaning descent, one complete cleaning takes $T_S = 2S/\Omega$. For a typical plant surface of $50m^2$, this yields around an 1 hour. Note however that this result is obtained using the parameters of the prototype, which is only meant to clean two panels, i.e. $3m^2$. This highlights the need to make the final product faster for a shorter cleaning time.

Water usage : Knowing that the brush of the system absorbs water and acts similarly to a sponge, there is no need for great amounts of water. It was thus decided to use 1 dL/m^2 as it was empirically confirmed that it suffices for this application. This means that for a 50 m^2 installation, 5 L of water are needed. Again, this suggests that the water tank could be further improved, as it can only hold 2L in the prototype version.

Winching motor : We will now compute the power needed for the DC (Direct Current) motor that lifts the robot up and down by winching the straps. As a worst case, the system was reduced to a simple motor, winching a mass vertically (see fig.28 in appendix). The mass is taken to be the robot's, and estimated at 5kg. Taking again the fact that our robot winches at a target speed of $0.1m/s$, the power needed to lift the robot is : $P = mg \cdot v \simeq 5W$. Note that friction was neglected as the robot is supposed to roll on the panels thus generating close to no force. The friction of the brush and squeegee against the panel is irrelevant as the bistable mechanism lifts the cleaning equipment before every ascent. With a safety factor of 2, the DC motor was bought with **10W** of mechanical power.

As DC motors are designed to have better efficiency at high speed and low torque, it was chosen to limit the diameter of the strap's spool to $D = 50mm$. Given our requirements, the angular speed of the motor should be by $n = \frac{60 \cdot v}{\pi D} \simeq 38$ rpm and its torque, $\tau > \frac{D}{2} \cdot F \simeq 1.25Nm \simeq 12.75kgf \cdot cm$. A 10W worm gear motor rated for those parameters was bought and it additionally features a worm drive gearbox, that ensures "infinite" locking moment. According to the datasheet [1], the motor is rated for rotation at 30 rpm with load of $32kgf \cdot cm$. The 10W of mechanical power are distributed at the desired speed and the motor is thus sufficient. The rated load of $32kgf \cdot cm$ is also at least two times higher than the previously calculated torque ensuring low current consumption on the system.

Cost-efficiency : Production of solar energy can vary drastically between locations. Using the calculator of [2] and inputting Ecublens as a location, the energy produced per year and surface unit on a 35° roof is 210 $kWhm^{-2}yr^{-1}$. Considering now a yield of 90% of the nominal energy output for a constantly clean panel (i.e. 10% losses in average due to soiling) it means that the potential for increasing energy output could be: 210(1-0.90) = 21 $kWhm^{-2}yr^{-1}$. At an average, electricity produced by a PV installation can be sold at around 20 $ctkWh^{-1}$ [3] thus allowing the owner to avoid a shortfall of 4.2 CHF per year, per square meter of solar panel. Given a typical plant of 40 m^2, this comes at around 168 CHF of savings per year. By defining a generous consumable budget of 8 CHF (including water and power consumption) per year for our product, we can conclude that the consumer can save around 160 CHF per year. According to [4] solar panels have typically a lifetime of twenty five years. Assuming our product is used for ten years, we obtain a total shortfall of 1600 CHF or 800 CHF for five years. This means that the production cost of the product should not exceed a heuristic of 200 CHF.

This paragraph highlights how the cost efficiency is not the best in a country such a Switzerland. California for example however, is much sunnier and solar panels produce more energy per year and surface. In addition to that, they also get dirtier due to less rain. The shortfall for such a plant is computed to be >500CHF per year in the same conditions, which is already more consequent, rendering our product more attractive.

2.4 Manufacturing choices and proposals for future production

2.4.1 Prototype fabrication

The prototype was built at the SPOT at EPFL. Various manufacturing process were used such as 3D printing, laser cutting, TIG (Tungsten Inert Gas) welding, soldering, along with the standard use of the lathe, drill and milling machines. As the part count is quite large, we will focus on the most interesting subsystems.

Carriage : The structural backbone for the mechanical elements on the carriage is an aluminium plate which was laser cut, with the DXF file shown in appendix D.8. Threaded holes were then tapped into it, to screw all other components on the plate. As shown on figure 10, the panels' side of the plate has seven small v-slot wheels that are part of the guide rail system that was already detailed in figure 6. On the other side, three spools, a DC motor, stepper motor, and finally the acrylic electronics board are screwed onto the plate. All the spacers, the spools and their mounts, and the supports for the motors are 3D printed, as it allows rapid prototyping at low costs for iterative designs and tests.

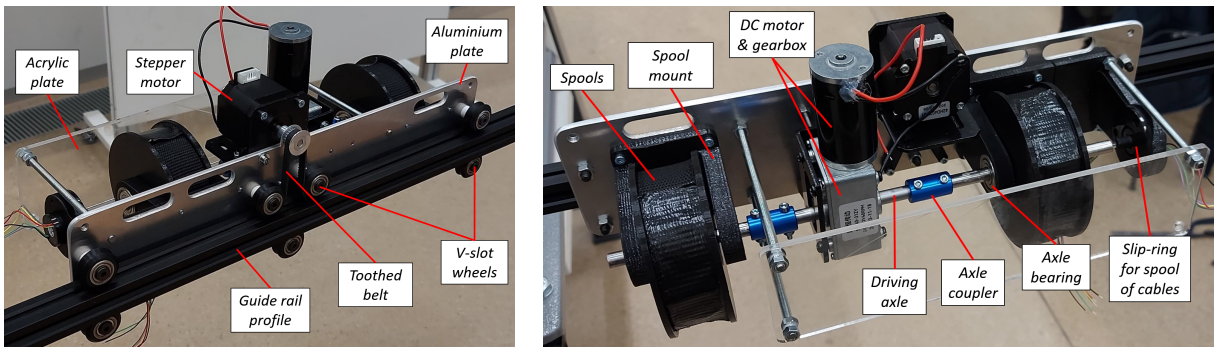


Figure 10: Detailed front and back of the carriage

In order to rotate the three spools to winch the straps that connect the carriage to the robot, we used a worm gear DC motor coupled to 8mm steel axles (see fig.10). Those components were all bought except for the axles that were machined. This entailed cutting them to size and drilling traversing holes in them to allow a screw to pass through and transmit the torque to the relevant component. As shown in figure 10, the axles are supported by four bearings embedded in the wheel mounts. The mechanical connecting straps that support the robot are 2m long and 25mm wide. They are made out of UV resistant polypropylene and are fixed to the spools using bolts.

Finally an acrylic plate is mounted at the back of the robot for the electronics, whose diagrams can be found in appendix 26 and 27. It was made using CO_2 laser cutting and acrylic scraps found in the recycling bin. The electronics platform is mounted offset to the aluminium plate using 120mm M5 rods. Again, see fig. 10.

One of the challenging points was to ensure that the robot's weight supported by mechanical straps attached to the carriage, and not in the electrical wires as this could otherwise damage them. On the other hand, without a little tension, cables do not wind around the spools correctly. A small tension was therefore still required and implemented by introducing compliance in the attachment of the electrical cables to the robot.

Cleaning robot : All components are mounted on a aluminium TIG-welded frame, shown in the left of figure 11. The wheels of the robot are mounted with plain bearings on axles that are also welded to the frame. The front wheels, are on a separate axles as part of the bistable mechanism. Wheels are secured axially with a washer and lockwire. A method that is used for every rotating axle on the robot and shown in the example of wheels, in the right of figure 11.

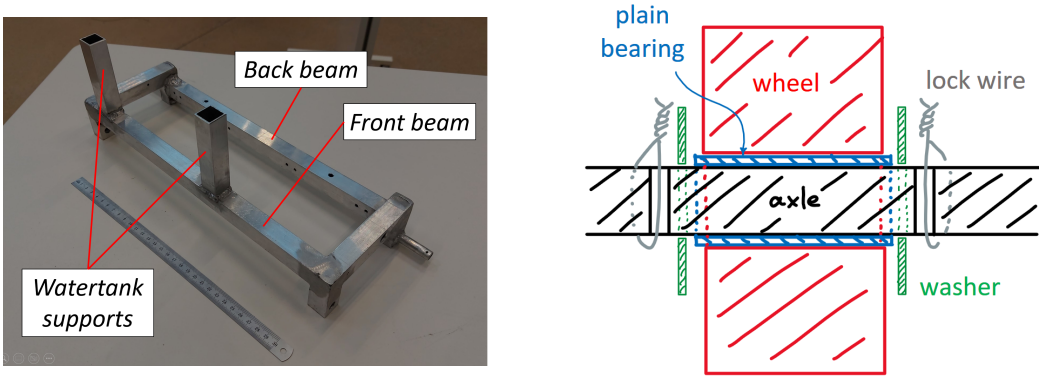


Figure 11: Robot's frame (left), schematic axial mounting of wheels (right)

The bistable mechanism shown in figure 12 is mainly made of 3D printed PETG parts. The use of different colors and transparent acrylic plate, lets us see the working principle. The black supports of the mechanism are clamped around the frame and in white, is the arm that supports the front wheel of the robot. It is free to rotate on a steel axis in the frame, such that, when reaching the end of the solar panels, the front wheels dangle off of the surface effectively rotating the white arm, such that the yellow spring loaded pin slides in the locking slot. At this point,

when the robot is winched back up in the ascent ③ stage, the bursh and squeegee are elevated. The mechanism is reset by the carriage pulling the robot hard against itself and pressing on the red reset cam, which lifts the yellow pin. The mechanism is mirrored to the other side of the robot too, as shown in further illustrations in appendix D.9.

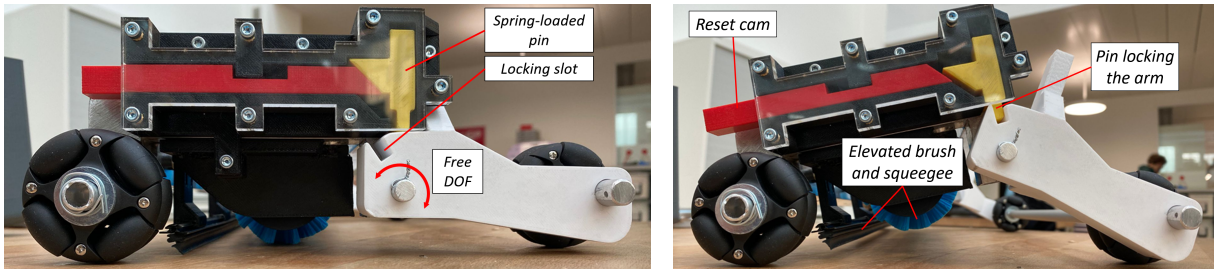


Figure 12: Bistable mechanism in OFF state (left) and ON state (right)

The brush and its motor are mounted on a spring-loaded suspension system shown in appendix D.6. This allows to have a vertical degree of freedom with 5mm clearance, so that the brush is always in contact with the panels. The contact pressure can be adjusted by inserting up to 16 helicoidal springs in parallel, in the system. All parts are 3D printed as they need to fit with the rotating brush and its DC motor, rendering them quite complex. The squeegee is a 300mm rubber wiper, mounted on the back beam of the robot's frame with adjustable 3D printed mounts, (shown in appendix D.6), for precise adjustment of the height of the wiper.

The water tank, shown in the right of figure 5, is also made with 3D printed PETG (Polyethylene terephthalate glycol). Although not soluble, this material is not ideal for water retention due to its porosity, but it is sufficient for the prototype. The bottom of the tank is made in such a way that all the water flows towards the outlet. It is mounted on the watertank supports of the frame and close shut with an acrylic lid that is laser-cut and engraved with the So'Clean logo. This lid has a hole to enable refilling of the tank at the idling station, with the future integration of the automatic tap. During the cleaning, a servo-valve is opened allowing water to flow into a perforated aluminum pipe that dispenses it on the rotating brush. The valve, visible in figure 5, is a simple 3D printed cam that is mounted on a servomotor. This enables the servo to pinch the plastic tubing through which the water passes, effectively blocking the flow.

With all mechanical elements assembled, electronic components can now be installed. A stop switch, determining the bistable mechanism's position, is mounted on the robot. Wires from the actuators are securely fastened along the frame using zip ties, and all wires are connected to a small electronic board. Both the cleaner robot and the carriage are grounded to prevent any potential safety issues.

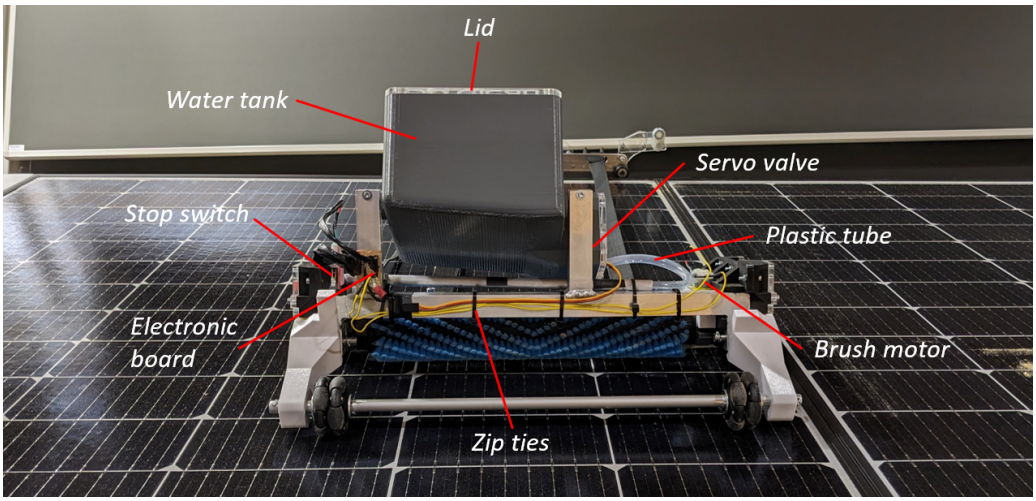


Figure 13: View of the cleaner robot from the back

Electronics : The choice of components for the electronics was based on the needs of the prototype and the availability of the components at the SPOT. One important choice is the one of the motors. The first one that translates the carriage laterally is a stepper motor (NEMA 17) which ensures an accurate and reproducible movement. As previously mentioned in 2.3.4, the motor chosen for the winch is a DC motor. We chose this particular one because it is powerful, affordable and features a worm gear which prevents the spools from unwinding as the motor is not powered. As it is a DC motor, the robot's vertical position could not be monitored. Hence, we added a distance sensor on the carriage that stops the motor once it is close enough to the carriage, before translating laterally. The brush motor was also a DC motor since it was the simplest choice regarding the cleaning system.

As for the other electrical components used, for ease of programming and access to the different components, we used an Arduino Mega as microcontroller. It was mounted onto the acrylic plate mentioned, along with all other electronic components. We used two motor drivers, one for the stepper motor specifically and one for the two DC motors. The drivers thus act as relays between the 12V power supply and the signals sent by the MCU. More components were used to enhance the functionalities of our prototype, namely current sensors for the DC motors. They allow us to stop the motor in case it consumes too much power, for instance if the robot is stuck. An RGB LED was also integrated to indicate operating states and problems. Additionally, a rain sensor was added to detect rain and avoid using additional water. Finally, as mentioned in the mechanical part, several limit switches were also mounted at each end of the rail and on the bistable mechanism. A more detailed electric diagram and components list can be found in appendix C and figure 29.

As for the software integration, we implemented test codes and modules all throughout the prototyping phase using the Arduino IDE Software. These modules were used to shape the final working code which can be consulted in appendix D.13.

2.4.2 Final product's estimated costs for manufacturing

For the final product all the parts should be reevaluated and redesigned to be improved in terms of working principle, cost and manufacturing.

Plastic parts: For the prototype, 3D printed PETG was used for many components. Injection molding using PP would be a suitable option for large-scale production due to its high tooling cost and rapid production rate. Compression molding could be more suitable for medium-scale production, while 3D printing remains a feasible choice for smaller-scale production.

Metallic parts: The only metallic parts are the frame and the axles. TIG welded aluminium beam used for the prototype could also be suitable for the final product. Alternatively, constructing the frame with polymers such as PP is another available option.

COTS parts: Some of the parts are standardized and it's usually cheaper to buy them on the market instead of manufacturing them ourselves. This is applicable to screws, omniwheels, straps, V-slot rails and wheels, rubber wipers, springs, pipes, motors, brushes, wires, spools, and possibly the water tank.

Electronics parts: All the electronics for the control of the system can be gathered on a PCB and produced by a subcontractor.

We can distinguish the cost of the product in three different categories: COTS components, parts to manufacture and assembly.

For the COTS components, one can estimate that the price could be reduced by at least a factor of 2 compared to what we used for the prototype. This reduction can be achieved through simplification, bulk ordering, and finding the best quality-price ratio. The brush is a good example of savings that can be done, we spent 58 CHF for it but we can certainly find one around 10 CHF. Therefore, considering our expenses for the prototype, we can estimate the cost

to be around 100 CHF. This estimate takes into account a sufficient quality that ensures the specifications are maintained, especially in terms of lifespan.

The source of cost for manufactured parts can be generally divided in three categories: raw materials, energy & consumables for processing and engineering & administration. The manufacturing cost formula (eq. 2) can be used to estimate the cost of manufactured parts [5].

One can consider that the parts are made out of PP and a price of 1.3 CHF/kg (see appendix D.10 and the following price parameters for injection molding: basic overhead 10 CHF/hour, injection molding time of 6 seconds, a mold cost of 40'000 CHF and a machine cost of 200'000 CHF (with a depreciation time of 10 years). We consider that 4 molds are used to manufacture all the parts. We can make the assumption that the cost is only due to the molds price because of the medium scale production. If we consider that molds never deteriorate and an amount of 5000 robots produced, the cost becomes 160'000/5000 = 32 CHF. For the metallic parts we can add an amount of 15 CHF.

$$C = \underbrace{\left(\frac{m}{1-f}\right) C_{rm}}_{\text{Material Cost}} + \underbrace{\frac{1}{\dot{n}}\left[\frac{1}{L}\cdot\frac{C_c}{t_{wo}} + \dot{C_{oh}}\right]}_{\text{Gross overhead}} + \underbrace{\left(\frac{C_t}{n}\right)\left\{1 + E\left(\frac{n}{n_t}\right)\right\}}_{\text{Dedicated cost}} \tag{2}$$

For the assembly, we can assume that the robot requires 4 hours of assembly time. Labor costs vary significantly depending on the country. These costs are detailed in Appendix 43. According to data from 2013, the hourly labor rate is approximately 54 CHF/hour in Switzerland, 36 CHF/hour in France, and less than 5 CHF/hour in China. Consequently, the assembly of the robot would cost approximately 220 CHF if conducted in Switzerland, 150 CHF in France, and less than 20 CHF in China. The assembly would certainly be done in China due to the difference of cost compared to Switzerland.

The total can then be estimated by summing the cost of COTS components, manufactured parts and assembly cost: 100 + 47 + 20 = **167 CHF**. Taking a margin of 20% the estimated price is 200 CHF. Note that eventual shipping price are taken into account in the margin.

2.4.3 Quality analysis and control

For quality analysis and control, the following points can be take in account.

Performance Testing : Conduct thorough performance tests to ensure that the robot cleaner meets or exceeds technical requirements. This aims to evaluate its cleaning efficiency.

Durability and Reliability Checks : Implement durability and reliability tests to assess the robot's ability to withstand continuous operation over an extended period. This involves testing various components, such as motors, sensors and bistable mechanism.

Sensor Calibration and Accuracy : Calibrate IR sensor to maintain accuracy in detecting the robot distance. Implement checks to verify the precision of sensor data and its correlation with the robot's actions.

Experience Testing : Conduct usability testing to assess the effectiveness of the overall user experience.

Error Handling and Recovery Testing : Simulate various error scenarios, such as getting stuck or encountering obstacles, to evaluate the robot's ability to handle and recover from such situations autonomously.

Noise Level Measurements : Measure and assess the noise levels produced during operation to ensure compliance with acceptable standards.

By incorporating these quality control measures, we aim to deliver a reliable and efficient cleaner.

3 Intellectual property analysis

3.1 Prior art search

There exists several patents on products that look similar to ours, and we will consider the most relevant ones, found on patent search engine *espacenet*, recommended by Dr. C. Schott [6].

JP2015150492A SOLAR PANEL CLEANER : The first one featuring a moving cleaning device guided by a linear rail that looks similar to our product is the patent JP2015150492A SOLAR PANEL CLEANER. This patent design is close to ours, but the biggest differences are that not only is the moving cleaning part conveyed by two rails (instead of just one for our product), but also the patent doesn't mention any use of water. Moreover there seems to be no way to lift the brush from the panel after cleaning.

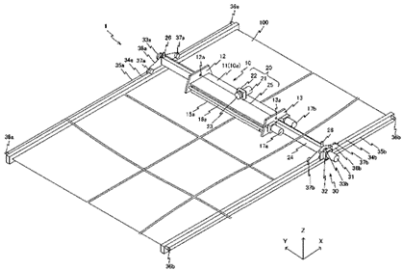


Figure 14: JP2015150492A [7]

CN209715830U Solar cell panel cleaning robot : Another patented product that has many similarities with ours is CN209715830U Solar cell panel cleaning robot shown in figure 15. This one has obvious similarities with our product, such as the water tank or the cleaning technique using liquid and a rotating cleaning brush. Unlike ours, this one is not autonomous, it cannot move sideways, and cannot lift the brush from the panel either.

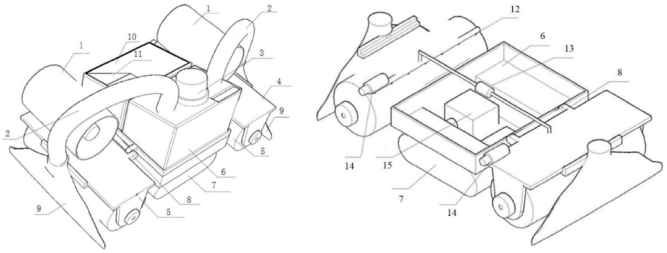


Figure 15: CN209715830U [8]

US2014109334A1 SOLAR PANEL CLEANING SYSTEM : The last patented product we are going to present is US2014109334A1 SOLAR PANEL CLEANING SYSTEM. The main common point of this cleaning system and ours is its topology : a track (rail), a cleaning carriage (robot) and a conveying carriage that moves the cleaning carriage from one row to another row. This product might be the most similar to ours, but this solution is not suitable for a domestic PV installation.

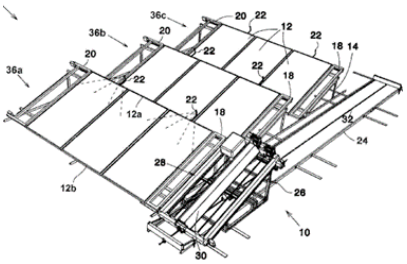


Figure 16: US2014109334A1 [9]

Existing commercialised solutions : Another point to address in this IP study is all the professional solar panel cleaning robots that are already on the market. Among many other similar ones, stands the SolarCleano M1 [10]. It is quite a bulky robot, which needs to be linked to a water supply and piloted by a professional, just like all the other similar cleaning robots. It is thus not very adapted to a domestic use. There is additional literature on water-free cleaning solutions [11], but none has been fully commercialized yet.

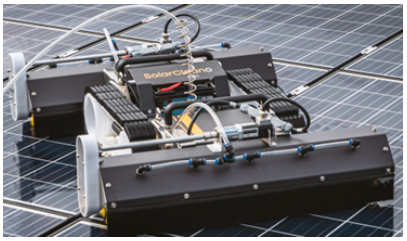


Figure 17: SolarCleano M1 [10]

3.2 Patent search

Provided the description of our product, three main categories of the International Patent Classification (IPC) fit our product well :

- **B08B1/002** : "Cleaning by methods involving the use of tools, brushes, or analogous members - brush".
- **Y02E10/50** : "Energy generation through renewable energy sources - Photovoltaic [PV] energy" considering that the selling point of our project is to save energy by cleaning the solar panels.
- **H02S40/10** : "Components or accessories in combination with PV modules, not provided for in groups - Cleaning arrangements" which makes a link between both previous ones.

We can also consider two other categories that have a direct link to our product, this time towards the domestic use of our product :

- **A47L11/38** : "Machines for cleaning floors, carpets, furniture, walls, or wall coverings - Machines, specially adapted for cleaning walls, ceilings, roofs, or the like".
- **B08B3/04** : "Cleaning by methods involving the use or presence of liquid or steam - Cleaning involving contact with liquid" justified by the use of water to clean the solar panels.

Looking at patents belonging to these main categories, and as we saw in the Prior art search, there isn't any autonomous cleaning robot, thus there won't be any problem that may prevent us from commercializing our product. We know that the most innovative part of our system is our bistable mechanism, but no patent category comes close. According to us, the most similar existing mechanism is the bistable mechanism found on pens. **B43K24/08** : "Mechanisms for selecting, projecting, retracting or locking writing units - operated by push-buttons", that regroups the retractable and blocking positions of a ballpoint pen. But once again, patents related to this category shall not prevent us to commercialize our robot since it's a long way from our solution.

3.3 Discussion on opportunities for IP and possible IP strategy

Among a variety of intellectual property strategies, two could be considered for our product, namely, the trademark and the patent. Copyrights and trade secrets are not suitable because firstly, the product does not fall in the category of authorship and secondly, it does not contain anything secret that can not be reverse engineered. The mechanical principles we employ are readily deconstructible and replicable, and our command algorithm can be easily reproduced without direct observation. We have not developed any confidential proprietary materials yet.

Figure 18: So'Clean logo and tagline

Trademark would be a choice for our visual identity, name, and tagline, shown in figure 18 above.

As far as existing solutions for cleaning solar panels are concerned, our product is already innovative in its bare concept. The system is unique in its architecture exploiting the slope of the house's roofs, and its combination of cleaning robot and conveying rail. It is the only autonomous cleaning robot adapted to houses or (almost) any roof installation. Patents could be used both as ground for economical negotiations and to protect our product from competitors. While our entire system can be patented, our most valuable innovation is the bistable mechanism, which allows us to lift the brush and squeegee off of the panels when not cleaning. The working principle, was already detailed in section 2.4.1 and below is a first draft of a possible patent claim.

Possible patent claim

A bistable mechanism for use in a solar panel cleaning device, the mechanism comprising: a means for selectively elevating and lowering a cleaning apparatus in relation to the surface of the solar panels; wherein the mechanism operates autonomously and is driven primarily by gravitational forces, and is configured to be activated at predetermined points during the cleaning cycle to prevent re-soiling of the solar panels. This elevation system enables the cleaning device to lift off from and make contact with the solar panels without manual intervention, optimizing the cleaning process and ensuring the cleanliness of the recently cleaned surfaces.

We could also think to patent other features of our product such as the mechanism to adjust the pressure of the brush, but it's far too similar to car suspensions to be patented, or even the self made electric cable system made with small cables and duct tape between the robot and the carriage, but the idea came from Jérémie, who used a similar process during his military service which means there is nothing really new either.

4 Aspects of a pre-Business plan

4.1 Market

As a first step of our product development, we target the Swiss/European market of individuals/small companies that own or plan to install, a domestic PV system. Switzerland is a fast-growing market for such installation. The International Energy Agency [12], counts already 200'000 PV installations, and with an additional 25'000 more being installed per year (see fig.19). In general, nearly 80% of new installations are on residential areas, which is the targeted segment. Our solution's ease of installation and the universal requirement for solar panel cleaning contribute to this vast market potential. Political incentives through preferential tariff to encourage investment in renewable energies are increasing in Switzerland, encouraging individuals, companies and public structures to invest in PV installations. In a second development step, as mentioned in 2.3.4, we wish to expand to all sunnier countries in the world where soiling shortfalls on solar panels are more significant. Our unique product would not have any direct competition, except for professionals that offer one-off washes, such as Karcher's iSolar [13], or SolarCleano M1 [10].

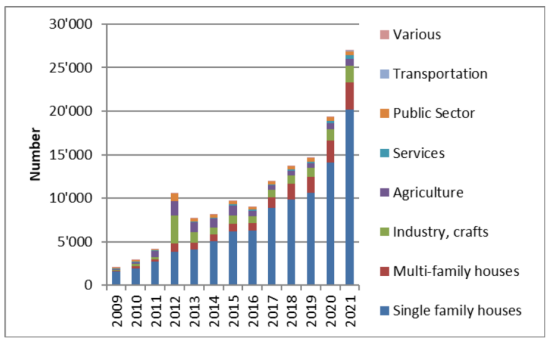


Figure 19: Number of PV systems installed per sector in Switzerland, from [12]

4.2 Strategy towards commercialization

Our initial approach focuses on B2B marketing, targeting solar panel companies. By addressing their specific needs for efficient and cost-effective maintenance, we not only provide direct value but also aim to leverage these partnerships for mutual promotion. We intend to encourage these businesses to become advocates for our autonomous solar panel cleaning robot, essentially turning them into ambassadors who can testify to the financial gains and sustainability benefits experienced firsthand. This aligns with our strategy of establishing credibility within the market, and will then enable us to target private customers directly. This will enable a natural development of an additional B2C model, where owners of existing plants will be able to purchase the cleaning system as a compatible add-on to their PV system.

This dual strategy, fosters partnerships for organic promotion and positions our product as a compelling choice for individuals or businesses looking to enhance their plants' efficiency while contributing to sustainability goals.

4.3 So'Clean organization

For a successful business plan and company, So'Clean needs a well-organized, balanced team with complementary skills. Essential to this project is a reliable design and prototype, supported by substantial plans and schemes to demonstrate feasibility for future investors. Our team requires proficient mechanical, electrical, and manufacturing engineers for this task. Additionally, a strong market analysis and effective communication strategy are crucial for targeting the right market segment. To ensure project efficiency and adherence to deadlines, a team leader is needed to coordinate these diverse functions. Our team's eclectic composition provides a solid foundation for appropriate work division.

4.4 Planning

Starting as a startup, So'Clean needs initial funding with investors and additional personnel, specifically a manufacturing engineer, a marketing expert, and an accountant. The team will

refine the prototype for industrial manufacturing and market it to solar panel installers, highlighting its financial benefits and efficiency. A large-scale canvassing campaign for solar panel installers is also to plan. Once the final product is ready, it will undergo extensive testing before being sold.

4.5 SWOT analysis

SWOT analysis involves examining the product's Strengths, Weaknesses, Opportunities, and Threats. This strategic planning technique helps understanding internal and external factors that can impact our product's success and plan for future actions.

Strengths	Uniqueness : No exisitng similar solutions in the market.
	Autonomy : The device functions with no need of user input.
	Versatility : Compatible with new and existing PV plants.
Weaknesses	Limited adaptability : May not be compatible with all plant topologies.
	Installation : Could require professionals for installing the system.
	Maintenance : Such requirements are not fully addressed.
Opportunities	Government incentives : Utilizing feed-in tariffs encouraging customers.
	Market growth : Capitalizing on the increase in new solar installations.
	Socioeconomic trends : Benefitting from public support for sustainability.
	Untapped market : Individual consumers is a less competitive segment.
Threats	"Do-Nothing" scenario : Customers considers natural rain as sufficient.
	Cost-benefit analysis : Costs do not align favorably with offered savings.
	DIY preference : Individuals are willing to manually clean plants.

4.6 Pricing and competition analysis

Pricing At the end of the prototyping phase, the budget we invested in our So'Clean prototype was 408,36CHF (see D.3 in appendix). We purchased many articles (electronic components wasted, budget for testing parts, reprinted parts etc.) and did not benefit from lot prices available for a company with a certain production rate. With growing production, we will subcontract the mechanical parts production and assembly, resulting in reduced costs for raw materials, parts, and assembly. We must then make a fair margin for our company, but also keep in mind the user will also have to pay for the installation. With all those factors taken into account and the dimensioning of point 2.3.4 'Cost efficiency' section in mind, we estimate a price of 200CHF at minimum (maximum 300) for the production cost of our product. Hence, with a margin of 3x this price for our company, we arrive at roughly 800CHF. As for the installation cost, the tariff of a PV plant installer is approximately 40CHF per hour in Switzerland [14], so depending on the time needed for installation we can add another 100 to 200CHF in, rounding our price up to 1000CHF. This is just an estimate as installations costs certainly need to be further discussed with the installers.

Competition As for competitors, we do not have much elements of comparison as the solution we propose is quite unique (refer to 3 IP study section). The price of a Karcher's iSolar double brush [13] is over 1000CHF, for a device the user has to manipulate himself, so our price, for a totally autonomous system, seems quite fair in comparison. But again, one could simply use soap and a brush or wait for the rain, which costs nothing. Checking Wondersuisse products [15], all PV brush cleaners (pilotable, so not autonomous like ours) are over 3000CHF and bigger robots (also pilotable and adapted to larger surfaces) are over 10000CHF. As we did not study a complete business plan for the moment, we have no detailed and exact calculations of the selling price of our product, only rough estimations, but we nevertheless are quite sure our product would be affordable comparing to other options available and largely paid off over time.

5 Project management

5.1 General strategy

The organization of the team continuously evolved throughout the project. During the first phase, every member was dedicated to coming up with new ideas and brainstorming. However, once the preliminary design was chosen, roles were assigned to everyone in a horizontal hierarchy, defined in table 4 below. The advantage of a horizontal hierarchy is that everyone is at the same level and it allows one more person to work on this challenging implementation. The disadvantage is that members need to be more proactive by defining their own tasks by themselves and that there is no manager to keep an overview of the progress. (CAD = Computer Aided Design)

	Blaise	**Antonin**	**Garance**	**Daniel**	**Jérémie**	**Matteo**
Design	IP Study	Electronic diagram	Choice of electronics components	Software Architecture	Mechanical CAD (Carriage)	Mechanical CAD (Robot)
Implementation	Communication Final presentation	Electronics hardware (soldering)	Software developing	Software developing	Manufacturing (Carriage)	Manufacturing (Robot)

Table 4: Summary of role distribution

5.2 Work breakdown structure

The following figure 20 represents an overview of the Work Breakdown Structure (WBS) used during this project. As the complete WBS is very large, it is dissected in the appendix for better readability D.11. For this project, the development of our product as a part of this course was split in three divisions: the technical division with all the product systems, the integration division, and finally the management division.

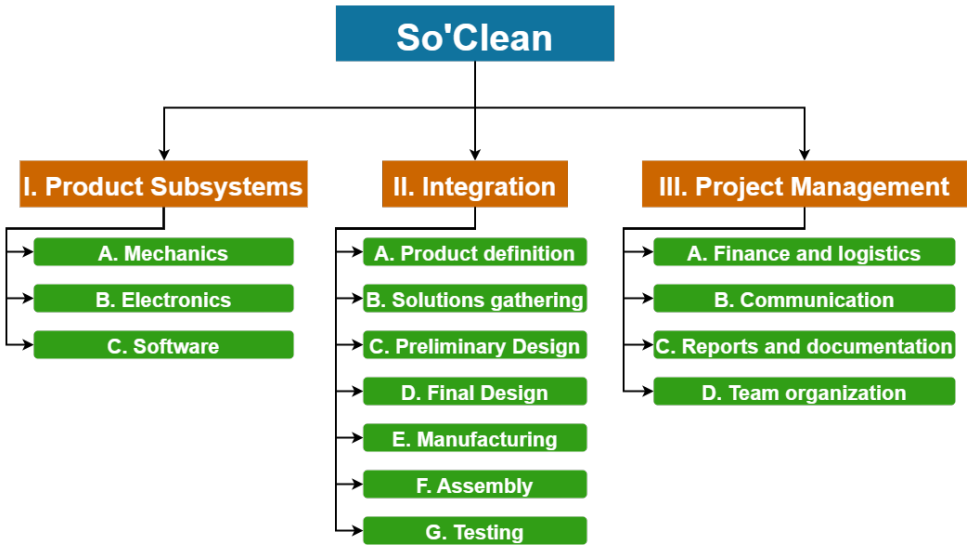


Figure 20: Overview of the WBS for So'Clean prototype development

For the **technical division**, it was chosen to create three systems : the mechanical , the electronic, and the software system. Each system was then further divided according to its properties. The leaves of the diagrams in the technical division need to follow a basic implementation process consisting of three tasks : design, manufacture and assembly.

The mechanical system was separated in three objects forming their own subsystems (figure 44). Then, the electronics system was also split in three functions forming their own subsystems (figure 45). Finally the software was separated in different steps of implementations (figure 46).

The **integration division** (figure 47) gathers an overview of the timeline of this project and describes the procedure of the development of the product.

Finally the **management division** (figure 48) defines all the management tasks that are needed for this project to work. The subdivisions are self-explanatory and represent different poles that, together, form a small company. It should be noted that the team organization leaves do not represent tasks but rather responsibilities.

5.3 Stakeholder analysis

Table 5 below, outlines key stakeholders involved in the deployment and use of our solar panel cleaning solution in Switzerland, highlighting their unique needs and how our product aligns with these requirements.

Stakeholders	**Needs**	**Actions**
Individuals/Small companies owning relevant PV plants	• Effective cleaning of solar panels • Adaptability to various installations	Provide a user-friendly interface
Swiss government, electrical grid	• Support for solar panel installations • Enhance power generation	Collaborate on clean energy incentives
Ourselves (the company)	• Design a market-driven product • Efficient market penetration	Develop targeted marketing strategies
PV systems installers	• Client satisfaction • Ease of installation	Offer training for installers and opportunities to gain commissions by recommending/selling our product

Table 5: Stakeholder Analysis for Solar Panel Cleaning Solution in Switzerland [1]

Overview: Figure 21 below, presents a comprehensive stakeholder analysis relevant to our product's market introduction in Switzerland. Stakeholders are categorized into six distinct groups, each color-coded for immediate recognition:

- **Institutions and Academic Guides (Brown):** This group comprises educational and regulatory bodies, including our professors, who establish the framework within which our project operates.
- **Project Team (Blue):** Representing our internal group, this includes all team members and the collective efforts of our project.
- **End Users (Yellow):** Highlighted are the direct beneficiaries of our product, encompassing individuals and businesses looking for solar panel solutions.
- **PV System Installers (Orange):** These are our primary commercial partners responsible for the installation of solar panels and direct purchasers of our product.
- **Swiss Electrical Grid (Green):** Encompassing the entire energy infrastructure of Switzerland, this group includes entities involved in power distribution and the broader energy market.
- **Manufacturing and Supply Partners (Gray):** Indicated here are the Discovery Learning Laboratories (DLL) facilities for prototype development and the suppliers providing us with Commercial Off-The-Shelf (COTS) components.

Each category is strategically identified so as to not only reflect their roles and importance but also to guide our engagement and communication strategies.

[1] Stakeholder needs were identified through a combination of surveys, expert consultations, and market analysis.

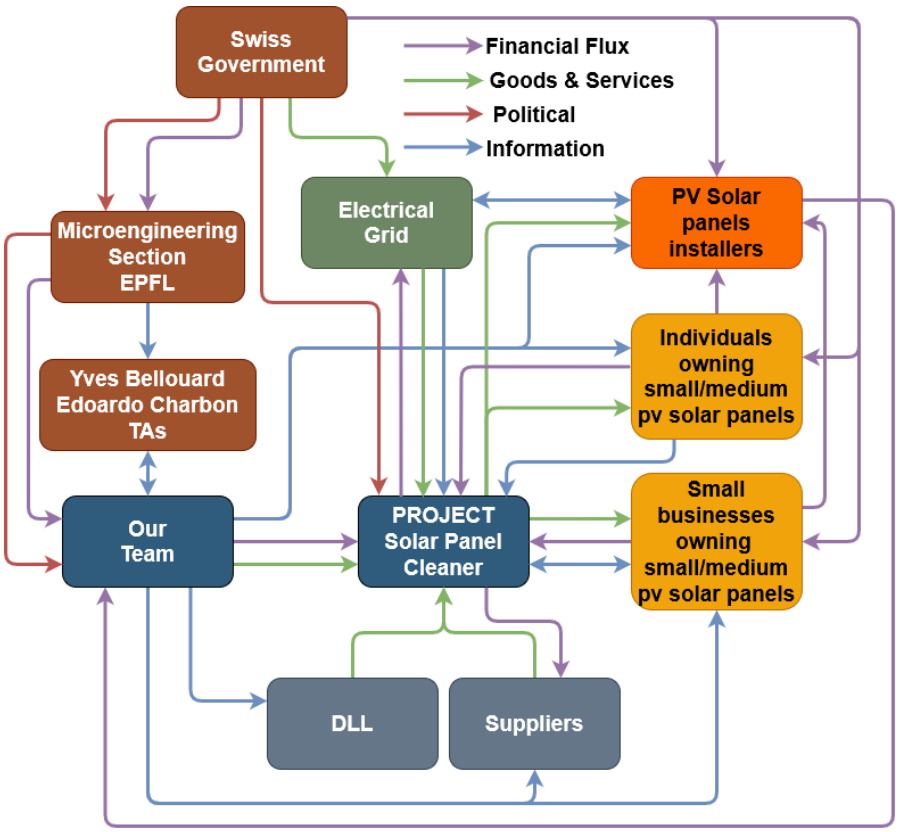


Figure 21: Stakeholder Analysis Chart

5.4 Gantt chart

This subsection will focus on the planning and the time management for this project.

5.4.1 Initial plan

The following figure 22 represents an overview of the Gantt chart for this project. As the expanded form takes too much space, it can be consulted in the appendix D.12 [2]. The approach for planning followed the organization laid out by the course including research, conception, implementation and finally prototyping.

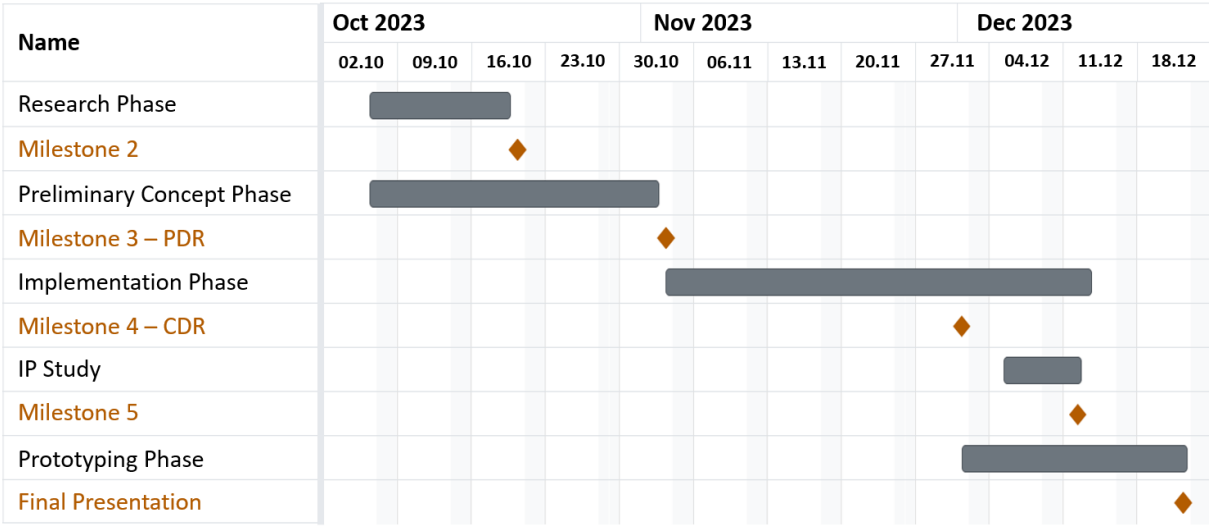


Figure 22: Overview of the Gantt chart

[2]Some dependencies are not represented on the full Gantt chart as it would overload the image

5.4.2 Deviations

Until milestone 4, the planning was thought to be followed quite closely. However, once the prototyping phase started, it was realised that the design of some mechanical sub-systems were not ready for integration and therefor caused a delay in the manufacturing and assembly of the device. As there were only two team members assigned to the mechanical system, it was not possible to speed up the process. For that reason the schedule, had to be shifted, which squeezed the testing and validation (ID 32) period to almost none. The software development (ID 31) was greatly impacted, forcing the use of a simplified code for the final demonstration.

5.5 Self reflection

As stated in the organization of the team 5.1, no manager was appointed for this project. In time, however, it was realised that the unforeseen complexity of the project made it difficult to handle the organization in this way. Members of the team (Jérémie during implementation and Antonin during manufacturing) therefore took the responsibility to act as a managers, which visibly improved efficiency.

An issue happened during the last days of the project concerning the product functionality. Indeed, once the system was loaded with water, it was not balanced anymore resulting in the system to be simply not usable. Fortunately, due to a good choice of cleaning system (brush and squeegee) the system was able to efficiently clean dust without water. The problem was thus mitigated by removing this additional feature of the prototype.

For the brush system, an electric DC motor was presented by a member of the group. This motor however had no identification thus making it hard to integrate in the system. After some experimental tests that were performed slightly late, it was realised that the motor speed was way too high (even by using a gearbox). It was thus chosen to use a different motor, directly available at the SPOT. However, this motor had little torque and a cheap plastic gearbox. Therefore, the suspension mechanism was not loaded with any springs to reduce the amount of friction generated by the rotation of the brush on the panel and thus reducing the torque that was applied on the new DC motor.

5.6 Work repartition

Antonin For this project, I had the opportunity to work on the electronic side and management. While I already had a bit of experience with electronics, it was still challenging to design and assemble such a system in this short amount of time. Having previous experience in soldering, I made several connector and "perf boards" solderings trying to make it easy to modify/fix and yet robust enough for a demonstration. During the whole semester I also contributed for management/logistics creating and editing spreadsheets for electrical components, budget/expenses, planning. I also filled up the minute reports and made a substantial part of the WBS.

I helped and supported all members of the team through their tasks, it was a good exercise to learn how to collaborate with a broad diversity of teammates.

Daniel I participated in the initial brainstorming sessions, contributing ideas that helped shape the project's direction. My role in the stakeholder analysis, alongside Garance, helped defining the stakeholder interests and align with them.

Technically, I was involved in developing the software's Work Breakdown Structure (WBS) as part of the overarching plan led by Antonin. I also played a role in designing electronic schematics and diagrams, collaborating with Antonin in this effort. In the realm of software development, I was responsible for coding the various software modules, with support from Garance, and I was involved in the assembly of the electronic components.

During the testing phase, in collaboration with Antonin and Garance, I focused on ensuring the functionality and reliability of both the software and electronic components, making sure they

fulfilled the project's requirements.

Garance During this project, I had the opportunity to work both on the electronic and software parts, alongside with Daniel and Antonin. At the beginning of the project, I participated, as every team member, at proposing ideas, and was in charge, together with Blaise, to establish a market analysis, and, together with Daniel, to establish the stakeholder analysis.

Once the solution was chosen, I was in charge of selecting the electronical components and their characteristics for our system. My role was, together with Daniel, to buy them, make measures, and test them separately to create test codes and modules for the different parts of the system.

Once the different components were mounted and wired, I made tests together with the mechanical team to make sure both the electronical and mechanical parts were working well together. Then, at the end of the project, together with Daniel, we worked on the software to produce a working code for the demonstration.

Jérémie During this semester, my main focus was on the mechanical aspect of the project. In the initial phase, I used my past experiences and tools from the EPFL Rocket Team to establish some system engineering elements, such as team organization, the Gantt chart, and the budget. I also played a role in constructing the WBS.

I actively participated in exploring ideas and mechanical solutions. I proposed and presented the solution we finally chose, and also key mechanical solutions like the bistable mechanism and brush suspensions. I was in charge with the help of Matteo of the carriage's design and its manufacture. I also contributed to designing and manufacturing some parts of the cleaner robot.

My past experiences with CAD design, mechanical workshop, 3D printing, and such, were very helpful for the project. Working with Matteo, who is involved in a different association (EPFL Racing Team), provided a great opportunity to share knowledge and learn from each other. I also collaborated with others team members, especially to define interfaces for subsystems.

Blaise At the beginning of the semester, like all the other members of the group, I was fully involved in the initial discussions on the design and the imagination of the project final solution.

Then, unlike some of my peers who brought specific skills to the table, my catalog of hard skills was more limited. However, I embraced the challenge by identifying where I could contribute most effectively. My primary responsibility centered around external communication, by talking to actors such as Romande Energie, the PV-lab, and CSEM, which enabled us to learn more about the subject, and also to obtain the solar panels we used for the demonstration.

During the semester, I facilitated discussions and coordinated various presentations, ensuring our slides reflected the group's meticulous design work. I also did the work on Intellectual Property and the risk analysis. Finally as the project progressed, I took on a supporting role at the DLL, assisting my colleagues with various manual tasks.

Matteo In the initial phase of the project, apart from brainstorming, I spent some time evaluating the feasibility of the dielectrophoretic cleaning concept from appendix D.5. Once our minds were set on the curtain cleaner, my role has been mostly technical and focused on the mechanics, in close collaboration with Jérémie.

In the main part of the project, I was responsible for the robot cleaner for the prototype. This entailed designing and interfacing various subsystems such as the chassis, bistable mechanisms, wheels, water system, etc. I designed using CAD software after systematically sharing my ideas with coaches of SPOT who provided important feedback on the feasibility of the concepts. During the integration phase, I started by assisting Jérémie in the manufacturing of his design of the carriage, only to then start producing the robot cleaner with his reciprocated help.

Outside this main responsibility, I also worked on the actuation of the system, by selecting the motors of the carriage as well as its guide rail and the attachment onto the PV plant.

6 Conclusions

Our project, part of the product design and system engineering course, exemplifies the multidisciplinary effort in developing a cohesive product idea and prototype. We successfully integrated mechanical, electrical, and software components, each contributing crucially to the overall design and functionality of the product. The project sharpened our skills in general management, product development, and system engineering, which proved crucial in navigating the complexities of the project. Our iterative and agile product development approach enabled us to adapt and refine our design through continuous testing and feedback.

After many hours of manufacturing, assembling, coding, and fixing, we managed to produce a convincing first demonstration showing that our system could not only clean dust off the solar panels but it could also operate smoothly without human interference.

Some aspects of this project could however be improved. The management could have been handled better if we had appointed a team leader. Furthermore, some mechanical issues already mentioned in 5.5 could have been avoided along with some minor electronics incidents.

For a commercial product, various aspects, both mechanical and electrical, need improvement. Each component should be able to withstand external weather conditions. An improved water system has to be designed, issues mentioned in 5.5 have to be resolved, and the bistable mechanism can be significantly optimized. Additionally, a robust and reliable integrated electronics system including various additional sensors, has to be implemented. Then the software has to be updated accordingly.

Despite the success of our demonstration, we do not intend to create a company out of this project. A market seems to be there but it is hard to know if PV installations owners are ready to integrate this product on their roof. Moreover, we still have concerns about the safety of this robot and its durability since it is an outdoor device.

In conclusion, this project ended successfully and the whole team is proud of the results. Although this project was rather challenging, we effectively cooperated using each member's strength and differences to create this innovative product. The valuable lessons learned from overcoming the challenges in this project will undoubtedly contribute to our engineering experience and enhance our capabilities for future endeavors.

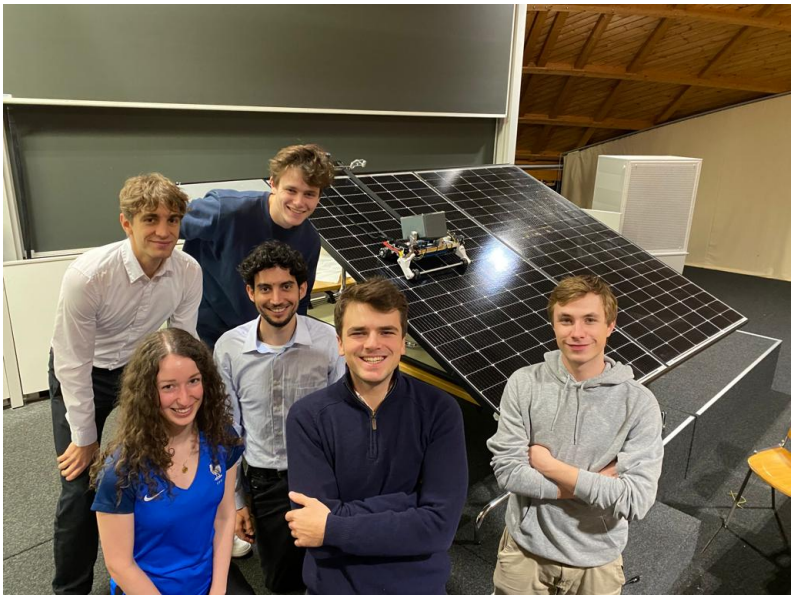


Figure 23: Team picture - (from left to right and top to bottom) Jérémie, Garance, Matteo, Daniel, Blaise, Antonin

Acknowledgements

We would like to extend our sincere thanks to our assistant Chrysoula Stathaki, who lent us consistent support and advices throughout the whole project, and to both the professors Dr. Yves Bellouard and Dr. Edoardo Charbon for their precious advices. We also address many thanks to the DLL coaches and assistants who gave us guidances, hints and support through the prototyping phase. Moreover, we would also like to express our gratitude to Mrs. Delphine Petri along with the CSEM staff, who took the time to hear about our project, and who provided us with two solar panels, without which our prototype could not have been complete. We would also like to thank Dr. Olivier Bernard from the Galatea lab who helped us transport the solar panels from Neuchâtel to the EPFL Campus. Another thank you goes to Mr B. Cuperly from Romande Energie who provided us with good insights about solar energy and gave us a tour of the PV installations of EPFL. Finally, we would like to thank the Microengineering section and EPFL for giving us the opportunity, the structure and the budget to realize such an insightful project.

7 Bibliography

[1] Datasheet worm gear motor. https://nfpshop.com/wp-content/uploads/2022/02/24V-Dc-Worm-Gear-Motor-5840-31Zy-Spec.pdf. Accessed: 2024-01-06.

[2] Suisse energie calculator. https://www.suisseenergie.ch/tools/calculateur-solaire/. Accessed: 2024-01-06.

[3] Pr Tobias Schmidt. Patchwork of issues limits solar expansion. Technical report, ETHZ, 2023. Accessed: 2024-01-06, at https://ethz.ch/en/news-and-events/eth-news/news/2023/02/press-release-patchwork-of-issues-limits-solar-expansion.html.

[4] Electrosuisse. Énergie solaire – évidemment photovoltaïque : technique et infrastructure. Technical report, Electrosuisse, 2019. Accessed: 2024-01-06, at https://www.electrosuisse.ch/wp-content/uploads/2019/10/Energie-solaire_Web.pdf.

[5] Yves Bellouard. Manufacturing technologies course. MICRO-301.

[6] Espacenet website. https://worldwide.espacenet.com/. Accessed: 2024-01-07.

[7] TAITOH TRADING CO LTD. Jp2015150492a. https://worldwide.espacenet.com/patent/search/family/053893323/publication/JP2015150492A?q=pn%3DJP2015150492A.

[8] SINOMACH ZHUHAI ROBOTICS ENG CO LTD. Cn209715830u. https://worldwide.espacenet.com/patent/search/family/068676624/publication/CN209715830U?q=pn%3DCN209715830U.

[9] SARAF MOSHE [IL]. Us2014109334a1. https://worldwide.espacenet.com/patent/search/family/047633774/publication/US2014109334A1?q=pn%3DUS2014109334A1.

[10] Official solarcleano m1 webpage. https://solarcleano.com/product/mini-solar-panel-cleaning-robot-m1.

[11] Petri Delphine Faes Antonin et al. Field test and electrode optimization of electrodynamic cleaning systems for solar panels. *Progress in Photovoltaics: Research and Applications*, 27:1020–1033, 2019.

[12] L. Lagay L. Bloch, L. Perret. National survey report on pv power applications in switzerland 2021. Technical report, International Energy Agency - PVPS, 2022.

[13] Kärcher isolar webpage. https://www.kaercher.com/ch-fr/professional/nettoyeurs-haute-pression/isolar-nettoyage-efficace-des-panneaux-solaires-et-augmentation
html.

[14] Solar energy system installer salary in switzerland. https://www.erieri.com/salary/job/solar-energy-system-installer/switzerland.

[15] Photovoltaic panel cleaner products available on wondersuisse.ch. https://wondersuisse.ch/categorie-produit/brosses/.

8 Appendix

A Initial project idea

At this stage early stage of the course, our exploration primarily focused on understanding the problem rather than delving into detailed solutions. However, we had begun to form some initial concepts. Our primary idea was centered on developing a handheld tool to assist in smoking cessation. This tool was designed to support a gradual reduction in nicotine consumption, helping individuals on their journey to quit smoking. Key features included:

- **Adjustable Dosing Mechanism:** This allows for personalized nicotine reduction, tailored to the user's needs.
- **Motivational Assistance:** The tool would offer rewards for reaching milestones, providing encouragement and support.
- **Substitution Strategy:** It would suggest alternatives to manage cravings effectively.

In addition to its primary purpose, the tool's design emphasized flexibility for on-the-go use, ensuring support for users at all times during their quitting journey. Furthermore, we recognized the potential for adapting this tool to address other forms of addictive behaviors, such as excessive phone usage, anxiety management, and gaming addiction, highlighting its versatile applications.

B Technical drawings

As explained in the report, most components for the system are 3D printed, and technical drawings thus are of little interest. Nevertheless, below are presented the drawings of the aluminum plate and the cleaner robot to provide an overview of the dimensions.

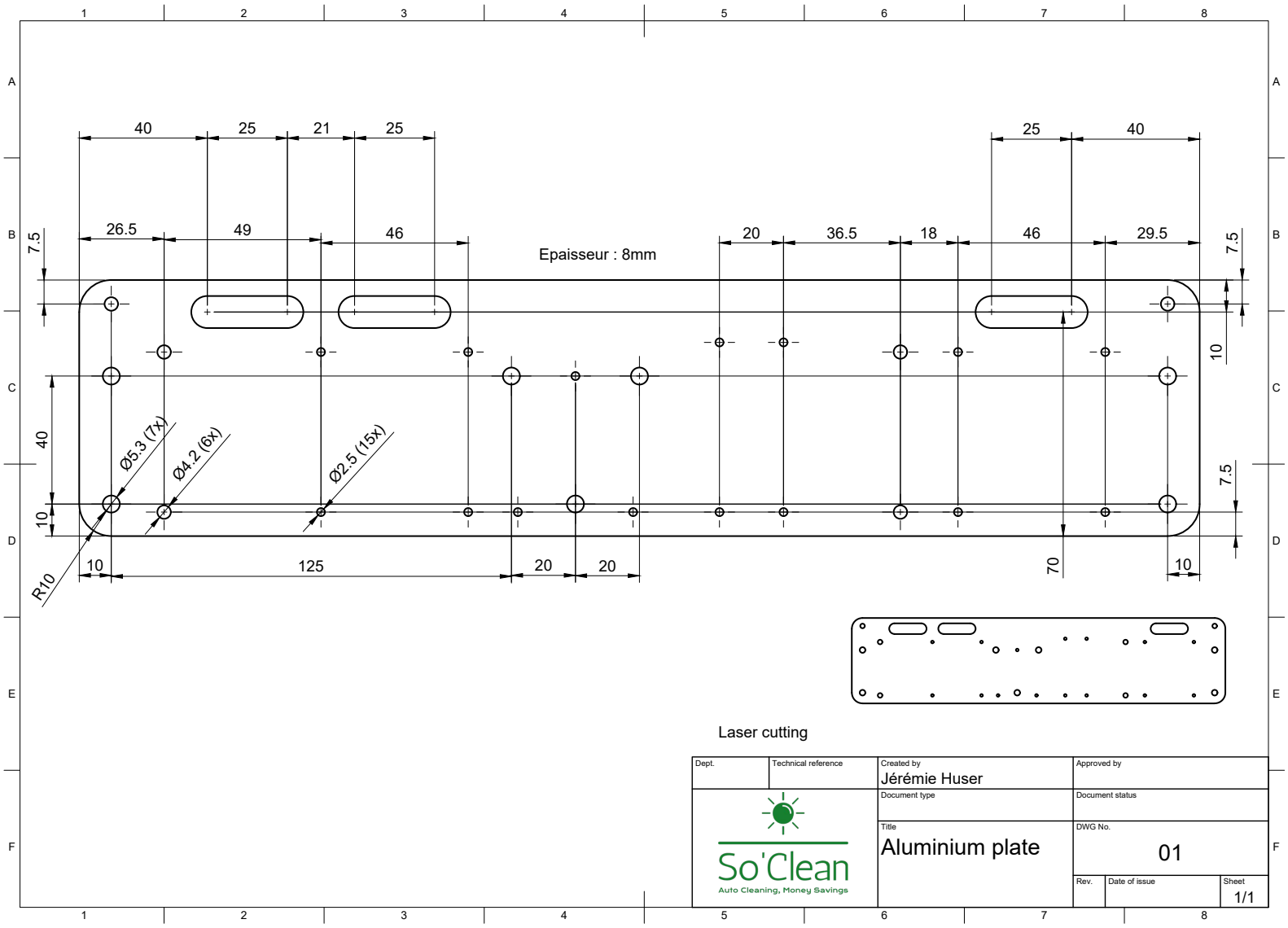


Figure 24: Aluminium plate drawing

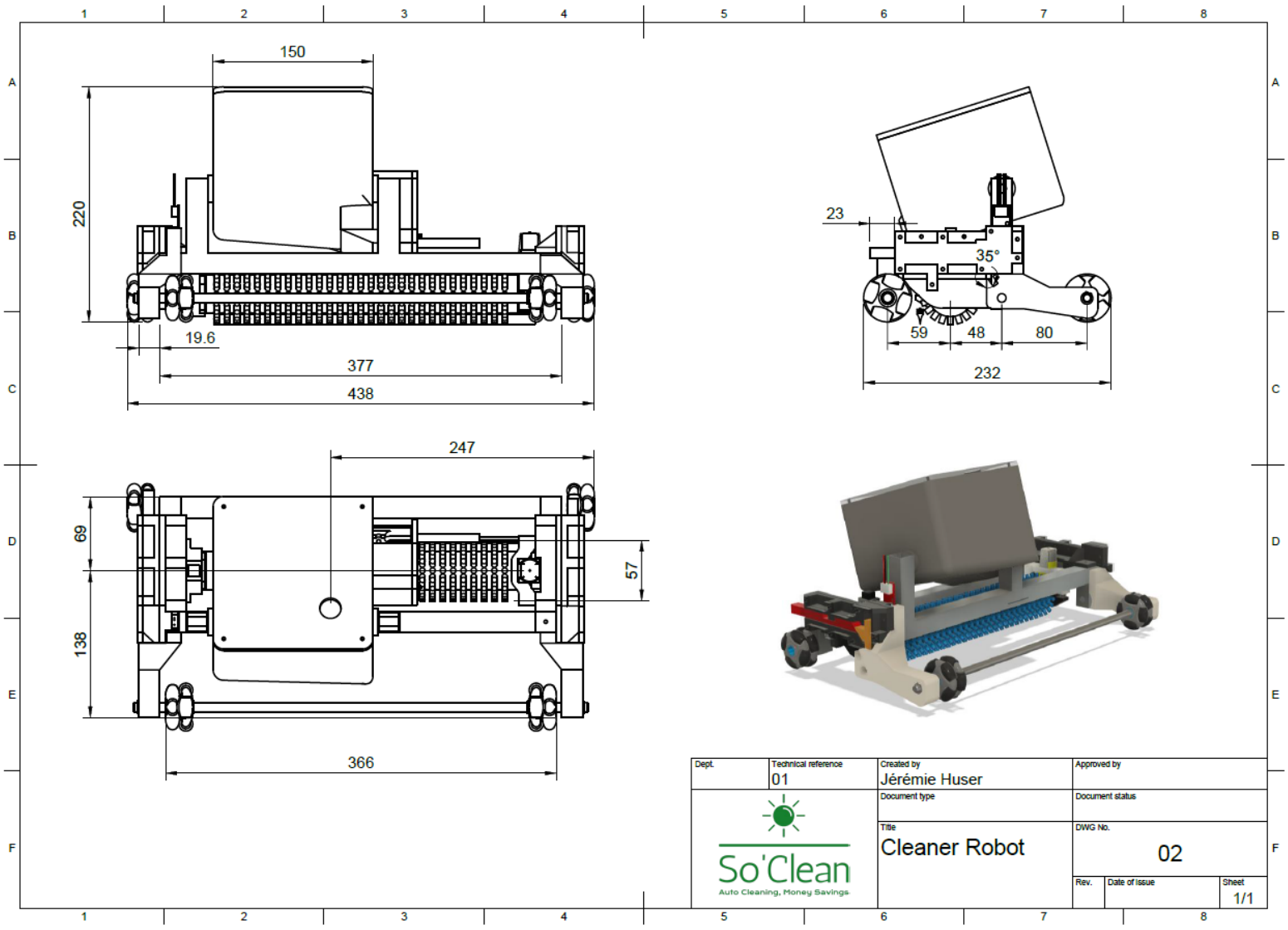


Figure 25: Cleaner robot drawing

C Circuit diagrams

In Figure 26, the electrical components of our system are depicted in a simplified manner. The diagram illustrates two separate parts: the carriage, housing most of the electronics, and the robot, equipped with essential electronics. The slip ring serves as the electronic interface, facilitating the connection between these two segments.

The system's electronic components include a microcontroller (Arduino Mega), various sensors and actuators, drivers, and an external power supply. It operates on power sources of 5 (supplied by the Arduino) and 12 volts (supplied the externally), with a common ground established for both the entire system and the individual carriage and robot components.

The actuators consist of a DC gearbox motor and a stepper motor located on the carriage, a DC motor for the brush on the robot, and an RGB LED on the carriage. Sensor-wise, the system includes an IR distance sensor on the robot, limit switches (two on the carriage and one on the robot), a rain sensor on the robot, and two current sensors on the carriage. As for the drivers, there is one L298N driver for the DC motors and one DRV8825 driver for the stepper motor.

The diagram also indicates typical current consumption and supply voltages for power lines and protocols or sampling method on the data lines.

For a detailed view, see the complete and rigorous electronic diagram of the system afterwards is shown in figure 27.

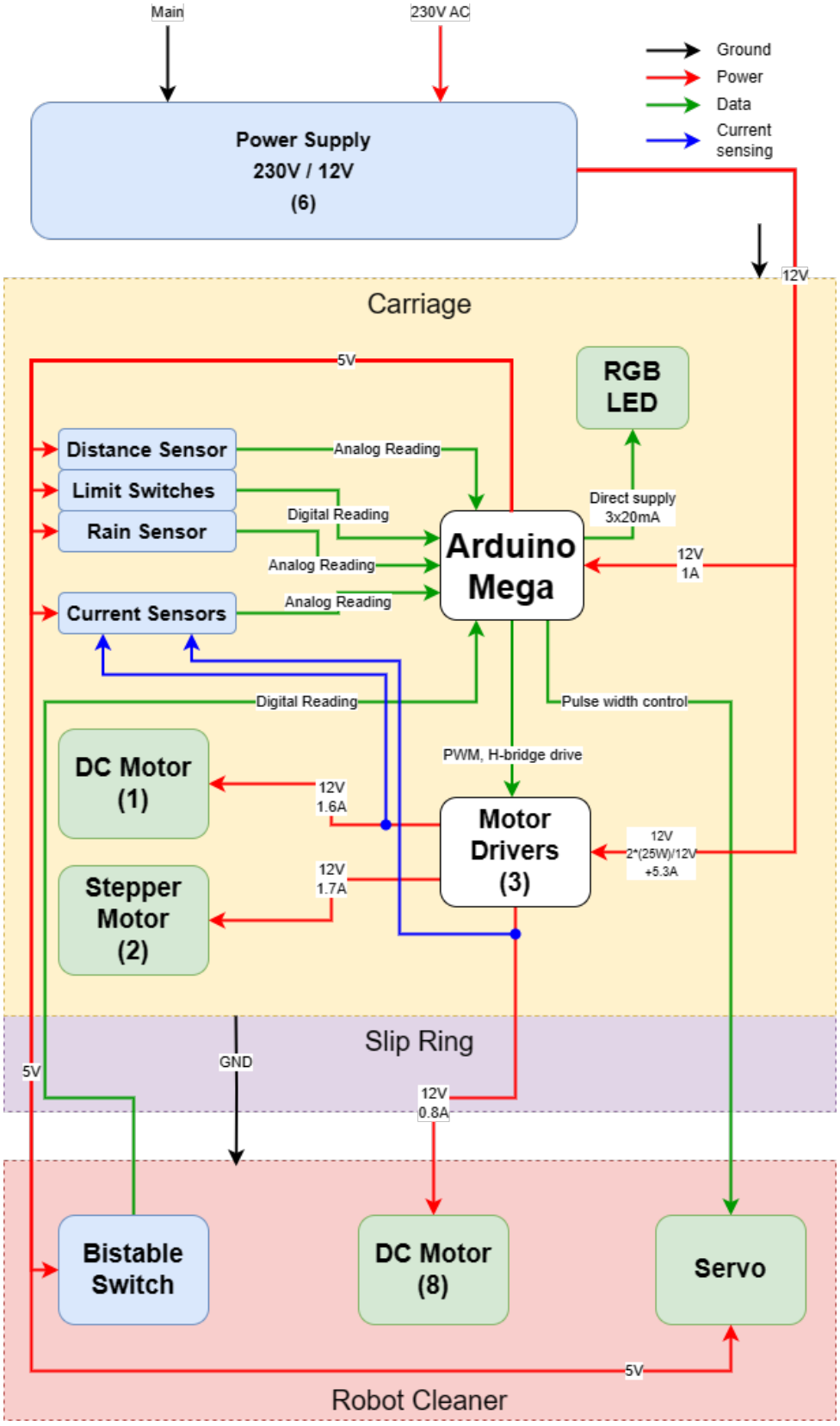


Figure 26: Simplified electrical diagram of the system

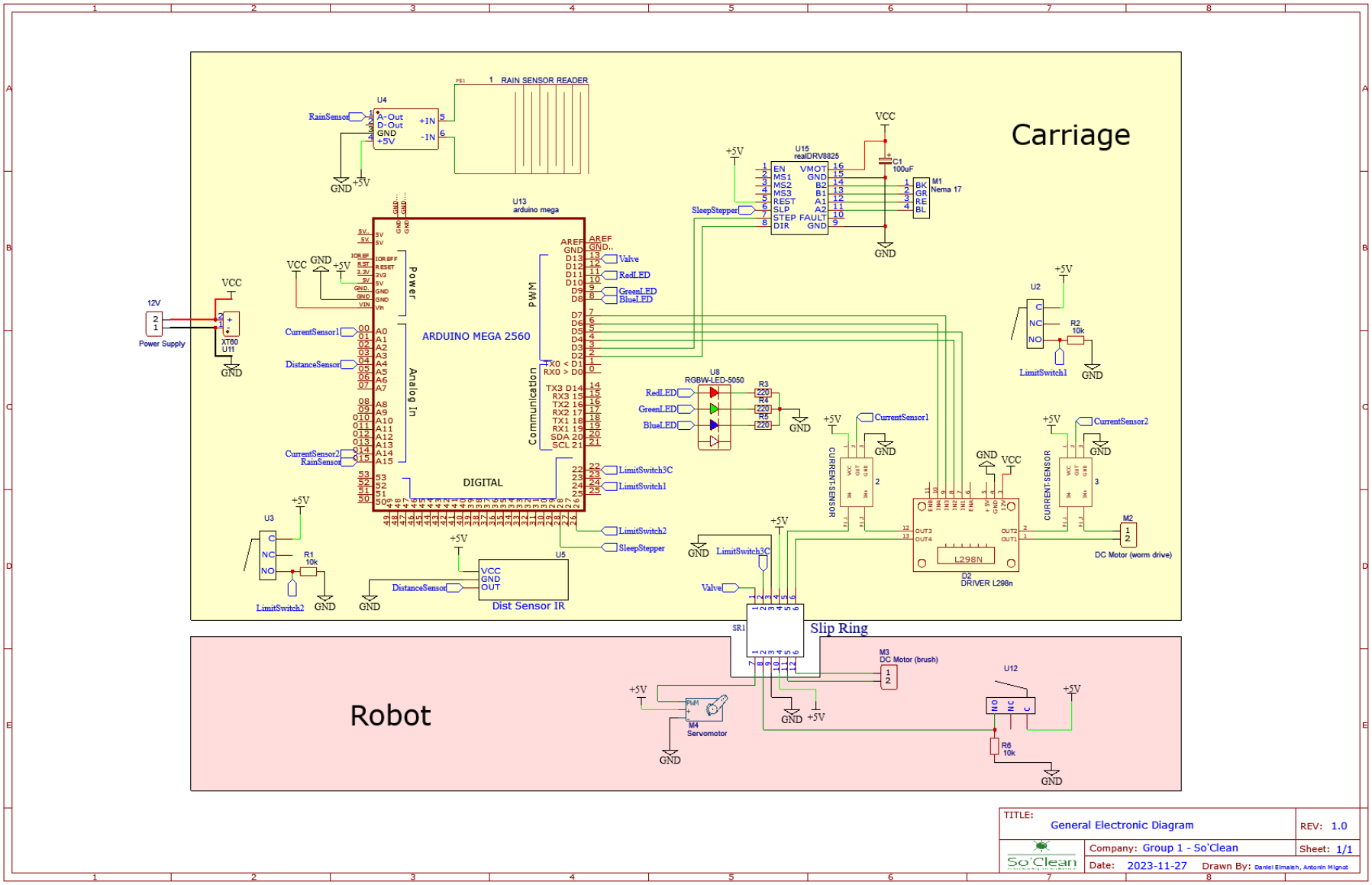


Figure 27: General schematic of the electronics of the system

D Additional Information

D.1 Schematic Winching System

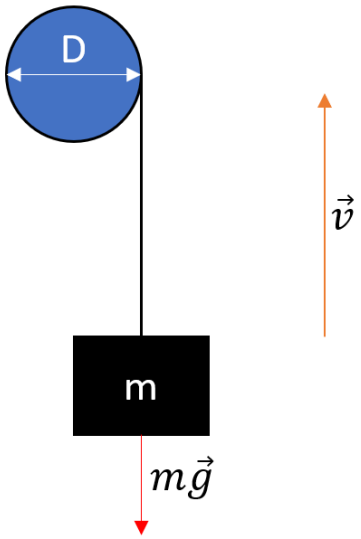


Figure 28: Model of winch system to estimate the power needed

D.2 Electrical Components

The following figure 29 contains a table of all the electrical components used in this project. Along with that, an estimation of the power consumed by the whole system has been performed which comes to a maximum of 50W. It should be noted though that a final product that would contain more sensors and be possibly bulkier would consume more power.

Electronic Components - Technical Details

ID	Component	Datasheet	Voltage Supply [V]	Current Consumption [A]	Power Consumption	Communication Protocol	Connected to ID	# Power pins	# Signal pins (IN)	# Signal pins (OUT)
				Carriage						
1	DC Worm gear motor	High Torque Self Locking 12V, 24V DC Gear Motor (nfpshop.com)	12,00	0,80	9,60	Direct supply	3.1	0	2	0
2	Stepper Motor Nema 17HS4401	17HS4401 Datasheet - 40mm, 2 Phase Stepper Motor (PDF) (datasheetcafe.com)	12,00	1,00	12,00	Direct Supply	3.2	0	4	0
3.1	Motor Driver L298N	L298N Motor Driver Module Pinout, Datasheet, Features & Specs (components101.com) https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf	12,00		5,00	H-bridge (PWM)	IN: 4, OUT: 1;8 Power 6	2	4	4
3.2	Motor Driver DRV8825	DRV8825 Stepper Motor Controller	12,00		5,00	H-bridge (PWM)	IN: 4, OUT: 2 Power 6	2	4	4
4	Arduino MEGA	A000067-datasheet.pdf (arduino.cc)	12,00	0,10	1,20	USB UART	3.1-2, 5.1-2, 6, 7, 9, 10?, 12.1-2	2 (Jack or Vin/Gnd pins)	4	11
5.1	Limit Switch		5,00	0,001	0,01	Measure the voltage directly	4	2	0	1
5.2	Limit Switch		5,00	0,001	0,01	Measure the voltage directly	4	2	0	1
6	Power Supply 12V	全漢pdf (fsp-group.com)	220,00		0,00	-	3.1-2,4	2 (12V, GND)	0	0
7	Rain Sensor	Rain drop Sensor Module Pinout, Datasheet & How to Use it in a Circuit (components101.com)	5,00	0,10	0,50	Analog/digital measure	4	2	0	1
9.1	Distance Sensor	SEN-IR01_Datasheet_2021-01-21.p	5,00	0,03	0,15	Analog Reading	4	2	0	1
12.1	LED		5,00	0,06	0,30	Direct Supply	4	1	1	0
				Robot						
8	DC motor for the brush	Unknown	12,00	0,50	6,00	Direct Supply	3.1	0	2	0
10	Servo Motor	datasheet-2267525-reely-standard-s	5,00	0,50	2,50	Pulse width	4	2	1	0
5,3	Limit Switch		5,00	0,001	0,01	Measure the voltage directly	4	2	0	1
				Others						
11	Electrical Cables		0,00	0,00	0,00	-				
				Power Consumption	**42,27**					
				equivalent current for 12V	**3,52**					

Figure 29: Summary of the electrical components used

D.3 Budget and expenses

The following spreadsheets present an overview of the budget and the expenses for this project.

Solar Panel Cleaner Budget

Subassembly	Part ID	Part Name	Quantity	Unit cost [CHF]	Cost Status	Contingency [%]	Cost w/ cont. [CHF]
Mechanics	ST_01	Raw Materials for protoype	1	CHF80,00	DLL Price Rough Estimate	10	CHF88,00
	ST_02	Test materials	1	CHF30,00	DLL Price Rough Estimate	10	CHF33,00
	ST_03	Screws, bolts, etc	1	CHF25,00	DLL Price Rough Estimate	10	CHF27,50
	OBI1	Chassis Alu Profile	2	CHF9,95	Catalog Price	50	CHF29,85
	3DW1	belt	3	CHF3,80	Catalog Price	0	CHF11,40
	3DW2	profile attachment nut	8	CHF0,30	Catalog Price	0	CHF2,40
	3DW3	belt gear	1	CHF3,80	Catalog Price	0	CHF3,80
	3DW4	vslot wheels	9	CHF2,10	Catalog Price	0	CHF18,90
	3DW5	profile 500mm	1	CHF9,95	Catalog Price	0	CHF9,95
	3DW6	profile 1000mm	2	CHF19,95	Catalog Price	0	CHF39,90
	3DW7	roulement	8	CHF0,90	Catalog Price	0	CHF7,20
	3DW8	coupleur	2	CHF2,30	Catalog Price	0	CHF4,60
	3DW9	locking nut	9	CHF0,20	Catalog Price	0	CHF1,80
	3DW10	screw	9	CHF0,30	Catalog Price	0	CHF2,70
	ST_10	Cables	2	CHF5,00	Rough Estimate	10	CHF11,00
	ST_11	Squegee	1	CHF13,50	Rough Estimate	10	CHF14,85
	KA1	Brush Plastic	1	CHF58,00	Catalog Price	0	CHF58,00
	AL1	Omniwheels	5	CHF2,00	Catalog Price	10	CHF11,00
	ST_14	Slip Ring	1	CHF14,50	Catalog Price	10	CHF15,95
Electronics	AL2	Motor DC vis sans fin	1	CHF15,00	Catalog Price	10	CHF16,50
	EL_01	Motor Stepper glissière	1	CHF10,00	DLL Price	10	CHF11,00
	FREE1	Motor DC brosse	1	CHF0,00	Récupération	10	CHF0,00
	FREE2	Solar Panels	2	CHF0,00	Récupération	10	CHF0,00
	FREE3	MCU, Arduino Uno	1	CHF2,00	Récupération	10	CHF2,20
	FREE4	Alimentation AC/DC	1	CHF0,00	Récupération	10	CHF0,00
	EL_02	Driver motor	2	CHF2,30	DLL Price	10	CHF5,06
	EL_03	Capteur de pluie	1	CHF0,80	DLL Price	10	CHF0,88
	EL_04	Electric cables	1	CHF5,00	DLL Price Rough Estimate	10	CHF5,50
	EL_05	Switch	1	CHF2,50	Catalog Price	10	CHF2,75
	EL_06	Capteur Fin course	2	CHF0,70	DLL Price	10	CHF1,54
	EL_07	Valve	1	CHF6,27	Catalog Price	10	CHF6,90
	EL_08	Capteur IR	1	CHF3,60	DLL Price	10	CHF3,96
	EL_09	Transistor/relais	1	CHF1,00	Catalog Price	10	CHF1,10
	EL_10	Diode	1	CHF0,50	DLL Price Rough Estimate	10	CHF0,55
	EL_11	RGB Led	1	CHF0,15	DLL Price	10	CHF0,17

	Finances [CHF]
Allocated	CHF500,00
Budget Spent	CHF449,90
Remaining	CHF50,10

Solar Panel Cleaner expenses

Order Nr.	Seller/Supplier	Products	Description	Subsystem	Unit Price	Quantity	Shipping	Total Price	Paid by	Status
1	Aliexpress	Bringsmart worm-drive motor	DC motor for robot height	Electronics	CHF12,31	1	CHF0,00	CHF12,31	Matteo	Received
		Omniwheels	omniwheel for robot	Mechanics	CHF1,86	5	CHF12,81	CHF22,11		Received
2	3dWare	GT2 Riemen 6mm breit	belt	Mechanics	CHF3,80	3	CHF0,00	CHF11,40	Matteo	Received
		Einzel T-Nut M5 Gleitstein für OpenBuilds	profile attachement nut	Mechanics	CHF0,30	8	CHF0,00	CHF2,40		Received
		GT2 Riemenscheibe 20 Zähne CoreXY 5mm Bohrung	belt gear	Mechanics	CHF3,00	1	CHF0,00	CHF3,00		Received
		OpenBuilds V-Slot Profil 20x20x500mm Schwarz	v-slot aluminium profile	Mechanics	CHF9,95	1	CHF0,00	CHF9,95		Received
		OpenBuilds V-Slot Profil 20x20x1000mm Schwarz	v-slot aluminium profile	Mechanics	CHF19,95	2	CHF0,00	CHF39,90		Received
		Für Openbuilds Kunststoffrad POM V Typ mit Lager	opebuilds wheel	Mechanics	CHF2,10	9	CHF0,00	CHF18,90		Received
		Kugellager 628ZZ	roulements à bille	Mechanics	CHF0,90	8	CHF0,00	CHF7,20		Received
		Z Achse Kupplung starr 8mm auf 8mm Aluminum	coupleur	Mechanics	CHF2,30	2	CHF0,00	CHF4,60		Received
		OpenBuilds Sechskant Kontermutter M5 schwarz	locking nut	Mechanics	CHF0,20	9	CHF0,00	CHF1,80		Received
		OpenBuilds Schraube mit niedrigem Kopf M5 30mm	screw	Mechanics	CHF0,30	9	CHF0,00	CHF2,70		Received
3	Kärcher	Brosse-rouleau, souple, blue, 300 mm	Brush	Mechanics	CHF58,00	1	CHF0,00	CHF58,00	Jérémie	Received
4	Teachers	Arduino UNO	Microcontroller	Electronics	CHF0,00	1	CHF0,00	CHF0,00	Free	Received
5	CSEM	Solar Panel	Solar Panel	Electronics	CHF0,00	2	CHF0,00	CHF0,00	Free	Received
6	Antonin	Power Supply FSP460-701UH	12V 35A power Supply	Electronics	CHF0,00	1	CHF0,00	CHF0,00	Free	Received
7	Matteo	DC Motor	DC Motor for brush	Electronics	CHF0,00	1	CHF0,00	CHF0,00	Free	Received
8	OBI	Combitech Tube carré Aluminium blanc 20 mm x 1 m	Aluminimum Profile	Mechanics	CHF7,95	2	CHF0,00	CHF15,90	Matteo	Received
		Sangle PP	carriage belt	Mechanics	CHF1,75	4	CHF0,00	CHF7,00	Matteo	Received
9		Caoutchouc de remplacement pour essuie-glaces standar	squeegee	Mechanics	CHF13,50	1	CHF0,00	CHF13,50	Blaise	Received
10	Garance	Arduino MEGA	microcontroller	Electronics	CHF0,00	1	CHF0,00	CHF0,00	Free	Received
11	DigiKey	SLIP RING 22MM 6WR MAX 240V 2A	Slip ring	Electronics	CHF14,17	1	CHF0,00	CHF14,17	Joël (GR8)	Received
12	DLL	Electronics	diverse electronics parts	Electronics	CHF26,15	1	CHF0,00	CHF26,15	Billed to the course	Received
13		Electronics	current sensors	Electronics	CHF2,80	1	CHF0,00	CHF2,80	Billed to the course	Received
14		Stepper frame		Mechanics	CHF4,80	1	CHF0,00	CHF4,80	Billed to the course	Received
15		insert fileté M3		Mechanics	CHF0,10	7	CHF0,00	CHF0,70	Matteo	Received
16		DC Brush + capteur courant + many cables		Electronics	CHF10,05	1	CHF0,00	CHF10,05	Billed to the course	Received
17		Pins female and male	connectors	Electronics	CHF0,25	2	CHF0,00	CHF0,50	Billed to the course	Received
18		Visserie & materials Atelier Meca (total Jérémie)		Mechanics	CHF16,05	1	CHF0,00	CHF16,05	Billed to the course	Received
19		Electronics (Jérémie)	Cables and servo	Electronics	CHF13,60	1	CHF0,00	CHF13,60	Billed to the course	Received
20		Electronics	Photoresistor	Electronics	CHF0,05	1	CHF0,00	CHF0,05	Billed to the course	Received
21		Electronics Stepper Driver	Stepper Motor Driver	Electronics	CHF1,40	1	CHF0,00	CHF1,40	Billed to the course	Received
22		3D printing	3D prints	Mechanics				CHF87,42	Not yet paid	Received

	Finances [CHF]
Allocated	CHF500,00
Total Spent	CHF408,36
Remaining	CHF91,64

D.4 Risk Analysis

The following section presents a risk analysis for the development of our prototype.

Risk Analysis

Risk ID	Risk	Category	Related requirements	Sub-category	Probability	Criticality	Risk Priority	Strategy	Risk Response
1	We do not receive our orders in time	External	SPC prototype shall be constructed before December 15, 2023, and presented by December 22.	Suppliers	4	8	32	Mitigate	Design backup plans if some critical functions cannot be built
2	One member cannot work because of personal issues	Project Management		Human ressources	2	4	8	Accept	Be ready to find time to work more
3	One of our components doesn't work	Technical		Performance	4	8	32	Transfer	Find a fast way to find another one to replace
4	Our final product has a last minute issue before the final presentation (doesn't work, or were dammaged)	Technical		reliability	2	8	16	Avoid	Keep the robot in a safe place, and be very cautious with it
5	Electronical assembly issues	Technical		Interface	2	2	4	Transfer	Find a solution to adapt each component with each other
6	Mecanical assembly issues	Technical		Interface	2	2	4	Transfer	Find a solution to adapt each component with each other
7	We can't take back the solar panels from Neuchatel	Organizational		ressources	2	4	8	Avoid	Find a solution to take them back
8	Cost overruns	Project Management	SPC prototype shall have a production cost of maximum 500 [CHF].	Estimating	1	4	4	Avoid	Be cautious in our expenses and plan everything
9	The panel isn't 95% efficient or above after the cleaning	Technical	SPC shall remove soiling on at least 95 [%] of the solar panels surface after cleaning.	Performance	2	8	16	Avoid	Focus and make intelligent choice to make or product useful
10	Our product isn't safe to use	Technical	SPC shall be handled without the risk of injury.	reliability	1	8	8	Avoid	Be cautious with the use of electricity and water with the same device
11	Water needs to clean the panels are underestimated, 2L isn't sufficient	Technical	SPC should be designed to use the least amount of water possible. SPC should not use more than 1 [l] of water per 10 [m²] of cleaning.	Performance	2	4	8	Mitigate	Test the importance of water, and what volume is needed, The software can be modified to dispense less water

12	The rail system can't be put on the solar panel	Technical	SPC shall be designed to be adapted for standard solar panels size (1 x 1.7 m). SPC shall be compatible with residential PVinstallations.	interface	1	8	8	Transfer	Find an adaptator
13	Our brush or our wheels damage the panel	Technical	SPC shall use materials and cleaning methods that do not damage the surface of solar panels.	Interface	1	4	4	Accept	Choose a soft brush
14	The robot can't clean 1 [m²/min]	Technical	SPC should be able to clean at a speed of at least 1 [m²/min].	Performance	1	1	1	Accept	Keep a security coefficient on our performances
15	Intellectual property concerns	External		market	1	2	2	Accept	Keep in the project limits, or verify if there isn't any IP concern
16	The Motors are not powerful enough to move the system	Technical	SPC shall remove soiling on at least 95 [%] of the solar panels surface after cleaning.	Requirements	4	8	32	Mitigate	Change the motor for a more powerful one
17	A malfunctionning component can't be replaced without damaging other components	Technical	SPC shall be designed to be easily repairable, each component can be changed without damaging any other one	Requirements	1	8	8	Mitigate	Use the least unreversible process to build the prototype

D.5 Cleaning action selection

As mentioned in the text, many options and combinations of systems were considered for the cleaning action of the product. Early on, the cleaning system was envisioned to be a combination of a variety of possible cleaning tools, of which a few are named here:

- A simple broom against the panel
- A motorized brush scrubbing the panel (wet or dry)
- A squeegee against the panel
- A water-free dust repellent using dielectrophoresis

While most components were easy to test in a practical setting, dielectrophoresis was of particular interest, given its added value of not needing water. Below is the concept for a cleaning system leveraging this principle.

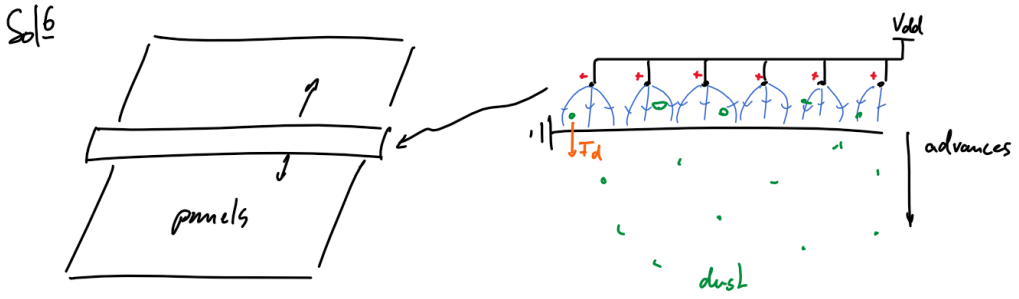


Figure 30: Concept for dielectrophoretic cleaning

A sort of "wiper" beam is swept across the panel, just like a simple squeegee. Inside it are electrodes as drawn in the figure, that, when under voltage, create a strong gradient electric field in between them. This induces a dipole in any dielectric body nearby, and in turn exerts a force, called dielectrophoretic force (DEP force). This enables trapping of particles in the electrodes and sweeping away of the dust on the panel.

To validate the topology of generated electric field and its gradient, simulations were performed in COMSOL, and a typical result is shown below.

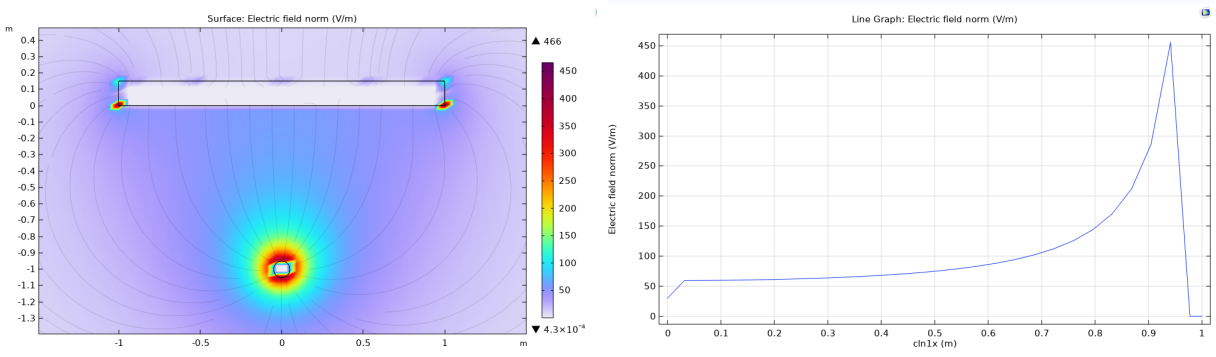


Figure 31: Electric field simulation for an electrode (left), field norm across its symmetry axis (rigth)

The bar element is grounded and the point, up to some potential. As shown in the line plot on the right, the electric field does present a strong derivative and could hence trap dielectric particles inside the electrode. To confirm this, a practical test was conducted with what was readily available for the group. Namely a 10kV DC voltage multiplier and some regular dust. When testing the electrode with dust in between, particles were indeed pushed away, but in every direction. This is easily explained by the DEP force being a second order phenomenon,

depending on the gradient of the field. While dust particles surely are dielectric, they are also often electrostatically charged. This means they react to the high electric field in the electrode and shoot out uncontrollably.

In parallel to this experimentation with the dielectrophoretic system, the practical tests of the less sophisticated cleaning components had been conclusive. Indeed the simple use of the wet brush and squeegee proved very effective during tests on a dirty glass surface as they left no visible soilings behind. While the system uses water, it is a lot simpler to integrate and with the limited time available to design and implement an already complex system, it was chosen to abandon the DEP system, in spite of its clever concept[3].

D.6 Brush suspension and squeegee mount

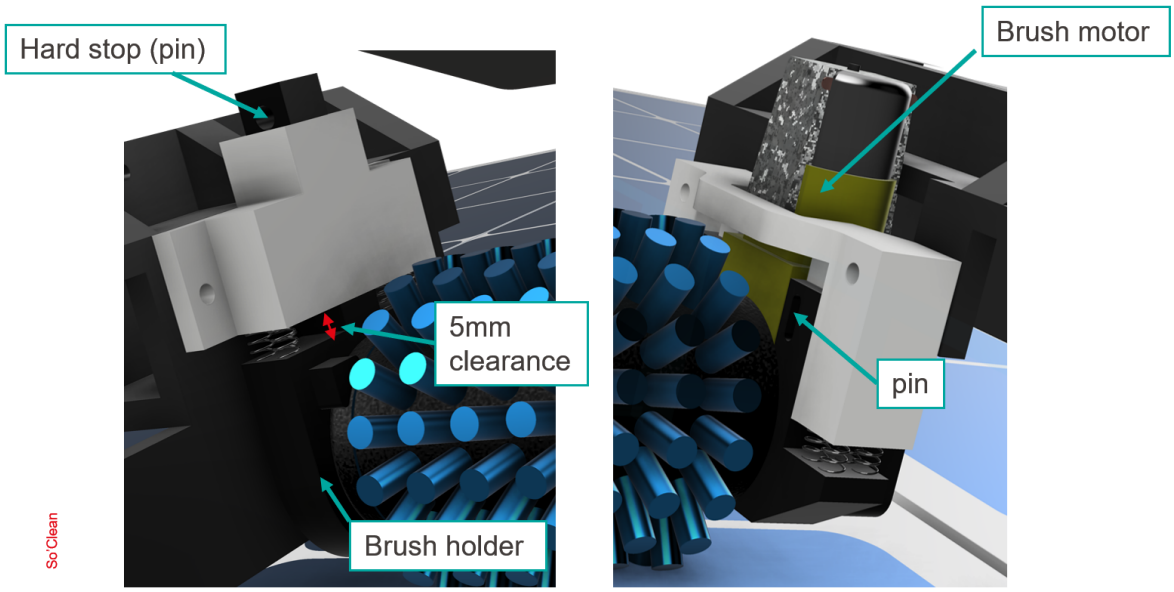


Figure 32: Spring loaded suspensions on either sides of the brush

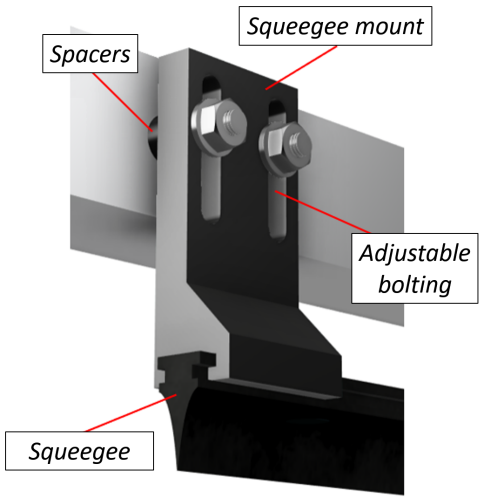


Figure 33: Squeegee mountings

[3]Which was hinted to the group during a lecture, by Prof. Y. Bellouard.

D.7 Installation manual

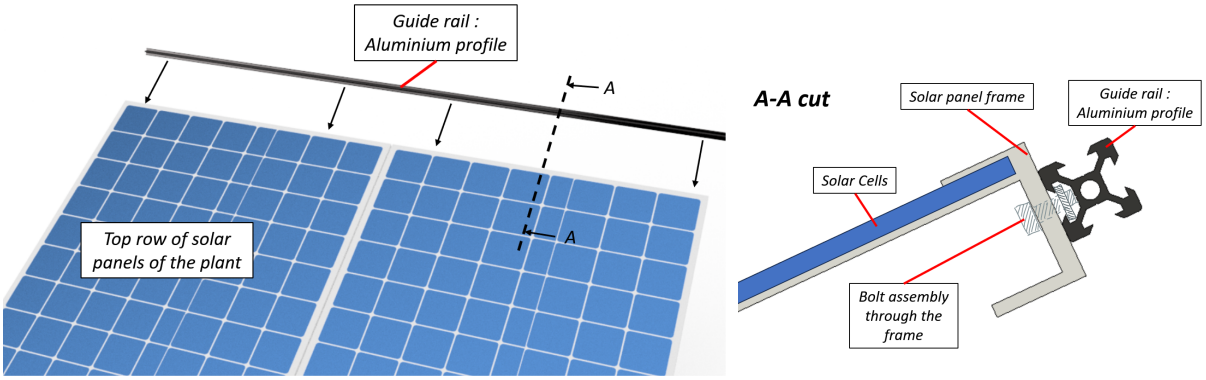


Figure 34: Installation of the guide rail

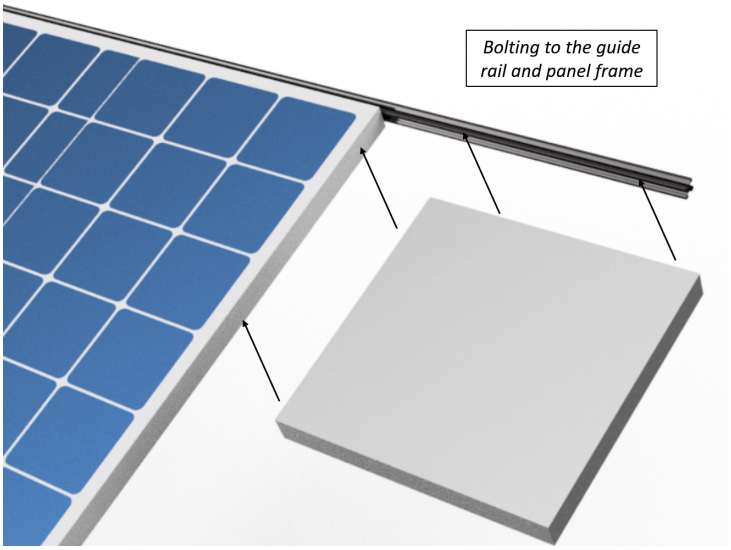


Figure 35: Installation of the idling platform (schematically)

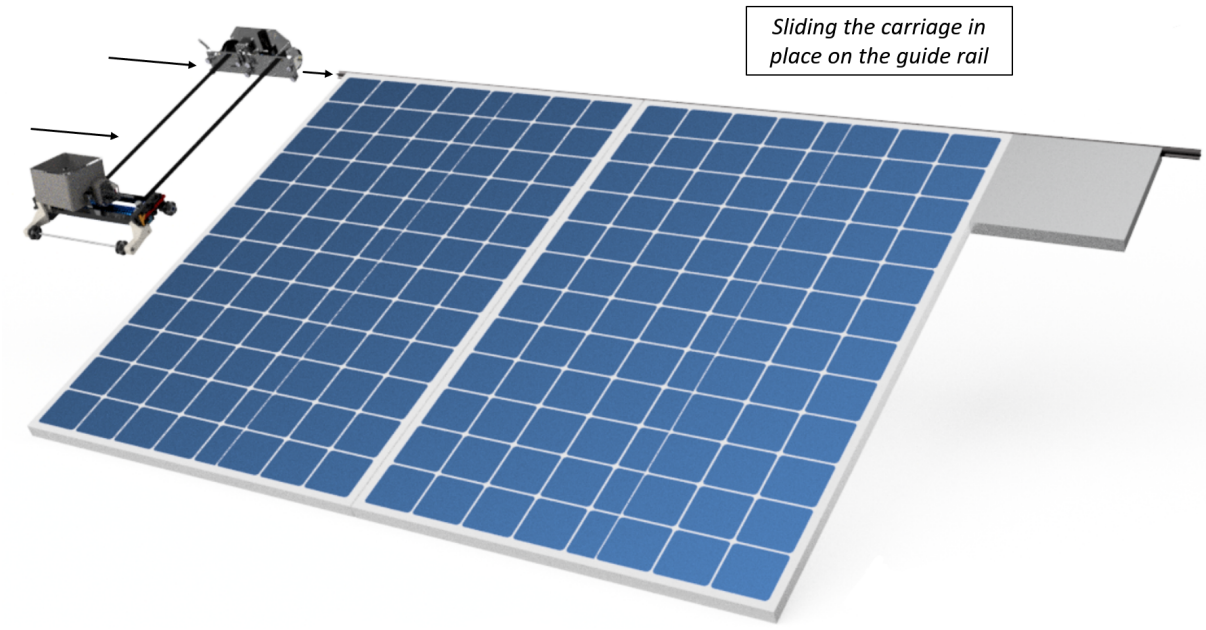


Figure 36: Final insertion of the carriage on the rail

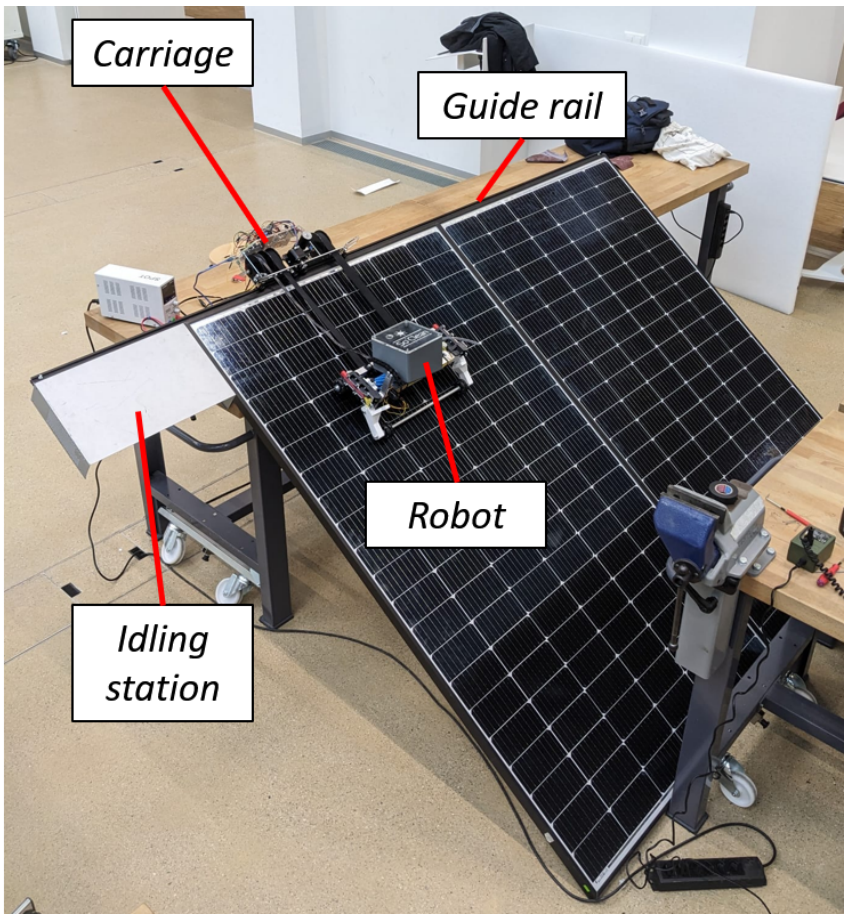


Figure 37: Finished prototype sitting at the SPOT

D.8 Aluminium Plate

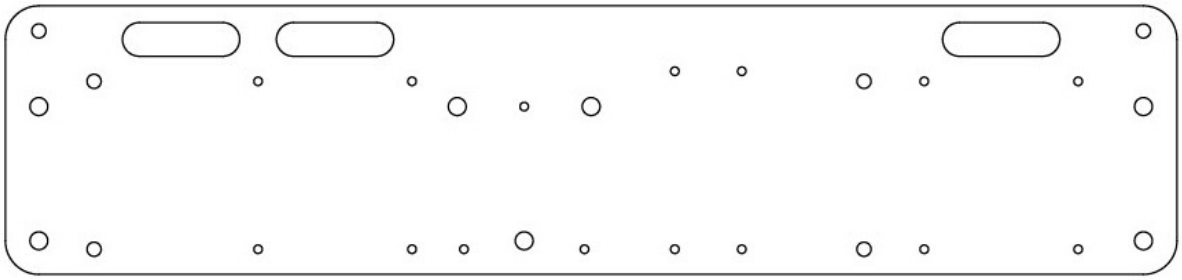


Figure 38: DXF file of Aluminium Plate for laser cutting

D.9 Cleaning Robot

Figure 39: Bistable mechanism mounted on the frame

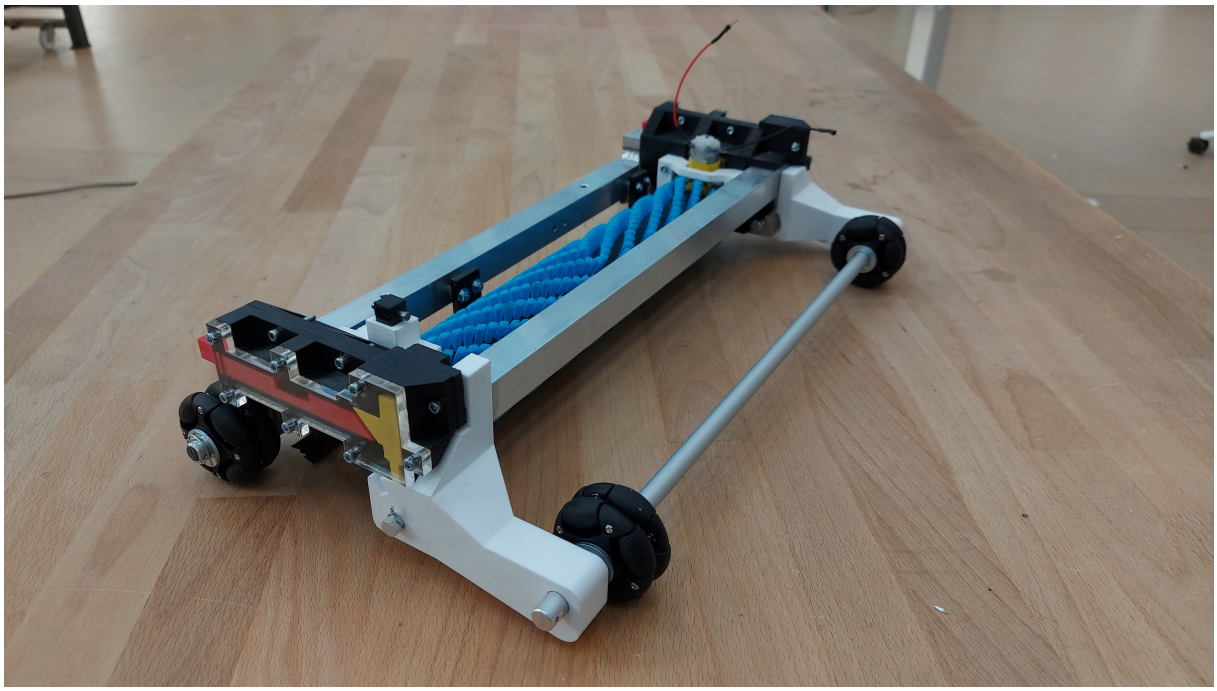


Figure 40: Next assembly step, with the mounted brush and squeegee

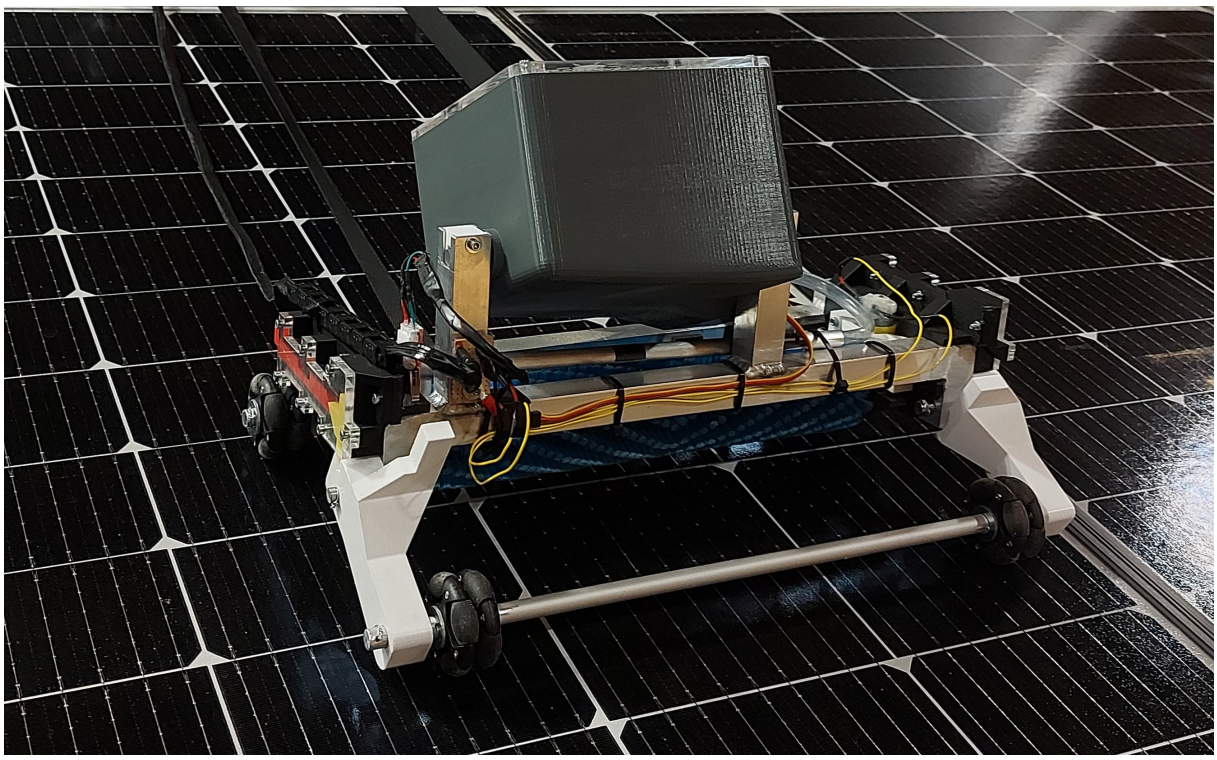


Figure 41: Finalized cleaning robot

D.10 Cost estimation

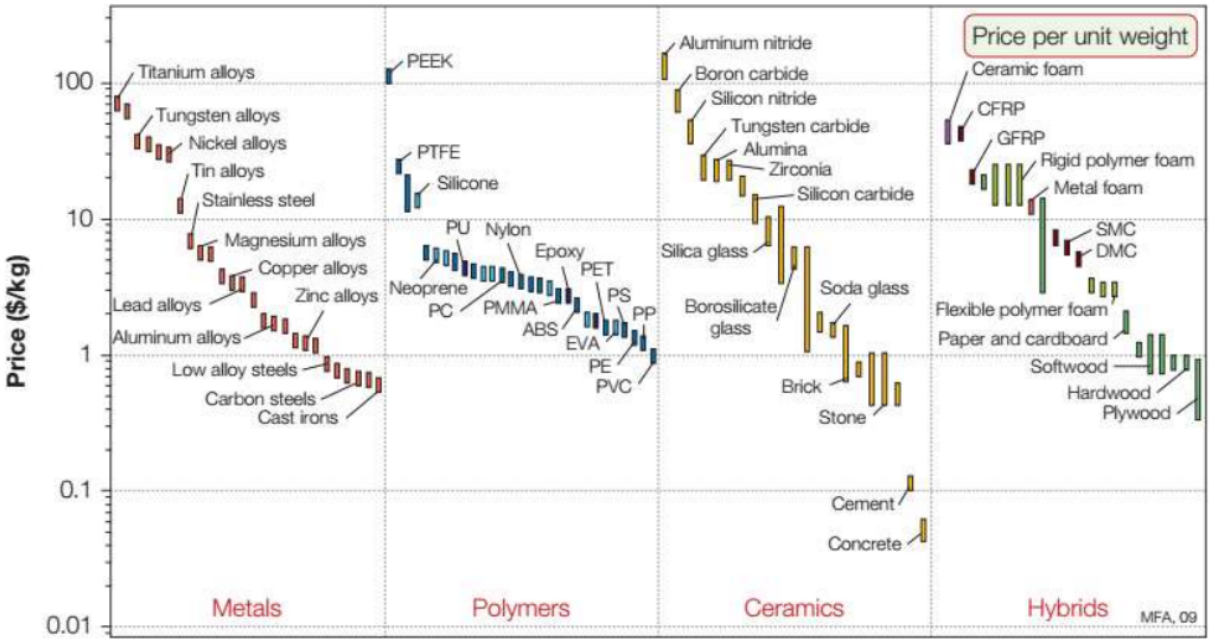


Figure 42: Typical cost of materials ($/Kg) [5]

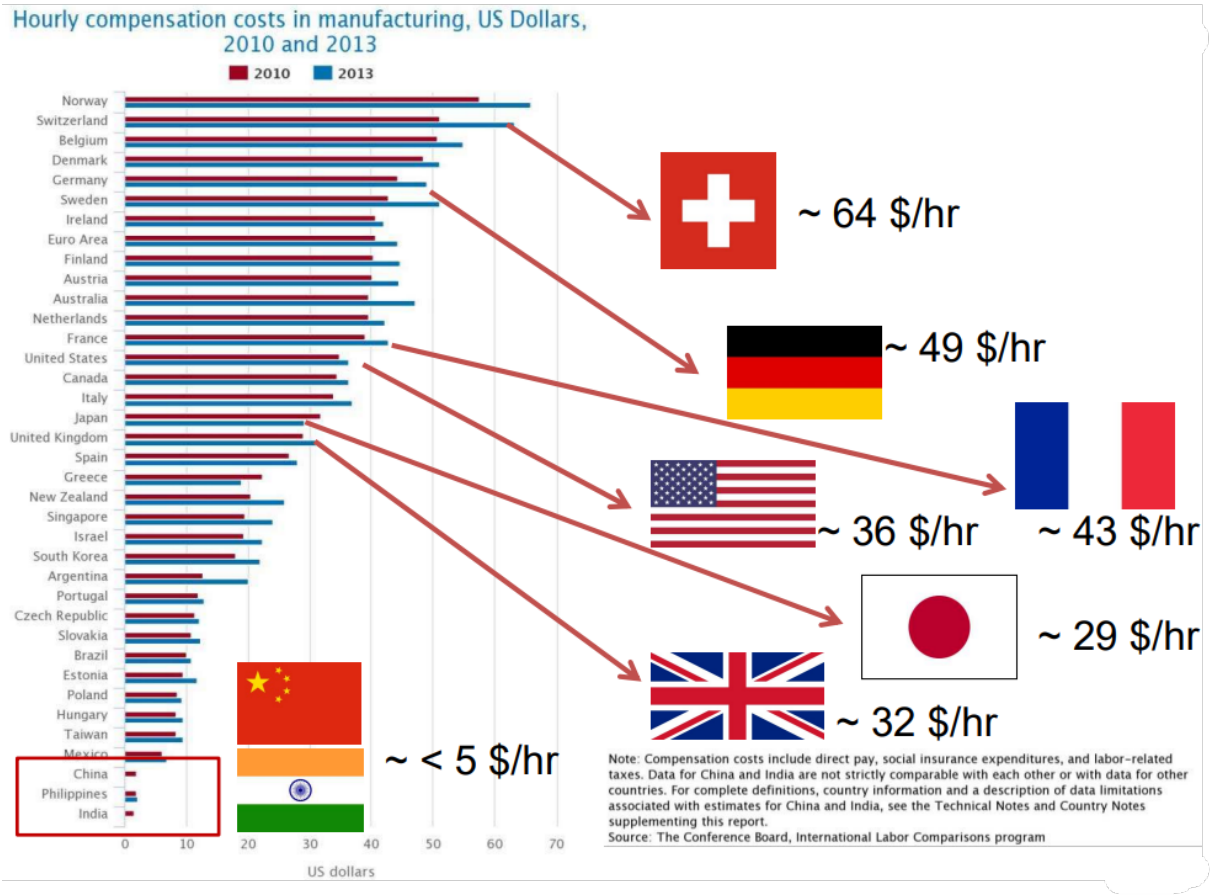


Figure 43: Labot cost in different countries [5]

D.11 Work Breakdown Structure

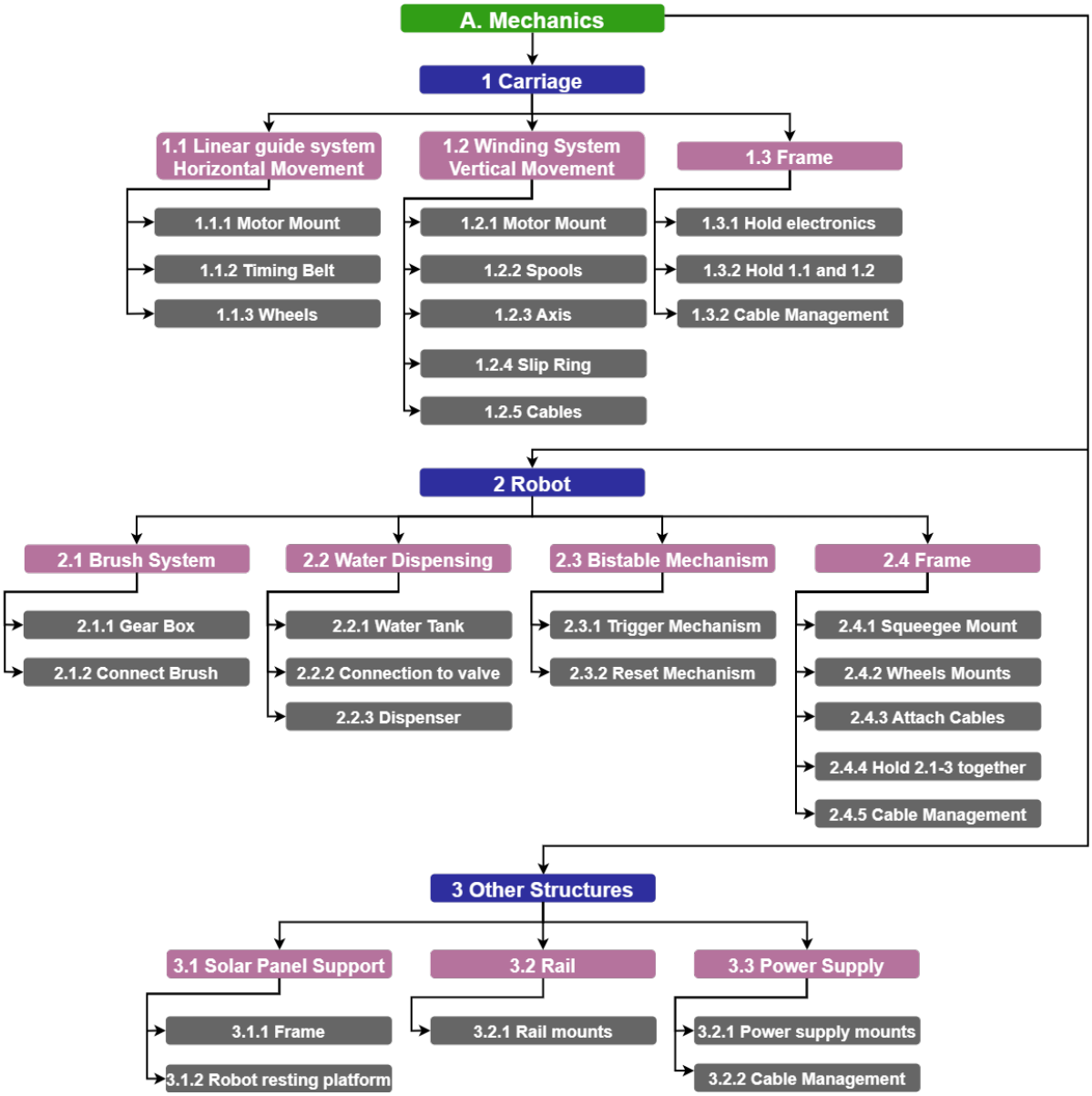


Figure 44: WBS - Mechanical part

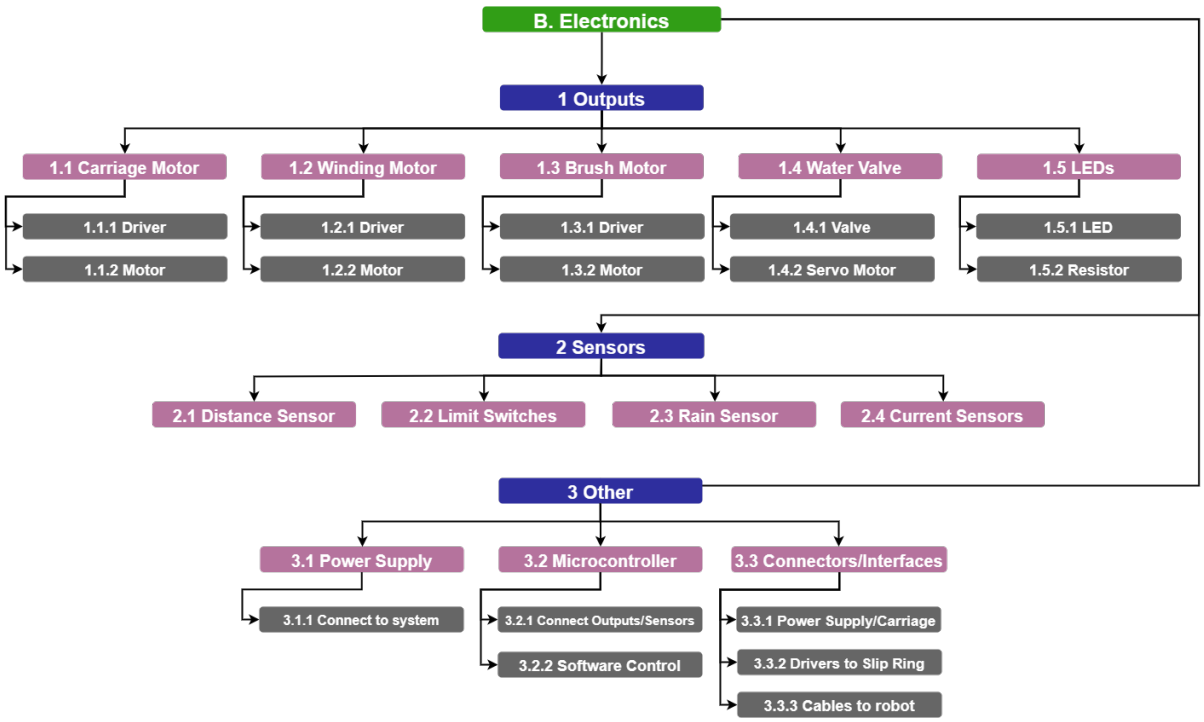


Figure 45: WBS - Electronics part

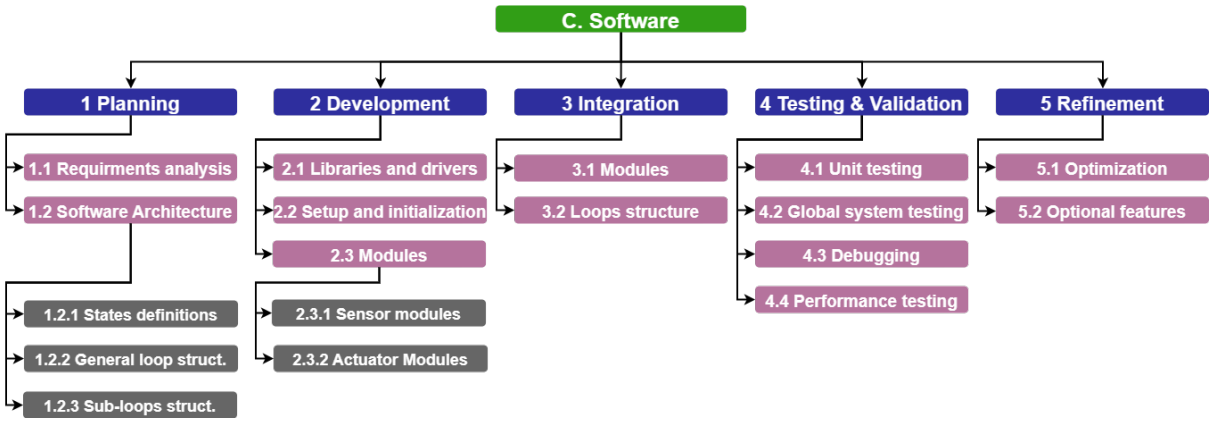


Figure 46: WBS - Software part

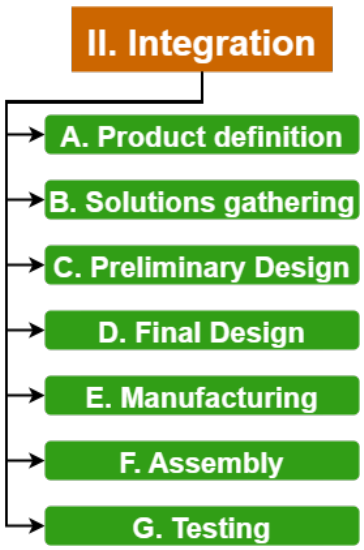


Figure 47: WBS - Integration

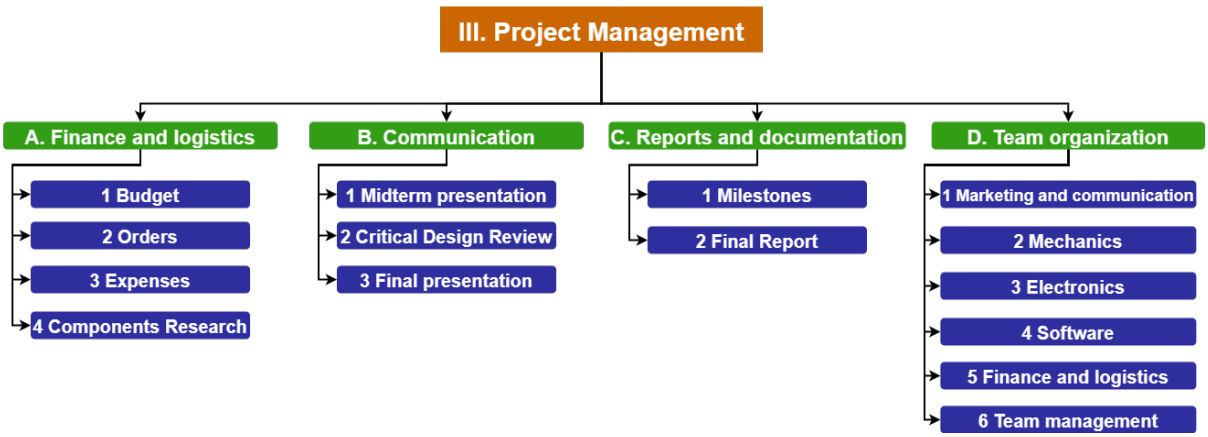


Figure 48: WBS - Management

D.12 Gantt chart

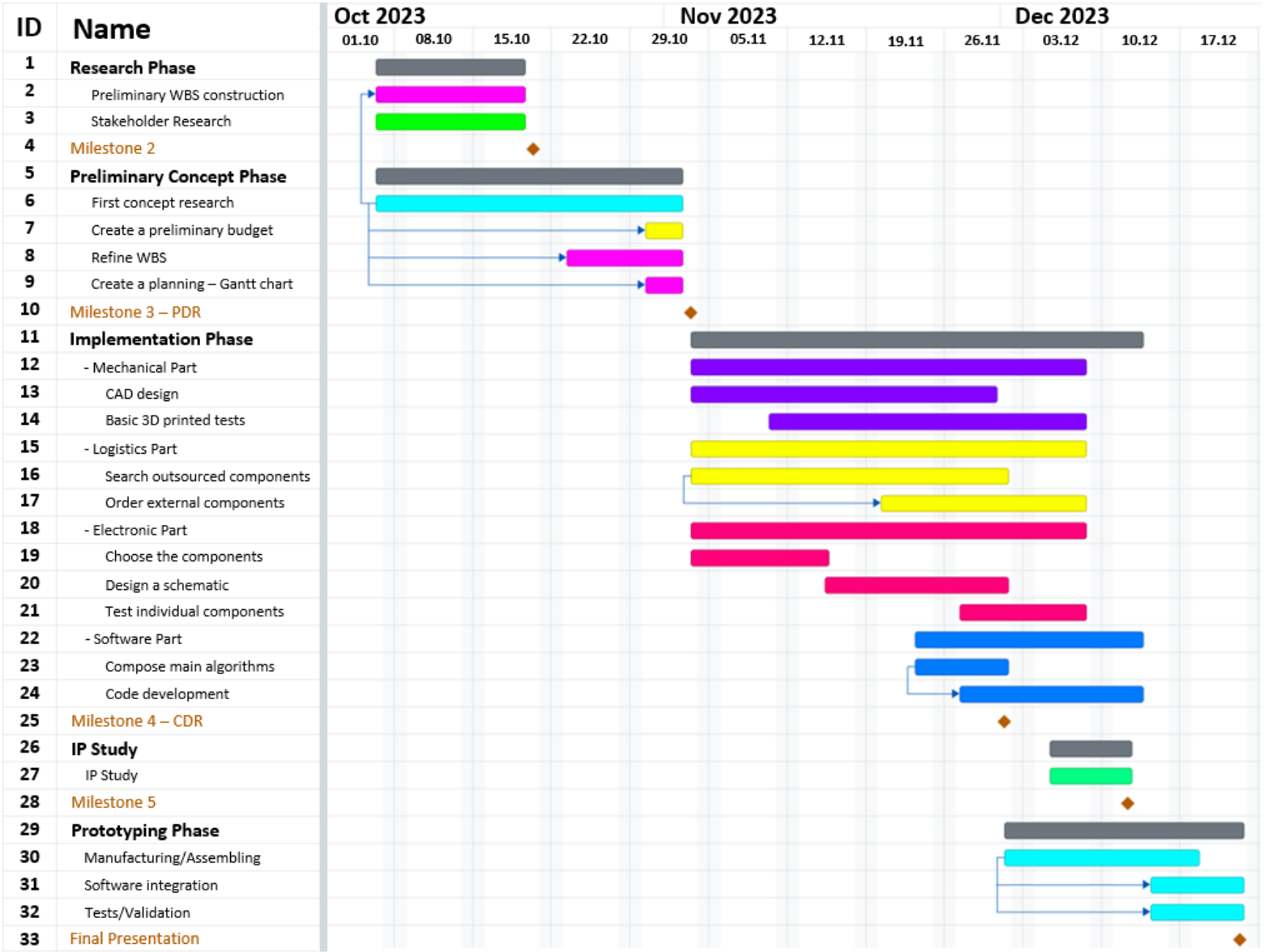


Figure 49: Complete Gantt chart for this project

D.13 Software

Pseudo code

Global Variables:

- *cleaning_width*: Effective cleaning width of the robot. Set as a CONSTANT.
- *cleaning_period*: Time between two cleanings. Set as a CONSTANT.
- *dist*: Total width of all panels. Set as a CONSTANT.
- *step*: Step width between 2 cleaning columns. Calculated from CONSTANTS.
  Calculated as E(dist / Cleaning_width) + (0 or 1 dep. Mod(dist / Cleaning_width)).
- *raining*: Boolean indicating if it is raining.
- *time*: Actual time.
- *cleaning_time*: Time of the last cleaning.
- *STATE*: Enumeration variable that represents the state of the system.

States [Enumeration]:

0. Rest (sleep)
1. Water Filling
2. Initial Position
3. Cleaning
4. Up Travel
6. Translation
8. Return Home
9. Problem

Setup and Initialization

Reset and initialization of all the electronical components. Setting outputs, inputs, and variables.

General Loop Structure:

General Loop ():

{

time update();

Edge cases and problem handling.

Check & update loops*();*

Switch(*STATE***):**

Case Rest**:**

STATE logic:

Action loops*()*

case transition.

Case Water Filling**:**

…

}

Sub-loops:

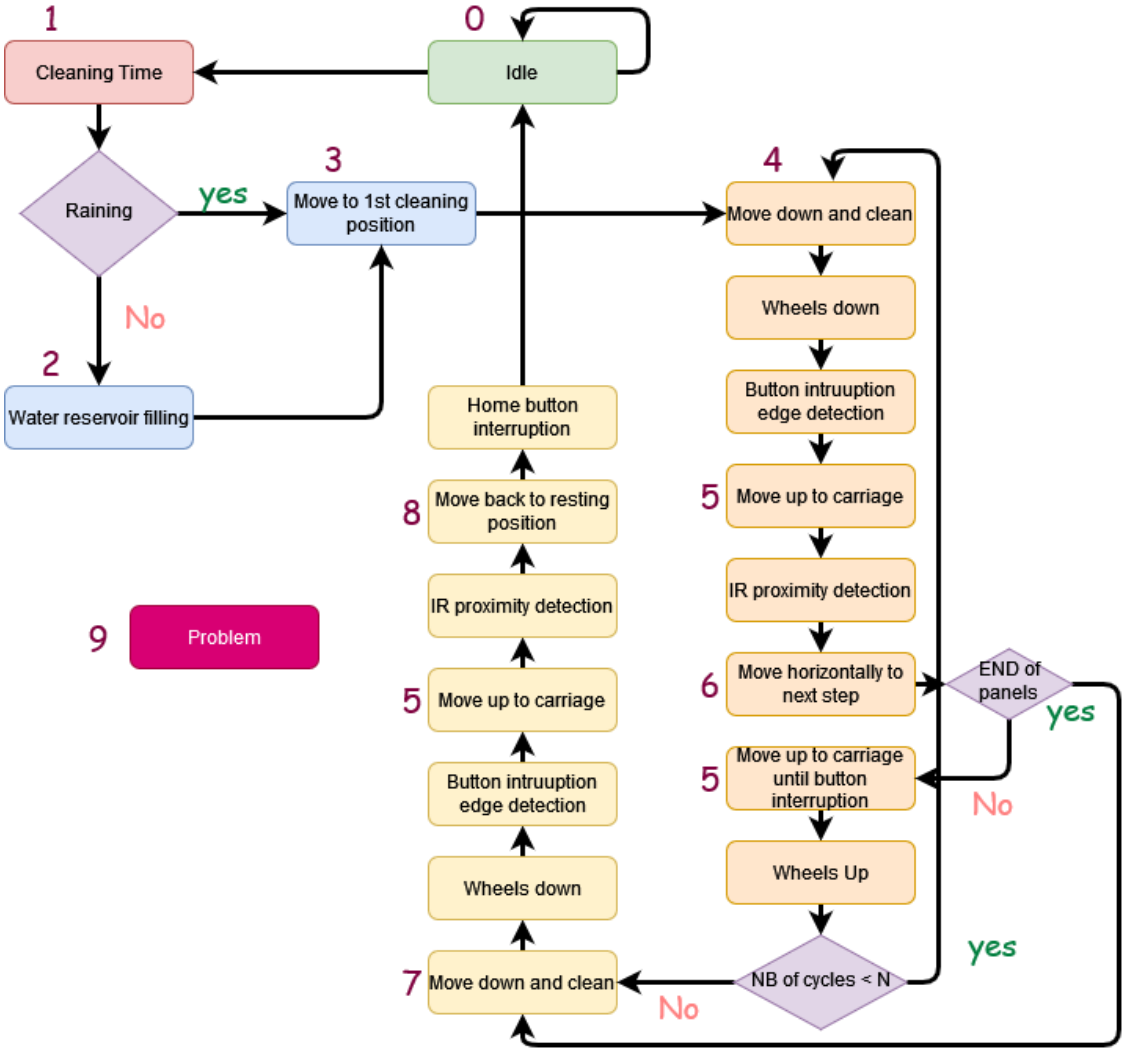
Check & update loops (State Transitions)	***Action loops***
Update_LEDs	*Valve_control*
IR_check	*DC_motor*
Button1/ Button2 / Button3 check	*Brush_motor*
Rain_detector_check	*Stepper_motor*

- ***Update_LEDs():*** State conditioned signaling of the RGB LED.
- ***IR_check ():*** Proximity sensors reading for relevant state and variables updates.
- ***Button1/ Button2/ Button3 ():*** Reading of the buttons for relevant state update.
- ***Check_rain_detector():*** Check and update of state.
- ***Valve_control():*** Control of the electronical water valve during the cleaning state.
- ***DC_motor():*** Control of the vertical (in the plane) robot positioning motor.
- ***Brush_motor():*** Control of the motor spinning the cleaning brush, while in cleaning state.
- ***Stepper_motor():*** Control of the horizontal (in the plane) robot positioning motor trough the carriage.

Sub-loop():

```
{
  time update();
  IF (time - last_time = loop_frequency_condition):
  {
    Code; // The appropriate code for this loop.
    last_time = time;
  }
}
```

Flow Diagram:



Note: this pseudo code is a simplified and vulgarized version of the real code which has additional variables, functions, and modules.

Main Program

PROJET_PDSE\PROJET_PDSE.ino

```
#include <Time.h>
#include "def_const_func.h"

// Setup and initialization
void setup() {
    Serial.begin(9600);
    myServo.attach(SERVO_PIN);
    void initializeMotors();
    void initializeButtons();
    void initializeLEDPins();
    void initializeRainSensor();
    void initializeStepperMotor();
    void initializeCurrentSensors();
    checkButtonChome();
    if (buttonStateChome == RELEASED) {
        currentState = RETURN_HOME;
    }
}

// Main loop
void loop() {
    if (currentState == PROBLEM) {
        updateLEDs(currentState);
        return;
    }

    if (buttonStateC1 == CLICKED){ // A voir si dernier netoyage apres ou avant clic
        stopAllMotors();
        currentState = RETURN_HOME;
        if (currentState != TRANSLATION) {
            currentState = PROBLEM;
        }
    }

    // Update the current time
    currentTime = millis();
    updateLEDs(currentState);
    checkCurrent(CURRENT_PIN1);
    checkCurrent(CURRENT_PIN2);
    checkButtonR();
    checkButtonC1();
    checkButtonChome();
    checkIRSensor();

    // Check the states and perform actions based on the current state
    switch (currentState) {
        case REST:
            Serial.println("REST");
            checkRainSensor();
            if (buttonStateR == CLICKED) {
                currentState = PROBLEM;
            }
```

```
            if (IRseen == false || buttonStateChome == RELEASED) {
                currentState = RETURN_HOME;
            }
            else if ((currentTime >= cleaningTime + CLEANING_PERIOD) || raining) {
                if (!raining) {
                    currentState = WATER_FILLING;
                }
                else {
                    currentState = INITIAL_POSITION;
                }
                cleaningTime = currentTime; // Update the last cleaning time
            }
            else {
                currentState = REST;
            }
        break;
        case WATER_FILLING:
            // If not HOME then RETURN_HOME
            if (buttonStateR == CLICKED) {
                currentState = PROBLEM;
            }
            if (IRseen == false || buttonStateChome == RELEASED) {
                currentState = RETURN_HOME;
            }
            // after filling done state = Initial_position
            else {
                // fill water reservoir
                delay(5000);
                currentState = INITIAL_POSITION;
            }
        break;
        case INITIAL_POSITION:
            // Logic to move to the initial position
            // steps to 1st position and when finished state = CLEANING
            moveMotor(RIGHT, DIST_TO_INITIAL_POSITION);
            stopAllMotors();
            while (buttonStateR == RELEASED) {
                moveMotor(UP, MOTOR_SPEED_GEAR);
            }
            stopAllMotors();
            currentState = CLEANING;
            delay(3000);
        break;
        case CLEANING:
            // Logic to clean the panels
            // If buttonR = 1  state = UP_TRAVEL, both for start and finish. delay for the finish.
            // Else go downward with dc motor and activate dc brush motor and valve (until buttonR=1).
            if (buttonStateR == RELEASED) {
                // Stop Brush motor
                digitalWrite(BRUSH_MOTOR_PIN1, LOW);
                digitalWrite(BRUSH_MOTOR_PIN2, LOW);
                controlValve(VALVE_ANGLE_CLOSE);
                delay(END_OF_CLEANING_DELAY);
```

```
106                 stopAllMotors();
107                 currentState = UP_TRAVEL;
108             }
109             else {
110                 moveMotor(DOWN, MOTOR_SPEED_GEAR);
111                 controlBrushMotor(HIGH);
112                 controlValve(VALVE_ANGLE_OPEN);
113             }
114         break;
115         case UP_TRAVEL:
116             // Logic to move up to the next position
117             // go upward with dc motor until IR sensor detects the robot
118             // nb_cycles = nb_cycles + 1
119             // if nb of cycles = N state = RETURN_HOME else state = TRANSLATION
120             if (buttonStateR == CLICKED) {
121                 currentState = PROBLEM;
122             }
123             else if (IRseen == false) {
124                 moveMotor(UP, MOTOR_SPEED_GEAR);
125             }
126             else {
127                 stopAllMotors();
128                 nb_cycles_counter = nb_cycles_counter + 1;
129                 if (nb_cycles_counter*step == DIST) {
130                     currentState = RETURN_HOME;
131                     break;
132                 }
133                 currentState = TRANSLATION;
134                 stopAllMotors();
135                 moveMotor(LEFT, step);
136             }
137         break;
138         case TRANSLATION:
139             // Logic to translate to the next position
140             // go sidways with stepper motor for step
141             // when stepper motor finished move upwards with dc motor until buttonR=0 and
    then state = CLEANING
142             if (buttonStateR == RELEASED) {
143                 moveMotor(UP, MOTOR_SPEED_GEAR);
144             }
145             else {
146                 stopAllMotors();
147                 currentState = CLEANING;
148             }
149         break;
150         case RETURN_HOME:
151             // Logic to return to the initial position
152             // if IRseen is false move upward with motor dc until IRseen is true
153             // if buttonChome=1 move sidways with motor stepper until buttonChome=0
154             if (buttonStateR == CLICKED) { // FIX THIS CONDITION WITHOUT PROBLEM STATE
155                 currentState = PROBLEM;
156             }
157             else if (IRseen == false) {
158                 moveMotor(UP, MOTOR_SPEED_GEAR);
159             }
```

```
            if (IRseen == true) {
              stopAllMotors();
              checkButtonChome();
              digitalWrite(STEPPER_SLEEP_PIN, HIGH);
              delayMicroseconds(2);
              digitalWrite(STEPPER_DIR_PIN, HIGH);
              while (buttonStateChome == RELEASED) {
                digitalWrite(STEPPER_STEP_PIN, HIGH);
                delayMicroseconds(1000);
                digitalWrite(STEPPER_STEP_PIN, LOW);
                delayMicroseconds(1000);
                checkButtonChome();
              }
              digitalWrite(STEPPER_STEP_PIN, LOW);
              digitalWrite(STEPPER_SLEEP_PIN, LOW);
              stopAllMotors();
              currentState = REST;
              delay(1000);
            }
        break;
        case PROBLEM:
            Serial.println("PROBLEM");
            // Logic to handle problems TO DO
            // stop everything
            stopAllMotors();
        break;
    }
}
```

Header File

test\def_const_func.h

```
// def_const_func.h
// Header file for the main program. "def_const_func" stands for "definitions, constants and functions".
// Description:
// This file contains all the constants and functions used in the main program.
// The functions are defined in the file 'def_const_func.cpp'.
// The constants are defined in the file 'def_const_func.h' and are used in the main program.
// We have the following modules: IR, rain, DC brush, DC gearbox, stepper, LEDs, current, buttons, valve (servo).
// this file is divided into: Constants, Global variables, Functions.
// The main program is divided into states. Each state has its own logic.
// The states are: REST, WATER_FILLING, INITIAL_POSITION, FIRST_POSITION_CLEANING, CLEANING, UP_TRAVEL, TRANSLATION, RETURN_HOME, PROBLEM.
// The main program is in the file 'main.ino'.

#ifndef DEF_CONST_FUNC_H
#define DEF_CONST_FUNC_H

//-----------------------------------GENERAL-------------------------------------
#define CLEANING_WIDTH 30 // Effective cleaning width in cm
#define CLEANING_PERIOD 1000*60 // Time between two cleanings in milliseconds
#define DIST 100 // Total width of one panel in cm
#define DIST_TO_INITIAL_POSITION 45 // Distance to the initial position in cm
#define END_OF_CLEANING_DELAY 2000 // Delay after cleaning in milliseconds, for the wheel to get down

// Global variables
#ifndef MYSERVO_H
#define MYSERVO_H

#include <Servo.h>

extern Servo myServo; // Declaration of the Servo object

#endif

extern int step;
extern int nb_cycles_counter; // Counter for the number of cycles
extern bool raining; // Boolean indicating if it is raining
extern unsigned long currentTime; // Actual time
extern unsigned long cleaningTime; // Time of the last cleaning

// Enumeration for states
enum State { REST, WATER_FILLING, INITIAL_POSITION, CLEANING, UP_TRAVEL, TRANSLATION, RETURN_HOME, PROBLEM };
extern State currentState;

enum MotorDirection {
    UP, DOWN, LEFT, RIGHT
};

enum ButtonState { CLICKED, RELEASED };
```

```
48
49
50  //-----------------------------------IR-----------------------------------
51  // GRADING:
52  // < 20 : nothing
53  // 150-160 = at 30cm
54  // 250-280 = at 20cm
55  // 460,470 = at 10cm
56  // >600 = < 5cm. Si trop proche la valeur rebaisse.
57
58  #define IR_THRESHOLD 620
59  #define IR_SENSOR_PIN A4
60
61  extern const float IR_PERIODE;
62  extern bool IRseen;
63
64
65  //----------------------------------RAIN----------------------------------
66  #define RAIN_THRESHOLD 700 // DEFINED
67  #define RAIN_SENSOR_PIN A15
68
69  // Frequency of the rain detector
70  extern const float RAIN_SENSOR_PERIODE;
71
72
73  //---------------------------------DC_BRUSH---------------------------------
74  #define BRUSH_MOTOR_PIN1 6
75  #define BRUSH_MOTOR_PIN2 7
76
77
78  //---------------------------------DC_GEAR----------------------------------
79  #define GEARBOX_MOTOR_PIN1 4
80  #define GEARBOX_MOTOR_PIN2 5
81  #define GEARBOX_MOTOR_SPEED_PIN 12
82
83  extern const int MOTOR_SPEED_GEAR;
84  extern const float MOTOR_PERIODE;
85
86
87  //---------------------------------STEPPER----------------------------------
88  // GRADING:
89  // 20cm in 5 rev. = 4 cm per rev.
90  #define CM_PER_REVOLUTION 2.0 // 4 cm per revolution
91  #define STEPPER_STEPS_PER_REVOLUTION 200 //remove??
92
93  #define STEPPER_DIR_PIN 2
94  #define STEPPER_STEP_PIN 3
95  #define STEPPER_SLEEP_PIN 28
96  extern int stepperStartSpeed, stepperEndSpeed, stepperAccelerationSteps;
97
98
99  //----------------------------------LEDs----------------------------------
100 #define LED_INTENSITY 150
101 #define RED_LED_PIN 11
102 #define GREEN_LED_PIN 9
```

```
103 #define BLUE_LED_PIN 8
104
105 extern const float LED_PERIODE;
106
107
108 //-----------------------------------CURRENT-------------------------------------
109 #define CURRENT_SENSITIVITY 0.185 // DEFINED
110 #define CURRENT_THRESHOLD_GEARBOX 1.5 // DEFINED
111 #define CURRENT_THRESHOLD_BRUSH 0.7
112 #define CURRENT_PIN1 A14 // current sensor for gearbox motor
113 #define CURRENT_PIN2 A0 // current sensor for brush motor
114
115 extern const float CURRENT_PERIODE;
116
117
118 //-----------------------------------BUTTONS-------------------------------------
119 #define DEBOUNCE_DELAY 50 // Debounce delay in milliseconds
120
121 #define BUTTON_PIN_R 22 // End of travel button on robot 3
122 #define BUTTON_PIN_C1 24 // End of travel button on carrige 2
123 #define BUTTON_PIN_Chome 26 // End of travel button on resting station 1
124
125 // Debouncing Variables
126 extern unsigned long lastDebounceTimeR, lastDebounceTimeC1, lastDebounceTimeChome;
127 extern bool lastButtonStateR, lastButtonStateC1, lastButtonStateChome;
128
129 extern bool buttonStateR, buttonStateC1, buttonStateChome;
130
131 extern const float BUTTON_PERIODE;
132
133
134 //-----------------------------------VALVE (SERVO)-------------------------------------
135 #define SERVO_PIN 13
136 #define VALVE_ANGLE_OPEN 180 // Angle of the valve in degrees open
137 #define VALVE_ANGLE_CLOSE 0 // Angle of the valve in degrees closed
138 extern const float VALVE_PERIODE;
139
140
141 //===================================FUNCTION DECLARATIONS================================
    =====
142 // Initialization functions
143 void initializeMotors();
144 void initializeButtons();
145 void initializeLEDPins();
146 void initializeRainSensor();
147 void initializeStepperMotor();
148 void initializeCurrentSensors();
149
150
151 // State functions
152 //-----------------------------------IR-------------------------------------
153 void checkIRSensor();
154
155 //-----------------------------------RAIN-------------------------------------
156 void checkRainSensor();
```

```
157
158 //-----------------------------------DC_BRUSH-------------------------------------
159 void controlBrushMotor(bool direction);
160
161 //-----------------------------------DC_GEAR--------------------------------------
162 void controlGearboxMotor(bool direction, int speed);
163
164 //-----------------------------------STEPPER--------------------------------------
165 void controlStepper(int distance, bool clockwise, int stepperStartSpeed, int
    stepperEndSpeed, int stepperAccelerationSteps);
166 void controlStepper(int distance, bool clockwise);
167
168 //-----------------------------------LEDs-----------------------------------------
169 void updateLEDs(State currentState);
170 void redLED(bool flashing);
171 void greenLED(bool flashing);
172 void blueLED(bool flashing);
173 void whiteLED(bool flashing);
174 void setColor(int red, int green, int blue, bool flashing);
175 void writeColor(int red, int green, int blue);
176
177 //-----------------------------------CURRENT--------------------------------------
178 void checkCurrent(int currentPin);
179
180 //-----------------------------------BUTTONS--------------------------------------
181 void checkButtonR();
182 void checkButtonC1();
183 void checkButtonChome();
184
185 //-----------------------------------VALVE (Servo)--------------------------------
186 void controlValve(int angle);
187 void rotateServo(int angle);
188
189
190 //-----------------------------------MOTORS MOUVEMENT-----------------------------------
191 void moveMotor(MotorDirection direction, float distanceOrSpeed);
192
193
194 //-----------------------------------STOP ALL MOTORS------------------------------
195 void stopAllMotors();
196
197 #endif // DEF_CONST_FUNC_H
```

Cpp File

test\def_const_func.cpp

```cpp
// def_const_func.cpp
#include <Arduino.h>
#include "def_const_func.h"
#include <Time.h>
#include <Servo.h>

//-----------------------------------GOBAL DEFINITIONS-----------------------------------
Servo myServo;  // Create a Servo object

int step = DIST / CLEANING_WIDTH + (DIST % CLEANING_WIDTH != 0); // In cm
int nb_cycles_counter = 0;
bool raining = false; // Boolean indicating if it is raining
bool IRseen = false; // Value of the IR sensor
unsigned long currentTime; // Actual time
unsigned long cleaningTime = 0.0; // Time of the last cleaning
int stepperStartSpeed = 1500, stepperEndSpeed = 1000, stepperAccelerationSteps = 500;
const float IR_PERIODE = 1.0; // Periode in milliseconds
const float RAIN_SENSOR_PERIODE = 1000.0; // Periode in milliseconds
const int MOTOR_SPEED_BRUSH = 200; // Brush motor speed
const int MOTOR_SPEED_GEAR = 200; // Gearbox motor speed
const float MOTOR_PERIODE = 100.0; // Periode in milliseconds
const float LED_PERIODE = 100.0; // Periode in milliseconds
const float CURRENT_PERIODE = 500.0; // Periode in milliseconds
const float BUTTON_PERIODE = 30.0; // Periode in milliseconds
const float VALVE_PERIODE = 1000.0; // Periode in milliseconds
bool buttonStateR = RELEASED, buttonStateC1 = RELEASED, buttonStateChome = CLICKED; // Current state of buttons. state is true when not pressed.
// Debouncing Variables
unsigned long lastDebounceTimeR = 0, lastDebounceTimeC1 = 0, lastDebounceTimeChome = 0; // Last debounce time for buttons
bool lastButtonStateR = RELEASED, lastButtonStateC1 = RELEASED, lastButtonStateChome = CLICKED; // Previous state of buttons


State currentState = REST; // Initial state

//-----------------------------------END GLOBAL DEFINITIONS-----------------------------------

void initializeMotors() {
    // Initialize gearbox motor pins
    pinMode(GEARBOX_MOTOR_PIN1, OUTPUT);
    pinMode(GEARBOX_MOTOR_PIN2, OUTPUT);

    // Initialize brush motor pins
    pinMode(BRUSH_MOTOR_PIN1, OUTPUT);
    pinMode(BRUSH_MOTOR_PIN2, OUTPUT);
}

void initializeButtons() {
    pinMode(BUTTON_PIN_R, INPUT);
    pinMode(BUTTON_PIN_C1, INPUT);
```

```
    pinMode(BUTTON_PIN_Chome, INPUT);
}

void initializeLEDPins() {
    pinMode(RED_LED_PIN, OUTPUT);
    pinMode(GREEN_LED_PIN, OUTPUT);
    pinMode(BLUE_LED_PIN, OUTPUT);
}

void initializeRainSensor() {
    pinMode(RAIN_SENSOR_PIN, INPUT);
}

void initializeStepperMotor() {
    pinMode(STEPPER_DIR_PIN, OUTPUT);
    pinMode(STEPPER_STEP_PIN, OUTPUT);
}

void initializeCurrentSensors() {
    pinMode(CURRENT_PIN1, INPUT);
    pinMode(CURRENT_PIN2, INPUT);
}


//-----------------------------------IR-------------------------------------
void checkIRSensor() {
    static unsigned long last_time = 0;
    if (millis() - last_time >= IR_PERIODE) {
        float irValue = analogRead(IR_SENSOR_PIN);
        if (irValue >= IR_THRESHOLD) {
            IRseen = true;
            Serial.println("Seen");
        } else {
            IRseen = false;
            Serial.println("Not Seen");
        }
        last_time = millis();
    }
}


//-----------------------------------RAIN-------------------------------------
void checkRainSensor() {
    static unsigned long last_time = 0;
    if (millis() - last_time >= RAIN_SENSOR_PERIODE) {
        int rainValue = analogRead(RAIN_SENSOR_PIN);
        if (rainValue <= RAIN_THRESHOLD) {
            Serial.println("Raining");
        } else {
            Serial.println("Not Raining");
        }
        Serial.println(rainValue);
        last_time = millis();
    }
}
```

```cpp

//-----------------------------------DC_BRUSH-----------------------------------
void controlBrushMotor(bool direction) {
    digitalWrite(BRUSH_MOTOR_PIN1, direction ? HIGH : LOW);
    digitalWrite(BRUSH_MOTOR_PIN2, direction ? LOW : HIGH);
}


//-----------------------------------DC_GEAR-----------------------------------
void controlGearboxMotor(bool direction) {
    digitalWrite(GEARBOX_MOTOR_PIN1, direction ? HIGH : LOW);
    digitalWrite(GEARBOX_MOTOR_PIN2, direction ? LOW : HIGH);
}


//-----------------------------------STEPPER-----------------------------------
void controlStepper(int distance, bool clockwise, int stepperStartSpeed, int stepperEndSpeed, int stepperAccelerationSteps) {
    digitalWrite(STEPPER_SLEEP_PIN, HIGH);
    delayMicroseconds(2);

    int totalSteps = distance / CM_PER_REVOLUTION * STEPPER_STEPS_PER_REVOLUTION;
    int stepDelay = stepperStartSpeed;
    int stepChange = (stepperStartSpeed - stepperEndSpeed) / stepperAccelerationSteps;

    digitalWrite(STEPPER_DIR_PIN, clockwise ? HIGH : LOW);

    // Accelerate
    for (int i = 0; i < stepperAccelerationSteps && i < totalSteps; i++) {
        digitalWrite(STEPPER_STEP_PIN, HIGH);
        delayMicroseconds(stepDelay);
        digitalWrite(STEPPER_STEP_PIN, LOW);
        delayMicroseconds(stepDelay);
        stepDelay -= stepChange;
    }

    // Constant speed
    for (int i = stepperAccelerationSteps; i < totalSteps - stepperAccelerationSteps; i++)
{
        digitalWrite(STEPPER_STEP_PIN, HIGH);
        delayMicroseconds(stepperEndSpeed);
        digitalWrite(STEPPER_STEP_PIN, LOW);
        delayMicroseconds(stepperEndSpeed);
    }

    // Decelerate
    for (int i = totalSteps - stepperAccelerationSteps; i < totalSteps; i++) {
        digitalWrite(STEPPER_STEP_PIN, HIGH);
        delayMicroseconds(stepDelay);
        digitalWrite(STEPPER_STEP_PIN, LOW);
        delayMicroseconds(stepDelay);
        stepDelay += stepChange;
    }

```

```
157     digitalWrite(STEPPER_STEP_PIN, LOW);
158     digitalWrite(STEPPER_SLEEP_PIN, LOW);
159 }
160 
161 void controlStepper(int distance, bool clockwise) {
162     digitalWrite(STEPPER_SLEEP_PIN, HIGH);
163     delayMicroseconds(2);
164 
165 
166     //int totalSteps = distance / CM_PER_REVOLUTION * STEPPER_STEPS_PER_REVOLUTION;
167     int totalSteps = distance*STEPPER_STEPS_PER_REVOLUTION;
168     digitalWrite(STEPPER_DIR_PIN, clockwise ? LOW : HIGH);
169 
170     // Constant speed
171     for (int i = 0; i < totalSteps; i++) {
172         digitalWrite(STEPPER_STEP_PIN, HIGH);
173         delayMicroseconds(stepperEndSpeed);
174         digitalWrite(STEPPER_STEP_PIN, LOW);
175         delayMicroseconds(stepperEndSpeed);
176     }
177 
178     digitalWrite(STEPPER_STEP_PIN, LOW);
179     digitalWrite(STEPPER_SLEEP_PIN, LOW);
180 }
181 
182 
183 //-----------------------------------LEDs-----------------------------------
184 void updateLEDs(State currentState) {
185     static unsigned long last_time = 0;
186     if (millis() - last_time >= LED_PERIODE) {
187         switch (currentState)
188         {
189         case REST:
190             blueLED(false);
191             break;
192         case WATER_FILLING:
193             blueLED(true);
194             break;
195         case INITIAL_POSITION:
196             whiteLED(false);
197             break;
198         case CLEANING:
199             greenLED(true);
200             break;
201         case UP_TRAVEL:
202             greenLED(false);
203             break;
204         case TRANSLATION:
205             whiteLED(true);
206             break;
207         case RETURN_HOME:
208             redLED(false);
209             break;
210         case PROBLEM:
211             redLED(true);
```

```
            break;
        default:
            setColor(0,0,0,false);
            break;
        }
        last_time = millis();
    }
}

void redLED(bool flashing) {
    setColor(LED_INTENSITY, 0, 0, flashing);
}

void greenLED(bool flashing) {
    setColor(0, LED_INTENSITY, 0, flashing);
}

void blueLED(bool flashing) {
    setColor(0, 0, LED_INTENSITY, flashing);
}

void whiteLED(bool flashing) {
    setColor(LED_INTENSITY, LED_INTENSITY, LED_INTENSITY, flashing);
}

void setColor(int red, int green, int blue, bool flashing) {
    const int flashDuration = 500; // Duration of each flash
    if (flashing) {
        for (int i = 0; i < 5; i++) {
            writeColor(red, green, blue);
            delay(flashDuration);
            writeColor(0, 0, 0);
            delay(flashDuration);
        }
    } else {
        writeColor(red, green, blue);
    }
}

void writeColor(int red, int green, int blue) {
    analogWrite(RED_LED_PIN, red);
    analogWrite(GREEN_LED_PIN, green);
    analogWrite(BLUE_LED_PIN, blue);
}


//-----------------------------------CURRENT-------------------------------------
void checkCurrent(int currentPin) {
    static unsigned long last_time = 0;
    if (millis() - last_time >= CURRENT_PERIODE) {
        float currentVal = analogRead(currentPin);
        currentVal = (2.5 - (currentVal * (5.0 / 1024.0))) / CURRENT_SENSITIVITY;
        currentVal = abs(currentVal);
        Serial.print("Current Value: ");
        Serial.print(currentVal, 5);
```

```
267         Serial.println(" [A]");
268         if (currentPin == CURRENT_PIN1)
269         {
270             if (currentVal >= CURRENT_THRESHOLD_GEARBOX) {
271                 Serial.println("Current threshold exceeded gearbox");
272                 currentState = PROBLEM;
273             }
274         }
275         else if (currentPin == CURRENT_PIN2)
276         {
277             if (currentVal >= CURRENT_THRESHOLD_BRUSH) {
278                 Serial.println("Current threshold exceeded brush");
279                 currentState = PROBLEM;
280             }
281         }
282         else
283         {
284             Serial.println("Error: Current pin not defined");
285         }
286         last_time = millis();
287     }
288 }
289
290
291 //-----------------------------------BUTTONS---------------------------------
292 void checkButtonR() {
293     static unsigned long last_time = 0;
294     if (millis() - last_time >= BUTTON_PERIODE) {
295         // Debounce button 1
296         bool readingR = digitalRead(BUTTON_PIN_R);
297         if (readingR != lastButtonStateR) {
298             lastDebounceTimeR = millis();
299         }
300         if ((millis() - lastDebounceTimeR) > DEBOUNCE_DELAY) {
301             if (readingR != buttonStateR) {
302                 buttonStateR = readingR;
303                 Serial.println("Button R changed");
304             }
305         }
306         lastButtonStateR = readingR;
307         last_time = millis();
308     }
309 }
310
311 void checkButtonChome() {
312     static unsigned long last_time = 0;
313     if (millis() - last_time >= BUTTON_PERIODE) {
314         // Debounce button 1
315         bool readingChome = digitalRead(BUTTON_PIN_Chome);
316         if (readingChome != lastButtonStateChome) {
317             lastDebounceTimeChome = millis();
318         }
319         if ((millis() - lastDebounceTimeChome) > DEBOUNCE_DELAY) {
320             if (readingChome != buttonStateChome) {
321                 buttonStateChome = readingChome;
```

```
                Serial.println("Button Chome changed");
            }
        }
        lastButtonStateChome = readingChome;
        last_time = millis();
    }
}

void checkButtonC1() {
    static unsigned long last_time = 0;
    if (millis() - last_time >= BUTTON_PERIODE) {
        // Debounce button 1
        bool readingC1 = digitalRead(BUTTON_PIN_C1);
        if (readingC1 != lastButtonStateC1) {
            lastDebounceTimeC1 = millis();
        }
        if ((millis() - lastDebounceTimeC1) > DEBOUNCE_DELAY) {
            if (readingC1 != buttonStateC1) {
                buttonStateC1 = readingC1;
                Serial.println("Button C1 changed");
            }
        }
        lastButtonStateC1 = readingC1;
        last_time = millis();
    }
}


//-----------------------------------VALVE (Servo)-----------------------------------
void controlValve(int angle) {
    static unsigned long last_time = 0;
    if (millis() - last_time >= VALVE_PERIODE) {
        rotateServo(angle);
        last_time = millis();
    }
}

void rotateServo(int angle) {
    // Convert angle to microseconds
    if (angle >= 0 && angle <= 180) {
        // You can adjust these values if your servo has a different range
        int minPulseWidth = 0;
        int maxPulseWidth = 5000;
        int pulseWidth = map(angle, 0, 180, minPulseWidth, maxPulseWidth); // Map angle to
microseconds
        myServo.writeMicroseconds(pulseWidth);
        Serial.print("Angle: ");
        Serial.println(angle);
    } else {
        Serial.println("Angle out of range");
    }
}


//-----------------------------------MOTORS MOUVEMENT-----------------------------------
```

```
376 void moveMotor(MotorDirection direction, float distanceOrSpeed) {
377     static unsigned long last_time = 0;
378     if (millis() - last_time >= MOTOR_PERIODE) {
379         switch (direction) {
380             case UP:
381                 controlGearboxMotor(true); // Fixed speed
382                 break;
383             case DOWN:
384                 controlGearboxMotor(false); // Fixed speed
385                 break;
386             case LEFT:
387                 // controlStepper(distanceOrSpeed, false, stepperStartSpeed,
    stepperEndSpeed, stepperAccelerationSteps); // Assuming false is left
388                 controlStepper(distanceOrSpeed, false); // Assuming false is left
389                 break;
390             case RIGHT:
391                 // controlStepper(distanceOrSpeed, true, stepperStartSpeed,
    stepperEndSpeed, stepperAccelerationSteps); // Assuming true is right
392                 controlStepper(distanceOrSpeed, true); // Assuming true is right
393                 break;
394         }
395         last_time = millis();
396     }
397 }
398
399
400 //-----------------------------------STOP ALL MOTORS-----------------------------------
401 void stopAllMotors() {
402     // Stop DC brush motor
403     digitalWrite(BRUSH_MOTOR_PIN1, LOW);
404     digitalWrite(BRUSH_MOTOR_PIN2, LOW);
405
406     // Stop Gearbox motor
407     digitalWrite(GEARBOX_MOTOR_PIN1, LOW);
408     digitalWrite(GEARBOX_MOTOR_PIN2, LOW);
409 }
```

Test/Demo File

test\test.ino

```
#include <Time.h>
#include "def_const_func.h"
int cont = 0;
int ir_thresh = 0;

void setup() {
    //void initializeSerialCommunication();
    Serial.begin(9600);
    //myServo.attach(SERVO_PIN);
    //controlValve(180);
    void initializeMotors();
    void initializeButtons();
    void initializeLEDPins();
    void initializeRainSensor();
    void initializeStepperMotor();
    void initializeCurrentSensors();
}

void loop() {
  //FINAL DEMO
  delay(2000);
  if (cont == 0) {
    calibrate_ir();
    delay(1000);
    // INITIAL POSITION
    updateLEDs(INITIAL_POSITION);
    moveMotor(LEFT, 11);
    // 10 rev = 41 cm
    // 43.5 cm de plateforme a première position
    stopAllMotors();
    delay(1000);
    float switchr = digitalRead(BUTTON_PIN_R);
    while (switchr == 1) {
      moveMotor(UP,0);
      switchr = digitalRead(BUTTON_PIN_R);
    }
    stopAllMotors();
    delay(1000);
    // CLEANING
    updateLEDs(CLEANING);
    moveMotor(DOWN,0);
    controlBrushMotor(true);
    //controlValve(0);
    switchr = digitalRead(BUTTON_PIN_R);
    while (switchr == 0) {
      moveMotor(DOWN,0);
      controlBrushMotor(true);
      switchr = digitalRead(BUTTON_PIN_R);
      Serial.println(switchr);
    }
    digitalWrite(BRUSH_MOTOR_PIN1, LOW);
    digitalWrite(BRUSH_MOTOR_PIN2, LOW);
```

```
    //controlValve(180);
    delay(200);
    digitalWrite(GEARBOX_MOTOR_PIN1, LOW);
    digitalWrite(GEARBOX_MOTOR_PIN2, LOW);
    // UP TRAVEL
    updateLEDs(UP_TRAVEL);
    float ir_value = analogRead(IR_SENSOR_PIN);
    while (ir_value < ir_thresh) {
      moveMotor(UP, 0);
      ir_value = analogRead(IR_SENSOR_PIN);
      Serial.println(ir_value);
    }
    stopAllMotors();
    cont = cont + 1;
  }
  if (cont == 6) {
    // TRANSLATION TO LAST POSITION
    float switchr = digitalRead(BUTTON_PIN_C1);
    digitalWrite(STEPPER_SLEEP_PIN, HIGH);
    delayMicroseconds(2);
    digitalWrite(STEPPER_DIR_PIN, HIGH);
    while (switchr == 1) {
      digitalWrite(STEPPER_STEP_PIN, HIGH);
      delayMicroseconds(1000);
      digitalWrite(STEPPER_STEP_PIN, LOW);
      delayMicroseconds(1000);
      switchr = digitalRead(BUTTON_PIN_C1);
    }
    digitalWrite(STEPPER_SLEEP_PIN, LOW);
    delay(1000);
    switchr = digitalRead(BUTTON_PIN_R);
    while (switchr == 1) {
      moveMotor(UP,0);
      switchr = digitalRead(BUTTON_PIN_R);
    }
    stopAllMotors();
    delay(1000);
    // CLEANING
    updateLEDs(CLEANING);
    moveMotor(DOWN,0);
    controlBrushMotor(true);
    //controlValve(0);
    switchr = digitalRead(BUTTON_PIN_R);
    while (switchr == 0) {
      moveMotor(DOWN,0);
      controlBrushMotor(true);
      switchr = digitalRead(BUTTON_PIN_R);
      Serial.println(switchr);
    }
    digitalWrite(BRUSH_MOTOR_PIN1, LOW);
    digitalWrite(BRUSH_MOTOR_PIN2, LOW);
    //controlValve(180);
    delay(200);
    digitalWrite(GEARBOX_MOTOR_PIN1, LOW);
    digitalWrite(GEARBOX_MOTOR_PIN2, LOW);
```

```
    delay(1000);
    // UP TRAVEL
    updateLEDs(UP_TRAVEL);
    float ir_value = analogRead(IR_SENSOR_PIN);
    while (ir_value < ir_thresh) {
      moveMotor(UP, 0);
      ir_value = analogRead(IR_SENSOR_PIN);
      Serial.println(ir_value);
    }
    stopAllMotors();
    cont = cont + 1;
    delay(1000);
    // GO BACK HOME
    updateLEDs(RETURN_HOME);
    switchr = digitalRead(BUTTON_PIN_Chome);
    digitalWrite(STEPPER_SLEEP_PIN, HIGH);
    delayMicroseconds(2);
    digitalWrite(STEPPER_DIR_PIN, LOW);
    while (switchr == 1) {
      digitalWrite(STEPPER_STEP_PIN, HIGH);
      delayMicroseconds(1000);
      digitalWrite(STEPPER_STEP_PIN, LOW);
      delayMicroseconds(1000);
      switchr = digitalRead(BUTTON_PIN_Chome);
    }
    digitalWrite(STEPPER_SLEEP_PIN, LOW);
    // END PROGRAM
    exit;
  }
  delay(1000);
  // TRANSLATION
  updateLEDs(TRANSLATION);
  moveMotor(LEFT, 7);
  stopAllMotors();
  delay(1000);
  float switchr = digitalRead(BUTTON_PIN_R);
  while (switchr == 1) {
    moveMotor(UP,0);
    switchr = digitalRead(BUTTON_PIN_R);
  }
  stopAllMotors();
  delay(1000);
  // CLEANING
  updateLEDs(CLEANING);
  moveMotor(DOWN,0);
  controlBrushMotor(true);
  //controlValve(0);
  switchr = digitalRead(BUTTON_PIN_R);
  while (switchr == 0) {
    moveMotor(DOWN,0);
    controlBrushMotor(true);
    switchr = digitalRead(BUTTON_PIN_R);
    Serial.println(switchr);
  }
  digitalWrite(BRUSH_MOTOR_PIN1, LOW);
```

```
  digitalWrite(BRUSH_MOTOR_PIN2, LOW);
  //controlValve(180);
  delay(200);
  digitalWrite(GEARBOX_MOTOR_PIN1, LOW);
  digitalWrite(GEARBOX_MOTOR_PIN2, LOW);
  delay(1000);
  // UP TRAVEL
  updateLEDs(UP_TRAVEL);
  float ir_value = analogRead(IR_SENSOR_PIN);
  while (ir_value < ir_thresh) {
    moveMotor(UP, 0);
    ir_value = analogRead(IR_SENSOR_PIN);
    Serial.println(ir_value);
  }
  stopAllMotors();
  cont = cont + 1;
  delay(1000);
  // GO TO NEXT POSITION
  //END FINAL DEMO
}

void calibrate_ir () {
  int mean = 0;
  for (int i = 0; i < 10; i++) {
    float ir_value = analogRead(IR_SENSOR_PIN);
    Serial.println(ir_value);
    mean = mean + ir_value;
    Serial.println(mean);
    delay(500);
  }
  ir_thresh = mean/10;
  ir_thresh = ir_thresh - 25;
  Serial.println(ir_thresh);
}
```