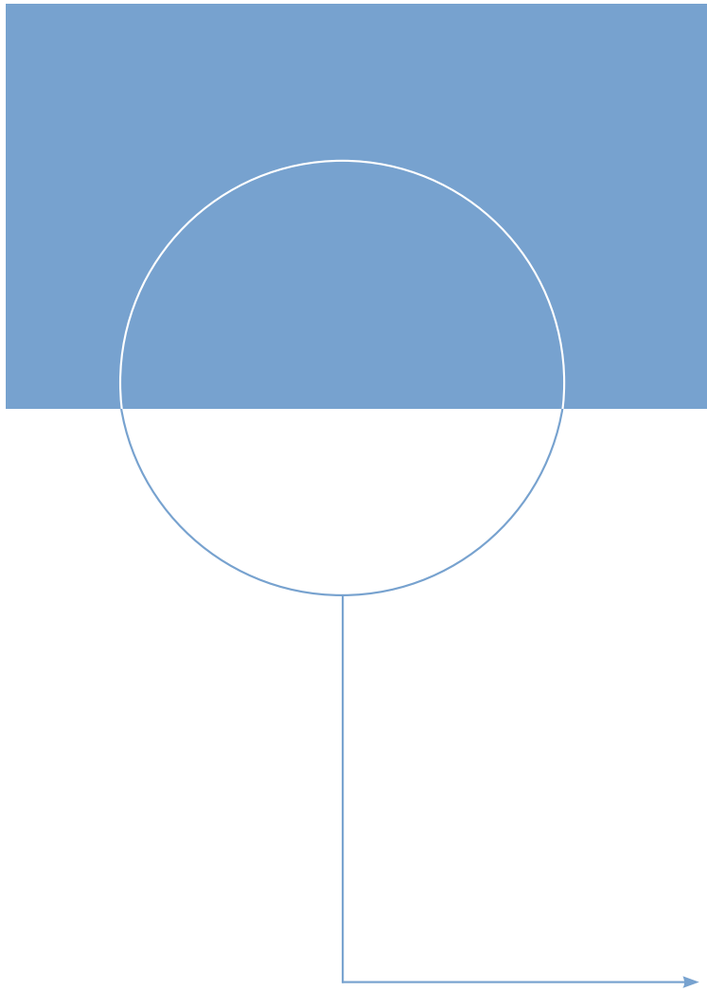


Master's thesis

2022

Victor Andreu Bañuls Ramirez

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Electric Power Engineering

Victor Andreu Bañuls Ramirez

DEVELOPMENT AND PROTOTYPING OF TWO-AXIS SOLAR TRACKER ARRAYS WITH A DECENTRALIZED DRIVE SYSTEM

July 2022

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

DEVELOPMENT AND PROTOTYPING OF TWO-AXIS SOLAR TRACKER ARRAYS WITH A DECENTRALIZED DRIVE SYSTEM

Victor Andreu Bañuls Ramirez

Erasmus plus
Submission date: July 2022
Supervisor: Steve Völler
Co-supervisor: Ángel Sapena Bañó

Norwegian University of Science and Technology
Department of Electric Power Engineering

SUMMARY

Solar trackers are a technology that improves the efficiency of solar farms. This is due to the fact that they allow for an increase in the peak power and the temporary spacing of the production. The main drawback of these types of trackers is their high manufacturing cost compared to fixed solar trackers. The disadvantages of these devices and their high cost make it pertinent to design an improvement proposal.

Existing two-axis solar trackers behave as mobile supports incorporating all the necessary components to work as complete units in themselves. Its components can be divided into three categories, control system, drive system and structure. If you have all the trackers in the same location they should move synchronously. This allows you to centralize the control and drive systems. It is by centralizing these systems that the manufacturing costs of the trackers are intended to be reduced.

A small-scale prototype was fabricated to prove that such a concept of centralizing systems is possible. For this purpose, a CAD software and a 3d printer were used to manufacture the structural parts. For the movement system it was used a system based on 4 stepper motors that transmit their motion to the structure through the use of cables and pulleys. An Arduino Mega was used for receiving and sending commands or data to sensors, web servers and motors. For the control and monitoring of the prototype a web platform was established. In the design process different tools and methods were used, among which are: sensor characterizations, vector simulations in Matlab, and programming of motion algorithms.

Performance tests were carried out on the prototype and on the web platforms. From these it was possible to demonstrate the functionality of the design as well as to obtain the maximum rotation angles. These turn angles were of has 360º of azimuth turn and 80º of elevation turn. The functionality of the web platforms was also successfully tested.

Once the design concept was demonstrated, it was necessary to prove that centralizing systems resulted in a cost reduction. To this end, a full-scale manufacturing cost model was created. For this purpose it was used a CFD and structural software. This allowed him to obtain the load and stress data needed for sizing. With this study it was possible to determine the relationships of the variables on which the cost estimation depended. Among the relationships it was able to determinate the impact of selecting an operating wind speed limit, or the relationship of the number of trackers in series with the relative improvement in manufacturing costs. Finally, in order to verify that these costs were reduced, the model created was compared with the manufacturing costs of the ST2408PH tracker model. Financially, this comparison yielded a 23.5% actual cost reduction.

SAMMENDRAG

Solar trackers er en teknologi som forbedrer effektiviteten til solfarmer. Dette skyldes det faktum at de åpner for en økning i toppeffekten og den midlertidige avstanden til produksjonen. Den største ulempen med disse typer trackere er deres høye produksjonskostnad sammenlignet med fast monterte solceller. Ulempene med disse enhetene og deres høye kostnader gjør det relevant å utforme et forbedringsforslag.

Eksisterende to-akse solcelletrackere oppfører seg som mobile støtter som inneholder alle nødvendige komponenter for å fungere som komplette enheter. Komponentene kan deles inn i tre kategorier, kontrollsystem, drivsystem og struktur. Hvis alle trackerene er på samme sted, beveger de seg synkront. Dette lar deg sentralisere kontroll- og drivsystemene. Det er ved å sentralisere disse systemene at produksjonskostnadene til trackerne kan reduseres.

En småskala prototype ble laget for å bevise at et slikt konsept med sentralisering av systemer er mulig. Til dette formålet ble CAD-programvare og en 3D-printer brukt til å produsere konstruksjonsdelene. For bevegelsessystemet ble det brukt et system basert på 4 trinnmotorer som overfører bevegelsen til strukturen gjennom bruk av kabler og trinser. En Arduino Mega ble brukt til å motta og sende kommandoer eller data til sensorer, webservere og motorer. For kontroll og overvåking av prototypen ble det etablert en webplattform. I designprosessen ble forskjellige verktøy og metoder brukt, blant annet: sensorkarakteriseringer, vektorsimuleringer i Matlab og programmering av bevegelsesalgoritmer.

Ytelsestester ble utført på prototypen og på nettplattformene. Fra disse var det mulig å demonstrere funksjonaliteten til designet samt å oppnå maksimale rotasjonsvinkler. Disse svingvinklene har 360º asimutsving og 80º høydesving. Funksjonaliteten til nettplattformene ble også testet.

Når designkonseptet ble demonstrert, var det nødvendig å bevise at sentralisering av systemer resulterte i en kostnadsreduksjon. For dette formål ble en fullskala produksjonskostnadsmodell opprettet. Til dette formålet ble det brukt en CFD og strukturell programvare. Dette tillot meg å få tak i belastnings- og stressdataene som trengs for dimensjonering. Med denne studien var det mulig å bestemme sammenhengene mellom variablene som kostnadsestimatet var avhengig av. Blant forholdene var det i stand til å bestemme virkningen av å velge en driftsvindhastighetsgrense, eller forholdet mellom antall trackere i serie med den relative forbedringen i produksjonskostnadene. Til slutt, for å verifisere at disse kostnadene ble redusert, ble modellen, som ble opprettet, sammenlignet med produksjonskostnadene til ST2408PH tracker-modellen. Økonomisk ga denne sammenligningen en kostnadsreduksjon på 23,5 %.

LIST OF FIGURES

LIST OF TABLES

ABREVIATIONS

PLA	Polylactide
.stl	File format native to the stereolithography CAD software
MPPT	Maximum power point tracking
LDR	Light Depending Resistor
AA	Azimuth angle
EA	Elevation angle
AI	Angle increment

AD	Azimuth angle destination
ED	Elevation angle destination
CFD	Computational fluid dynamics
PP	Panel profile
EA	Elevation angle
PD	Maximum distance between upper and lower pulleys
SD	Distance between supports
RS	Number of supports at the rear
FS	Number of supports at the front
NM	Number of motors
CS	Control system
NS	Nema stepper
SC	Steel cable
D	Drum
DP	Double pulley
SS	Steel structure
SN	Seafreight to Norway
n	Number of solar trackers per array
TCR	Total cable required
BPD	Budget for the proposed design
BMM	Manufacturing budget of the ST2408PH (n)
F_D	Drag force
C_D	Drag coefficient
ρ	Density
A	Area

CONTENTS

CHAPTER 1: CONTEXT AND JUSTIFICATION

The use of solar trackers is becoming more and more frequent in photovoltaic plants, as the solar industry has been able to verify the great advantages they offer. Solar trackers can significantly increase energy production, and therefore improve the profitability of projects and the return on investment.

Before going into the objectives and operation of solar trackers, it is essential to explain the sun's movements and understand how they directly affect solar energy production. The rotation and translation movements of the earth are responsible for the seasons, the succession of days and nights, and the temperature differences between different points of the planet. Solar radiation depends on these movements and will vary according to latitude and time of year.

The position of the sun, which directly affects the angle of incidence of the sun's rays, is determined by the elevation and azimuth angles.

Depending on the season of the year, the Sun draws different trajectories. To analyze this movement, a coordinate system with two angles is used:

- Altitude angle: it is the angle formed by the horizontal of the site with the apparent position of the Sun. Its value varies between 0 and 90º. The solar height determines the optimum tilt of the panels. Hereinafter it will refer to this angle as elevation angle.
- Azimuth angle: is the angular value with the deviation of the normal to the surface from the local meridian, the origin of angles being South (azimuth angle = 0º), taking East and West as positive and negative respectively. That is, the angle determined by the projection of the solar vector on the horizontal plane and the South direction. Its value varies between 0 and ±180º. Thus, it has positive values of azimuthal angle before the measured solar day and negative values after the half solar day. The azimuth angle is the one that determines the orientation that the panels should have.

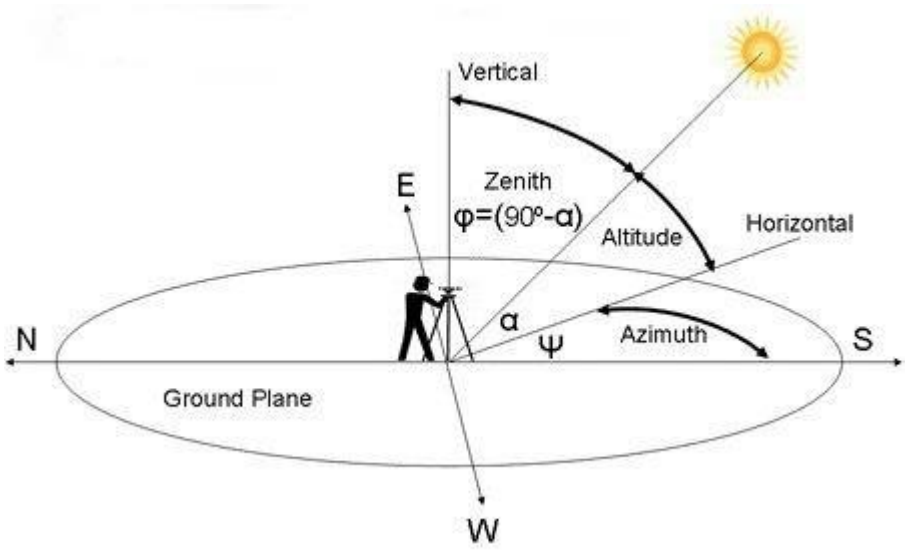


Figure 1: Azimuth and altitude for northern latitudes[1]

Figure 2 reflects the incidence of the sun's rays in the summer and winter seasons, both at noon and on a horizontal surface. As it can be seen, in summer the sun follows an orbit that causes the sun to be at a very high point in the sky and to remain visible for more hours of the day. Thus, the sun's rays strike more perpendicularly and with a much higher energy yield. On the contrary, in winter the inclination is maximum, and the energetic power of the sun's rays is minimum. Moreover, due to the rotational motion, the sun remains at a very low point on the horizon and is visible only a few hours a day.[2]

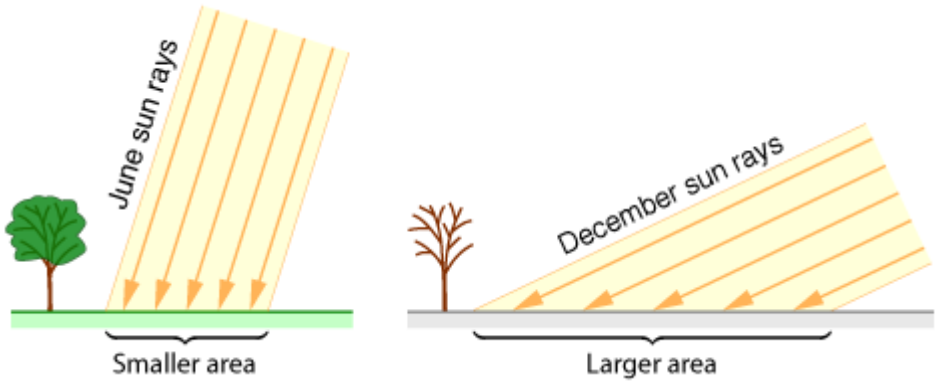


Figure 2: Incidence of the sun's rays in summer and winter seasons[2].

This effect is accentuated as one moves away from the equator. That is, in regions close to the polar circles, the daylights are very long in summer and very short in winter, while in other areas closer to the equator, there is much less variation between night and day.

Thus, the daily and annual solar trajectories of a location vary according to latitude and directly affect the angle of incidence of radiation[3]. Latitude has a very significant effect on the range of azimuth angles that the sun adopts, reaching a variability of 360 degrees at some points of the planet. The higher the latitude, the greater the azimuth angle variability. This has a resulted that in certain periods of the year and certain regions of the planet the sun does not set in summer or does not appear in winter. This means that in regions of the planet that are located especially to the north or south, the incorporation of solar supports that follow the sun substantially increases the overall efficiency of producing photovoltaic energy.

As far as this master thesis is concerned, the design will be specialized for the city of Trondheim, Norway. This city is located at 63 degrees north, and 10 degrees east. In Figure 3, it can be seen the different variability of the sun's position in Trondheim on June 21, compared to another location 35 degrees south on the same day. The yellow crescent-shaped area represents the path of the sun in a horizontal plane. The thickness of this moon depends on the angle of elevation of the sun, the greater the thickness the higher the elevation. In the interior of the moon one can see the trajectory of the

different azimuth angles taken by the sun. The orange lines mark the azimuth angle limits that the sun would take for the particular day and locations.

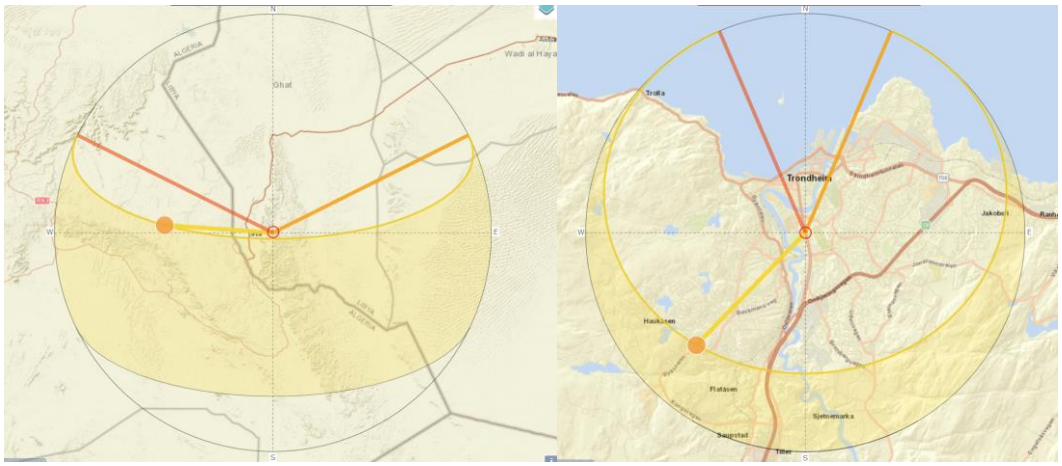


Figure 3: Variability of the sun's position for two different locations on June 21, midsummer[4]

Solar trackers are composed of a fixed and movable part, to follow the movement of the sun and orient the area of the modules perpendicular to the sun. In this way, the collection of solar radiation and consequently the energy supplied by the installation can be increased. There are two types of trackers depending on the type of movement they perform and their tracking algorithm:

- Single-axis trackers: the rotation of the collection surface is made on a single axis, which can be horizontal, vertical, or oblique. Thus, these trackers move along the azimuth from east to west during the day.
- Dual-axis trackers: in addition to moving along the azimuth, they also track the sun's elevation angle, thus achieving full tracking.

This thesis will focus on two-axis solar trackers.

1.1 DUAL-AXIS TRACKERS

There are two predominant two-axis solar tracker designs in the market. The slew driver-based design and the actuator-based design. Both designs are formed by a mechanism type structure made of normally steel and aluminum. These structures allow for two degrees of rotation. In addition, these designs incorporate their own controller, sensor system and motion system. The main differences that characterize each design are described below.

1.1.1 SLEW DRIVER BASED DESIGN

This design uses two different types of drive systems. For azimuth axis motion, it incorporates slew drivers in what would be the main tower. A slew driver is a gearbox that can safely support radial and axial loads as well as transmit torque. Slew drives are made by manufacturing gears, bearings, seals, housings, motors and other ancillary components and assembling them into a finished gearbox. By adopting the slewing bearing as its core component, the slewing drive can support axial force, radial force and tilting moment simultaneously. This gearbox normally allows turns of between 280/310 degrees. A linear actuator is used for the lift slewing. This is an electrically powered hydraulic piston. These pistons have a high power to size ratio. In addition, its power consumption is very low, partly because it does not require energy expenditure to stay in

a particular position. It has a degree of freedom of movement and has a continuous movement. Its implementation in solar trackers allows about 80 degrees of rotation on one axis.

This design is more intended to incorporate a large number of panels in series thanks to the incorporation of a slew driver which allows it to withstand large loads. In the following image one can see an example of this design:

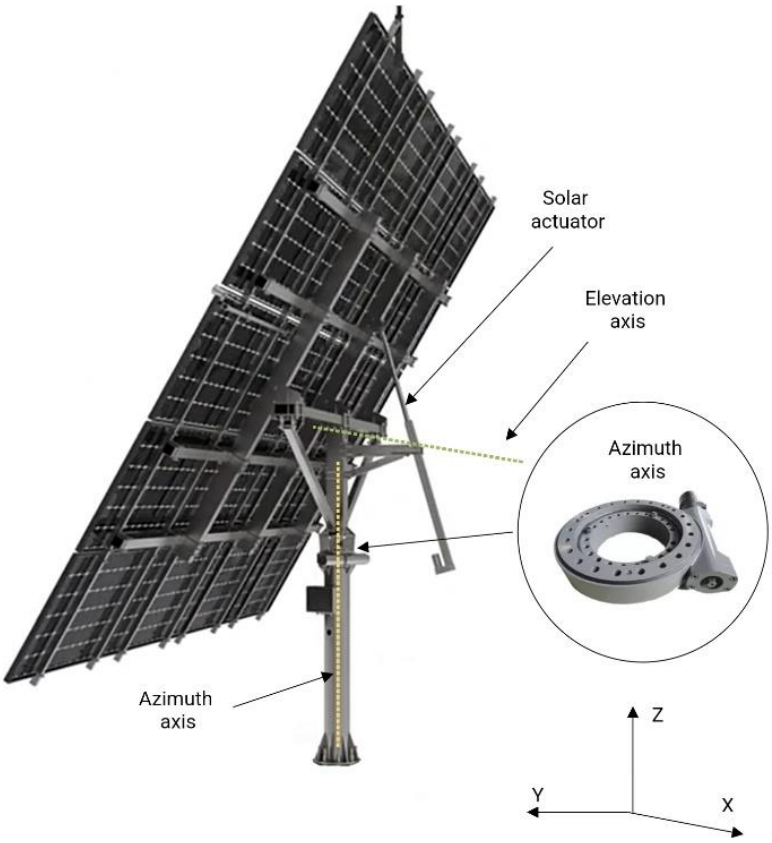


Figure 4: Slew driver based solar tracker[5]

1.1.2 LINEAR ACTUATORS BASED DESIGN

The drive system is based on two linear actuators, also called hydraulic ram. Each linear actuator is responsible for one axis of rotation. These are the same actuators that can be found in the design described above. For this reason they usually have smaller degrees of rotation than the previous design. Specifically, they usually have 90-100 degrees of azimuth rotation and 75-80 degrees of elevation. They support lower loads so they usually have capacity for fewer panels in series than the previous design. Their simplicity also makes them less expensive than the previous ones. An example of this design can be seen in the following image.

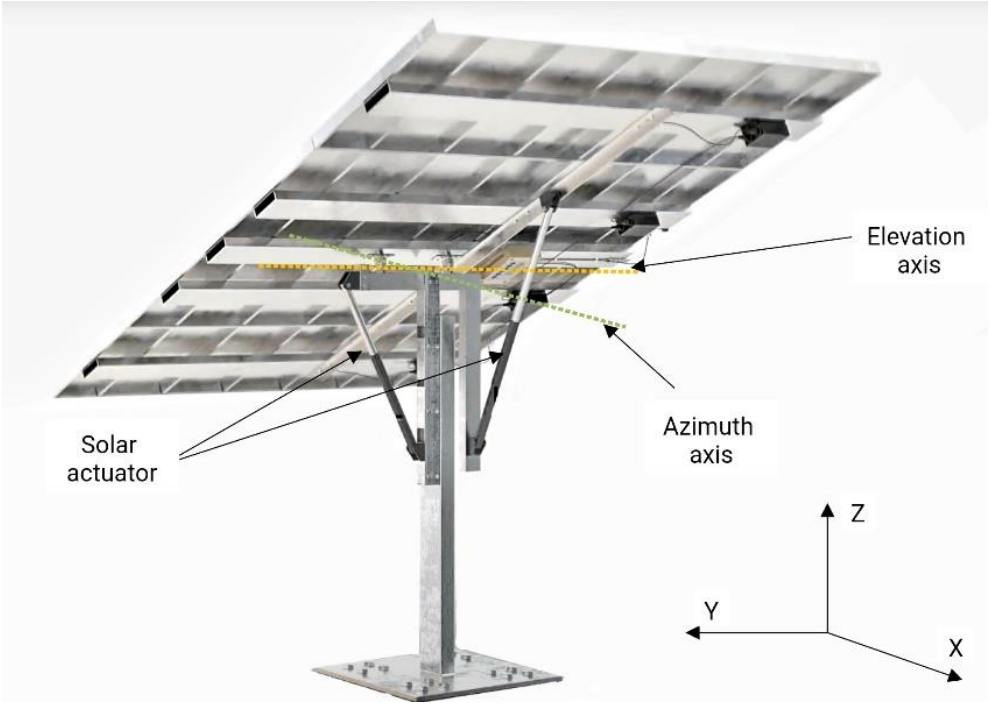


Figure 5: Linear actuators based solar tracker[6]

1.2 ANGLES TRACKERS

Thanks to these movements, electricity production can be increased with respect to fixed installations by up to 20% with single-axis trackers and up to 30% with dual-axis trackers [7]. Although these values may vary according to geographic location

At this point, it is clear that thanks to the movement of the solar trackers, the amount of radiation received by the panel and consequently the energy produced increases significantly. In the following figure it can be seen better.

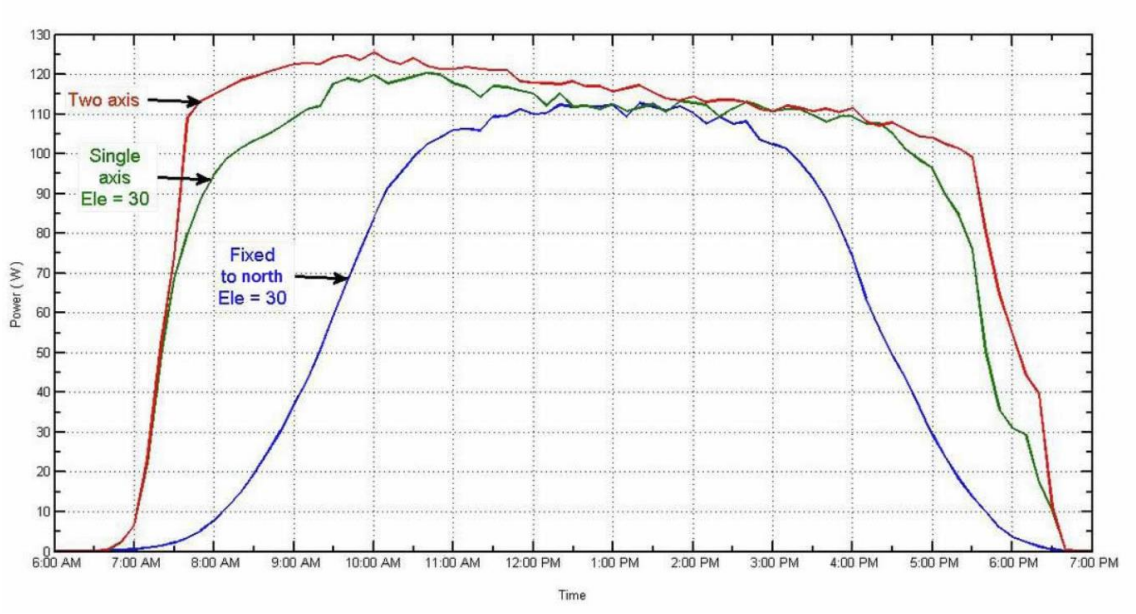


Figure 6: Energy production curves produced by solar tracked and non-solar tracked supports.[8]

The Figure 6 shows the energy produced during a day with a fixed photovoltaic installation and with an installation with solar trackers that move in 1 and 2 axes. The data is compiled by micropowergrids australia. This data corresponds to energy production tests for the 3 types of mounts, all with one solar panel of 150W. These tests were carried out on 21/7/2017 in Sydney. The blue curve is the result obtained in the static installation, the green curve is the result of the installation with 1-axis trackers and finally, in the red curve is the result of the installation with 2-axis trackers. All installations use the same panels.

As it can be seen, the panels installed with solar trackers have received more solar radiation, and therefore have produced more energy throughout the day. The yield has increased by about 30% for the 2-axis tracker compared to the fixed installation. This difference in efficiency can be observed by comparing the areas enclosed by these curves with the X-axis. These areas show the Wh production, so it can be extrapolated that more area means more energy produced and therefore more efficiency.

1.3 HIGHER-QUALITY POWER OUTPUT

Another important aspect to highlight is that solar tracking not only increases the energy production of the photovoltaic plants but also improves how the output power produced is delivered[3]. With solar tracking it is possible to extend the time of maximum power and thus produce at a higher capacity for more hours per day.

Going back to Figure 6 this concept can be better understood. In the blue curve there is an evolution of the power output throughout the day; it gradually increases until it reaches its peak at noon, then gradually decreases again. On the other hand, the green and red curves are approaching the maximum power from early in the morning and the production is maintained until late in the afternoon. These differences in power output between supports vary during the year, depending on how the fixed tilt angles are optimized. If these angles are optimized for winter months the panels will have a high tilt since the sun takes its lowest average elevation values in winter. This would result in lower yields in summer. This can also be applied in reverse if the fixed angles are optimized for the summer months. On the other hand, two-axis solar trackers always take the maximum energy from the sun.

Having commented on all the advantages of implementing these supports, it remains to explain why they are not as standardized as fixed supports at the time of writing this thesis. The main problem with solar trackers is their cost, which normally increases the total budget by 50% compared to an equivalent fixed support [9] . This 50% increase would have an impact of an increase in efficiency of about 30%, as mentioned above. This would make it often more convenient, at least in the short term, to invest in more panels with lower unit production but higher overall production. This has two main implications. The first is that in some ways it would be deliberately lowering the efficiency of the system, also without considering the increase in future environmental problems due to the extra panels. On the other hand, as solar energy captures more of the market, it will be evident that it will not be necessary to increase peak solar energy production, but it will be more interesting to have production over longer periods[10].

Finally, this thesis aims to improve this disadvantage by proposing a presumably more efficient large-scale design that is decentralized and has a two-axis of movement.

CHAPTER 2: OBJECTIVES

The main objective in this thesis is to develop an alternative two-axis solar tracker design proposal that has lower manufacturing costs than existing two-axis solar trackers on the market. A prototype of the design is to be built to demonstrate that the design works. This prototype, in turn, will also be used for research, education and presentation purposes. This design must be easy to assemble, but it must also be robust to be able to be outdoors in the Nordic climate. This main objective can be subdivided into these sub-objectives:

- Design of a tower with a frame for holding the solar panel.
- Design of a motorized system for movement of the solar panel.
- The system must follow the sun (tracking), but it must also be able to be remotely controlled by digital tools (Internet, SMS...) or follow the predicted sun path thoughout the day/year
- The panel must tilt from 0 degrees (horizontal) to 90 degrees (vertical) and the system must rotate 360 degrees.
- The design has to withstand the Nordic climate (ice, snow, wind, wind, rain)
- The system should be built within the period of the master's thesis, if possible, with the help of the department
- Incorporation of a safety system, e.g. in case of high winds.
- The design should incorporate an energy production measurement system and a data collection tool should be incorporated.
- Develop a full-scale manufacturing budget for the proposed design.
- Compare manufacturing costs of the proposed design with market designs.

CHAPTER 3: DESIGN PROPOSAL CONCEPT

There are several methods to reduce the manufacturing costs of a product such as the standardization of parts, optimization of means of transport, optimization of required material, etc. This thesis proposes to reconceptualize the designs of two-axis solar mounts existing in the market. As already mentioned, all two-axis solar mounts function as complete units in themselves. This means that each solar mount is equipped with everything necessary for its operation. This would include a sensor system, a motor system, a controller capable of guiding the motors and finally the structure itself. The inclusion of all these components in each of the solar trackers of a solar farm could presumably be redundant as all the solar mounts would have to move synchronously. This is because their optimal angular azimuth and elevation inclination depend on their location and time, and all trackers would share the same. In this way, it is proposed to reduce to the maximum all the systems of which a solar tracker is composed, transforming the design into a centralized one. In this way it is proposed to centralize the motion system and the control system (this includes the sensor system and the controller). In this way the final design would be composed of a centralized motor system which would be in charge of moving multiple structures synchronously. These motors would be guided by a centralized control system. For the transmission of the forces of the movement system to the structures, the use of cables and pulleys is proposed. Cables and pulleys are well established tools in the market. For this reason, they should presumably not contribute large extra costs and their operation should be robust.

In order to estimate the maximum cost reduction that a centralized system could provide, the production cost data of a two-axis solar support model on the market will be taken. The percentage distribution of manufacturing costs for a two-axis tracked solar mount can be seen in Figure 7. These percentages have been provided by the company Suntrack Nordic AS [11]. It can be seen that the costs associated with the control system plus the motion system total 38% of the manufacturing costs of a solar tracker. Therefore, these two systems make up a large part of the manufacturing costs of a solar tracker.

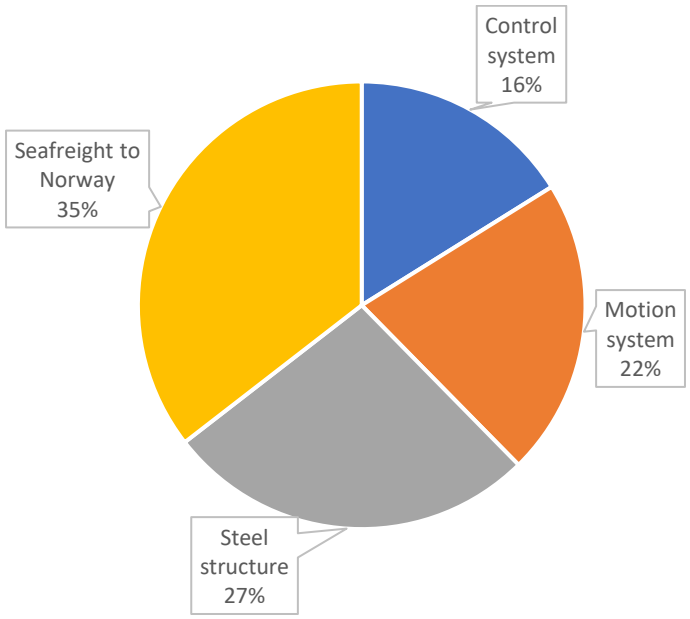


Figure 7: Relative manufacturing cost graph of two-axis solar trackers

CHAPTER 4: PROTOTYPE

This chapter will describe all the parts of which the prototype is composed, emphasizing their characteristics, design versions, and purpose. The explanation of how the different parts interact and how are they calculated will be given in the char of "Methodology". The reason for this is that in order to give a detailed explanation of why each part takes the shape it does, a previous explanation of multiple concepts is required.

The prototype can be divided into four different systems:

- Solar stand structure
- Wired motorized motion system
- Computing system
- Feedback system

In Figure 8, one can see an image with the different parts of the prototype.

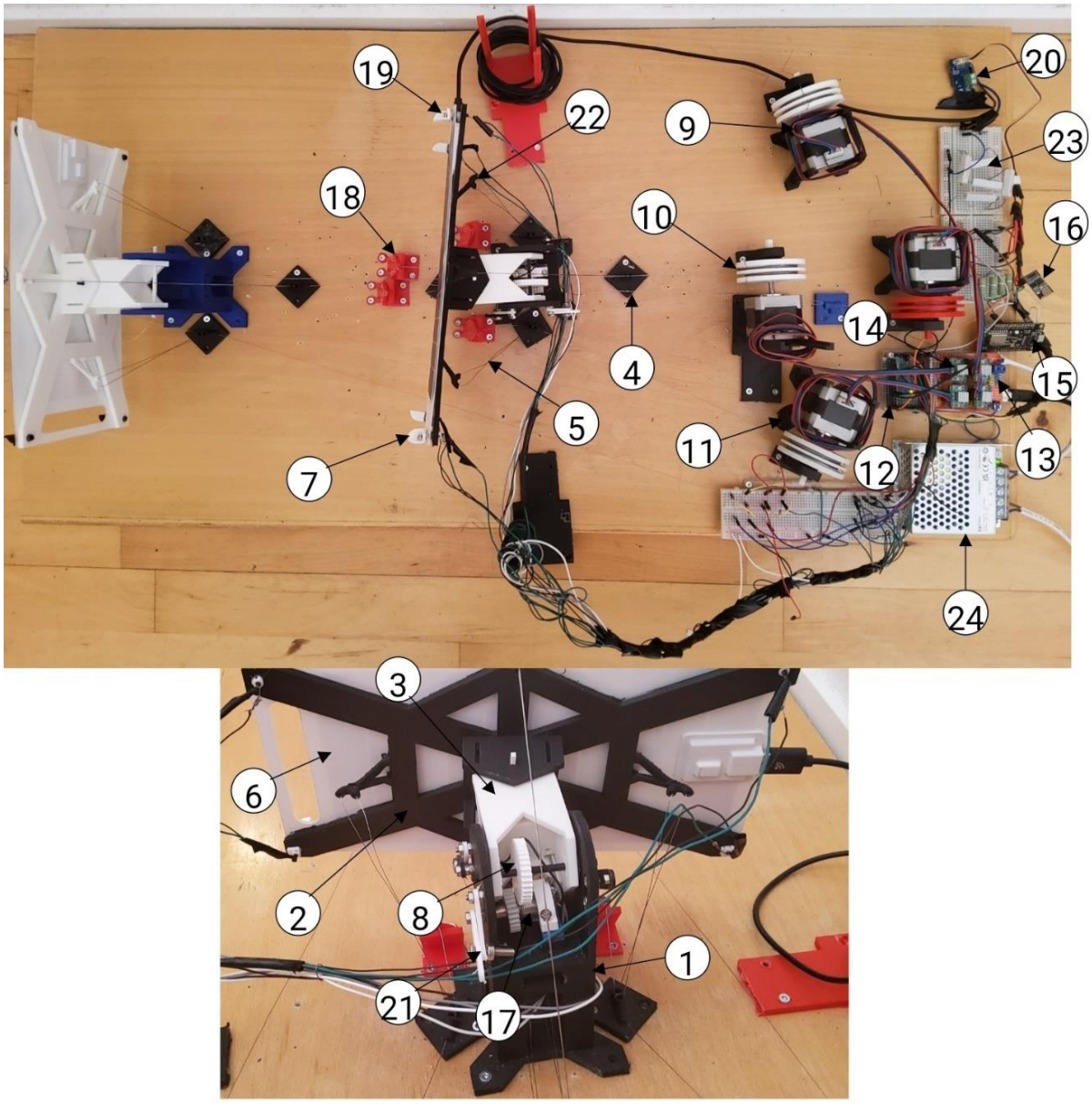


Figure 8: Prototype schematic parts

Solar stand structure	
1	Main tower
2	Support for the solar panel
3	Rotula
4	Lower pulleys
5	Cable
6	Solar panel
7	Panel-mount connectors
8	Gears
Motorized wire control system	
9	Electric motor
10	Drum
11	Support for motors
Computing system	
12	Microcontroller
13	CNC shield v3.0
14	A4988 stepper motor driver
15	NodeMCU v3.0 Wi-Fi module
16	ESP01 Wi-Fi module
Feedback systems	
17	Angle detection system
18	Wire tension detection system
19	Light source detection system
20	Energy production detection system
Extra components	
21	Angle limiter
22	Higher pulleys
23	Energy burn resistors
24	Power supply

Table 1: Prototype parts list

4.1 SOLAR STAND STRUCTURE

This part consists of the structural components of the prototype. Most of the components that will be shown in this section are printed in PLA. "*Polylactide (PLA) is a biodegradable, aliphatic polyester derived from lactic acid"[12]*. The process of materialization of the designs is as follows:

1. The development begins with the realization of the designs of the 3D models. These have been made with Inventor, the 3D design tool of Autodesk.
2. Once the designs are finished, they are exported in ".stl" format and implemented in CURA, a free 3D printing software. This program is compatible with the "Ultimaker S5" printer, the 3D-printer provided for the development of this thesis.
3. Then the parts go through a post-processing process in which the 3D supports are removed (these are complements incorporated by the CURA program to ensure a satisfactory printing).
4. Finally, the parts are assembled, and the tolerances are checked.

In Figure 9 one can see this production process.

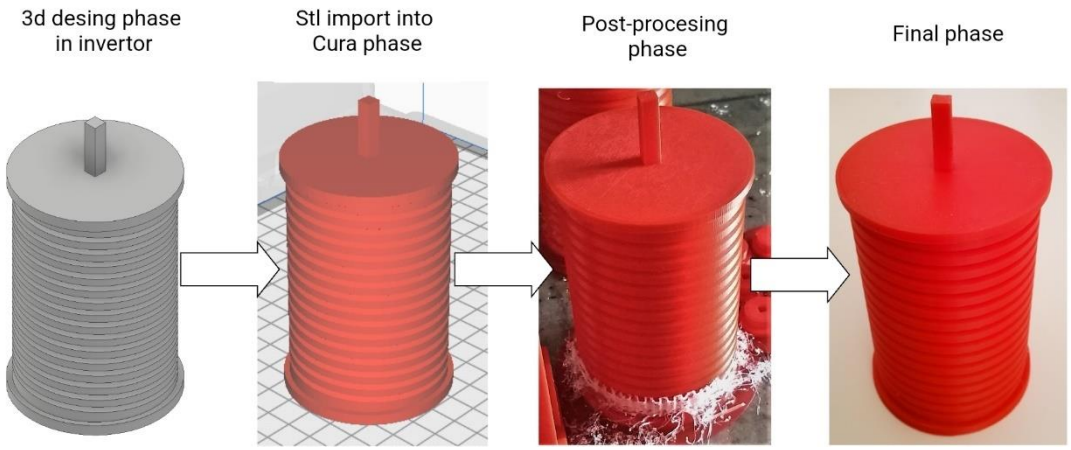


Figure 9: Process of materialization of the design

This segment is characterized by having an iterative nature, since, as it will be shown, to reach the final design of each part, different versions have been passed through. The solar stand structure parts can be highlighted in this segment, and they are the following:

4.1.1 MAIN TOWER.

The main purpose of this part is to elevate the panel so that it can move freely. There are two main differences from the first versions. The first one is the incorporation of rotation angle limiters. The purpose and operation of these angle limiters are explained in more detail in section 5.1.The second is the reduction of the tower height. The reasons for the tower readjustment are again explained in same section. In Figure 10 one can see the evolution of the versions from left to right.

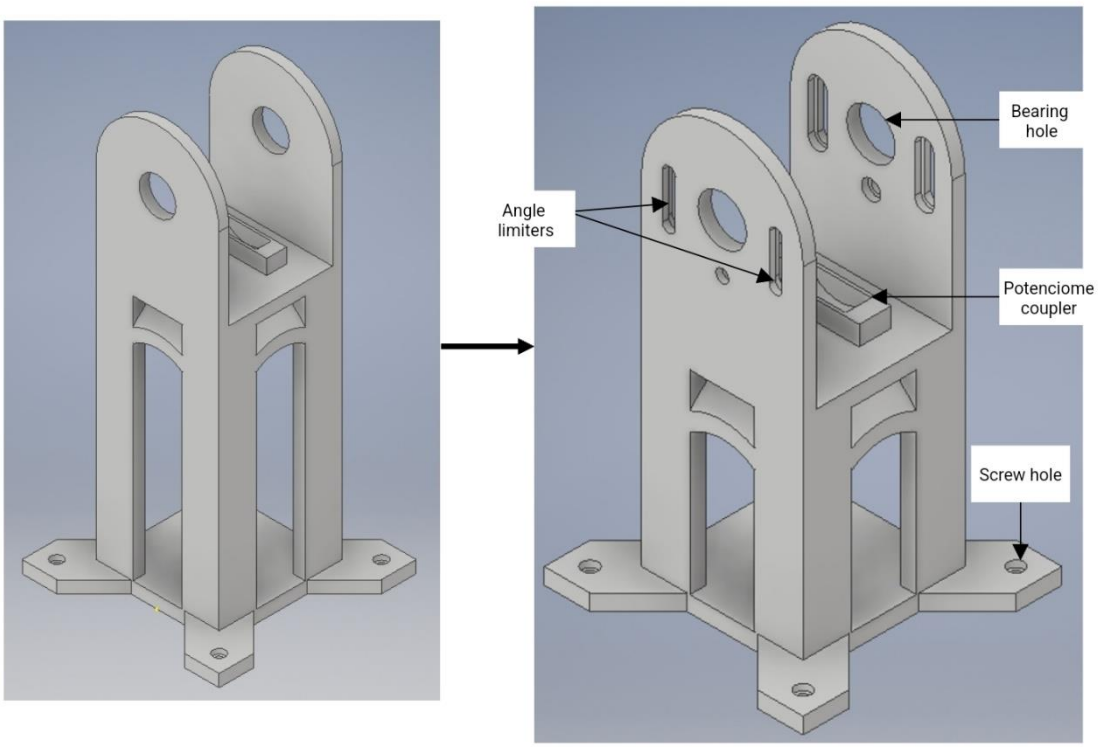


Figure 10: Versions of the main tower

4.1.2 SUPPORT FOR THE SOLAR PANEL

The purpose of this part is to fix the panel to the structure. It is connected to the main tower by means of the bracket. In the first version a design was chosen in which the pulleys were placed in an "x" shape, following the decisions of the panel fixings, as opposed to a "+" shape. This resulted in the azimuth and elevation axes of rotation not being linked to a separate set of motors, so that the movement of one motor affected both axes, rather than just one axis. This had the advantage that the associated with the elevation swing were not located in a plane that could come into contact with the structure for turns greater than 80º. The main problem associated with this design was the extra calculation of adding motion matrices to reflect these axis/motor ratios. This was a problem for two reasons: The first was that these matrices returned non-integer values, which could result in cumulative errors, since the motors move discretely and do not have continuous motions. The second was the extra computation that can result in a slow down the drive algorithm. To avoid the spin axis cables touching the structure it was chosen to lower the height of the structure and lower the plane of the pulleys. These pulleys will also be referred to as upper pulleys hereafter. This made it possible to opt for a + shape. This can be seen in the second version of the panel support. For the final version 3 more improvements were added. The first of these was the incorporation of stiffeners in the structure. The second of these was to corporation the holes to add angle limiters as in the case of the main tower. And finally, the plane containing the sheaves of this support was brought closer to the axis of elevation rotation. As already mentioned, this helped to keep the cables associated with the lifting of the panel from touching the structure and extended the limits of rotation. This extension of the angles of rotation will be explain in section 5.1. In Figure 11 one can see the evolution of the versions from left to right.

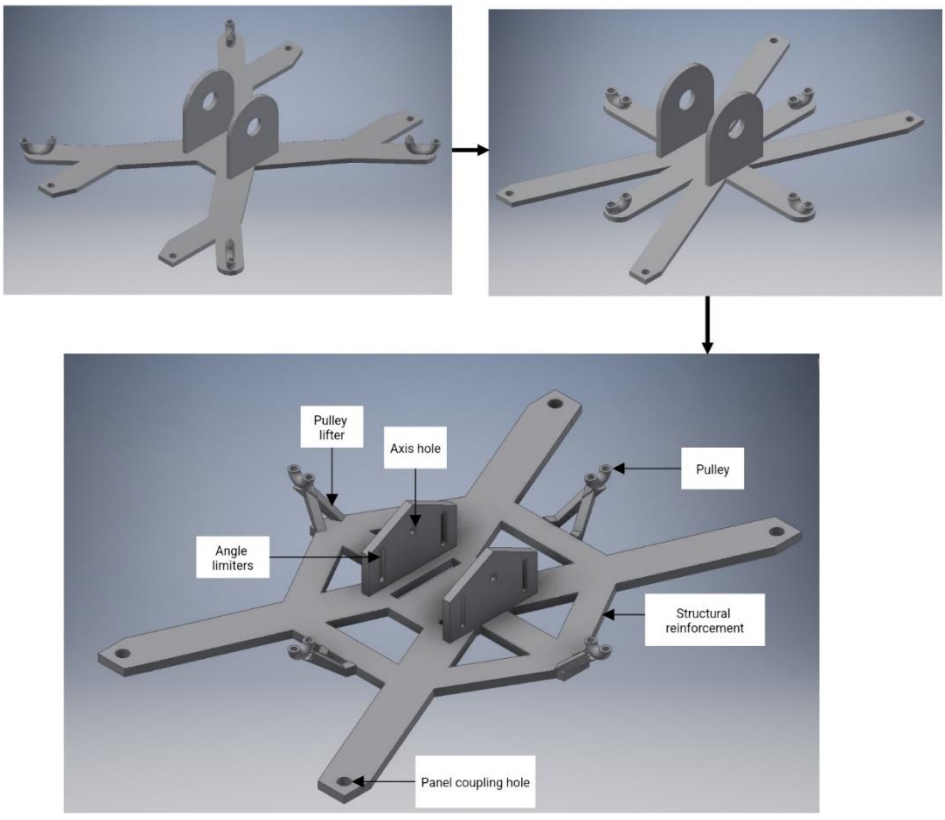


Figure 11: Versions of panel support

4.1.3 ROTULA

This has the function of giving two degrees of freedom to the support assembly. More specifically these degrees of freedom are translated in the two angles of rotation of elevation and azimuth. It connects the panel support to the main tower through the use of bearings and shafts. The main change to highlight between versions has been the adjustment of the wheelbase. This allowed minimizing the distances between upper and lower poles. This is related to the "Null moment limit state" explain in section 5.1. In Figure 12 one can see the evolution of the versions from left to right.

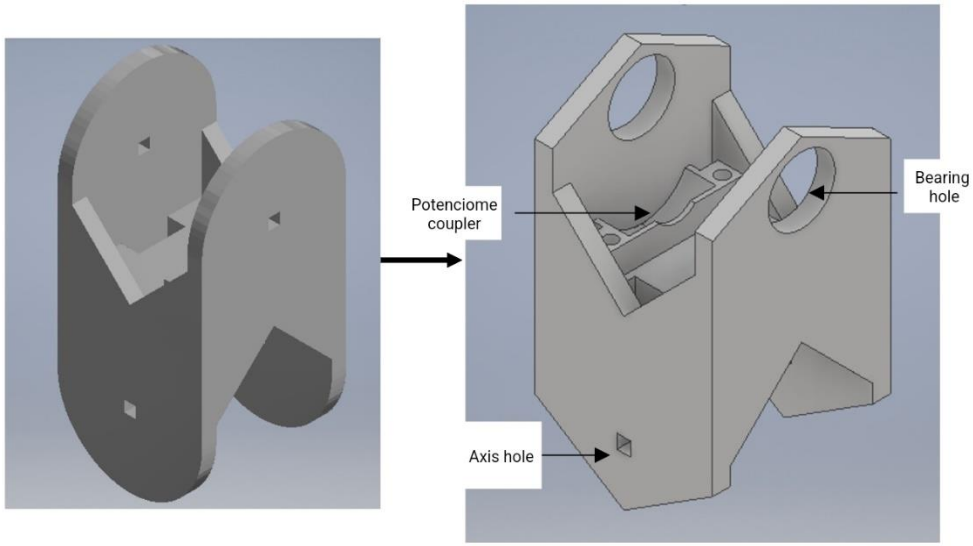


Figure 12:Versions of the rotula

4.1.4 LOWER PULLEYS

The purpose of these pulleys is to guide the cable so that it transmits correctly the forces created by the movement of the motors. They also try to minimize friction. In spite of not being pulleys (since they do not have movable parts), they have been denominated this way since in the model to real scale they try to replace for real pulleys. The main reason for not having bearings is the low relative definition of the 3D printer. This is due to the fact that a bearing is formed by balls and rings, both with very low tolerances. In Figure 13 one can see the different bearings that can be found in the prototype.

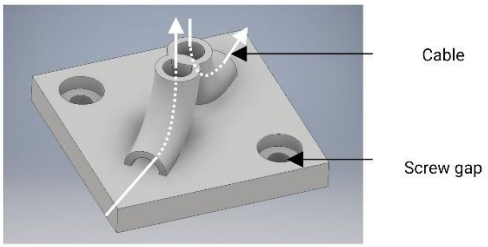


Figure 13: Pulleys

4.1.5 CABLE

The purpose of this cable is to transmit the forces created in the drum connected to the motor with the support for the solar panel. In the first version of this cable it was chose to use a Nylon fishing line. In the final version a steel cable has been chosen. The reason

for this decision can be found in "CHAPTER 6:, since its reasoning requires the previous explanation of multiple concepts.

4.1.6 SOLAR PANEL

The PRT-16835 panel was chosen, featured by SPARKFUN ELECTRONICS. It has a maximum output of 10W and 5V. Its power output is via an USB connector. It lacks a MPPT. "*The MPPT is a charge controller that compensates for the changing Voltage Current characteristic of a solar cell. The MPPT fools the panels into outputting a different voltage and current allowing more power to go into the battery or batteries by making the solar cell think the load is changing when you really are unable to change the load"[13].*

4.1.7 PANEL-MOUNT CONNECTORS

They have two main purposes. The first is to attach the panel to the bracket. The second is to serve as a light blocker for the photoresistor-based feedback system. Figure 14 shows these connectors.

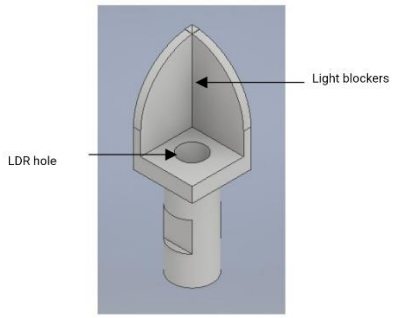


Figure 14: Panel-mount connector

4.1.8 GEARS

The purpose of these gears is to transfer the rotation of the shafts to the potentiometers. The methodology followed for their calculation and design is explained in section 5.2. In Figure 15 one can see the gears for the elevation axis on the left, and the azimuth axis on the right.

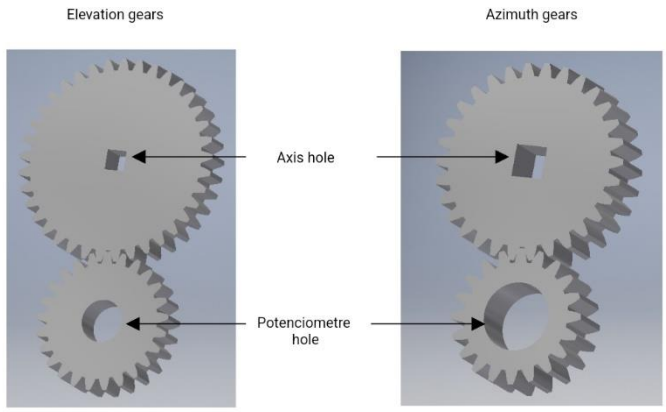


Figure 15: Gears

4.2 MOTORISED WIRE CONTROL SYSTEM

4.2.1 ELECTRIC MOTOR

The selected family of electric motors is the stepper motor[14]. The stepper motor is a brushless DC motor in which the rotation is divided into a certain number of steps resulting from the motor structure. The reasons for this decision are as follows:

- High precision: a complete 360° shaft revolution is divided into 200 steps, which means that a single shaft stroke is performed every 1.8°.
- Reliability: there are no brushes in the motor construction, which results in high mechanical durability and increased reliability.
- Easy control: fast starting due to high torque, easy stopping due to high holding torque and the ability to quickly change the direction of rotation. These are controlled by a controller that supports digital direction and direction inputs.

This type of motor also has certain disadvantages, which must be considered when designing the motion algorithms. The main problems of this type of motors are the following:

- Danger of slippage: if the torque is insufficient, a phenomenon called slippage or missing steps occurs. Therefore, a feedback mechanism is required for reliable motor control, which can be realized, for example, on the basis of an encoder or other type of sensor. Thanks to it, the motor controller can "make sure" that the motor has performed the indicated number of strokes.
- Energy requirement: The motor requires energy both when it is in motion and when it is stationary. This disadvantage will be taken into account in the maintenance cost calculations for this design.

Among the models on the market, the 17HS15-1504S from the manufacturer STEPPERONLINE has been selected. It has a clamping torque of 42 Ncm. It belongs to the NEMA17 family of steppers with a standard housing size of 42 x 42 x 39 mm. For more details on this model the specification sheet can be found in p.75. In Figure 16 one can see a picture of the model.

Figure 16: Stepper motor 17HS15-1504S

4.2.2 DRUM

The purpose of this part is to collect the cable in order to transmit the movement of the motors to the structure. In the first option it was opted for a threaded design, in order to avoid overlaps in the cable. This was necessary since the first prototype design was based on the use of two motors instead of four and therefore a drum was needed to collect the cable while transferring it to the other end. This explanation can be found in

more detail in section 5.1 .For the new design with 4 motors the above mentioned condition was no longer needed, now the drum only needed to be able to either pick up or yield the cable and not both at the same time. In this way it has been possible to simplify the design and thus add the cable reels (one for each support to be moved). In Figure 17 one can see the evolution of the versions from left to right.

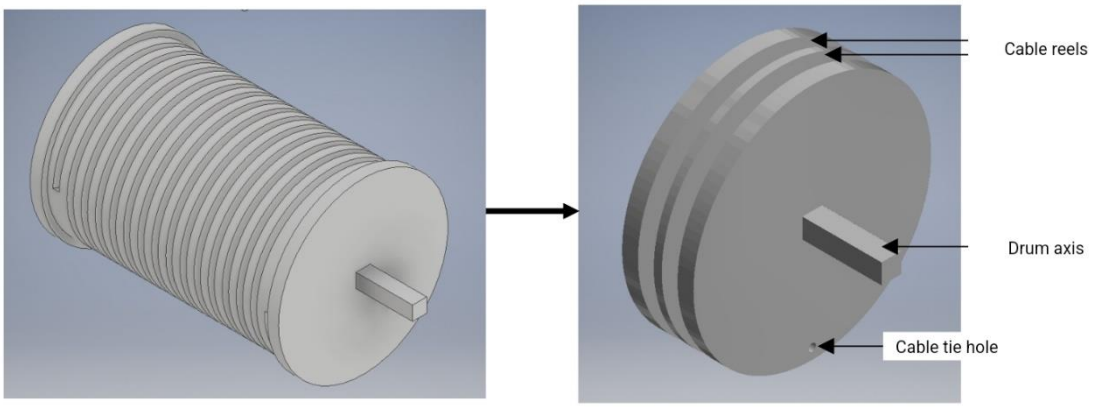


Figure 17: Versions of the drums

4.2.3 SUPPORT FOR MOTORS

These allow the motor and drum assembly to be fixed to a surface. In the first version it allowed the attachment of 2 motors and two drums (one on top of the other). This was used for the prototype version based on two motors instead of 4. These versions will be explained later in section 5.1 .The final version reduces the amount of plastic and allows adjustment of the clearance between the motor and the drum. Figure 18 shows the evolution of the versions from left to right.

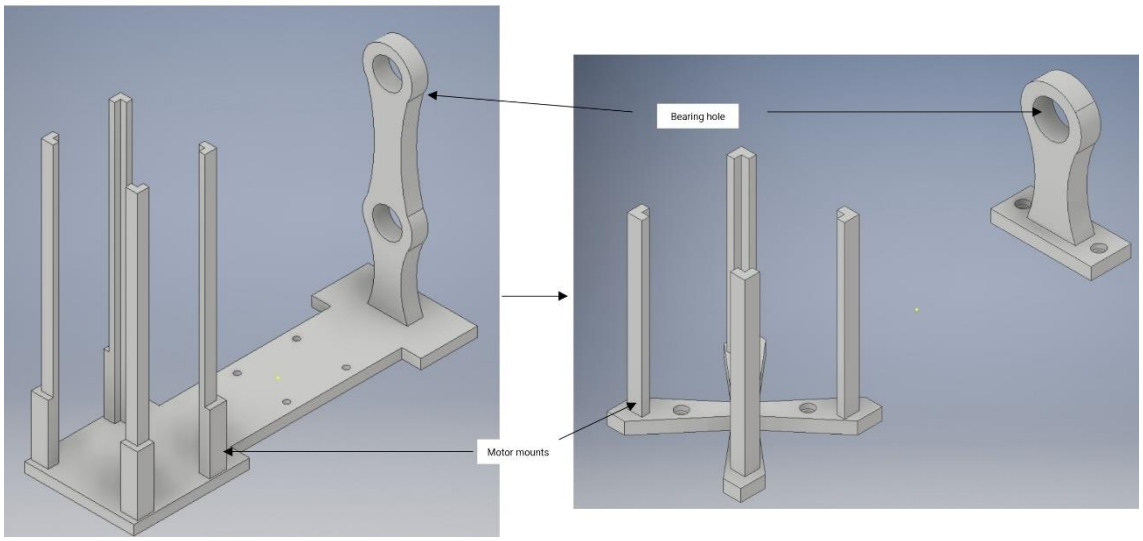


Figure 18: Versions of the motor supports

4.3 COMPUTING SYSTEM

This system oversees processing, receiving and sending data and commands. Its main component is a microcontroller. It is compatible with a series of modules that allow to extend its functionalities. These are its components:

4.3.1 MICROCONTROLLER

It is the device in charge of carrying out the instructions defined by the person who programs the device autonomously, in an electronic system the microcontroller is the most important component, it is composed of a central processing unit, a memory and a series of input and output pins. Among the different families of microcontrollers on the market it was chosen the Arduino family.

Arduino is a family of free hardware boards that incorporates a reprogrammable microcontroller AVR type (family of microcontrollers of the American manufacturer Atmel). There are different Arduino boards depending on the use it is needed which is given depending on the size, memory, number of pins and type of microcontroller[15]. For the development of the project it was used an Arduino Mega 2560 board which due to its characteristics fits the requirements of the project. Specifically, the Arduino Mega 2560 has been chosen for two reasons: for its greater number of inputs and outputs, and for its greater ROM memory, with respect to the rest of Arduino boards. To perform the programming an integrated environment known as IDE was used, that allows modifications and corrections in the code before they are executed by the microcontroller. This environment uses C and C++ code language[16]. In Figure 19 one can see these inputs and outputs.

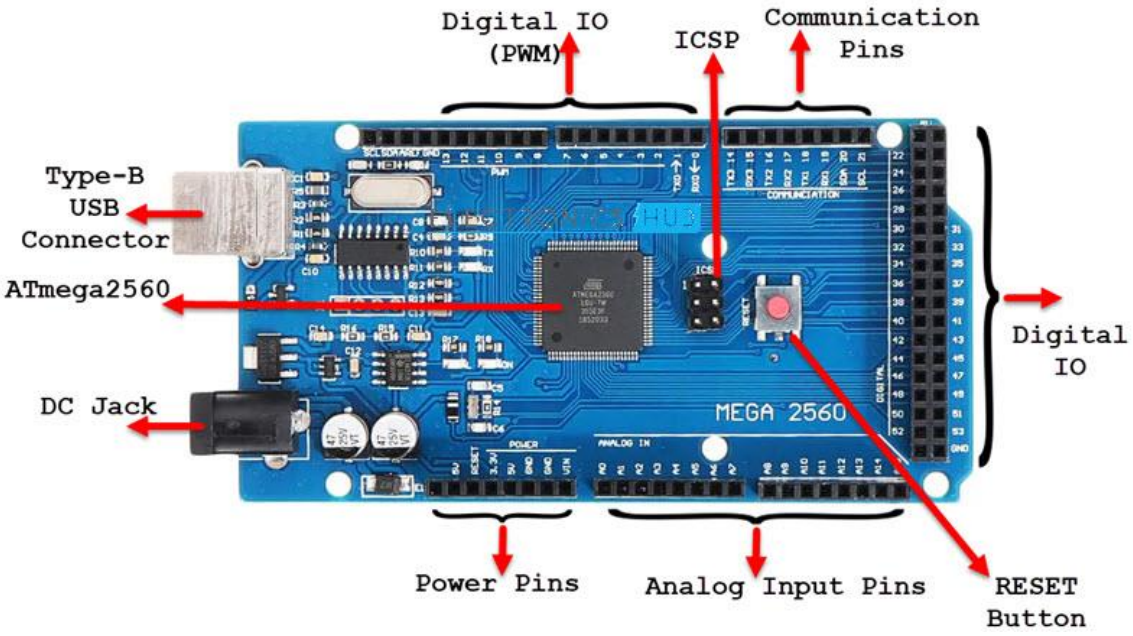


Figure 19: Layout of Arduino Mega Board[17]

- DC Jack: This is a positive center connector. It is used to power the board when it is not connected to the computer or a power supply higher than 5 volts is used. The power applied to this connector should be between 7 and 12 volts. This is because the Arduino Mega incorporates a voltage regulator.
- RESET Button: This button allows to reset the system. This means that all the programmed code will be executed again, just as if the system had just been connected to the power supply.
- Type-B USB: this allows communication with the computer and is powered by 5V.
- Power Pins: this is made up of pins that are used to power other devices when making a project. These are:
    - GND: Ground pins.

    - 5V: 5 volt power supply for sensors or other devices.
    - 3V3: This pin provides a voltage of 3.3 volts for devices that require it (It can supply up to 150 mA, although it is recommended not to exceed 50 mA).
    - VIN (Similar to the Jack connector): Allows powering the board when it is not powered using the USB port. This pin can be used to obtain the voltage present in the Jack connector.
- Analog Input Pins: these are 16 analog inputs (10 bits resolution) denoted A0, A1,...,A15, which can be used with analog sensors (such as temperature or humidity probes).
- Communication Pins: these are 6 serial ports that allow the board to communicate with a computer or other device that supports this protocol.
- Digital IO: these are formed by the 64 pins and can read values of 0 (0 V) and 1 (5V).

To these inputs and outputs have been connected various devices that are detailed in the following sections.

4.3.2 CNC Shield V3.0

CNC Shield V3.0 is an expansion module mainly used in engraving machines and 3D printers. It allows to manage 3 stepper motors independently. These are called by the acronyms "X", "Y" and "Z". There also one additional motor call "A" that works as a duplicate of one of the previous ones. These are controlled by 4 A4988 drivers. It allows a DC power supply between 12 to 36 volts. The incorporation of this shield is not completely necessary, but it simplifies the communication with the motors and provides a cleaner design. It also incorporates capacitors that are used for protecting the A4988 driver from voltage spikes.

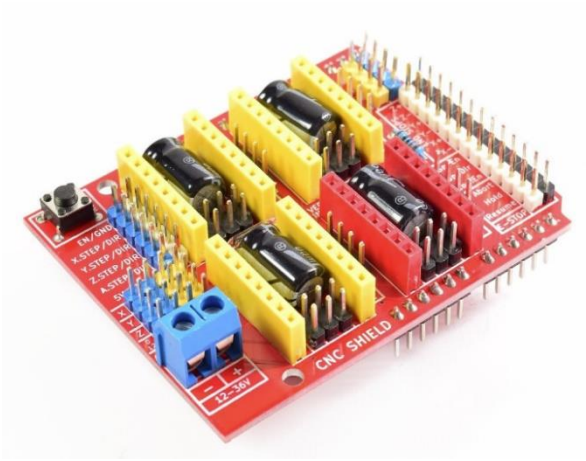


Figure 20: CNC Shield V3.0[18]

4.3.3 A4988 STEPPER MOTOR DRIVER

It is a controller to handle the high voltages and currents needed to move stepper motors such as those used in this project. It also allows to limit the current flowing through the motor and provide protections to prevent damage to the electronics. For its control only require two digital outputs, one to indicate the direction of rotation and another to communicate the number of steps. They also allow micro-stepping, a technique to achieve accuracies higher than the nominal step of the motor[19]. If these drivers were connected to the 4 inputs of CNC Shield V3.0 only 3 steppers could be moved independently. For this reason one of the A4988 has been modified so that its address

pins are not connected to the Shield. In this way they are connected to other free digital inputs and thus be able to control 4 motors independently. In Figure 21 one can see on the left the A4988 driver and on the right its modified version.

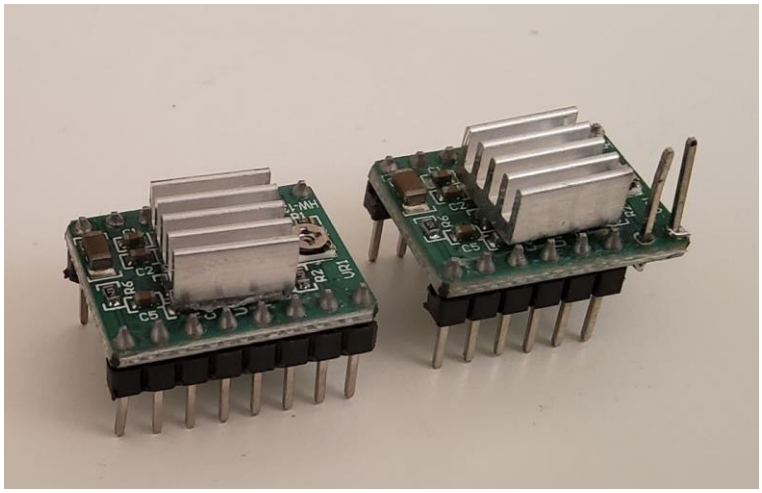


Figure 21: A4988 stepper motor driver and its modification

4.3.4 WI-FI MODULE NODEMCU V3.0

The NodeMCU module is a small Wi-Fi board. It is mounted around the ESP8266 chip which offers a complete Wi-Fi networking capabilities, allowing it to host the application or to serve as a bridge between the Internet and a microcontroller. This module exposes its pins on the sides. It also features an integrated voltage regulator as well as a USB programming port. It can be programmed using the Arduino IDE.

The ESP8266 has powerful on-board processing and storage capabilities that allow it to integrate with sensors and application-specific devices through its GPIOs with minimal development and minimal run-time overhead. Its high degree of on-chip integration allows for minimal external circuitry, and the entire solution, including the module is designed to occupy the minimum area on a PCB[20]. In Figure 22 the module can been seen.

Figure 22: NodeMCU V3.0 module

4.3.5 WI-FI MODULE ESP01

This module is a simplified version of the previous module. This one has integrated the same ESP8266 chip, but this one lacks the USB input for the connection with the computer. The incorporation of this second chip is due to the impossibility to connect to different web pages at the same time. This is due to problems with the http and https protocols in Figure 23 the module can been seen.

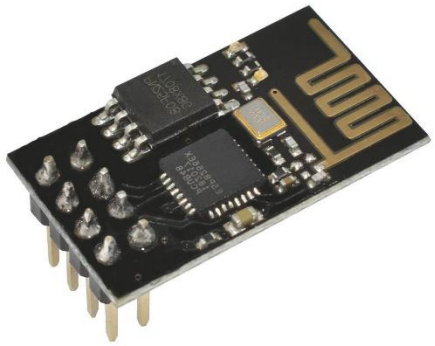


Figure 23: ESP01 Wi-Fi module

4.3.6 MULTIPLEXERS

Multiplexers are combinational circuits with several inputs and a single data output. They are equipped with control inputs capable of selecting one, and only one, of the data inputs to allow transmission from the selected input to that output. Due to the limited number of analog inputs of the microcontroller, a multiplexer was required to measure all the sensor values.

4.4 FEEDBACK SYSTEMES

In this segment the set of sensors that are connected to the Arduino Mega will be described. All of them have analog outputs and are composed of 4 types:

4.4.1 ANGLE DETECTION SYSTEM

This is composed of two rotary potentiometers. One potentiometer has a variable resistance at each end and a third connection to a slider, which will allow us to increase or decrease the resistance itself. In the case of the rotary potentiometers, this resistance varies according to the angle of rotation of an axis. The potentiometer selected is the 3852A-282-103AL of the Bourns brand. This has a turning range of 280º with which it can go from a value of 2Ω to one of 1kΩ. It has a total resistance tolerance of ±10% and an independent linearity of ±10%. These tolerances are relatively large since the potentiometers are not intended for use as precision angular sensors, but rather as an analog control tool. Assuming a linear equation that relates the rotated angle to an output voltage value would result in an unacceptable absolute error. For this reason this sensor has been characterized to increase the average pressure. This characterization is detailed in section "POTENCIOMETER". In Figure 24 one can see the selected potentiometer.

Figure 24: 3852A-282-103AL potentiometer

4.4.2 WIRE TENSION DETECTION SYSTEME

This is intended to detect the tension in the cable to serve as feedback. This is necessary since the cable is intended to be placed in a range of tensions. Below this range, it would not be possible to ensure that the panel is sufficiently restrained to maintain a position in case of dynamic wind loads. Above this range, the motor would be overloaded. It is not desirable to set this limit much above the minimum. The lower the upper voltage limit, the more solar mounts can be attached to the motor system because the motor would somehow be dividing its maximum torque by the number of mounts connected.

The first version was composed of a push button, a spring, a pulley and support for these. The method for the regulation of this spring followed this procedure. First the tension value in the cable is determined. With a simple force study, the selected tension is related to a value of force applied to the spring. Knowing the desired elongation, Hooke's law can be applied to calculate the desired spring elastic constant. With this constant the desired spring can be selected. The main problem of this version was the friction of the pulley with the support, preventing to give an accurate signal. In Figure 25 one can see the 3D design of this first version.

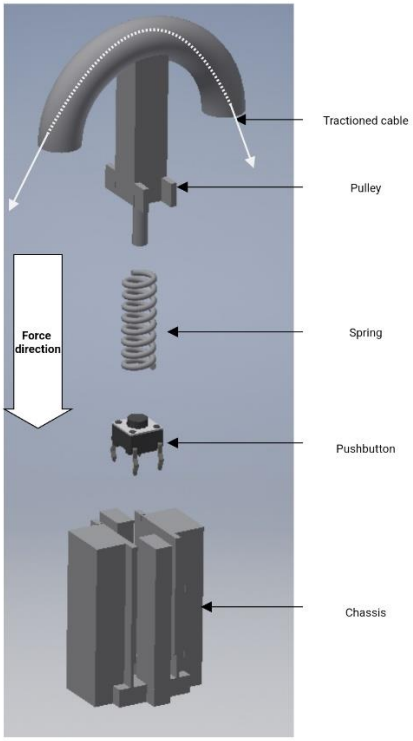


Figure 25: Wire tension sensor Version 1.0

The second version consisted of the use of a force-sensitive resistor. This resistor is able to modify its resistivity depending on the pressure applied to it. When no pressure is applied its resistance rises above 1MΩ. Knowing the area of this sensor it is possible to transform a pressure signal into a force signal. Unlike the previous version, this one returns analog pressure values, thus providing more information on the actual cable tension at any given moment. In one Figure 26 can see the 3D design of the final version.

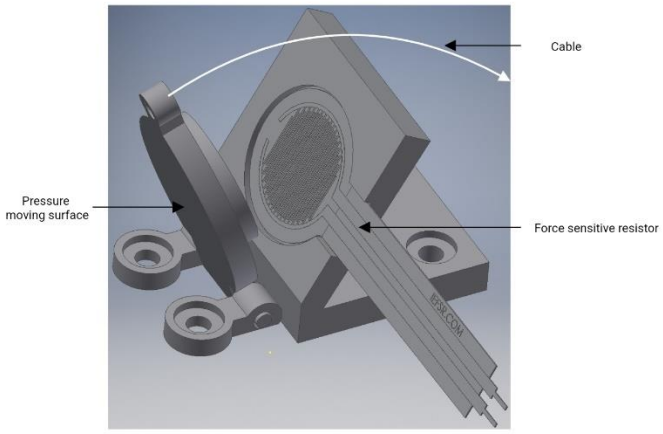


Figure 26: Wire tension sensor Version 2.0

4.4.3 LIGHT SOURCE DETECTION SYSTEME

This system is based on the use of 4 photoresistors. A photoresistor is an electronic component whose resistance decreases with increasing incident light intensity. It is also called light-dependent resistor or by its acronym LDR. Its body consists of a photoreceptor cell and two pins. The electrical resistance value of an LDR is low when light is shining on it, and very high when it is dark. These LDRs are installed in the inside of the panel-mount connectors. As already mentioned these panel-mount connectors allow to block the light coming from two planes. By installing these connectors pointing to 4 different quadrants the photoresistors can be used to detect the direction of a light source. This is done by comparing the different resistivities of the LDRs. This is explained in more detail in section "PHOTORESISTORS DIRECTION ALGORITHM". In the following image one can see these LDRs.

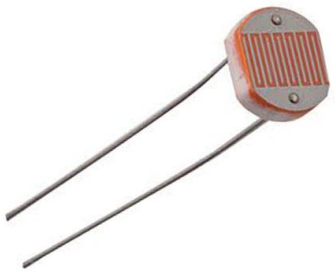
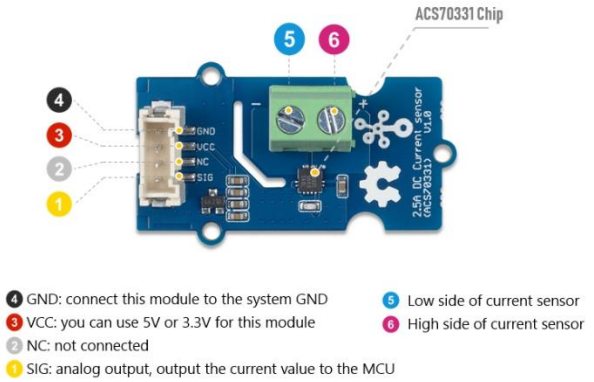


Figure 27: LDR

4.4.4 ENERGY PRODUCTION DETECTION SYSTEM

To obtain an energy value, voltage and current values must be collected. Since the Arduino is capable of measuring voltage values from 0 to 5V, the same range as the solar panel, it is only necessary to incorporate a current sensor.

The transducer in charge of translating current values to voltage is the Hall effect current sensor ACS70331, from the manufacturer Allegro Microsystems. The data sheet can be found in p.79. This informs about some of its characteristics, such as a maximum measurement current of 2.5A, a resistance in the conductor of 1.1mΩ, a response time of less than 550ns and a sensitivity of 800mV/A. Figure 28 shows such a sensor.



4 GND: connect this module to the system GND
3 VCC: you can use 5V or 3.3V for this module
2 NC: not connected
1 SIG: analog output, output the current value to the MCU
5 Low side of current sensor
6 High side of current sensor

Figure 28: Current sensor ACS70331

CHAPTER 5: METHODOLOGY

This section the different methodologies and reasoning applied in the development of the prototype will be explained.

5.1 METHODOLOFY AND ITERATIONS OF THE SOLAR STAND

As shown in section "CHAPTER 4:" ,this segment has had many iterations. This is due to two main reasons.

The first is due to the refinement of tolerances. 1 mm can be the difference between a free bearing or a bearing with a lot of friction. 1 mm can also be the difference between a shaft with a lot of play that does not transmit the movement to the gears well, thus producing systemic error, with one that does transmit the movement well.

The second one is due to the assumption of an erroneous hypothesis at the beginning of the development of this thesis. This assumption was based on the possibility of finding a design that could be guided by two motors and two cables, since the support to be designed had two degrees of freedom. In the first stage of development, a prototype was built based on this assumption. It consisted of the first versions of the segments belonging to the solar stand structure and the wired motorized motion system. It resembles notoriously the operation of the final prototype but with a main difference, based on the use of two motors, one for each axis. When one of the steppers was activated, it in turn moved the 1.0 drum which served to transmit the motion to the cable. When the drum rotated in one direction, the cable was picked up at one end and released at the other end, allowing the panel to rotate on one of the axes. Figure 29 shows a simplified diagram of this operation.

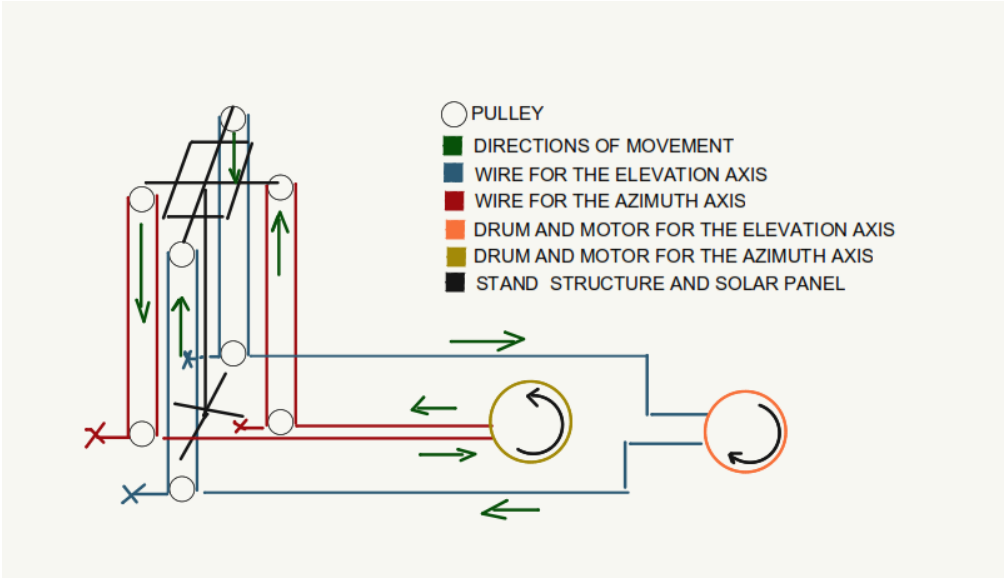


Figure 29: Frist design diagram

If the above hypothesis were true, an invariance would follow. This is that the tension in the cable would remain constant, it should simply be pre-tensioned only once at the time of assembly (assuming a perfect grip of the cable on the drum and therefore no slippage).

In the testing of the first prototype it could be seen that the tension will not be constant, when the panel rotated more than 45º in one axis. The cable started to untension. This is not acceptable in the design since the tension in the cables is essential to transmit the rotation, and without tension both the positioning accuracy and the stability of the panel are lost.

In order to be able to state that the constant tension hypothesis was wrong (since it could be due to other factors such as elasticity in the cable), the next step was to develop a Matlab simulation of the structure[21]. This simulation was intended to calculate the distances between the pulleys located on the panel support and the pulleys anchored to the ground, for different angles of elevation and azimuth. In the case of verifying that the sum of the distances between sheaves is constant with the movement of the panel, it could be derived that the cable should not slacken. This can be reasoned since all the extra cable needed to raise one of the ends should be the same with opposite sign to lower the opposite end (according to the hypothesis presented at the beginning). This is further supported by the fact that the distance from the drum to the sheaves is constant. The distances between sheaves were calculated and not the tensions in the cable since their calculation is simpler.

This simulation makes use of two mathematical tools:

- Rotation matrices: are matrices that allow to calculate the position resulting from the rotation of a vector. In the following formulas one can see these matrices of rotation being θ the angle of rotation

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\,\theta & -\text{sen}\,\theta \\ 0 & \text{sen}\,\theta & \cos\,\theta \end{pmatrix}$$

$$. \; R_y(\theta) = \begin{pmatrix} \cos\,\theta & 0 & \text{sen}\,\theta \\ 0 & 1 & 0 \\ -\text{sen}\,\theta & 0 & \cos\,\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\,\theta & -\text{sen}\,\theta & 0 \\ \text{sen}\,\theta & \cos\,\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Vectorial calculation of distances between two points A and B.

$$A(x_1, y_1) \text{ y } B(x_2, y_2)$$
$$d(AB)^2 = AP^2 + BP^2$$
$$d(AB)^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$
$$\sqrt{d(AB)^2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$
$$d(AB) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}2$$

Figure 30 shows the simulation plot for an elevation angle (axis of rotation z) of 30º and an azimuth angle (axis of rotation x) of 30º. In the upper part and in red one can see the vectors that simulate the ends of the panel support, in these ends the upper poles would be arranged. At the bottom and in red the main tower would be simulated. Blue and magenta would be simulating the vectors of rotation of the rotula. Green simulates the cable needed to move the panel with the azimuth orientation. Yellow would be simulating the cable needed to move the panel with the elevation orientation. Finally, it would be these yellow and green vectors that would be calculated to check the hypothesis of continuity of length in the cable.

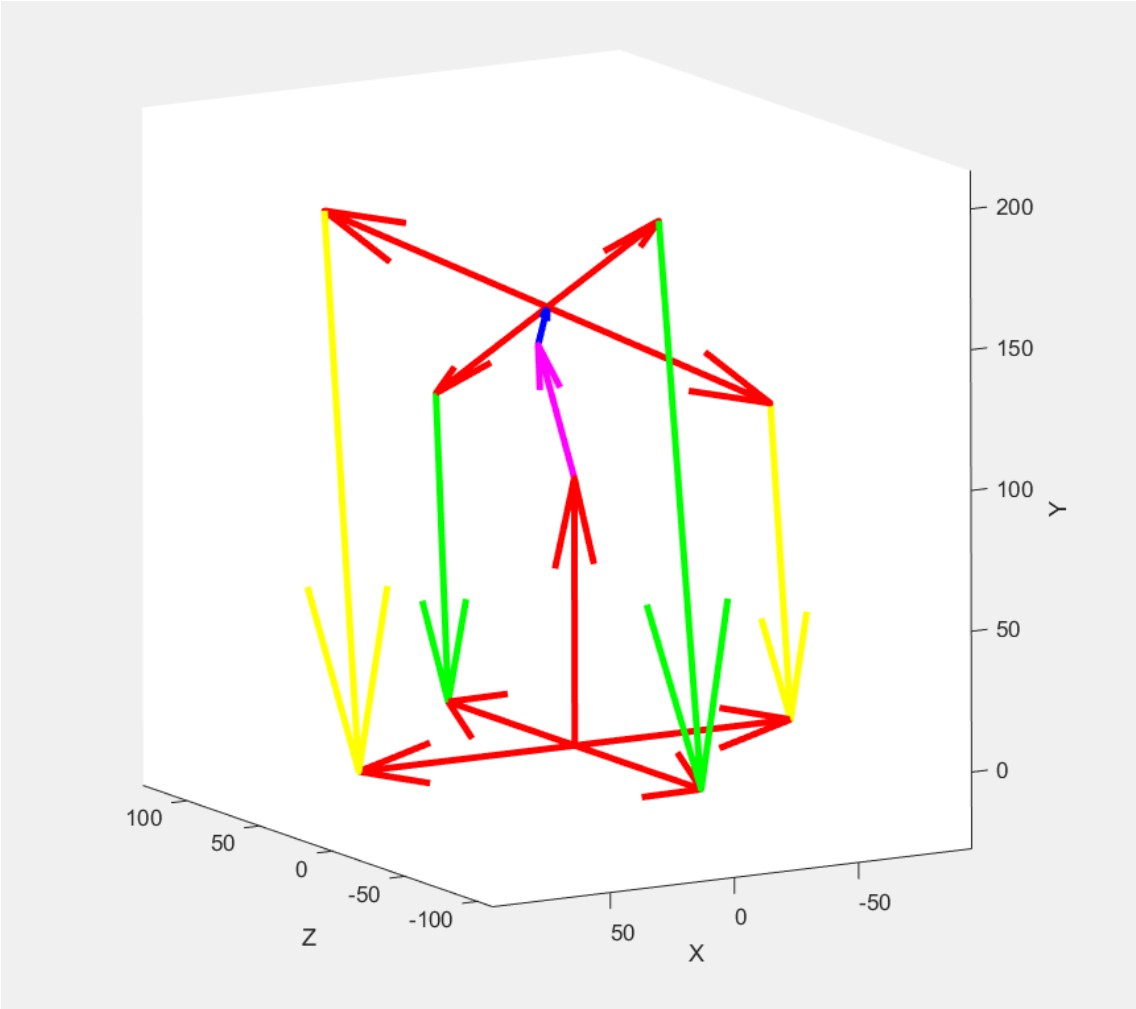


Figure 30: Matlab simulation graph

A study of variables has been carried out on this Matlab simulation and several conclusions can be drawn from it, these are the most relevant ones:

- The distance obtained from the sum of the distances between the upper and lower poles is not constant and therefore the cable is untightened with the movement of the panel.
- The height of the tower affects the percentage increase of cable necessary for the movement of the panel, the higher the height, the lower the percentage increase of cable.
- The distance from the lower pulleys to the center of the main tower directly affects the degrees of freedom of rotation that the prototype has. This conclusion will also apply to the final prototype. This limitation can be explained by two limit states. For the explanation of the limit states, the system will be view statically, the forces applied on the panel support generate fictitious "torsional moments". Non-zero torsional moments would thus indicate a movement of the structure or, in other words, that the static analysis is not stable. In a real dynamic model, the torsional moments generated by the cables are compensated and result in a fixed position, where the torsional moment of the shaft is zero (since it is a rotula). These are the two limit states:
    - **Null moment limit state**: for there to be torsional moment there must be forces applied at non-zero perpendicular distances from the axes of rotation.

These forces comes from tension in the cable (force vector) and distances are those ranging from the cable to the swivel axis of the hinge. For a fixed dimension of the panel, there is a limit state in which the cable in charge of moving the panel in one of the orientations comes into contact with the axis of rotation of the same orientation. At this position the distance at which the forces are applied is 0, so that forces applied by the cable would not result in a torsional moment and therefore would not generate motion (regardless of the magnitude of these forces). The angle formed by the plane containing the panel with the surface planes (horizontal plane) and the plane of rotation of the axis (axis that has come into contact with the cable), marks the limit angle of rotation for the zero moment. At this moment the support is not stable and collapses. This is the main reason why angle limiters have been added.

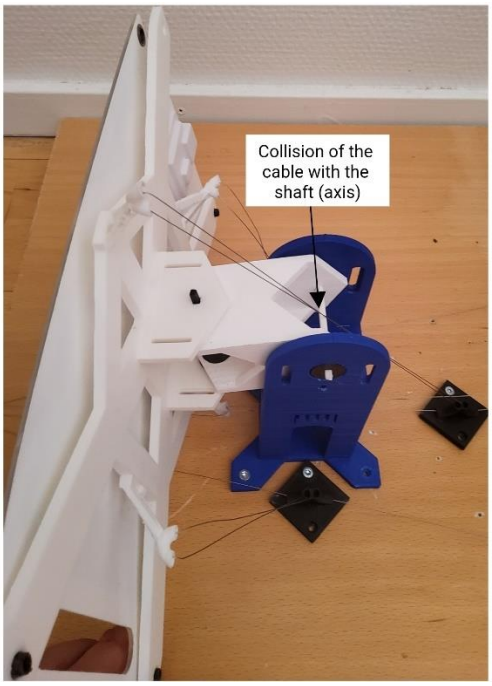


Figure 31: Null moment limit state

- **Limit state at maximum torsional moment**: in the position in which the cable is contained by the plane containing the support of the panel, a maximum fictitious torsional moment state is reached. In this position the cable cannot generate a force vector perpendicular to the panel plane and can therefore no longer a rotation. The angle formed by the plane containing the panel with the surface plane, marks the limit angle for maximum torsional moment. This limit state can be seen in the following picture.

Figure 32: Limit state at maximum torsional moment

The distances between the axes of rotation contain in the rotula and the panel directly affect the maximum angle for the limit state of zero moment. The greater the distance of these axes, the less degrees of rotation the null moment limit allows (keeping all other variables constant).

The distance between the lower sheaves and the main tower is the variable directly affecting the determination of the maximum angle of rotation by null moment. The greater the distance, the more degrees of rotation the null moment limit allows (keeping all other variables constant).

The distance between the lower sheaves and the main tower is the variable directly affecting the determination of the maximum angle of rotation by maximum moment. The greater the distance, the less degrees of rotation the maximum moment limit allows (keeping all other variables constant).

The sum of distances that are kept constant are the distances from the upper sheaves to the surface plane. If the lower sheaves could be automatically placed under the upper sheaves, all of the above assumptions would be fulfilled.

All these conclusions are scalable to any size since these conclusions are drawn from relative positions and sizes, and not from absolute measurements. Finally, all the above conclusions have been taken into consideration to select the positioning and size measures for the "Solar stand structure" segment. It should be added that the only variable that has not been determined by the previous conclusions has been the width of the tower and the break. This width has been determined by the size of the potentiometer.

Once it was proven that the tension is not constant for prototype 1.0, it remained to study the solutions to solve the untensioning problem of the cable. These were the 4 proposed solutions:

- Movable lower pulleys: this proposal is based on designing a mechanism that somehow allows the movement of the lower pulleys to be located just below the upper pulleys. This solution has not been tested, due to the complexity of designing such a mechanism (this will be mentioned in the limitations section).
- Dead weight: this proposal is based on the use of 4 suspended weights anchored at the 4 ends of the 2 cables. These weights would be in charge of pulling the rope when it is untensioned. The main advantage of this proposal is the ease of controlling the tension of the rope. On the other hand, the main disadvantage is the increase in the unit price of the support, since there would have to be 4 extra devices per panel support. This proposal has been tested, and its effectiveness has been demonstrated. Finally it has been discarded because of its main disadvantage, since the idea of the global design is to lower as much as possible the unit costs of the "Solar stand structure" segment.

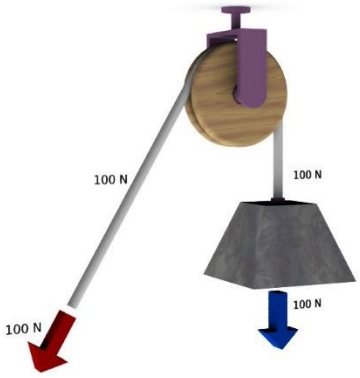


Figure 33: Pulley force distribution[22]

- Constant tension spring: this proposal has an operating principle very similar to the previous one, but instead of using the gravitational force, it makes use of the force of a spring. It has the same advantages and disadvantages. Finally, its implementation has been discarded for the same reasons as the previous proposal.

Figure 34: Constant force spring[23]

- 4 motorized system: this is based on the use of 4 steppers instead of two as proposed in prototype 1.0. The main disadvantage is the increase of the system price, but unlike the previous 3 proposals, the cost of this proposal does not scale with the increase of steppers. This would be because 4 motors could theoretically move "n" supports simultaneously. This has finally been the chosen solution applied in the final prototype.

5.2 METHODOLOGY FOR THE GEAR DESING

For the calculation of the gear design, an internal inventor tool was used. For this purpose, 4 variables had to be determined:

- Center-to-center distance: this distance is imposed by the rotula design. This would correspond to the distance from the center of the potentiometer to the axis of rotation.
- Gear ratio: this is the ratio between the number of teeth in coupling gears. It is equivalent to the gear ratio, how many turns one gear turns relative to the other. As the degrees of freedom of rotation of the potentiometer is 280º and the maximum angle of rotation of the panel is 180º, the optimum gear ratio would be 280/180= 1,55. It is optimal from the point of view of the measurement accuracy of the potentiometer. The Arduino Mega has a 10-bit analog definition, or in other words it can measure 1024 different values from 0 to 5V. If the gear coefficient of 1.55 is applied, 180 degrees of panel rotation would be equivalent to 280 of the potentiometers, this would be turned to 5V output, so the Arduino would measure a maximum value of 1023 (since it goes from 0 to 1023). This would translate to the Arduino Mega being able to measure increments of 0.175º of rotation (180 degrees/1024). If the coefficient was 180 degrees of panel rotation would be equivalent to 180 of the potentiometers, this would be turned to 3.22V output, so the Arduino would measure a maximum value of 660. This would mean that the Arduino Mega could measure increments of 0.27º of rotation (180 degrees/660). Ultimately, a value of 1.5 gear coefficient was chosen as this was the closest value that the Inventor program allowed. It should be noted that the closest possible lower value should be selected, otherwise the potentiometer would not be able to measure the 180 degrees of rotation.
- Number of teeth: this depends mainly on the definition of the 3D printer, the higher the number of teeth the smaller they are and the worse the printer can print them. On the other hand, an excessively low number of teeth can cause a bad transmission of forces. The number of teeth of one of the two gears associated with a shaft has to be completed (since the tooth calculation of the remaining gear can be derived by means of the gear module). In this case it was chosen to set the value of the number of teeth of the gear that is coupled to the potentiometer. In order to select the best option, different values have been tested, of which a value of 25 teeth has been chosen.
- Thickness of the gear: this has been determined in 5 mm.

There are more variables such as pressure angles or propeller angles, these have been selected automatically by the inventor software.

5.3 CHARACTERIZATION AND CALIBRATION OF THE SENSORS

5.3.1 AMPEREMETER

Due to discrepancies of 7.8% between the current values measured by the analog input of the Arduino Mega and values measured with a multimeter, it has decided to characterize the sensor.

The output voltage at zero current is 250mV. Its nonlinearity error is ±0.2%, its sensitivity error is around ±1.5%, its offset error is ±15mV and its total output error is between ±2% and ±5%. Its power supply is 3.3 V with a typical consumption of 4.5mA.

Based on the data provided by the manufacturer, the sensor has been tested in MAKENTNU's laboratory[24]. The test consists of arranging circuits with different resistance values fed by a power supply of known voltage. By applying ohm's law the current value can be derived. The amperemeter should be connected in series to the circuit and the output voltage values should be measured using a multimeter. Standard resistors of 4.81 Ω with tolerances of less than 0.05% have been selected. The selected power supply has been a 12V power supply with a maximum current of 5A (twice the current of the amperemeter's measuring range). In this way, the reliability of the currents produced can be assured. Figure 35 shows the results of the input currents and their respective output voltages in the test performed.

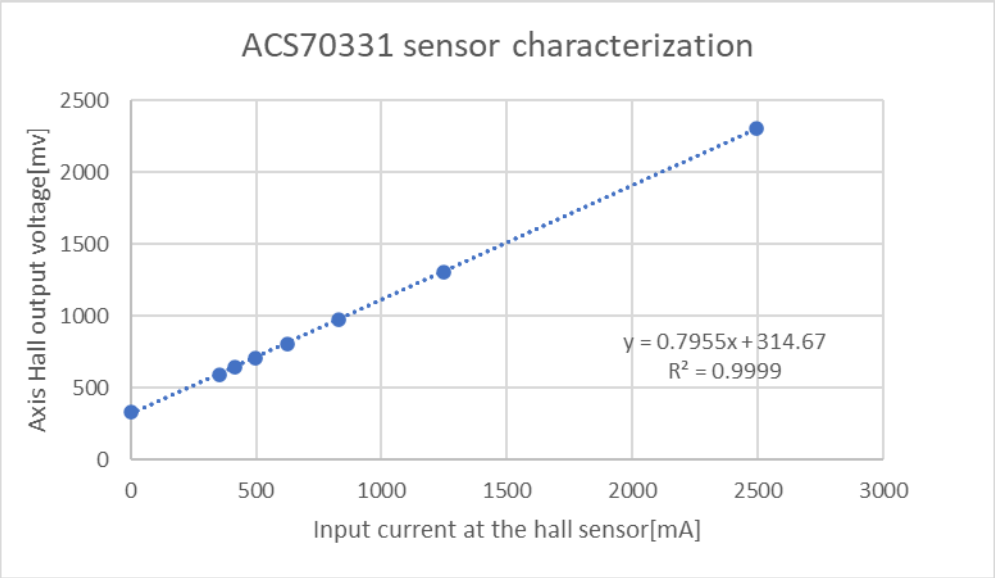


Figure 35: ACS70331 sensor characterization

It can be seen at a glance and through the coefficient of determination (R), that it is a purely linear function, with a cut on the y-axis of 314.64mV. This value is far from the offset published by the manufacturer. Nevertheless, the linearity and sensitivity are confirmed to be fairly similar to that found in the spec sheet.

With the new characterization of the sensor, the absolute measurement error by the Arduino is reduced by 7.4%, so there are still one or more factors that are generating discrepancies in the measurements. Analyzing the data collected by current and voltage measurement of the Arduino Mega it has been observed that there is a fluctuation of the measurements (maintaining constant boundary conditions of the solar panel). This could be explained by the existence of electrical noise produced in the internal circuits of the solar panel. During the amperemeter characterization tests, a buzzing sound coming from the solar panels was detected. This buzzing sound would be produced by variations of the magnetic fields generated in turn variations of electric potentials. In this way, electrical noises can generate magnetic fields that cause magnetic materials to vibrate[25]. For the measurement of the noise frequency, an audio software capable of performing a spectral analysis has been used. It makes use of a microphone placed close to the solar panel. Figure 36 shows the results of the test. There are 3 axes, the time axis (vertical axis), the frequency axis (upper horizontal axis) and the wave amplitude axis (color axis).It can be observed that the main frequency of the noise is located at 3094hz. To the right of this frequency, the harmonics of this main frequency can be observed.

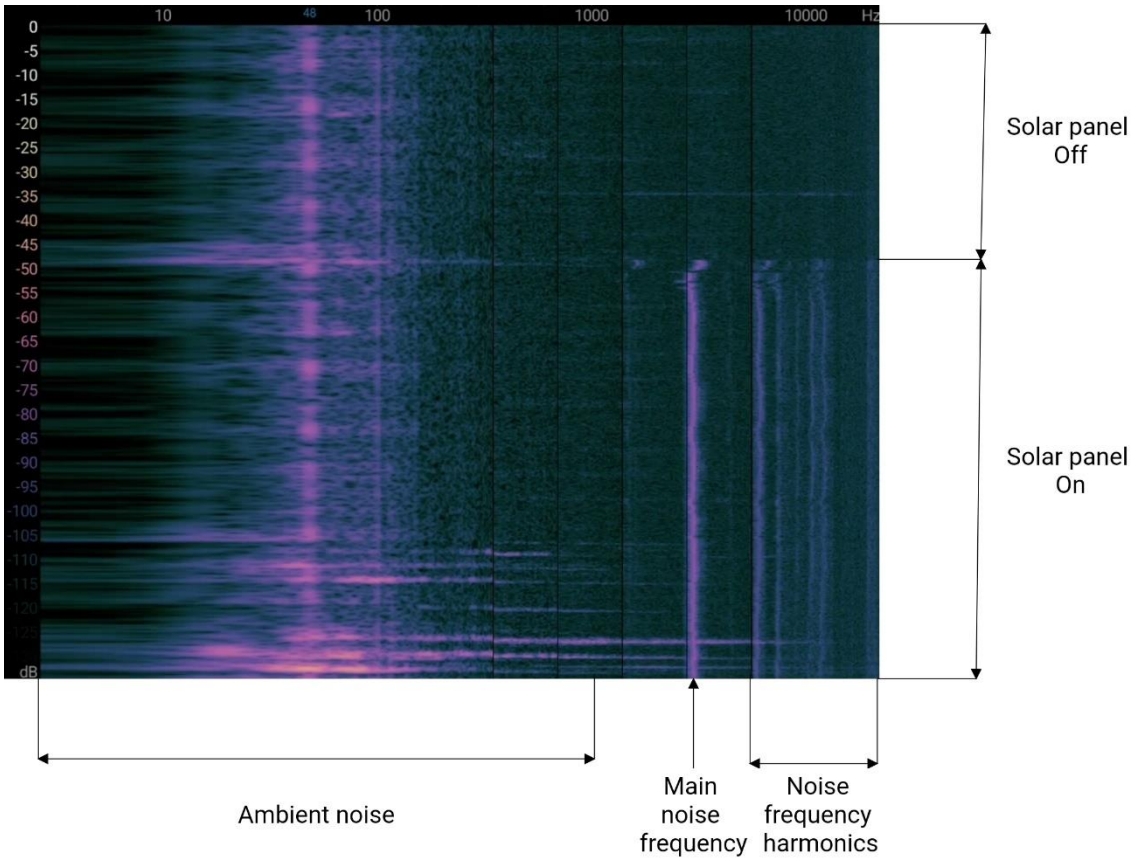


Figure 36: Electric noise sound analysis

Two alternatives have been proposed to solve this problem. The first one consists of incorporating a low pass filter that eliminates the high frequencies. This would incorporate a series resistor and a parallel capacitor. The second is to perform an arithmetic average of the collection of "N" measurements taken in series in a time interval.

The maximum sampling frequency of the Arduino Mega is 980 hz. This would in principle rule out the second option due to the Nyquist-Shannon sampling theorem. This theorem shows that accurate reconstruction of a continuous baseband signal from its samples is mathematically possible if the signal is band-limited and the sampling rate is greater than twice its bandwidth. Since it is not necessary to measure the current waveform but its average value, the sampling rate of the Arduino Mega is valid since it does not coincide with any multiple of the electrical noise frequency.

Between these two options, the second option has been chosen since no additional components are required. For data acquisition, the value of N measurements has been set to 100, and the acquisition frequency to 980hz (depending on the analog port to which it is connected).

By incorporating the above settings, the absolute error of the Arduino Mega measurements has been reduced to 0.82% with respect to the measurements provided by the multimeter. Finally, this error is acceptable for the framework of this project.

5.3.2 POTENCIOMETER

As mentioned in section 5.3.2, the potentiometer has a total resistance tolerance of ±10 % and an independent linearity of ±10 %. This magnitude of error is not acceptable for

the use of a potentiometer as a precision sensor. For this reason it has been proposed to characterize this component. For this purpose, a test has been carried out in which different values of rotation have been given to the potentiometer and measured by means of the analog input of the Arduino Mega. To measure the angle of rotation of the potentiometer, an angle transporter has been used. In Figure 37 one can see a representation of the acquired data.

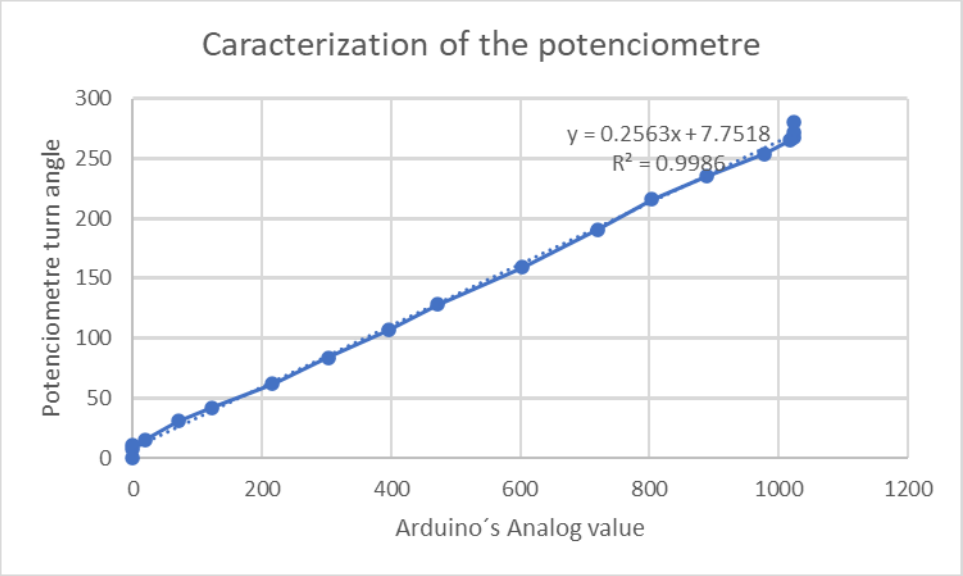


Figure 37: Characterization of the potentiometer

Two conclusions can be extracted from this graph:

- The first 11º and last 12º of potentiometer rotation do not show any change in resistivity.
- The R^2 index is very close to 1, so the linearity of the component can be assured.

From this characterization it is extracted that the degrees in which the potentiometer really varies its resistivity are 257º (280º-11º-12º) and a linear behavior can be assumed within this 257º. This has been taken into account in the coupling of the gears and the potentiometer.

5.4 WEBSITES DESIGN

This segment will expose the three web connections to which the Arduino mega exchanges data. Being a web connection and not local, all of these allow access to the data from anywhere on the planet.

5.4.1 MAIN WEB PAGE

This allows real-time monitoring of analog values measured by the Arduino mega. It is connected through the ESP 01, with which it executes requests in PHP language[26]. These requests allow the exchange of information with a MySQL database. It is here where the data accessed by the web page itself and the Arduino mega will be temporarily stored. There are three types of variables:

- Input boolean variables: these take on values of 0 or 1. These act as virtual buttons to trigger an action on the Arduino mega. It is the user who modifies them. The first button "Movement Capability" is used to activate or not the

motors. Whenever off the panel will remain static. The second button "Safe Mode" is used to position the panel in a preset safe position. The third button "Automatic mode (ON) - Manual mode (OFF)" is used to select an automatic or manual panel positioning mode. If manual is selected, the angles must be entered manually. This manual mode is mainly used to calibrate and test the prototype. The next two buttons "Photoresistors (ON) \ Sun position calculation (OFF)" and "Step tracking mode (ON) \ Potentiometer mode (OFF)" allow to select the panel movement algorithms. These algorithms are further detailed in section 5.5.

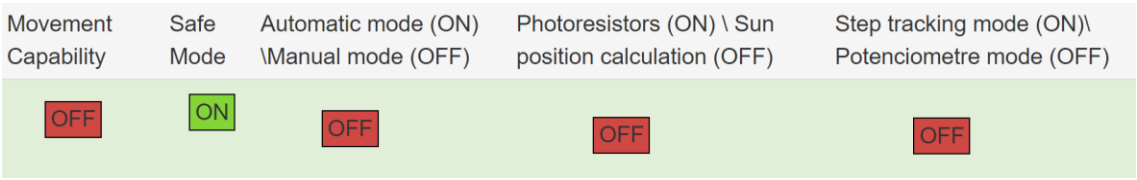


Figure 38: Buttons associated to the output boolean variables

- Input Integer variables: these send the value stored in their boxes to the Arduino Mega. They are used to set the manual rotation angle of the panel. For these to work the third button must be off.

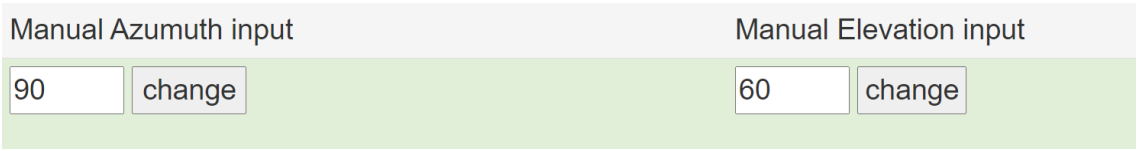


Figure 39: Boxes for manual input of angular values

- Output integer variables: these integer variables are used to display analog data measured by the Arduino Mega in real time.

Azimuth Angle	Elevation Angle	Voltage	Current
329	1022	0	139

Figure 40: Output integer variables

The URL to access the main website is:
https://solartrackertrondheim.000webhostapp.com/index.php

The source code of the web page is written in PHP and HTML language. It can be found explained in p.95.

5.4.2 SPREADSHEET FOR THE CALCULATION OF THE POSITION OF THE SUN

This spreadsheet automatically calculates the position of the sun in degrees azimuth and elevation. For this purpose only the positioning coordinates of the solar panels have to be determined. It is automatically synchronized with the satellite clock to perform the necessary calculations. The calculation of the sun's position is based on the equations shown by J.J. Michalsky in "Astronomical Algorithms"[27]. The accuracy is 0.01 deg, the observed values may vary from the calculations since they depend on: the composition of the atmosphere, temperature, pressure and other conditions.

The communication with the Arduino Mega is done by the ESP8266 Wi-Fi module. It sends a data request to the Spreadsheet and it sends the three variables compressed in

a string format. These variables are the azimuth angle, the elevation angle and the time. This string format will later be separated in the Arduino Mega. In this way only one request has to be sent to receive the three variables.

The URL to access this spreadsheet is: https://docs.google.com/spreadsheets/d/1g75Y8BZ386HefE1u6Tbkg5Fr1V0LoHuQvh8PYUYIn2Y/edit?usp=sharing

The source code that allows the connection of the Arduino Mega is written in JavaScript[28] and can be found explained in the annexes in p.84 .

5.4.3 SPREADSHEET FOR DATA COLLECTION

This spreadsheet stores the voltage, current, elevation angle and azimuth angle data (the angles provided by the potentiometers). Along with this data, the date and time of receipt of the data is also recorded. In addition, with the voltage, current and time data, the energy production is calculated. This spreadsheet allows further analysis of the data.

	A	B	C	D	E	F	G
1	Time	Hour	Voltage(V)	Current (mA)	Energy Production (Wh)	Azimuth	Elevation
2	11/05/2022	0:18:43	4,56	140	0,19152	25	90
3	11/05/2022	1:18:43	4,56	123	0,729144	89	89
4	11/05/2022	2:18:43	4,6	350	3,703	73	34
5	11/05/2022	3:18:43	4,56	356	5,357088	88	90

Figure 41: Spreadsheet for data collection

The URL to access this spreadsheet is: https://docs.google.com/spreadsheets/d/1fAW-WAu-lRaT28DQSrIhMrFxW3-LFiHu42EQR5wUw1s/edit?usp=sharing

The source code that allows the connection of the Arduino Mega is written in JavaScript and can be found explained in the appendix p.95.

5.5 ALGORITHMS

For the development of the prototype, 4 algorithms have been developed for the movement of the structure. In the "prototype testing" section, these algorithms will be compared. They make use of different sensors and intrinsic characteristics of the steppers. They can be grouped into two categories. The first is the algorithms used to know the position of the panel. These are the potentiometer-based algorithm and the motor step tracking based algorithm. The second category is based on algorithms that provide a direction of rotation. These are the algorithm based on photoresistors and the algorithm based on mathematical calculation of the sun position. These two categories can be combined using the switches found on the main web page, thus producing 4 different "packages" of different algorithms (one package consists of one algorithm from each category). The development of these 4 packages has not only been designed to shop them against each other and see which one is the most suitable. There is also the possibility of incorporating all the algorithms in the final code, as this provides robustness to the design. When designing a product that is intended to be installed outdoors, multiple problems can occur with the sensors, such as: partial or total fouling of one or more of the photoresistors, increase of clearances in the gears and thus loss of pressure in the potentiometers, etc. In this way, if there is a problem with any sensor, there would be a replacement method. In addition, these can be used to monitor each other, selecting one package for panel movement and another to check that the panel is within an error range.

5.5.1 DIRECTION ALGORITHM

These algorithms are used to determine in which direction to move the motors and are categorized into two types:

5.5.1.1 SUN PREDICTION DIRECTION ALGORITHM

This algorithm is based on using the spreadsheet described in section 5.4.2, to receive a target angle.

5.5.1.2 PHOTORESISTORS DIRECTION ALGORITHM

This algorithm makes use of 4 photoresistors located at the 4 corners of the panel. These photoresistors are able to vary their electrical resistance depending on the light they are receiving. By transforming the resistivity into a voltage value that the Arduino can read, their values can be bought to determine in which direction the panel should be moved. This way whenever there is a discrepancy between the photoresistor resistivities the board will move until they all have the same value (within an error range).

Due to the limited number of analog inputs and as explained in previous sections, a multiplexer has been used to read the photoresistors. Figure 42 shows a flowchart of how this algorithm works.

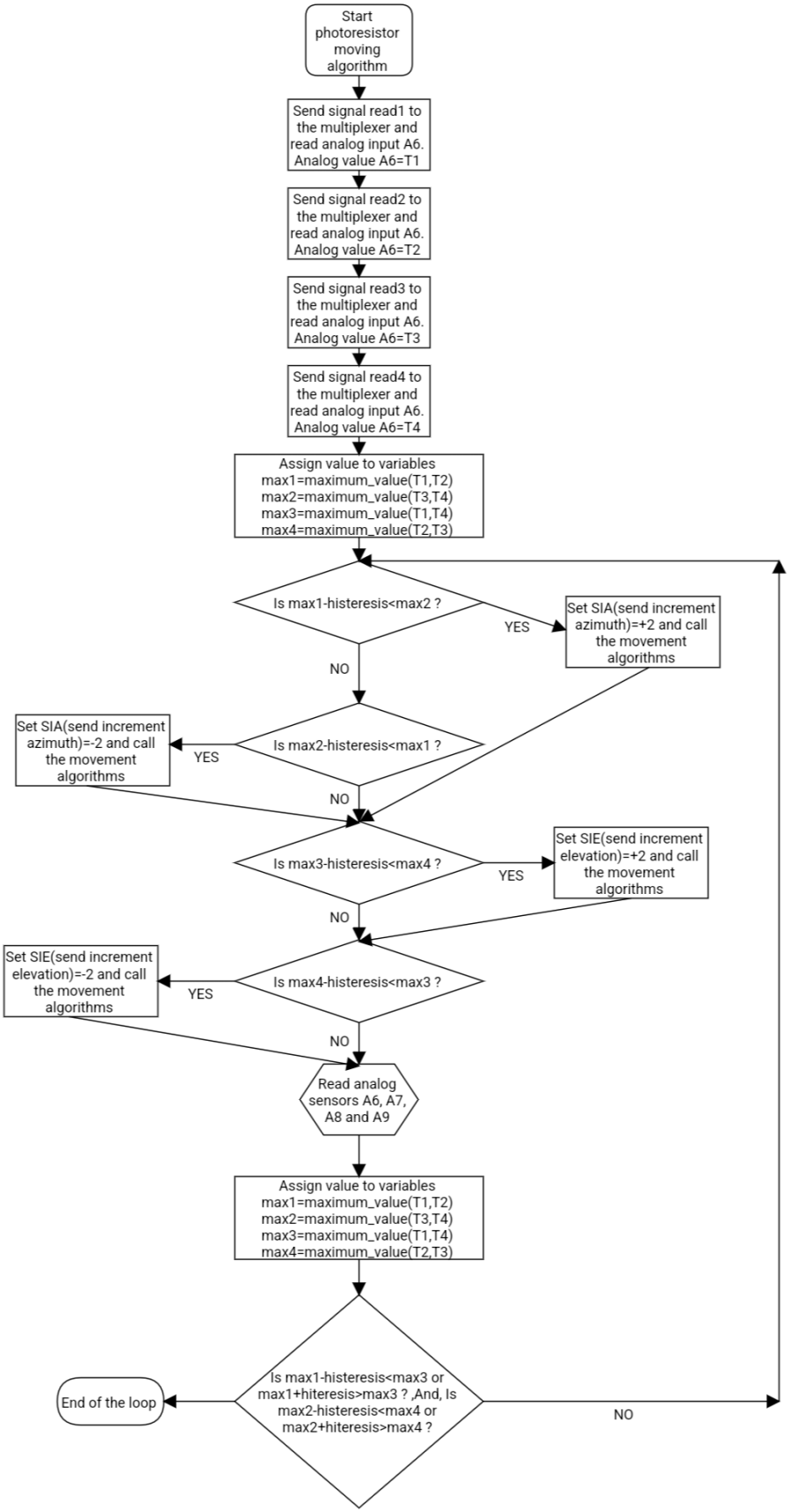


Figure 42: Flow diagram of photoresistors direction algorithm

5.5.2 FEEDBACK POSITION ALGORITHMS

These algorithms allow gathering information on where the panel is positioned at any given moment. They are fundamental since they are the way that the steering algorithms know where to be in order to know how to move the panel. They are divided into two types: potentiometre position feedback algorithm and step tracker position feedback algorithm, which will be explained below.

5.5.2.1 POTENTIOMETRE POSITION FEEDBACK ALGORITHM

This algorithm makes use of two potentiometers located in the main tower. These can vary their resistivity according to the angle at which they are rotated. For the transmission of the panel rotation, gears are used to connect the rotation axes with the potentiometers. More detailed information on these gears can be found in section 5.2. If the resistivity is converted into a proportional voltage value the Arduino can measure the turn angle of the potentiometer and therefore in the panel. In addition, it makes use of "WIRE TENSION DETECTION SYSTEME" detail on section 4.4.2. In this way the Arduino can also measure the tension of the cable at any time and thus it can adjust the movement of the motors to keep the cable tension within a range. In Figure 43 one can see its flowchart.

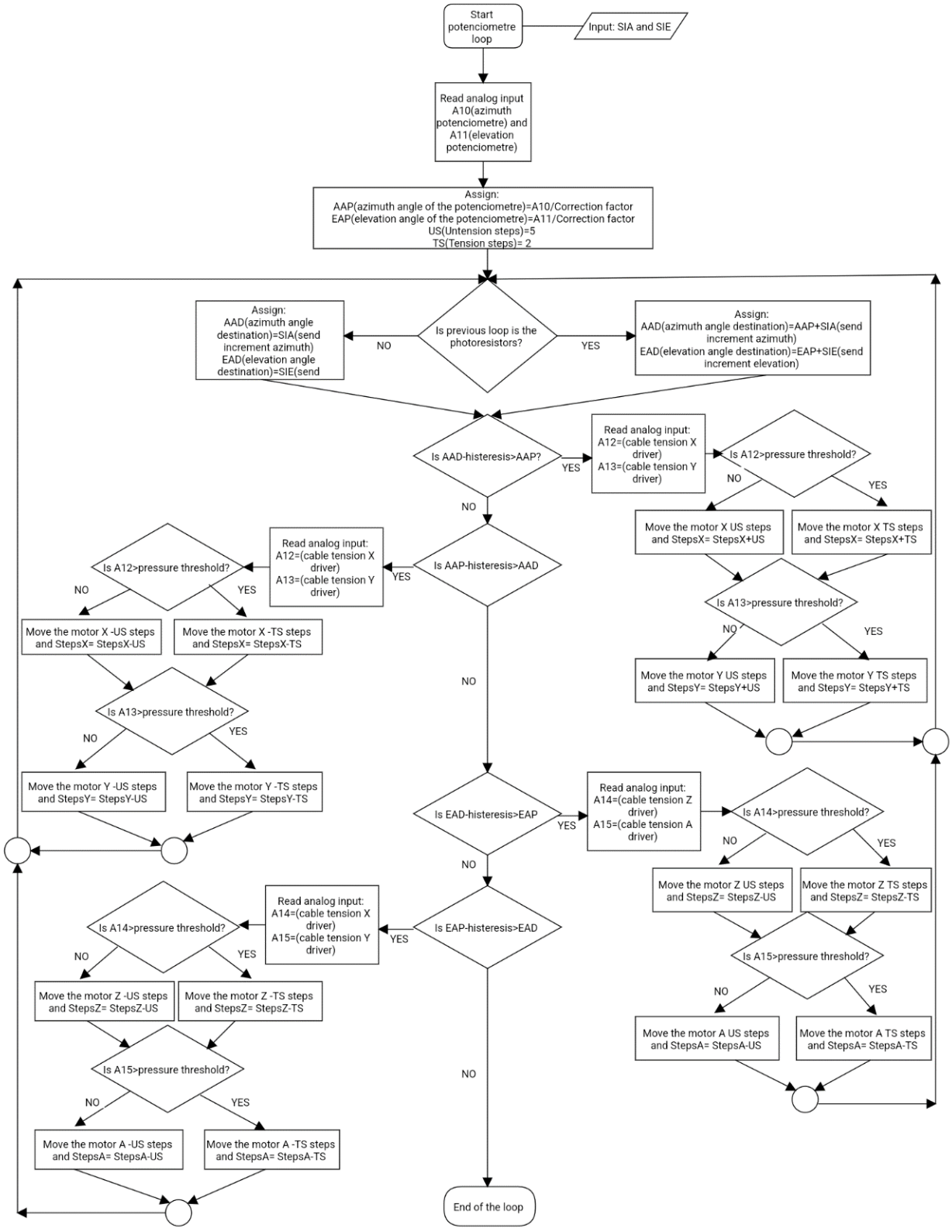


Figure 43: Flow diagram of potentiometer position feedback algorithm

5.5.2.2 STEP TRACKER POSITION FEEDBACK ALGORITHM

This makes use of the steppers' ability to move in 1.8° angular increments. If one start from a known position, one can record the steps. A relationship between steps and panel position angle can then be found. This would be the principle of operation of 3d printers. For this purpose this algorithm makes use of 3 different functions or sub-algorithms.

The first of them is one used for the calibration of the panel position. This is based on the positioning of the panel at a known elevation angle and azimuth. As will be explained in the limitations, the implementation of this positioning is done manually for reasons of scope of the thesis. A possible solution for the calibration function would be the incorporation of push buttons on the main tower, which are activated when the panel is in a specific position. With the final code it should be manually calibrated each time this step tracker position feedback algorithm is chosen.

The second one is an algorithm that translates angles into steps. To do so, it makes use of a database that allows to relate a given angle to a given number of steps. For the development of this database the Matlab simulation described in section 5.1 was used. From this simulation the distances between lower and upper sheaves for each possible azimuth and elevation angle can be found. With these distances the rope lengths required for these positions can be calculated. Then, knowing the drum diameter and the angle increment for each motor step, a relationship between these steps and a panel angle increment can be found. The use of a database or data table is based on two principles. The first is that the step/angle increment ratio is not constant. The second is that the computation of this ratio has a very high computational overhead for a single-core, low-frequency processor, such as the Arduino Mega. In this way the ratio calculations would be done only once, thus speeding up the algorithm. In Figure 44 one can see its flowchart.

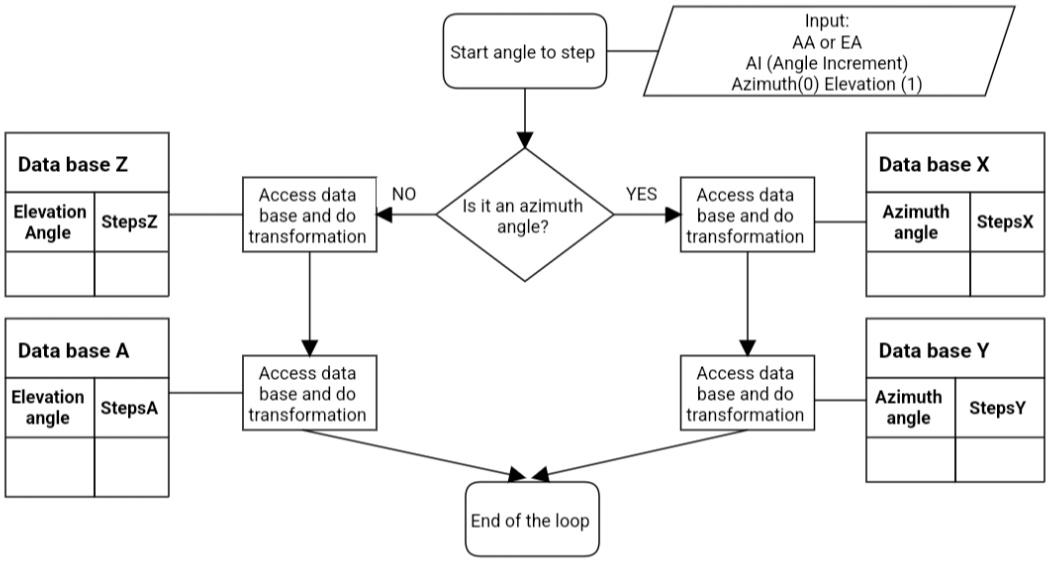


Figure 44: Flow diagram of the angle-to-steps conversion algorithm

The third would be to perform a reverse transformation, from steps to angles. This is based on the same principles as the previous one. In Figure 45 one can see its flowchart.

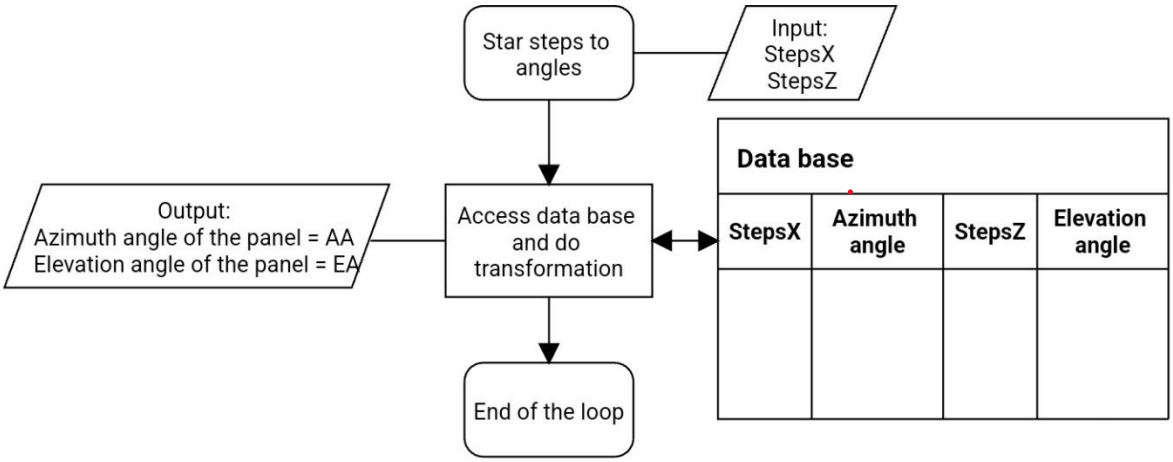


Figure 45: Flow diagram of the steps-to-angle conversion algorithm

In Figure 46 one can see the steps/angle elevation ratios calculated.

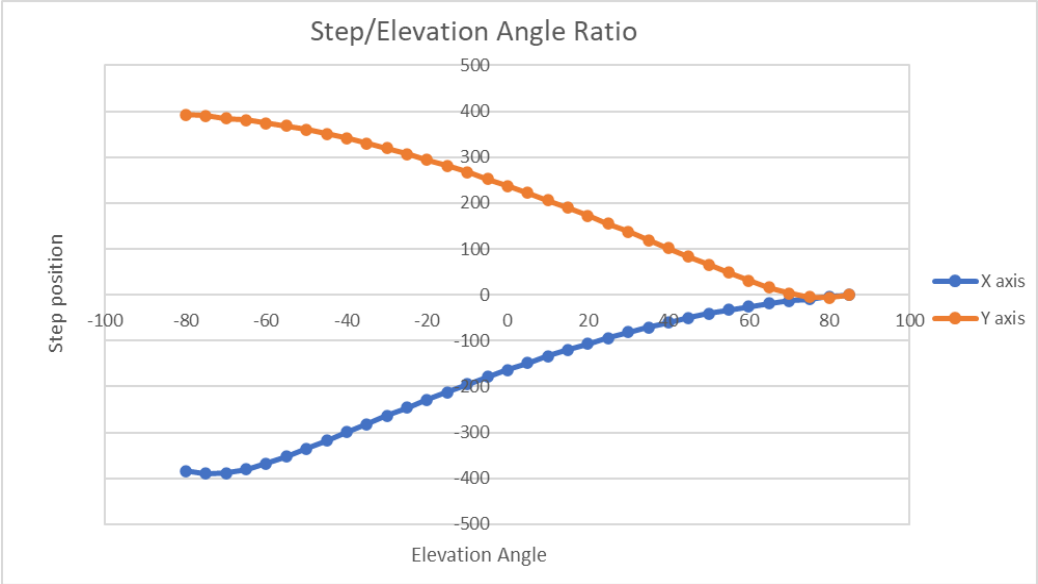


Figure 46: Step/Elevation angle ratio

Finally, the flowchart of the step tracker position feedback algorithm can be seen in the Figure 47.

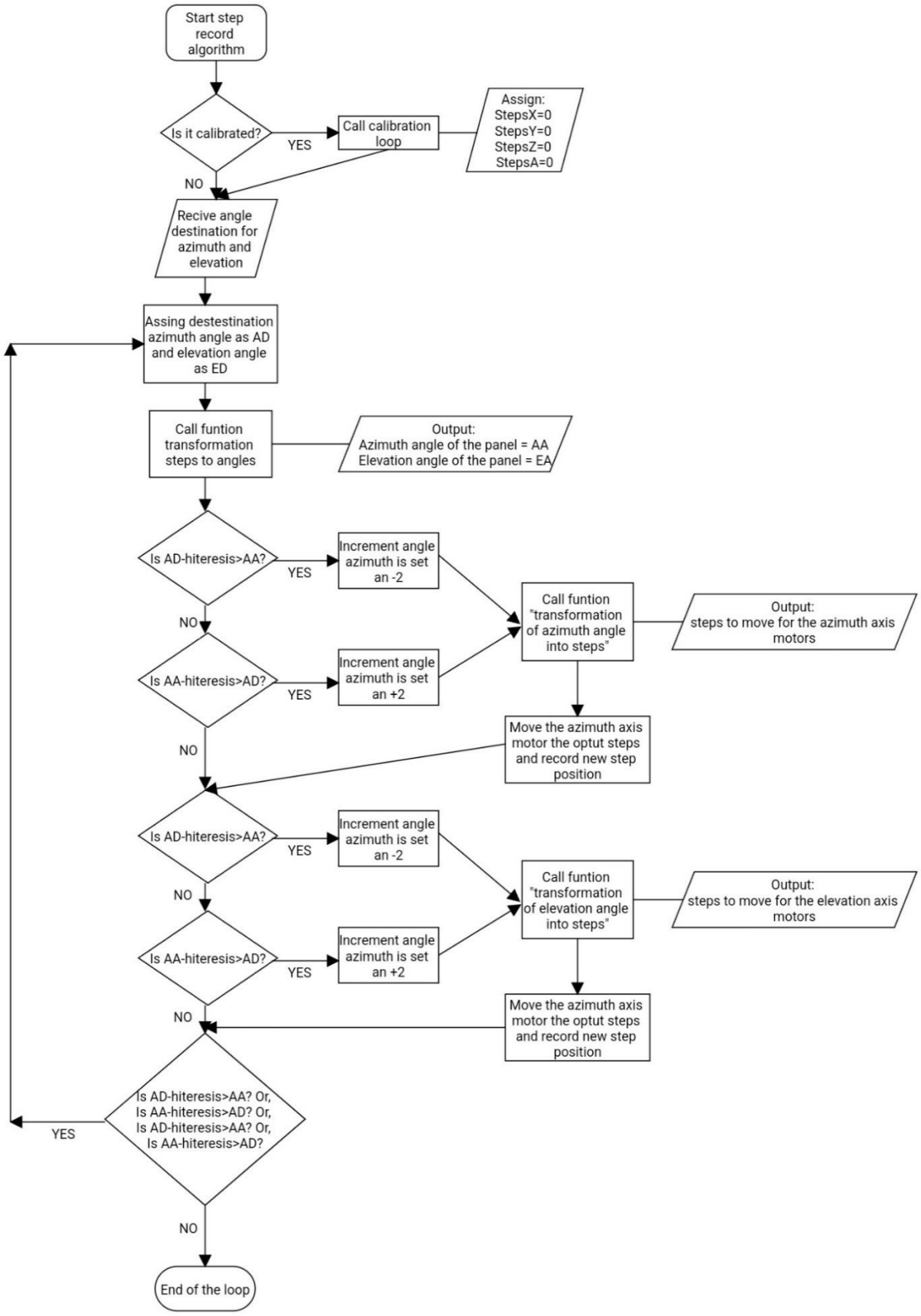


Figure 47: Flow diagram of step tracker position feedback algorithm

CHAPTER 6: PROTOTYPE TESTING

This section will start explaining in a summarized way how the prototype works to provide an overview. Subsequently, the conditions under which the prototype was tested will be defined. This segment will end with the results and conclusions of the tests.

6.1 OVERVIEW OF THE PROTOTYPE'S OPERATION

To start up the prototype, the power supply must be connected and the Wi-Fi modules must have internet access. To start up the tracking system, the main web page must be accessed first. From there it must be activated and selected from the different operating modes. Depending on which operating mode is selected, different sensors are used. The Arduino Mega, the control center, will receive this information via the ESP 01 Wi-Fi module. The Arduino Mega then reads the values from the different sensors and sends them to both the main web page (via the ESP01) and the data collection spreadsheet (via the NodeMCU). The purpose of this is both for visualization and for gathering information for monitoring the prototype and for further analysis of its operation. Once this data is sent, the Arduino proceeds to move the 4 steppers. To do this the CNC Shield V3.0 was used. This shield makes use of the drivers and the power supply to translate the commands of the Arduino into movement. To the axis of these steppers are connected drums that allow winding multiple cables. It is these cables that, with the help of pulleys, transmit the movement to the structure. It is this ability to add multiple connections that allows the control of as many supports as the motors can move.

The following image shows the connection and wiring scheme. A more detailed figure with the naming of each component can be found in appendix p.75.

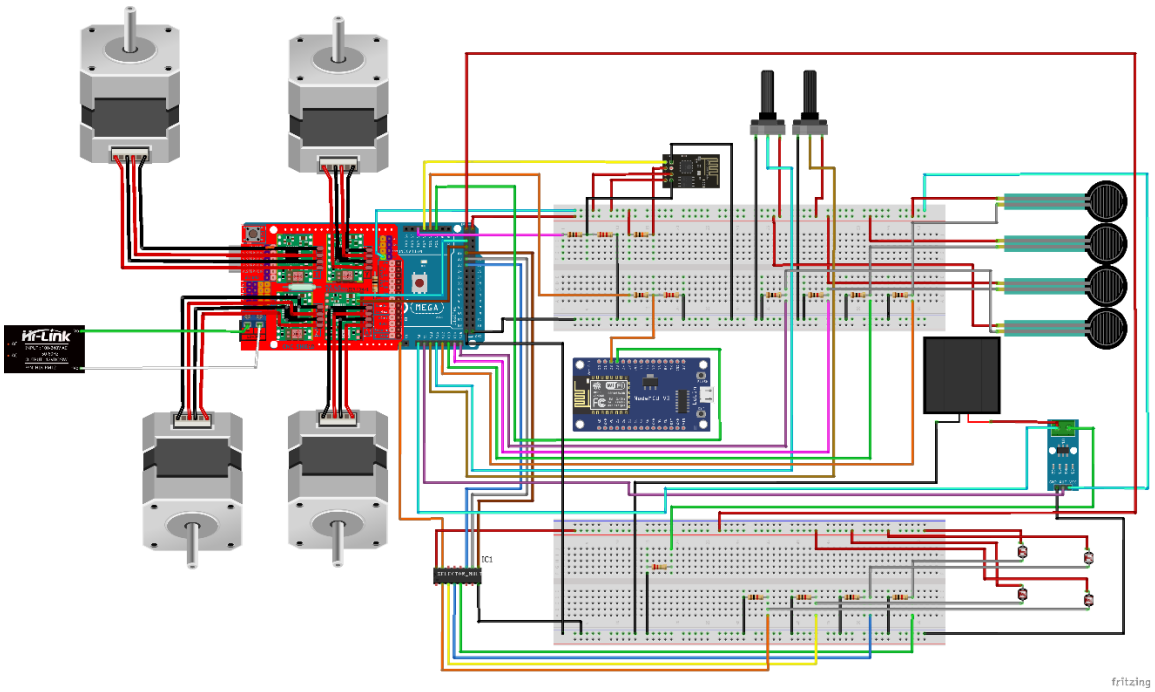


Figure 48: Connection scheme of the prototipe[29]

An image of the finished prototype is shown in the following picture.

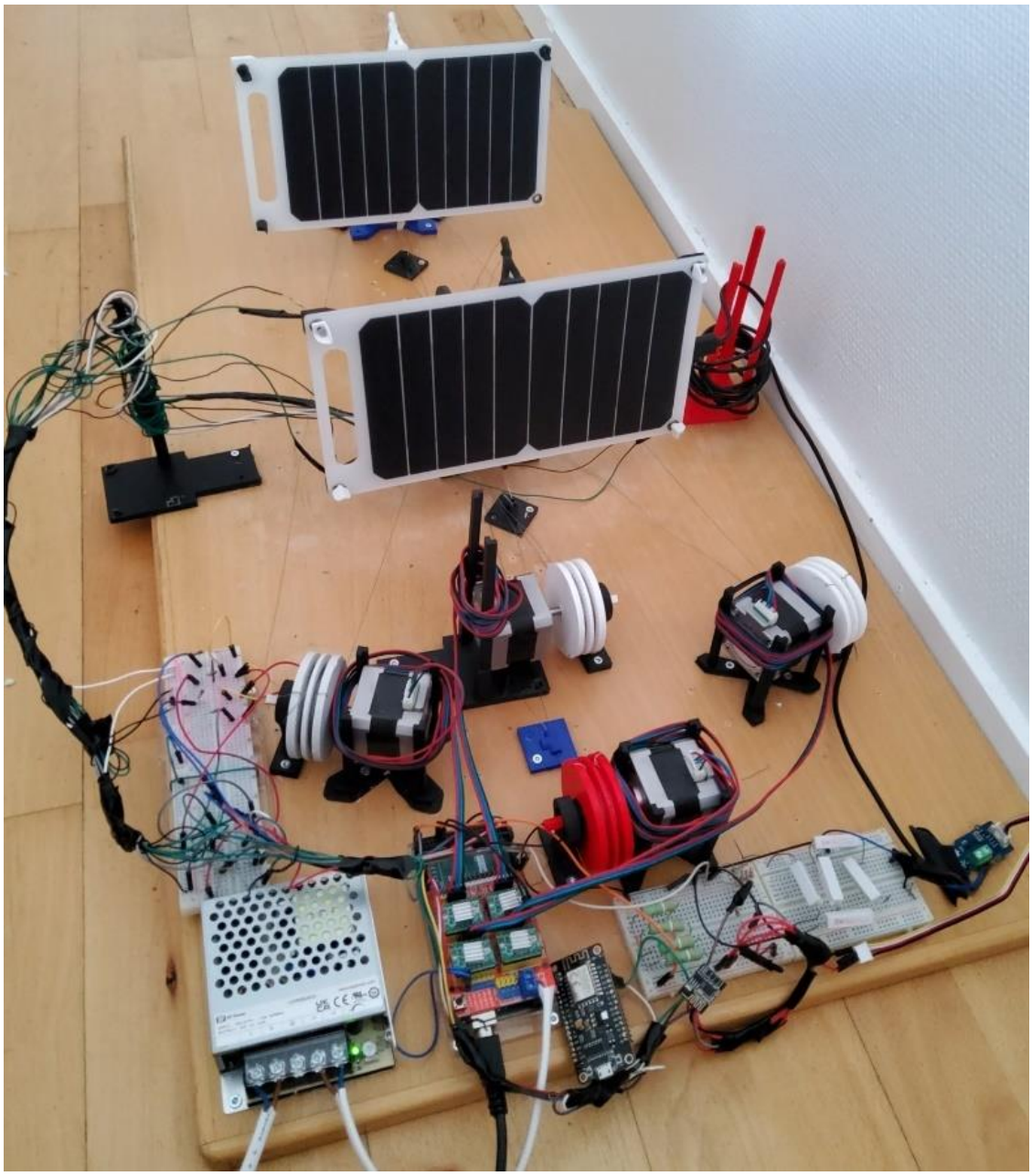


Figure 49: Small scale finish prototype

6.2 PROTOTYPE TEST CONDITIONS

The tests have been carried out with both cable versions mentioned in section 4.1.5 and several conclusions have been drawn. These revolve around the elasticity of both cables and how this characteristic affects the behavior of the system. Depending on the algorithm used for the motor motion the elasticity quality is either an advantage or a disadvantage.

For algorithms based on the use of pressure sensors this elasticity was an advantage since it provides more continuous tension readings, i.e. the minimum and maximum tension step in the cable is less pronounced. This is because the elasticity of the cable itself absorbs part of the force applied to it as a spring does. This lowers the risk of the motor exceeding its torque limits, thus preventing it from going out of step.

On the other hand, for algorithms based on step tracking, the elasticity characteristic would not present an advantage. It would complicate the calculations performed by the Matlab simulation. Recall that this simulation is used for the creation of the conversion table from steps to angles and vice versa. For the created reaction obtained by the simulation of an increase of distance between pulleys in a number of steps of the motor, an infinite modulus of elasticity or Young's modulus will be assume. This would be the same as saying that the cable would not bind due to changes in cable tension. In order to incorporate an elastic cable for this type of algorithm, a force study should be incorporated in the simulation. This study should add the frictional forces on the pulleys and the forces required to move the panel. These forces would change according to the angle of rotation since the distance of the force vector containing the tension in the cable is not constant. Therefore, a new conversion table from angles to steps would have to be created. This is beyond the scope of this thesis. The steel has Young's modulus of 205.000MPa and the fishing line 1.300MPa. Finally a steel wire rope has been chosen for the rest of the tests.

The prototype has can turn 170º in the elevation axis and 160º in the azimuth axis. The reference position 0º Azimuth and 0º Elevation can be seen in Figure 50Figure 50.

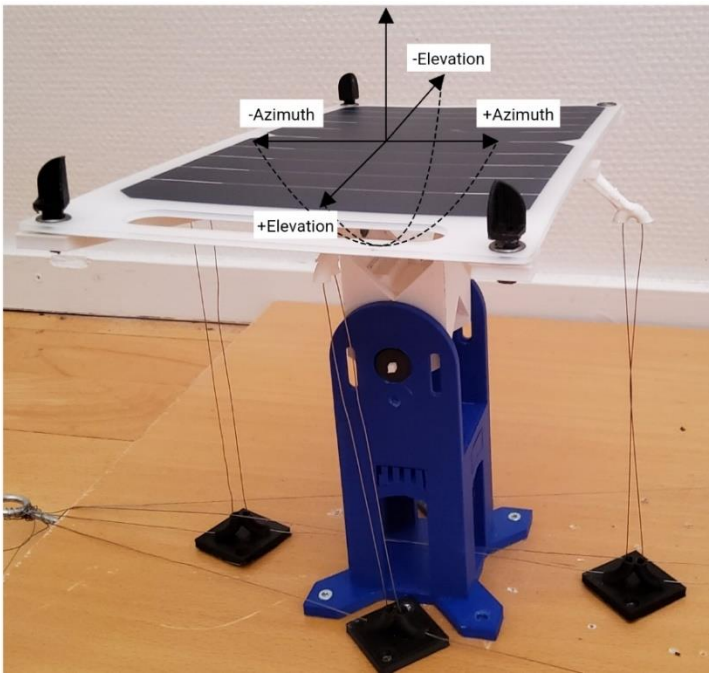


Figure 50: Angle reference position

These were the procedural conditions under which the performance tests were performed:

- These tests have been performed under cover since the prototype is not prepared for outdoor conditions. Therefore, a 500W lamp has been used to simulate the sun.
- The incorporation of the multiplexer define in section 4.3.6, has not been possible due to time constraints. As mentioned, this multiplexer was intended to apply the analog inputs of the Arduino Mega, for this reason the different motion algorithms have been tested separately.
- When testing indoors, a path sequence was stipulated and that consists of moving the panel in this order: 0º Azimuth, 0º Elevation →0º Azimuth, 80º

Elevation →0º Azimuth, -80º Elevation→ 0º Azimuth, 0º Elevation → 85º Azimuth, 0º Elevation →-85º Azimuth Elevation 0º →0º Azimuth, 0º Elevation. This sequence of movements can be seen in Figure 51.

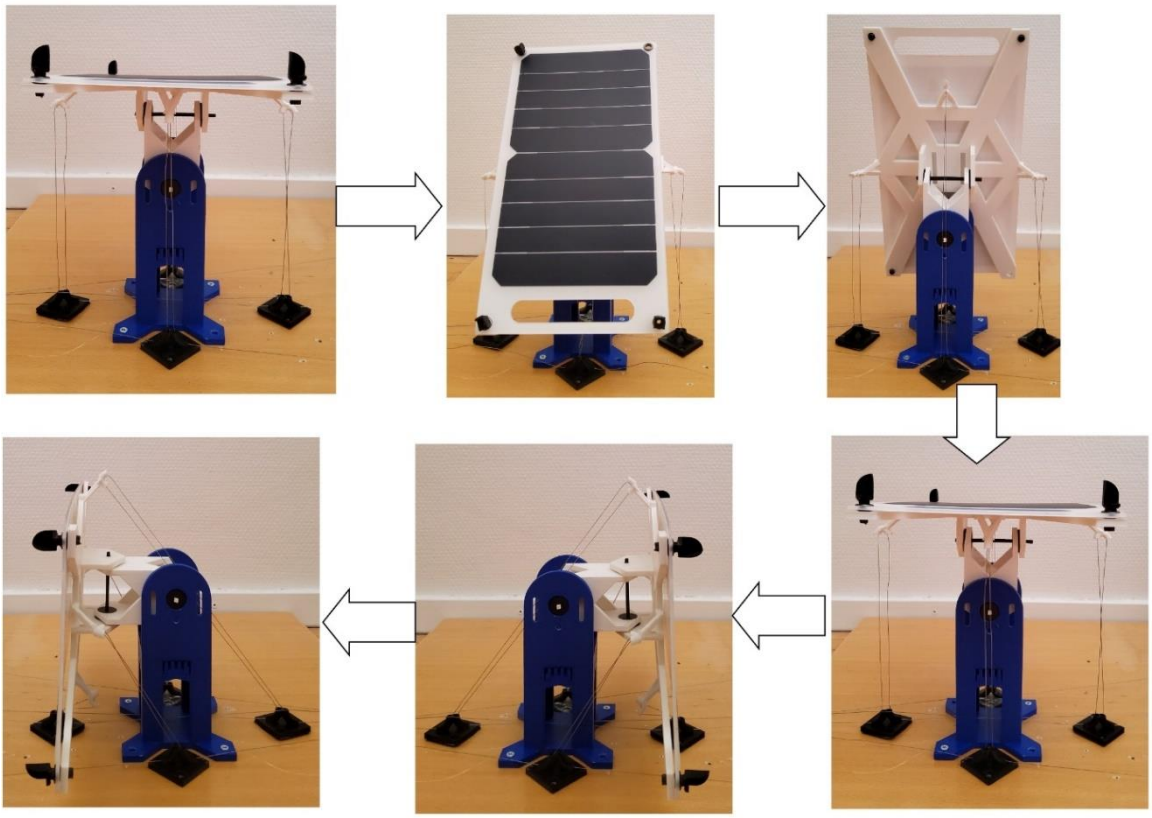


Figure 51: Test path sequence

- Multiple iterations of the travel sequence will be performed to test the ability to maintain accuracy.
- A "precision test" of the prototype will be performed in which the movement sequence, "0º Azimuth 0º Elevation 0º Azimuth 85º Elevation", will be performed 5 times. At the beginning of each iteration the prototype will be calibrated by manually positioning it in the reference position.

To test the functioning of the prototype, the 6 ways of functioning mentioned in section 5.5 (two manual and 4 automatic) have been tested and the functioning of the web capabilities have been checked. These were the results and conclusions of the tests:

- The algorithms based on potentiometers are 50% slower than those based on step tracks. The following table shows the time it takes for each algorithm to complete the first iteration of the run sequence.

	Potentiometer	Step track
Manual	181 s	124 s
Photoresistors	187 s	126 s
Sun prediction	181 s	123 s

Table 2: Operation times in seconds per algorithm

- Potentiometer-based algorithms have proven to be more robust than step tracker-based algorithms. The step tracker based algorithms have predicted their calibration after 3rd iteration contrary to the potentiometer based algorithms. This

may be due to slippage problems in the stepper due to reaching the maximum torque limit of the stepper.

- Step track based algorithms are more valuable than potentiometer based algorithms when starting from calibration. In the following table one can see the elevation degree data obtained from the precision test.

	Potentiometer	Step track
1º iteration	85º	85º
2º iteration	82º	85º
3º iteration	81º	85º
4º iteration	84º	85º
5º iteration	83º	85º

Table 3: Precision test results

- The connections to the main web page require an estimated time of 38 seconds to send, receive and send all the variables. 30 of these 38 seconds are spent in sending the analog values since they are sent one by one in different php requests, contrary to the boolean variables that are received by the Arduino all in one request.
- The sending of the data to the data collection spreadsheet, takes 10 seconds, from serial communication (Arduino Mega-NodeMCU) to web communication (NodeMCU-Google servers). Due to serial communication problems between the data sent from the Arduino Mega and the NodeMCU, 1 out of 6 data transmissions are made with erroneous data. This is due to random reading errors in the NodeMCU. The quality of this is measured based on the error rate BER (Bit Error Rate) which is obtained as the result of measuring the number of received erroneous bits between the total number of transmitted bits. For the data package that the Arduino Mega sends to the NodeMCU is 144bits, so it is estimated a VER rate of 1bit/864 bits. This is not acceptable, so it is proposed to incorporate error detection and correction protocols (such as v.42 or MNP in modems).
- The reception of data from the sun position spreadsheet takes all 9 seconds. This code execution contemplates reception of web data (Google-NodeMCU servers) up to serial communication (NodeMCU-Arduino Mega). No communication errors have been detected with respect to this spreadsheet.
- As it lacks an integrated MPPT it was opted to incorporate a variable resistor with ranges from 0 Ω to 10 Ω. This range has been obtained by the following formula:

$$Resistante = \frac{(Avarage\ voltage\ output\ of\ the\ solar\ panel)^2}{Rated\ power\ output} = \frac{5^2}{10} = 2{,}5\Omega$$

- It should be noted that the azimuth and elevation axis angles shown in the tests do not correspond to actual azimuth and elevation angles of the sun. The names of these axes have been given by taking a fixed reference to the sun when it is located in a southerly orientation (azimuth angle 0°). In this position the angles of the azimuth axis correspond to the degrees of the sun's azimuth axis, the same for the degrees of elevation. When the sun is placed in an east or west orientation the axes are reversed, so that the angles of the elevation axis correspond to the degrees of the sun's azimuth axis and vice versa. This makes the actual degrees of rotation of the design to be 360º azimuth and between 80º and 85º elevation rotation. As will be discussed in the limitations, the angle conversions could not be implemented in the final model.

CHAPTER 7: DISCUSSION

In this section, an objective analysis of the proposed design will be made and compared with the two-axis solar tracker designs in the market.

A cost analysis of the different designs described will be carried out. This analysis will be based on the simulation of the designs by means of different software. This will be the procedure to be followed in this cost analysis:

First, a study will be carried out with a Computational Fluid Dynamics software or CFD. From it, the external forces due to wind loads experienced by the structure can be extracted. These forces will be used to create a state of loads. These will be incorporated into a structural calculation software. This software will allow to calculate the stresses in the cables which will be used to calculate the number of supports that can be placed in parallel. In addition this software will be used to calculate the optimal structural profiles. In other words, this software will select the necessary tower sections for each design and will allow to check if there are differences between the different designs.

Once the simulations have been carried out, the different variables that affect the calculation of the manufacturing cost estimate and the relationship between them will be defined. The cost analysis will end with the determination of a manufacturing cost estimate for the proposed full-scale design.

7.1 COMPUTATIONAL FLUID DYNAMICS ANALYSIS

Numerical simulations play a crucial role in the analysis of many outdoor applications and, more specifically, in the safety measures of these applications. As one of the objectives is to perform a strength analysis of the structure, it is necessary to perform extensive and accurate simulations of the air flow around the solar panel.

ANSYS Fluent software has been selected as the CFD program to perform the multiple simulations. Choosing the right approach to simulate the flow is essential and is a compromise between high accuracy and low computational cost. The following section describes the complete process to obtain the results.

7.1.1 GEOMETRY AND BOUNDARY CONDITIONS

Before making the settings and choices in the CFD program, the environment in which the solar panel will operate has to be chosen.

To do this it will start by setting the simulation speed. For ensuring that the resistance of the structure the maximum air velocity value measured in Trondheim in the last 20 years will be taken as the simulation air velocity value. The weather station selected for data

collection is located in Voll Trondheim. These maximum wind gust values can be seen in Figure 52.

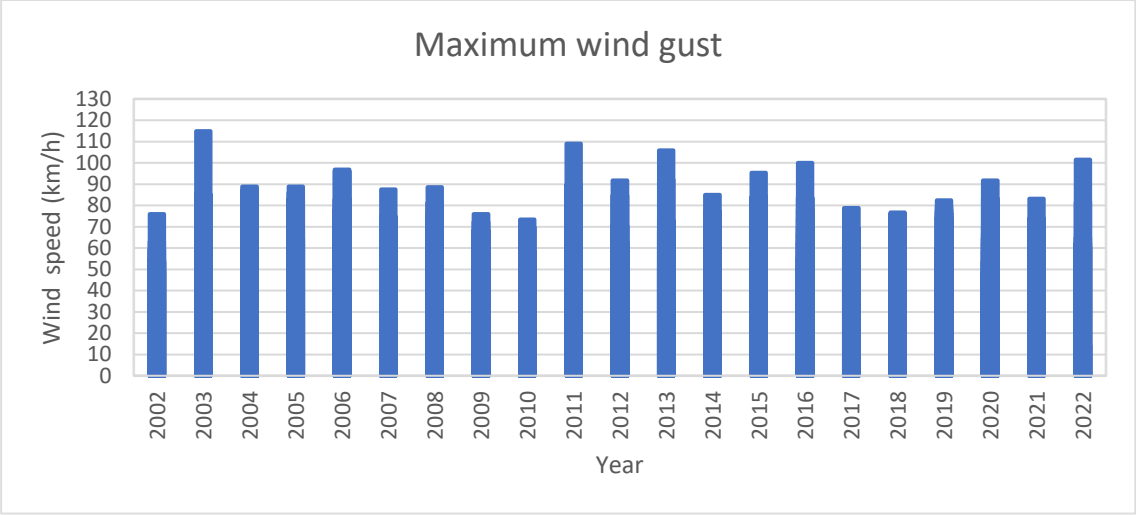


Figure 52: Historical of the maximum wind gust in Trondheim

The maximum wind speed recorded in Trondheim in the year 2021 has been 115 km/h .Taking a safety margin of 5%, a wind speed of 120 km/h was assumed for the simulations.

To simulate the environment around the structure, a limited volume can be made. This can be seen in Figure 53.

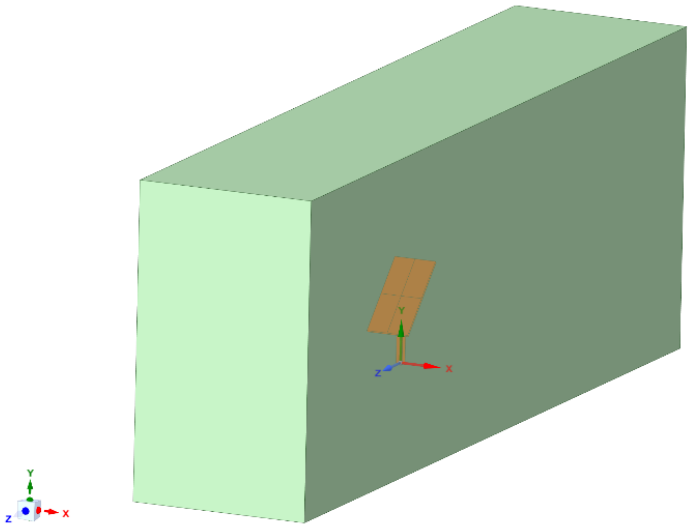


Figure 53: Geometry model of the Ansys simulation

7.1.2 CHARACTERISTICS OF THE MESH

The accuracy, convergence and speed of the CFD simulations are highly dependent on the mesh size and type. The mesh divides complex geometries into small elements used to compute local approximations of the total domain. Choosing the fine mesh, will result in higher accuracy, but leads to higher computational cost.

The CAD program is linked to the ANSYS Fluent meshing program[30]. This is possible by exporting the geometry to IGES files, and generating a collection of faces. As a result, the geometry must be repaired through the Spaceclaim software tool, which can now be fully utilized in Fluent.

In this simulation, it is decided to generate a mesh refinement zone around the panel that allows to obtain a finer mesh in the region of interest, while maintaining a reasonable computational cost. The overall mesh size is chosen as 0.1m while the refinement zone refines the mesh down to 0.05m, resulting in a total of 1.3 million mesh elements. A cross section of the final mesh can be seen in Figure 54. A mesh check is performed to ensure that the mesh is of sufficient quality.

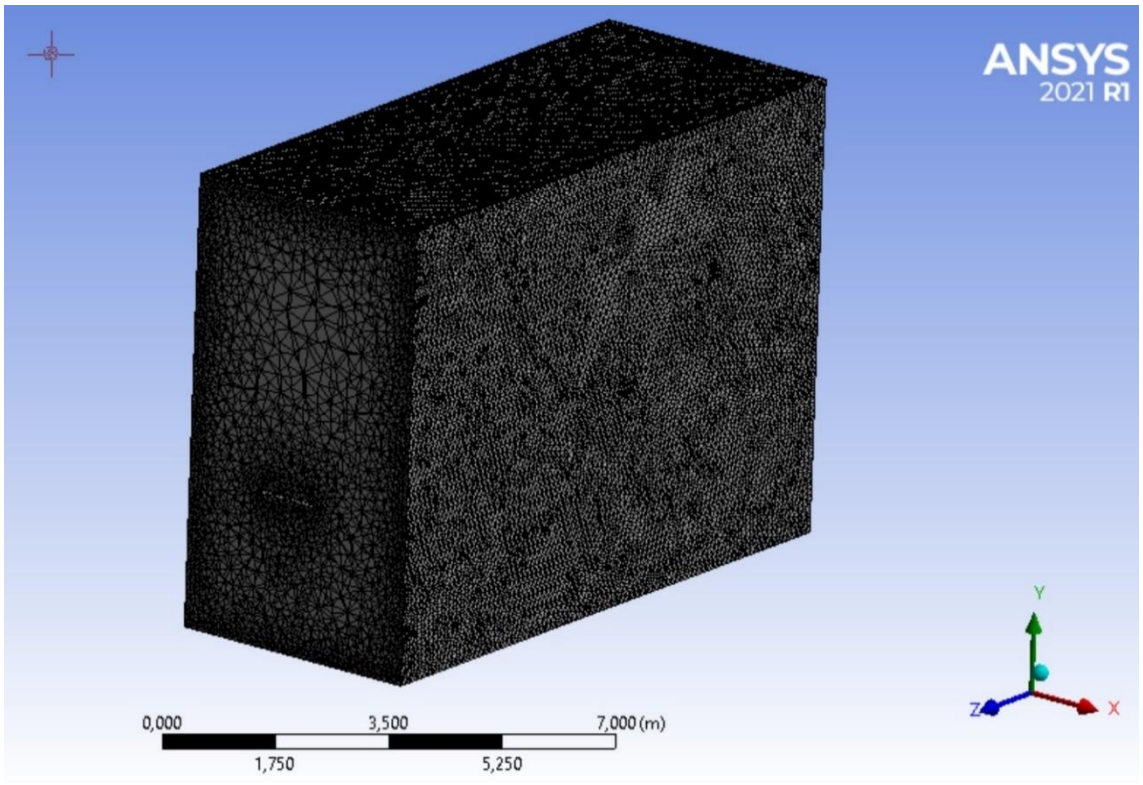


Figure 54: Ansys simulation mesh

7.1.3 FLUID CONFIGURATION

A stable solver was chosen. Since the main interest is in the effect of air on the plate, transient solvers are not necessary. With a stable solver, the calculation does not take time into account and solves until convergence is reached. For more detailed studies on the effect of the solar plate on the flow, a transient solver must be used to correctly represent the time-varying behavior of the turbulence.

In addition, a decision must be made between a pressure-based or density-based solver. Since the inlet velocity is kept below 120 km/h, the air could be treated as an incompressible fluid. Since pressure-based solvers show higher accuracy in simulations of incompressible flows, this option will be chosen. To converge the solution, different methods can be selected. The default method for coupling velocity with pressure is SIMPLE. Fluent also offers COUPLED. In cases where pressure and velocity are closely coupled in a scenario, as in rotating machines, then COUPLED works best. In our case, SIMPLE will be the appropriate numerical approach. For spatial discretization, the parameters are kept by default. Only the gradient method is changed to Least Squares

Cell-Based which has higher accuracy than Green-Gauss Cell-Based. It also has comparable accuracy to the Green-Gauss Node Based method for skewed and unstructured meshes, but lower computational cost. The Green-Grauss Cell-Based method could reduce the computational cost. However, in the end the computational cost was low enough that higher accuracy in this method was opt to used.

In addition, our goal is to compute the forces on the plate due to wind. For this purpose, multiple report definitions are performed. The faceplate is subdivided into four equal squares and the average force on the face is calculated over 300 iterations in the longitudinal direction. Finally, the simulation is initialized and run through 500 iterations and checked for convergence.

7.1.4 RESULTS

These simulations have been repeated for 3 different combinations of elevation inclination, 30°, 45° and 60°. In addition, two different wind directions have been tested. One of them, both directions parallel to the horizontal plane. The first one crosses the face where the panels are located and the other one is its opposite direction. In the following image one can see the graphical pressure representation of the results of the 45º of inclination.

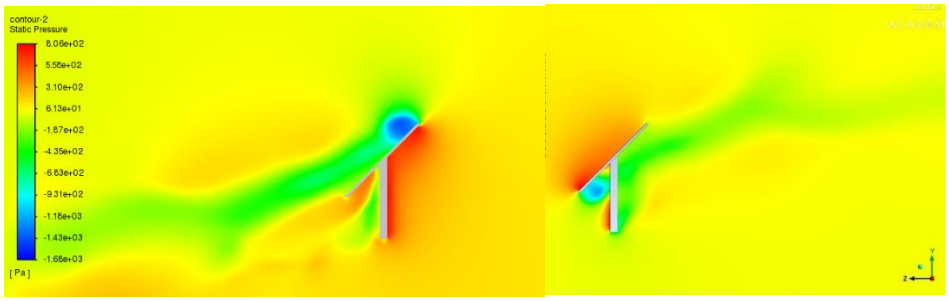


Figure 55: Ansys simulation pressure results

7.2 SAP2000

Once the data of the forces applied to the panel is obtained, the solar tracker designs are modeled in SAP2000. SAP2000 is a finite element program, with a 3D object-oriented graphical interface, prepared to perform, in a fully integrated way, the modeling, analysis and dimensioning of the widest range of structures[31]. Due to the similarities in the degree of rotation characteristics, the design based on Slew drivers and the design proposed in this thesis will be modeled.

These have been the considerations that have been applied in both modalities:

- Like the simulation in ANSYS Fluent, the surface has been divided into 4 quadrants. This has totaled a surface area of 20m^2. For the incorporation of the forces applied in each quadrant uniform surface loads have been assumed. In this way the applied force was divided by the surface area of each quadrant.
- The joint joining the structure to the floor was assumed to be a perfect embedment.
- The materials selected were a combination of aluminum for the frames supporting the panel and S355 steel for the main tower.
- The self-weights have been considered for the calculation of the loads.
- In addition to the wind loads, a uniform gravity load of 40 cm of snow was assumed. For this purpose, a snow density of 200kg/ m^3 has been assumed. This

would result in a uniform load of 0.8KN/m^2. So, it has been incorporated as in combination of loads together with the wind and self-weight loads. For this purpose, the linear sum of loads has been chosen.
- For the selection of frames, the European standard has been chosen.

These have been the considerations that have been applied in the modulization of the slew driver design:

- A perfect transmission of forces and moments in the slew driver has been assumed, so it has been modeled as a continuous element.
- The join joining the panel to the main tower has been implemented as a hinge that allows rotation in the Y-axis.
- The join joining the structure to the floor has been assumed as a perfect embedding.
- The join between the actuator and the panel has been implemented as a hinge that allows rotation in the Y-axis.

These have been the considerations that have been applied in the modulization of the design proposed in this thesis:

- The joins joining the panel to the main tower have been implemented as a swivel that allows rotation in the Y and X axis.
- The joins joining the cables to the ground have been implemented as ball joints allowing rotation in the Y and X axis.
- The joins connecting the cables to the panel are implemented as swivel joints allowing rotation in the Y- and X-axis.
- For the cable geometry it has been chosen to select a relative undeformed length of 1.01. This ensures that the cable fixes the position of the panel. This also means a maximum cable elongation of 1%. In this way the resulting axial in the cables would reflect the tension in the cable that the motor would have to apply to maintain the support tilted for the simulated load and angle conditions.
- The loads have been simulated as nonlinear, as this is necessary for the cables to behave as such and not as frames. In this way, the cables only work at traction. This simulation of non-linear loads means that the simulation has an iterative character until a convergent solution is found.

These models have been repeated for each of the three different angle combinations 30º, 45º and 60º with the six load combination cases. In the following image one can see themeshing of both designs for the 45º inclination models

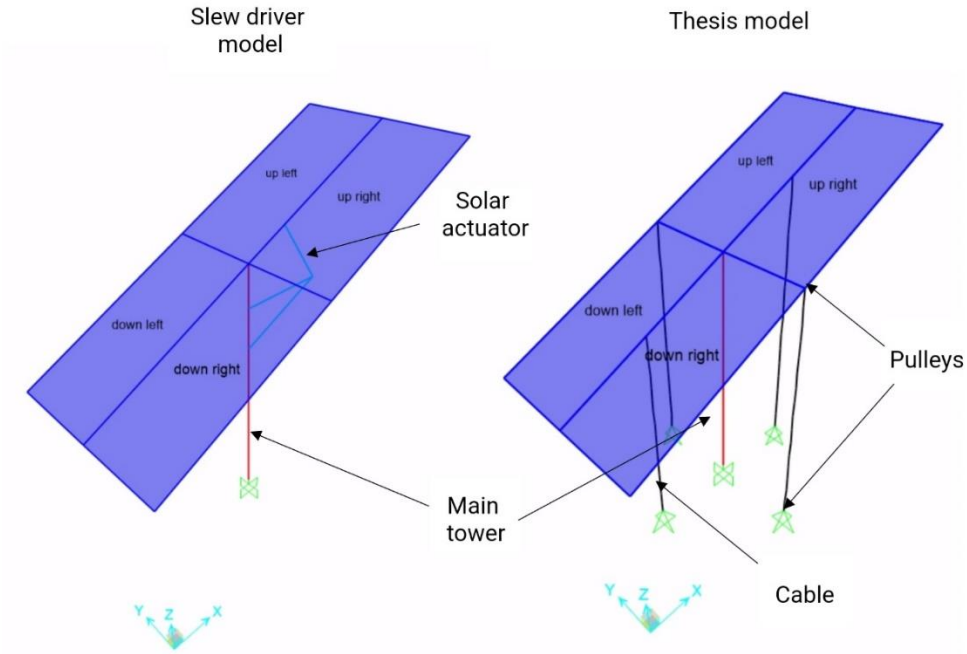


Figure 56: Sap2000 geometry simulation models

7.2.1 SIMULATION RESULTS

The results of the simulations allow us to observe the moment, deformation, and stress distributions of both models.

The following image shows the deformed shape of both models with a scale factor applied so that the deformations can be seen. Both images correspond to the cases of headwind and 45° tilt, for both designs.

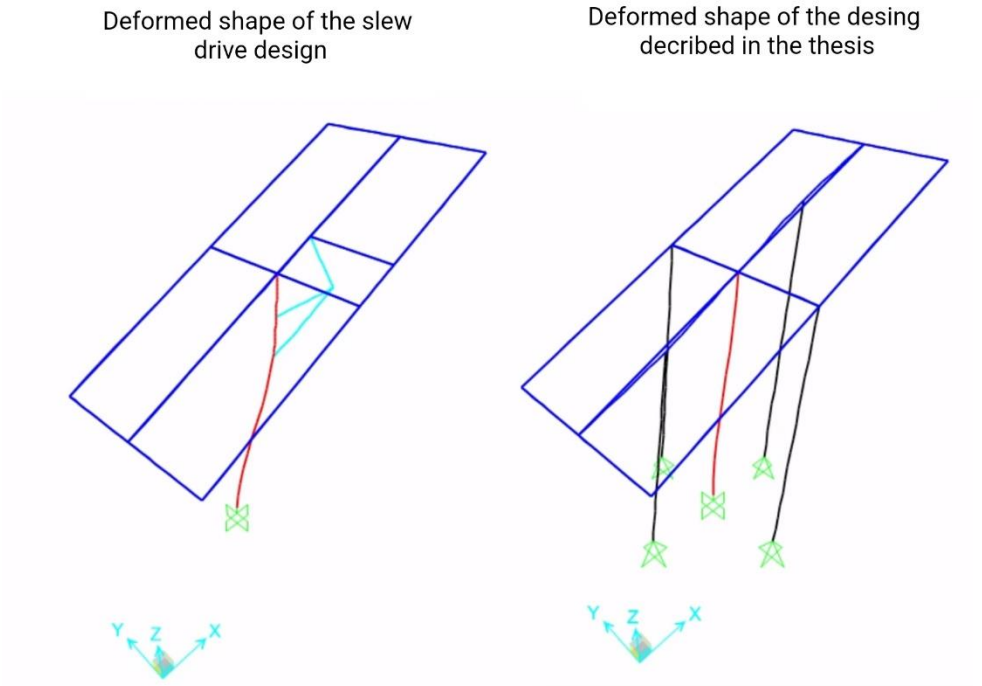


Figure 57: Deformed shape results of the structural models

In the following one can see the optimal sections selected within the European standard of square frames for the cases of headwind and 45° tilt for both designs.

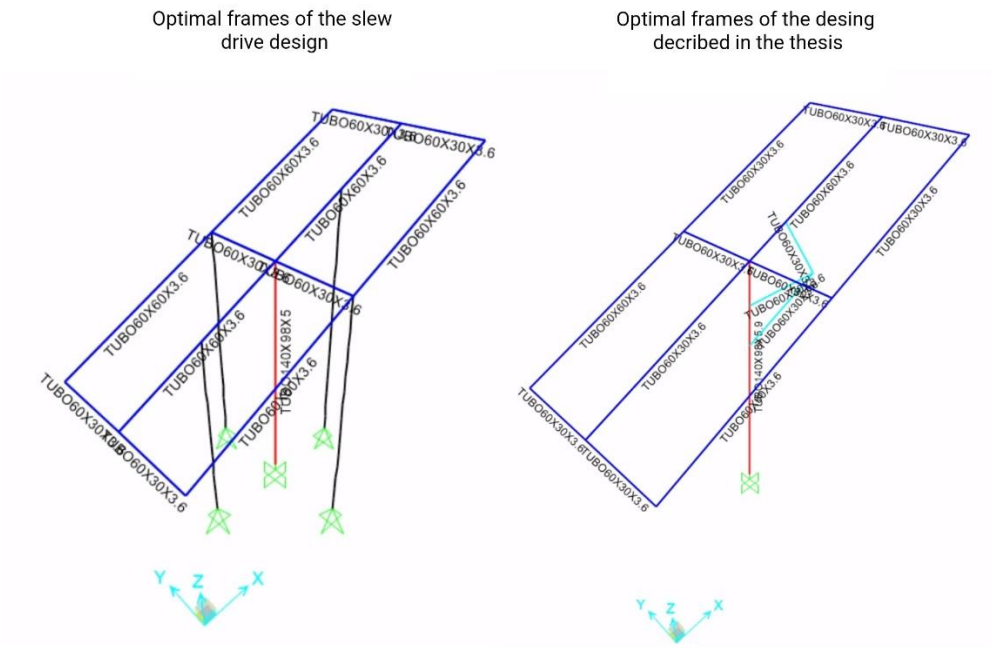


Figure 58: Optimal frame results of the structural models

The following image shows the axial force diagrams for the case of frontal wind and 45º degrees of inclination:

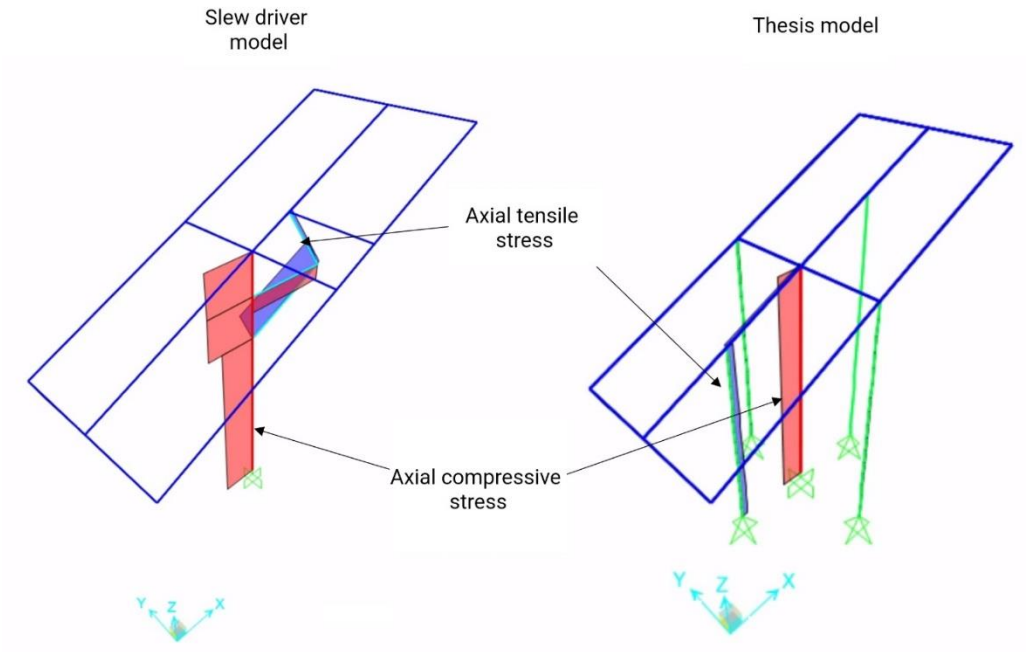


Figure 59: Axial force diagrams

The following table shows the maximum axial loads measured in the cables for the 6 types of load cases analyzed in the solar tracker model proposed in this thesis.

	Headwind Case	Backwind Case
60º	6.12 kN	6.33 kN
45º	6.25 kN	6.58 kN
30º	5.7 kN	6.01 kN

Table 4: Maximum cable forces results

Two main conclusions can be drawn from the simulations in Sap2000:

- There are no major differences in the calculated optimum cross-sections between the different calculated scenarios, so it can be inferred that the structural requirements are the same and therefore similar material costs can be assumed.
- The largest axial stress suffered in the cables was 6.58 kN, so this is the maximum force to be applied by a motor per support.

7.3 MANUFACTURING COST ANALYSIS

There are multiple variables that affect the final manufacturing costs. For this reason, it is necessary to analyze how these variables interact with each other, in order to select an optimal budget.

To do this it will start by determining the equation that will define the maximum number of supports that can be connected to an array. Secondly, the equation that determines the relationship between the number of solar trackers in an array and the amount of cable needed will be determined. Third, the relationship between the wind speed and the maximum axial load on the cables will be defined. In this way a maximum operating speed can be determined. Fourth, the manufacturing costs of a commercially available two-axis solar tracker will be presented. Part of these costs will be taken as variables for the determination of the manufacturing costs of the proposed design. Fifth, the equation relating all the variables will be determined in order to obtain a manufacturing cost estimate. Sixth, the variables will be set in order to obtain a final manufacturing cost estimate.

7.3.1 NUMBER OF SOLAR TRACKERS PER ARRAY

For the calculation of the number of solar trackers that can be moved per array, 3 conditions will be assumed:

- Model 52HT102-5003S will be taken as the stepper motor. This has the highest torque in the market at 50Nm.
- A reduction gearbox can be included. This will increase the torque by a factor that can vary between 1 and 100.
- The inside diameter of the drum will be taken as 10cm.

In addition, it must be considered that the Sap 2000 model does not consider the double sheave system present in the proposed design. These systems allow to double the applied force. Thus, the axial load obtained from the simulation on the cables must be divided by two to obtain the real tension in the cable.

Taking these considerations into account, these are the equations:

$$Maximum\ torque\ on\ drum = Maximum\ torque\ of\ the\ steppermotor \times Gear\ box\ reduction\ factor$$

$$Cable\ tension(N) = \frac{Axial\ force\ at\ the\ cable(N)}{2}$$

$$Maximum\ number\ of\ panels\ per\ array\ = \frac{Maximum\ torque\ on\ drum(Nm)}{Drum\ diameter(m) \times Cable\ tension(N)}$$

$$Maximum\ number\ of\ panels\ per\ array\ = \frac{50 \times Gear\ box\ factor}{0.01 \times Cable\ tension(N)} = \frac{5000 \times Gear\ box\ factor}{Cable\ tension(N)}$$

7.3.2 CALCULATION OF THE AMOUNT OF CABLE REQUIRED

Calculating the distance between supports is fundamental for the sizing of the required meters of cable.

The distance between supports is determined by the condition of not blocking each other's sunlight between solar supports. This condition is in turn determined by the elevation turning limit of the support and the dimensions of the support. The elevation turning limit of the prototype is 80, therefore, it can work optimally for solar elevation degrees between 10 and 90 degrees. The lower the degree of elevation of the sun the more shade to produce, so 10 degrees has been taken as the angle for the calculation of distances between panels. As for the dimensions of the supports, these will be the same as those implemented in the Sap2000 models mentioned above. The following image shows a representation of the dimensions involved in the calculations.

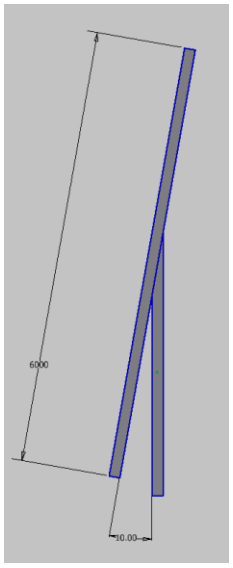


Figure 60: Profile dimensions of the full-scale model

Dimensions:

- Panel profile (PP)= 6 m
- Elevation angle (EA)= 10º.

$$Distance\ between\ supports = \frac{6 \times (\mathrm{Cos}(10))^2}{\sin(10)} + 6 \times \sin(10) = 34.55m$$

For the calculation of the required cable it will opt for an in-line model in which the motor system is positioned in the middle of the 14 supports. In this way there will be 7 supports at the rear and 7 at the front.

Necessary variables for the calculation of the required cable:

- Maximum distance between upper and lower pulleys (PD)=4m

- Distance between supports (SD)=34.55m
- Number of supports at the rear (RS)
- Number of supports at the front (FS)
- Number of motors (NM)=4

$$\frac{Total\ number\ of\ supports\ (n)}{2} = RS = FS$$

$$Total\ cable\ required\ (TCR) = NM \times (\sum_{i=0}^{RS}(n \times SD + 2 \times PD) + \sum_{i=0}^{FS}(n \times SD + 2 \times PD))$$

$$Total\ cable\ required(TCR) = 2 \times NM \times \sum_{i=0}^{\frac{n}{2}}(n \times SD + 2 \times PD)$$

$$Total\ cable\ required(TCR) = 2 \times 4 \times \sum_{i=0}^{\frac{n}{2}}(n \times 34.55 + 2 \times 4)$$

7.3.3 MANUFACTURING COST OF THE MARKET MODEL ST2408PH

To perform the cost comparison, the manufacturing cost data of the solar tracker model ST2408PH, from Suntrack Nordic AS, will be used[11]. This is a two-axis solar tracker. It supports a maximum solar panel area of $20m^2$, like the simulated models. This model is an example of a design based on Slew drivers. It supports 300 degrees of azimuth rotation and 70 degrees of elevation. The specification sheet for this model can be found on p.109 for more information. The following table shows the manufacturing costs for each ST2408PH:

Number	Resource description	Quantity	Unit	Unit cost	Total cost
1	Control system	1	Ud.	15.000 kr	15.000 kr
2	Actuators (Circular and linear)	1	Ud.	20.000 kr	20.000 kr
3	Steel structure	1	Ud.	25.000 kr	25.000 kr
4	Seafreight to Norway	1	Ud.	33.000 kr	33.000 kr
					93.000 kr

Table 5: Manufacturing costs of the two-axis solar tracker model ST2408PH

BUDGET EQUATION

For the generation of a construction budget for the design proposed in this thesis, the following considerations have been taken:

- For the specifications, a 4 mm diameter cable with a breaking carcass of 16 kN (more than 4 times the maximum tension in the cable) has been dimensioned.
- For the poles, double pole models with maximum loads of 10 kN have been selected.
- The same unit costs have been taken for the control system. This is because both designs can be controlled by very similar systems. The main difference is that in the design of this thesis only one support is controlled and the others function as mirror supports, requiring one control system for every n supports. In the event that one of the supports was not aligned correctly, this could be detected by individually monitoring the energy production of each support.

- The same unit costs have been taken for the steel structure. This is mainly because the optimal structural cross-sections calculated in Sap2000 are the same.
- The same unit costs have been taken for the seafreight to Norway.

These are the variables that affect the budget:

- Control system (CS): 15000kr per array.
- Three phase Nema 52 stepper (NS): 20000kr per stepper,4 per array.
- 4mm braided steel cable (SC): 3.7kr per meter.
- Gearbox reducer (GR): 4000kr per gearbox, 4 per array.
- 10 cm lathed drum(D): 4000kr per drum, 4 per array.
- Double pulley 50 mm capacity 10000N (DP): 312.2kr per pulley, 8 pulleys per solar tracker.
- Steel structure (SS): 25000 kr per structure, one steel structure per solar tracker.
- Seafreight to Norway (SN): 33000 kr per Seafreight, one per solar tracker.
- Number of solar trackers per array (n)
- Total cable required (TCR)
- Budget for the proposed design (BPD)

$$BPD = CS + NS \times 4 + SC \times TCR + GR \times 4 + D \times 4 + DP \times 8 \times n + SS \times n + SN \times n$$

$$BPD(n) = 15000 + (20000 + 4000 + 4000) \times 4 + 3.7 \times 2 \times 4 \times \sum_{i=0}^{\frac{n}{2}} (n \times 34.55 + 2 \times 4) + (312.2 \times 8 + 25000 + 33000)$$

7.3.4 BUDGET COMPARATION

Since the manufacturing budget of the proposed design does not vary linearly with the number of solar trackers (n), it will be compared with its market counterpart with respect to n.

As the market model analyzed incorporates everything necessary for its operation, it can be assumed that the manufacturing cost will vary linearly with the number of solar tracks. This is the following equation represents the manufacturing budget of the ST2408PH (BMM) model versus n:

$$BMM = 93000 \times n$$

The following graph shows the evolution of the manufacturing costs of both models as a function of the number of solar trackers.

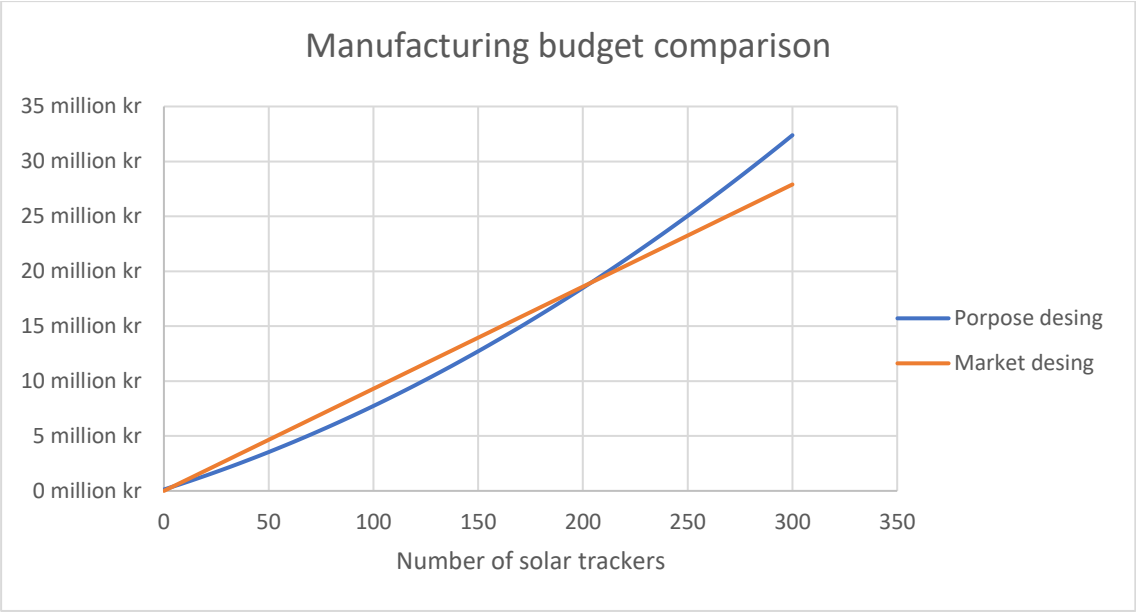


Figure 61: Variation of the manufacturing costs with the number of solar trackers

If a relative comparison is made between the manufacturing costs of the proposed model and the costs of its market counterpart, the improvements in cost reduction can be better observed.

The following graph shows this relative improvement in manufacturing costs.

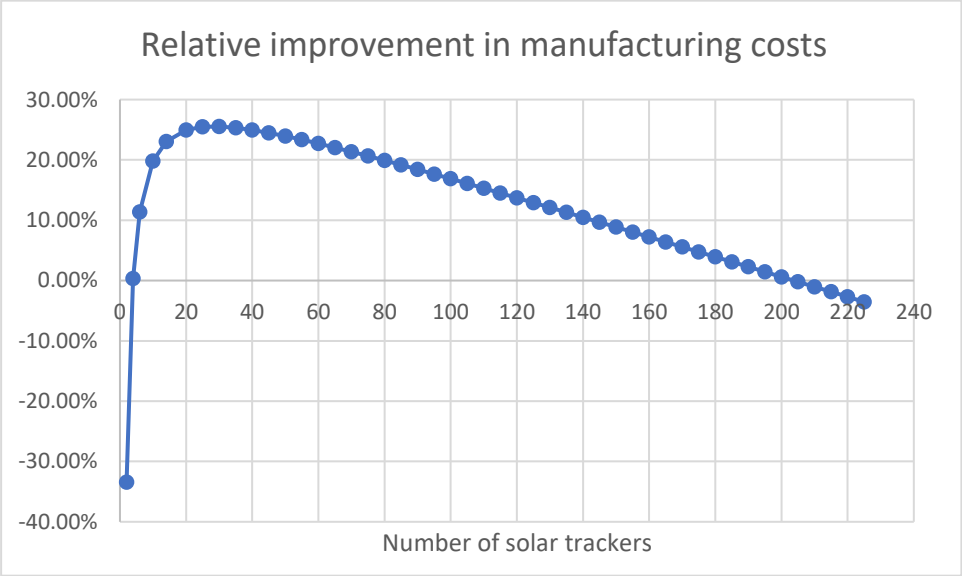


Figure 62: Variation of the relative improvement in manufacturing cost with the number of solar trackers

Two main conclusions can be drawn from this relative comparison:

- The proposed model is cheaper to manufacture for arrays ranging from 4 to 212 solar trackers in series.
- The relative cost improvement has a maximum of 25.55% when reaching 30 trackers per array.

7.4 MANUFACTURING BUDGET PROPOSAL

For the realization of a cost estimate, the number of solar trackers per array (n) must be determined. This is determined by the balance between the forces that can be exerted by the driving system and the forces experienced by the support when exposed to a given load case. For this reason, in this section it will be analyzed how the characteristics of the drive system and the determination of the limiting operating loads affect n, and therefore the manufacturing costs. In $BPD(n)$ formula it can be seen that the determination of the maximum number of panels per array depends on the gear factor and the cable tension.

This analysis will define the relationship between wind speed and cable tension, an operating speed and the relationship between the gearbox reduction factor and the number of trackers.

7.4.1 EFFECT OF WIND SPEED ON THE LOADS

The wind speed directly affects the wind load applied on the panels. This in turn directly affects the maximum simulated axle in the cables, and therefore the cable tension. And as already shown, this tension affects the number of panels that can be connected per array. Finding the relationship between these variables would allow us to know how the determination of a wind speed limit affects the final manufacturing cost. Above this limit the panels would be positioned in a more aerodynamic and therefore safer position. This would involve running multiple simulations in Ansys Fluent for each wind speed. Then their results would have to be implemented in Sap2000, in order to obtain the tensions in the cables. Due to the high computational cost of running the simulations in Ansys, it has chosen to assume that the forces experienced by the panels are as follows with respect to the wind speed:

$$F_D = \frac{1}{2}\rho A C_D v^2$$

F_D represents the magnitude of the c, which is proportional to the square of the wind speed (for static objects). C_D is the drag coefficient, this depends on the geometry o the object. *A* is the projected area on the wind direction plane. *ρ* is the density of the fluid. The variables C_D, A and *ρ* will be considered as constant. In order to keep the same force distribution as in the simulated one for a wind speed of 120 km/h, 4 different drag coefficients will be calculated (one for each quadrant).

$$C_{Dq} = \frac{F_{max,q}}{\frac{1}{2}\rho A_Q {v_{max}}^2}$$

$$\mathrm{A}_q = \frac{total\ surface\ area}{numbre\ of\ quadrants} \times \mathrm{Cos(Elevation\ angle)} = \frac{20\mathrm{m}^2}{4} \times Cos(45º)$$

$$F_{DQn} = \frac{1}{2}\rho A_q C_{Dq} {v_n}^2$$

Once the force values for each quadrant and each speed have been obtained, these forces must be translated into uniform loads so that they can be added in Sap2000. In

this way the tensions in the cable can be obtained. The following graph shows the relationship between wind speed and maximum cable tension.

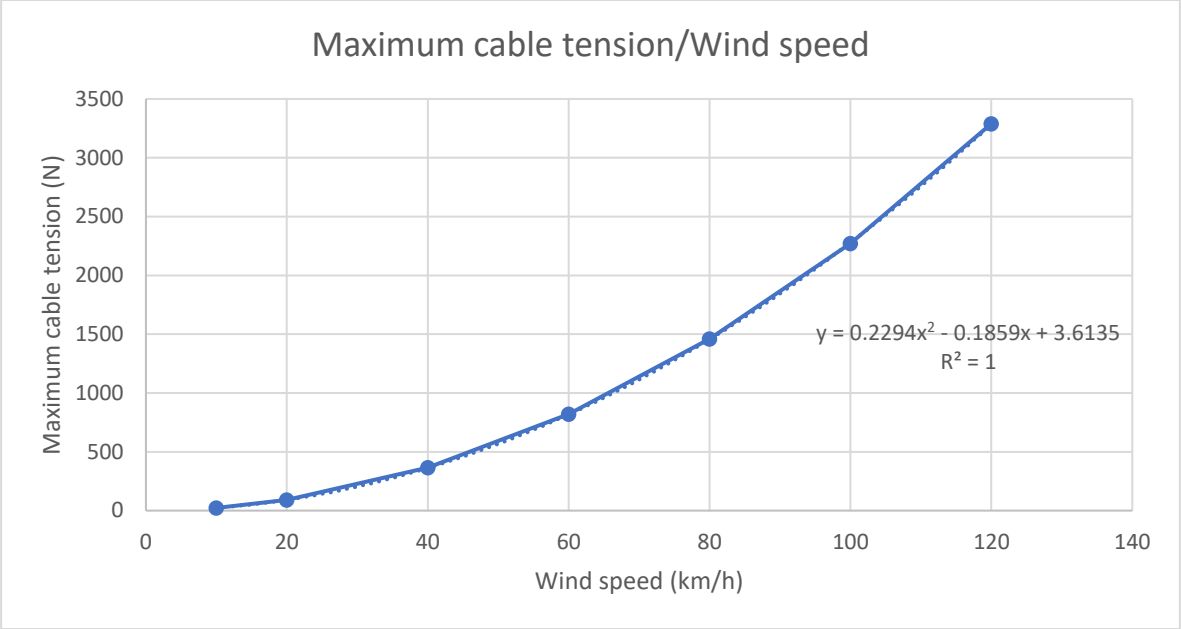


Figure 63: Effect of the wind speed in the maximum cable tension

7.4.2 DETERMINATION OF THE SPEED LIMIT FOR OPERATION

The maximum mean wind speed (24h) data for the last 20 years have been collected to select a limiting operating speed. In other words, these data would indicate the maximum mean sustained wind speeds that have been measured in a day, and not the wind gust data (tgust<<<tmaximun mean). The distribution of these data was then analyzed by grouping the data into ranges with wind speed increments of 2.5 km/h. This distribution can be seen in the Figure 64. The left axis shows the probability that the maximum mean wind speed of any given day will take a value within the range. The right axis shows the probability that the solar tracker has to stop at least once a day. The weather station selected for data collection is in Voll Trondheim.

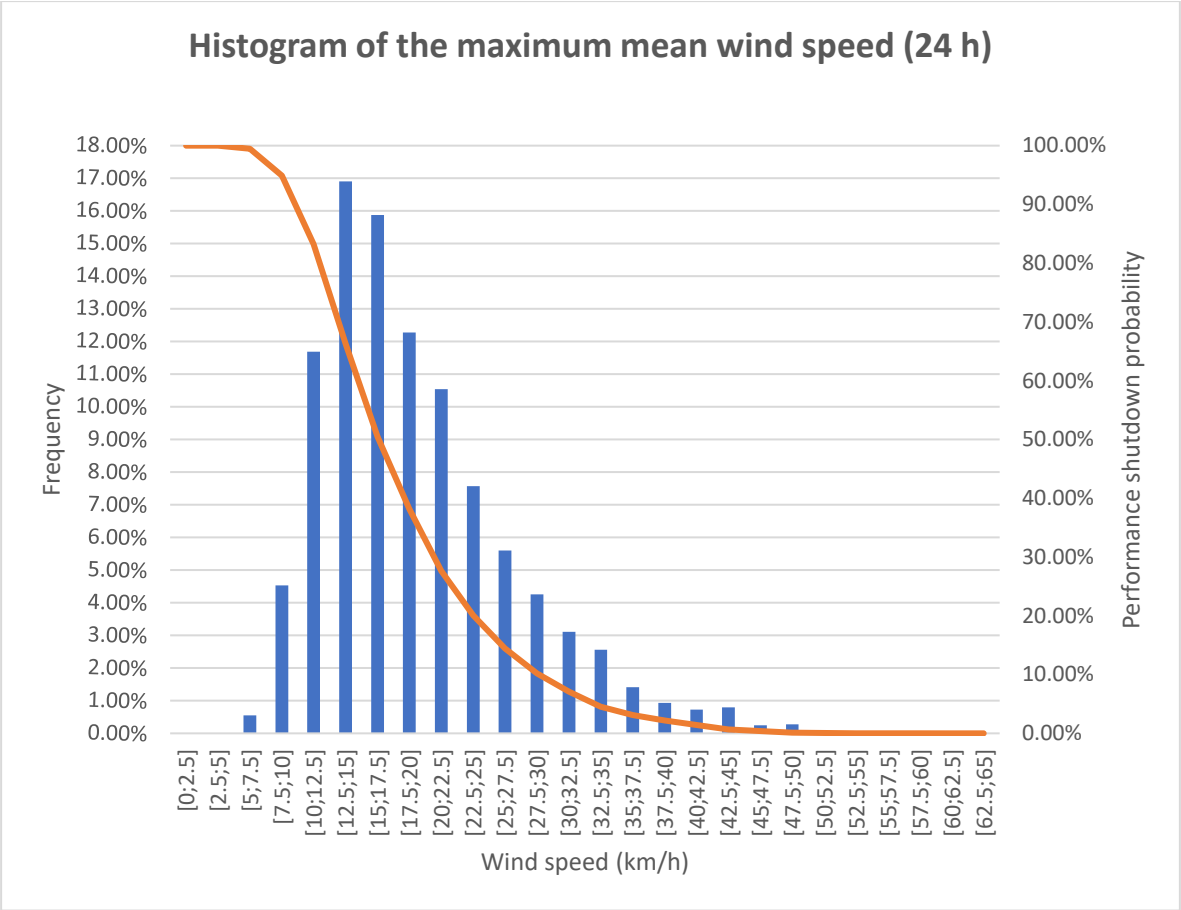


Figure 64: Histogram of the maximum mean wind speed (24 h) of Trondheim

For the determination of the final budget, the operating speed limit will be set at 30km/h. With this speed, there is a 92% probability of uninterrupted daily operation. In addition, this operating speed is similar to the operating speed limit of the ST2408PH model, which makes the cost comparison fairer.

7.4.3 SELECTION OF GEARBOX REDUCTION FACTOR

Taking the results of the last two sections, it can be deduced that for + a wind speed of 30 km/h, the tension in the cables is 209.88N. Setting this tension in the cables as the limiting operating tension, it is possible to infer a straight relation between the gear factor and the number of trackers in an array.

$$Maximum\ number\ of\ panels\ per\ array\ = \frac{50 \times Gear\ box\ factor}{0.01 \times Cable\ tension(N)}$$

$$Maximum\ number\ of\ panels\ per\ array\ = \frac{5000 \times Gear\ box\ factor}{209.88}$$

$$= 2.392 \times Gear\ box\ factor$$

With this equation the Figure 65 has been obtained. It shows the relationship between the gear box factor, the number of solar trackers per array and the wind speed.

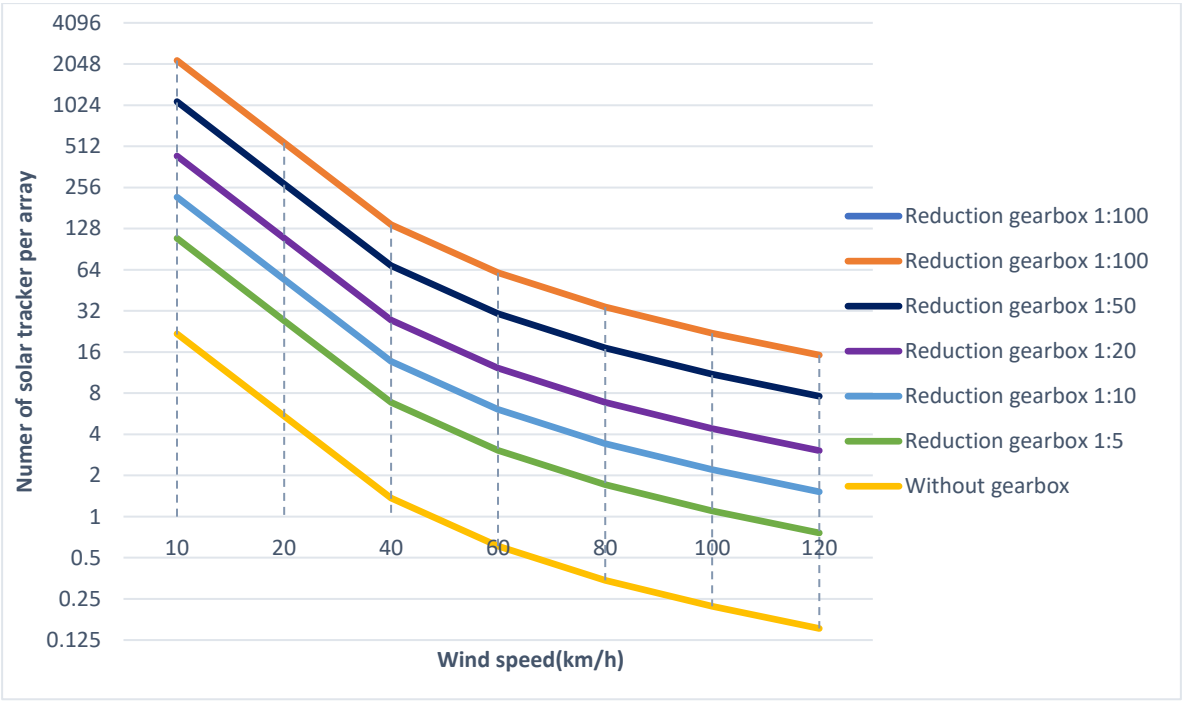


Figure 65: Effect of the gearbox reduction factor in the number of solar trackers per array

For the determination of the gear box reduction factor, two aspects will be taken into account

- The price of the gear box increases with the gear box factor. This is because the higher the reduction factor, the greater the complexity and the greater the forces experienced by the internal gears.
- The graph of sown in Figure 62 will be taken into account, which reflects the relative improvement in manufacturing costs with respect to the number of followers in an array.

For an operating wind speed limit of 30 km/h and considering the two factors mentioned above, the gearbox factor has been set at 1:5. Finally, a total of 16 followers per array were obtained.

7.4.4 PROPOSED MANUFACTURING BUDGET

Number	Resource description	Quantity	Unit	Unit cost	Total cost
1	Control system	1	Ud.	15.000 kr	15.000 kr
2	Nema 52 3-phase Stepper Motor Bipolar 50 Nm	4	Ud.	20.000 kr	80.000 kr
3	4mm steel braided cable	41872	m	3,7 kr	154.926.4 kr
4	Gearbox reducer 1:5	4	Ud.	4.000 kr	16.000 kr
5	10 cm lathed drum	4	Ud.	4.000 kr	16.000 kr
6	Double pulley 3” capacity 1000Kg	128	Ud.	312,2 kr	39.961.6 kr
7	Steel structure	16	Ud.	25.000 kr	400.000 kr
8	Seafreight to Norway	16	Ud.	33.000 kr	528.000 kr
					1.249.888 kr

Table 6: Proposed manufacturing budget

Following the line of reasoning described above, the proposed budget would be 23.5% cheaper than its market counterpart.

CHAPTER 8: LIMITATIONS

8.1 LIMITATIONS IN THE DEVELOPMENT OF THE PROTOTYPE

The limitations that have affected the development of the prototype can be seen reflected in the following factors:

- Only one 3D printer was available for the development of a small-scale model. This lengthened the iterative process described in the section SOLAR STAND STRUCTURE4.1.
- There have been weeks of delays in the arrival of purchased parts, especially the force sensitive resistors.
- It was not possible to incorporate the multiplexer in the final testing of the prototype. The purpose of the multiplexer was to increase the analog inputs of the Arduino Mega. For this reason, it was not possible to incorporate all the combinations of the motion algorithms with a single code.
- It has not been possible to build a scale prototype due to problems with NTNU's manufacturing laboratory being overloaded with other projects. Because of this, it has not been possible to demonstrate the effectiveness of a full-scale prototype outdoors.

8.2 IMPROVEMENT PROPOSALS

The following improvements have been proposed for the proposed design.

- Incorporation of a wind sensor: this sensor would allow the behavior of the support in different wind speeds. The minimum tension of the cables could be modified in cases of low wind speeds, thus reducing the consumption of the servomotor. It would also allow to set a wind speed from which the support could be positioned in a safe state.
- Design of a storage system that allows to feed back the energy produced to move the servomotors for isolated photovoltaic plants.
- Design to implement a positioning check system that uses the positioning algorithm that is not in use to check the other one, and in case of discrepancy implement a warning system.
- Implement a visual web system based on graphs that allows one to check the performance of the solar tracker. The Google spreadsheet created to collect the data could be used as a database.
- Implement improvements in the electrical connections of the sensors in order to test the outdoor design.
- Implement a code that transforms the elevation and azimuth angles of the prototype axes into real values of angular positions of the sun. For this it is proposed to use the same scheme used in the development of the Matlab simulation.

8.3 ASSUMPTIONS IN THE DEVELOPMENT OF THE DISCUSSION

The discussion described in this thesis should be taken as a preliminary study and not as a complete study. This is due, in part, to the various assumptions that have been made

in performing the analyses. These assumptions can change the balance sheet costs. These are some these assumptions:

- The fluid and structural analyses have been based on simplified models and only 3 different panel positions have been considered. To increase the reliability of these analyses, more combinations of both wind positions and wind directions should be simulated. The reason for the limited number of simulated combinations was the high computational demand of the fluid simulations. Each simulation required an average of 50 hours of computation. This added to the fact that these simulations have been performed in the last stage of the development of the thesis and it has been decided to limit the number of different simulations to 6. This could presumably increase the costs of the proposed design if a more critical cable stress case is encountered.
- Structural analyses do not include dynamic load studies. For this reason, the fatigue resistance of the calculated optimum sections cannot be assured.
- In the budgeting of the design proposed in this thesis, costs associated with sensors have not been directly detailed. These have been assumed to be within the control system costs. This could presumably modify the costs of the proposed design, since both designs make use of different types of sensors.
- The comparison has been carried out with only one actual model of solar tracker on the market. This in a very small sample size. From a statistical point of view this does not give a high reliability in the extrapolation of the comparison to the total market designs.
- Although the performance of the prototype has been demonstrated on a small scale, it has not been tested on a full scale. In the process of full-scale construction of the design, various problems may be discovered that require the incorporation of new devices, which would increase the cost.

Future lines of research should be conducted to demonstrate the superiority of this model.

CHAPTER 9: CONCLUSION

The centralization of motion and control systems has proven to be a viable concept. This has been demonstrated by building and testing the performance of a small-scale model. In addition, it has been possible to successfully establish a web platform for the control and monitoring of the prototype. The prototype also meets the turning degree targets because it has 360º of azimuth turn and 80º of elevation turn. From the testing and prototype design, the following findings should be highlighted:

- The premise of being able to use two motors due to tension conservation has been proven incorrect. In contrast, a drive system design based on 4 different motors has been chosen.
- The relative positioning of the pulleys and shafts directly affects the degree of rotation of the tracker.
- The performance of the algorithm based on photoresistors has been tested.
- The performance of the algorithm based on the online calculation of the position of the sun based on geographical position data has been proven.
- The performance of the algorithm based on the use of potentiometers has been proven.
- The performance of the position algorithm based on motor step tracking has been proven.
- Position algorithms based on potentiometers are slower and less accurate than position algorithms based on step tracking
- Position algorithms based on potentiometers are more robust than those based on step tracking.

From the dimensioning of the full-scale model the following findings can be highlighted:

- Of the six scenarios simulated in the discussion, the most critical one turned out to be the case of back wind and a 45° inclination.
- For a simulation speed of 120km/h, maximum cable stresses of 3.29kN were measured.
- The structural section requirements of the presented model and the ST2408PH model are identical.
- The number of panels in series that can be arranged in an array varies quadratically with the limiting operating wind speed.

Regarding the described conditions the presented design reduces the manufacturing costs with respect to its market counterparts. This is based on two findings:

- The manufacturing cost reduction depends on the number of trackers per array. This has a maximum peak relative cost reduction of 25.5% when the number of trackers is 30. For manufacturing budget of the proposed full-scale model reduces by 23.5% its relative manufacturing costs with respect to ST2408PH model
- The number of solar trackers in series should be between 5 and 212 in order to have lower manufacturing cost values than the solar tracker model ST2408PH.

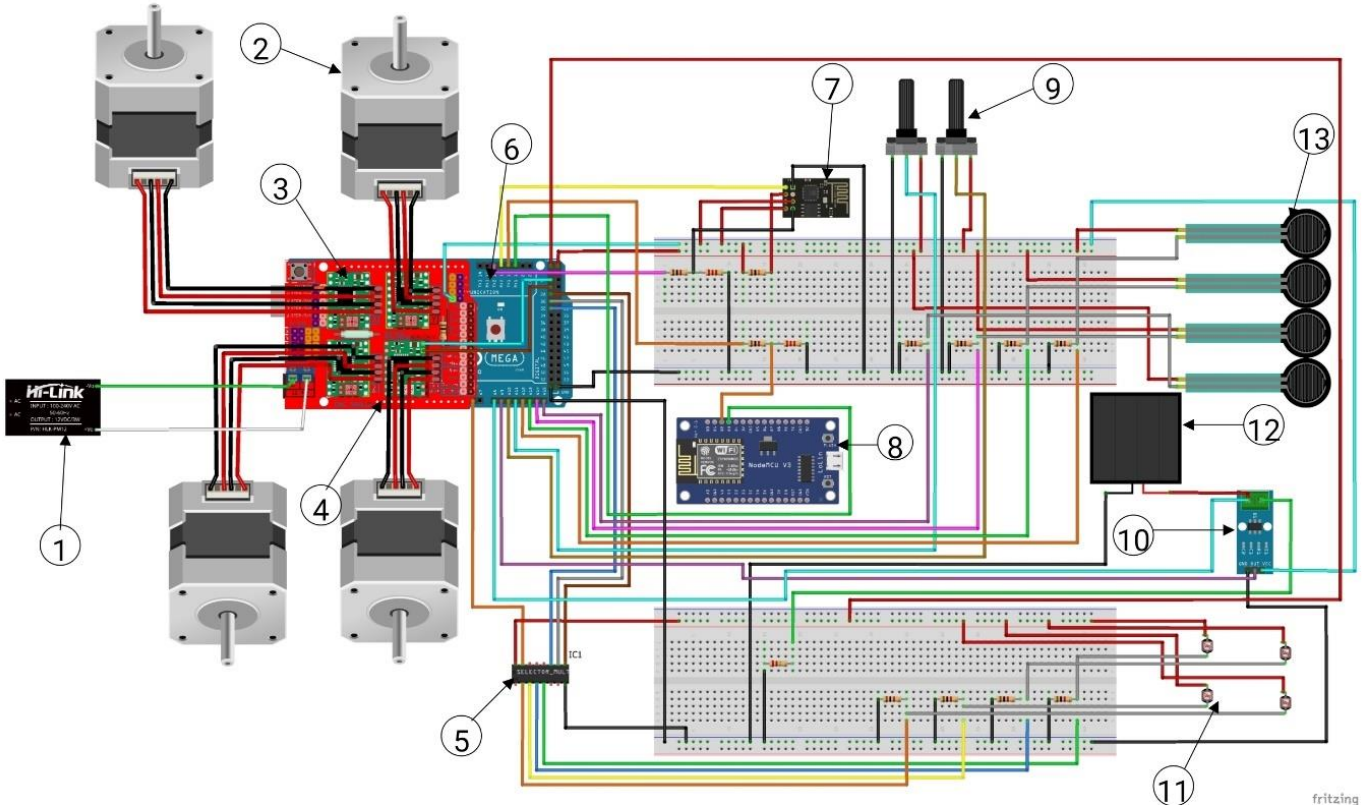
For future lines of research, improvements are proposed in section 8.2.

CHAPTER 10: BIBLIOGRAPHY

1. Babota, F., *INCREASE ENERGY EFFICIENCY AND COMFORT IN HOMES BY INCORPORATING PASSIVE SOLAR DESIGN FEATURES.* Bulletin of the Polytechnic Institute of Jassy, CONSTRUCTIONS. ARCHITECTURE Section, 2014. **60**: p. 175-186.
2. Schroeder, D.V., *The Sun and the Seasons*. 2011, Weber State University: https://physics.weber.edu/.
3. Patil, K., et al., *Dual Axis Solar Tracker with Cleaner.* International Journal for Research in Applied Science and Engineering Technology, 2022. **10**: p. 1253-1256.
4. Position, S.; Available from: https://www.sunearthtools.com/.
5. huayue, *HYS-12PV-78-LSD Solar Tracking Solar Controller Dual Axis Solar Pv Tracker Sun Tracking*. 2016.
6. saVRee, *3D Technical Animation - Dual Axis Solar Tracking System*. 2019.
7. Lewandoski, C., R. Ferreira Santos, and A. Ikpehai, *THE ADVANTAGES OF SOLAR TRACKER.* International Journal of Environmental Resilience Research and Science, 2021. **3**.
8. Council, C.E., *Solar Tracking in Australia*.
9. Hashim, I., A. Ismail, and M. Azizi, *Solar Tracker.* International Journal of Recent Technology and Applied Science, 2020. **2**: p. 59-65.
10. Soni, V. and N. Singh, *Solar Energy Pricing*. 2021. p. 217-229.
11. Løvmo, E., *Manufacturing costs of the two-axis solar tracker model ST2408PH*. 2022: Suntrack Nordic AS.
12. Pang, X., et al., *Polylactic acid (PLA): Research, development and industrialization.* Biotechnology journal, 2010. **5**: p. 1125-36.
13. Beriber, D. and A. Talha, *MPPT Techniques for PV Systems*. 2013.
14. *Types of Electric Motors – Classification of AC, DC & Special Motors*. Electrical Technology.
15. Banzi, M., *Getting Started with Arduino: The Open Source Electronics Prototyping Platform* 2015: Make Community, LLC;.
16. Stroustrup, B., *The C++ Programming Language*. 2013: Addison-Wesley Professional;.
17. Teja, R., *Arduino Mega Arduino Mega Pinout | Arduino Mega 2560 Layout, Specifications*. 2021: Electronics Hub.
18. Fahad, E., *Arduino CNC Shield V3.0 and A4988 Hybrid Stepper Motor Driver + Joystick*. 2020.
19. *A4988 Stepper Motor Driver Carrier*.
20. Koyanagi, F., *Arduino MEGA 2560 With WiFi Built-in - ESP8266*. 2015: Instructables.
21. *MATLAB Programming Fundamentals for R2022a*. 2022: Mathworks.
22. Goupillière, *Pulleys*. Lego maquinas alm jm.
23. Automotive, J., *SCHERDEL: constant force and power springs*.
24. *MAKE NTNU*. 2022.
25. Sealey, W., *The Audio Noise of Transformers.* American Institute of Electrical Engineers, Transactions of the, 1941: p. 109-112.
26. Forbes, A., *The Joy of PHP: A Beginner's Guide to Programming Interactive Web Applications with PHP and mySQL*. 2012: BeakCheck LLC;.
27. Michalsky, J., *The Astronomical Almanac's algorithm for approximate solar position (1950–2050).* Solar Energy, 1988. **40**: p. 227-235.
28. Portnoy, A., *Learning Google-app-script*. 2018, riptutorial.com: Strack Overflow contibutors.
29. Commanderfranz, *Fritzing - a Tutorial*. 2017, instructables.
30. *ANSYS Fluent Tutorial Guide 2021 R1*. 2021: Ansys Europe.
31. *Linear and Nonlinear Static and Dynamic Analysis and Design of Three-Dimensional Structures* Vol. Introductory Tutorial for SAP2000. 2011: Computers and Structures, Inc.

CHAPTER 11: APPENDIX

11.1 PROTOTYPE PARTS



List of components:

1. LCW50US12 - Switched-Mode Power Supply
2. Stepper motor 17HS15-1504S
3. A4988 stepper motor driver
4. CNC Shield V3.0
5. 2717 - I2C Multiplexer
6. Arduino Mega 2560
7. ESP01 Wi-Fi module
8. NodeMCU V3.0 module
9. Potentiometer 3852A-282-103AL
10. Current sensor ACS70331
11. LDR
12. Solar panel PRT-16835
13. Force Sensitive Resistor SEN-09375

11.1.1 1. LCW50US12 - SWITCHED-MODE POWER SUPPLY

LCW50 Series

AC-DC POWER SUPPLIES

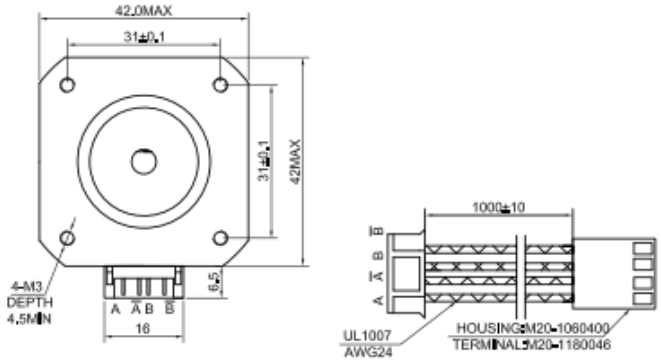
Input

Characteristic	Minimum	Typical	Maximum	Units	Notes & Conditions
Input Voltage - Operating	85	115/230	305	VAC	Derate output power linearly from 100% at 100VAC to 80% at 85VAC and from 100% at 277VAC to 80% at 305VAC
	120		430	VDC	Alternative input. Not to be used in addition to AC input. DC input not included in safety approvals, external DC rated fuse required. Derate output power linearly from 100% at 120VDC to 80% at 100VDC and from 100% at 390VDC to 80% at 430VDC
Input Frequency	47	50/60	63	Hz	
Input Current - Full Load			1.2	A	115VAC
			0.8		230VAC
No Load Input Power			0.5	W	
Inrush Current		30		A	115VAC cold start at 25°C ambient
		60			230VAC cold start at 25°C ambient
Earth Leakage Current			0.75	mA	277VAC/50Hz (Typ)
Input Protection	T3.15A/300VAC Internal fuse fitted in line				

Output

Characteristic	Minimum	Typical	Maximum	Units	Notes & Conditions	
Output Voltage	4.5		52.8	VDC	See Models & Ratings table	
		±2		%	Full load	LCW50US05
		±1				All other models
Voltage Adjustment		±10		%		
Minimum Load	0			A	No minimum load required	
Start Up Delay	58		130	ms	115VAC full load	
	60		138		230VAC full load	
Hold Up Time		8		ms	115VAC	
		30			230VAC	
Drift			±0.03	%	After 20 minutes warm up, 230VAC, 0°C to 50°C	
Line Regulation		±0.5		%	100-264VAC, full load	
Load Regulation			±1.0	%	0-100% load	LCW50US05
			±0.5			All other models
Transient Response			10	%	Recovery within 1% in less than 5ms for a 50-75% and 75-50% load step	
Ripple & Noise				mV pk-pk	See Models & Ratings table	
Over/Undershoot			10	%	Full load 5ms recovery	
Overvoltage Protection			6.3	%	LCW50US05	Hiccup mode, auto recovery
			16.2		LCW50US12	
			21.75		LCW50US15	
			33.6		LCW50US24	
			49.0		LCW50US36	
			60.0		LCW50US48	
Overload Protection	110		200	%	Nominal output current, auto recovery	
Temperature Coefficient		±0.03	5	%/°C		
Short Circuit Protection	Continuous, hiccup with auto recovery					

STEPPER MOTOR 17HS15-1504S

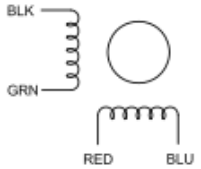


SPECIFICATION / CONNECTION	BIPOLAR
AMPS/PHASE	1.50
RESISTANCE/PHASE(Ohms)@25°C	2.30±10%
INDUCTANCE/PHASE(mH)@1KHz	4.40±20%
HOLDING TORQUE(Nm)[lb-in]	0.45[3.98]
STEP ANGLE(°)	1.80
STEP ACCURACY(NON-ACCUM)	±5.00%
ROTOR INERTIA(g-cm²)	54.00
WEIGHT(Kg)[lb]	0.28[0.62]
TEMPERATURE RISE:MAX.80°C（MOTOR STANDSTILL:FOR 2PHASE ENERGIZED）	
AMBIENT TEMPERATURE -10°C~50°C[14°F~122°F]	
INSULATION RESISTANCE 100 Mohm (UNDER NORMAL TEMPERATURE AND HUMIDITY)	
INSULATION CLASS B 130°C[266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1MIN.(BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX.85%(NO CONDENSATION)	

TYPE OF CONNECTION (EXTERN)		MOTOR	
PIN NO	BIPOLAR	LEADS	WINDING
1	A —	BLK	A
2	A\ —	GRN	A\
3	B —	RED	B
4	B\ —	BLU	B\

FULL STEP 2 PHASE-Ex. ,
WHEN FACING MOUNTING END (X)

STEP	A	B	A\	B\		
1	+	+	-	-	↓ CW	CCW ↑
2	-	+	+	-		
3	-	-	+	+		
4	+	-	-	+		



STEPPERONLINE®

		APVD		STEPPER MOTOR
		CHKD		
1:1		DRN		17HS15-1504S-X1
SCALE		SIGNATURE	DATE	

11.1.2 9. POTENTIOMETER 3852A-282-103AL

Features

- Single-turn (3851 and 3852)
- 3-3/4-turn (3856)
- Linear and audio tapers
- Wide resistance range
- Minimal depth package
- Good resolution

3851/3852/3856 - 3/4 " Diameter Panel Control

Initial Electrical Characteristics[1]

	3851 Conductive Plastic Element	3852/3856 Cermet Element
Standard Resistance Range		
Linear Tapers (A, B, E, and H)	1 K to 1 megohm	100 ohms to 1 megohm
Audio Tapers (C, D, F, and G)	1 K to 1 megohm	1 K ohms to 1 megohm
Total Resistance Tolerance	±10 % or ±20 %	±5 % or ±10 %
Independent Linearity	±10 %	(A & H tapers) ±5 %
Absolute Minimum Resistance	2 ohms maximum	2 ohms maximum
Effective Electrical Angle	(Linear tapers) 250 ° ±5 °	(Linear tapers) 250 ° ±5 °
	(Audio tapers) 225 ° ±5 °	(Audio tapers) 225 ° ±5 °
Contact Resistance Variation	±1 %	±3 % of total resistance or 3 ohms (whichever is greater)
Dielectric Withstanding Voltage (MIL-STD-202, Method 301)		
Sea Level	900 VAC minimum	900 VAC minimum
70,000 Feet	350 VAC minimum	350 VAC minimum
Insulation Resistance (500 VDC)	1,000 megohms minimum	1,000 megohms minimum
Power Rating (Voltage Limited By Power Dissipation or 350 VAC, Whichever Is Less)		
+70 °C	(Linear tapers) 1 watt	(Linear tapers) 2 watts
	(Audio tapers) 0.5 watt	(Audio tapers) 1 watt
+125 °C	0 watt	
+150 °C		0 watt
Theoretical Resolution	Essentially infinite	Essentially infinite

Environmental Characteristics[1]

Operating Temperature Range	-1 °C to +125 °C	-1 °C to +125 °C
Storage Temperature Range	-65 °C to +125 °C	-65 °C to +150 °C
Temperature Coefficient Over Storage Temperature Range	±1,000 ppm/°C	±150 ppm/°C
Vibration	20 G	20 G
Total Resistance Shift	±2 % maximum	±2 % maximum
Voltage Ratio Shift	±5 % maximum	±6 % maximum
Shock	100 G	100 G
Total Resistance Shift	±2 % maximum	±2 % maximum
Voltage Ratio Shift	±5 % maximum	±6 % maximum
Load Life	1,000 hours	1,000 hours
Total Resistance Shift	±10 % maximum	±3 % maximum
Rotational Life (No Load)	100,000 cycles	20,000 cycles
Total Resistance Shift	±15 % TRS maximum	±5 % or 5 ohms TRS whichever is greater
Contact Resistance Variation	±3 %	±3 %
Moisture Resistance (MIL-STD-202, Method 103, Condition B)		
Total Resistance Shift	±10 % maximum	±2 % maximum
IP Rating	IP 40	IP 40

[1] At room ambient: +25 °C nominal and 50 % relative humidity nominal, except as noted.

11.1.3 10. CURRENT SENSOR ACS70331

COMMON ELECTRICAL CHARACTERISTICS [1]: Valid over full range of T_A, and V_{CC} = 3.3 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Unit
ELECTRICAL CHARACTERISTICS						
Supply Voltage	V_{CC}		3.0	3.3	4.5	V
Supply Current	I_{CC}	$V_{CC}(min) \le V_{CC} \le V_{CC}(max)$, no load on VIOUT, fuses powered down	–	4.5	6	mA
	$I_{CC\_START\_UP}$	$V_{CC}(min) \le V_{CC} \le V_{CC}(max)$, no load on VIOUT, fuses powered (from time when V_{CC} rises above $V_{CC}(min)$ to t_{PPD})	–	–	7.5	mA
Primary Conductor Resistance	R_{IP}	QFN-12 package, T_A = 25°C	–	1.1	–	mΩ
		SOIC-8 package, T_A = 25°C	–	1.7	–	mΩ
Primary Conductor Inductance	L_{IP}	QFN-12 package	–	1.7	–	nH
		SOIC-8 package	–	4	–	nH
Power On Time	t_{PO}	T_A = 25°C	–	5	–	µs
Fuse Power Down Time [2]	t_{PPD}	T_A = 25°C	–	60	120 [3]	µs
OUTPUT CHARACTERISTICS						
Output Resistive Load	R_L	VIOUT to GND or VIOUT to VCC	22	–	–	kΩ
Output Capacitive Load	C_L	VIOUT to GND, output is stable, slew rate and bandwidth are reduced	–	–	100	pF
		VIOUT to GND, maintains BW	–	–	50	pF
Source Current	I_{SOURCE}	T_A = 25°C	–	0.4	–	mA
Sink Current	I_{SINK}	T_A = 25°C	–	0.5	–	mA
Saturation Voltage [4]	V_{SAT_HIGH}	$V_{CC}(min) < V_{CC} < V_{CC}(max)$; R_L = 22 kΩ to GND	2.8	V_{CC} – 0.15	–	V
	V_{SAT_LOW}	$V_{CC}(min) < V_{CC} < V_{CC}(max)$; R_L = 22 kΩ to VIOUT	–	20	200	mV
Bandwidth	BW	–3 dB bandwidth	–	1	–	MHz
Response Time	$t_{RESPONSE}$	1 V swing on VIOUT, 50% to 50%	–	535	–	ns
Rise Time	t_r	1 V swing on VIOUT, 10% to 90%	–	460	–	ns
Propagation Delay	t_{pd}	1 V swing on VIOUT, 20% to 20%	–	220	–	ns
Noise Density	I_{ND}	Input referred noise density	–	6	–	$\mu A_{RMS}/\sqrt{(Hz)}$
Noise	I_N	Input reference noise; T_A = 25°C, Bandwidth = 1 MHz	–	6	–	mA_{RMS}
Hysteresis	I_H	T_A = 25°C; change in the output at zero current after a ±10 A pulse of current through the sensor	–	10	20	mA
		T_A = 25°C; change in the output at zero current after a ±100 A pulse (~20 ms in duration) of current through the sensor	–	20	–	mA

Continued on next page...

ALLEGRO microsystems
Allegro MicroSystems
955 Perimeter Road
Manchester, NH 03103-3353 U.S.A.
www.allegromicro.com

11.2 PROGRAMMING CODES

11.2.1 MATLAB SIMULACION CODE

```
%Rotation angles in degrees
THETAz =0;
ALPHAx =0;
BETAY =0;
%Reference points for the position THETAz =0; ALPHAx =0;
BETAY =0;
Panel_reference_point=88.66;
P1 = [ 0 ; 0 ; -Panel_reference_point];
P2 = [ Panel_reference_point  ; 0 ; 0 ];
P3 = [  -Panel_reference_point ; 0 ; 0];
P4 = [ 0; 0 ; Panel_reference_point];
%Definition of lower pulley positions
Pulleydistance=92/2;
Pulley1=[ 0 ; 0 ; -Pulleydistance ];
Pulley2=[Pulleydistance ; 0 ; 0];
Pulley4=[ 0; 0 ; Pulleydistance ];
Pulley3=[-Pulleydistance; 0 ; 0 ];
%Definition sice of elements
MainTower_height=[ 0  95 0];
Distace_rotula_Yaxis_Xaxis=[0;49.52;0];
Distace_rotula_Xaxis_panel=[0;3.27;0];
%Rotational matrices
Rx = [ 1       0             0       ; ...
       0   cosd(ALPHAx)  -sind(ALPHAx) ; ...
       0   sind(ALPHAx)   cosd(ALPHAx)];


Rz = [ cosd(THETAz)  -sind(THETAz)   0 ; ...
       sind(THETAz)   cosd(THETAz)   0 ; ...
           0             0         1];
%Vector rotation
Pf1 = Rx*Rz*P1
Pf2 = Rx*Rz*P2
Pf3 = Rx*Rz*P3
Pf4 = Rx*Rz*P4
Axisf1= Rx*Distace_rotula_Yaxis_Xaxis
Axisf2= Rx*Rz*Distace_rotula_Xaxis_panel
hold on;
%Calculation and drawing of the vectors from the main
tower to the X-axis of the rotula
Vectoraxis1=[ Axisf1(1)  Axisf1(2) Axisf1(3)];
Plotaxis1=drawVector(MainTower_height, Vectoraxis1, 'm');
Endpointaxis1=[(MainTower_height(1)+Vectoraxis1(1))
(MainTower_height(2)+Vectoraxis1(2))
(MainTower_height(3)+Vectoraxis1(3))];
```

```
hold on
%Calculation and drawing of the vectors from the X-axis to
the panel of the rotula
Pos_rotaxis2=[ Endpointaxis1(1)  Endpointaxis1(2)
Endpointaxis1(3)];
Vectoraxis2=[ Axisf2(1)  Axisf2(2) Axisf2(3)];
Plotaxis2=drawVector(Pos_rotaxis2, Vectoraxis2, 'blue');
Endpointaxis2=[(Pos_rotaxis2(1) +Vectoraxis2(1))
(Pos_rotaxis2(2) +Vectoraxis2(2)) (Pos_rotaxis2(3)
+Vectoraxis2(3))];
hold on
%Calculation and drawing of the turned panel vectors
Pointspanel=[Endpointaxis2(1) Endpointaxis2(2)
Endpointaxis2(3) ;Endpointaxis2(1) Endpointaxis2(2)
Endpointaxis2(3) ;Endpointaxis2(1) Endpointaxis2(2)
Endpointaxis2(3) ;Endpointaxis2(1) Endpointaxis2(2)
Endpointaxis2(3)];
Vectorpanel=[Pf1(1) Pf1(2) Pf1(3);Pf2(1) Pf2(2) Pf2(3)
;Pf3(1) Pf3(2) Pf3(3) ;Pf4(1) Pf4(2) Pf4(3)];
Plotpanel=drawVector(Pointspanel, Vectorpanel, 'red');
hold on
%Calculation and drawing of the vectors that contain main
tower and the pulleys positions
Ptower= [ 0 ; 0 ;0];
Pos2=[0 ; 0 ;0];
Pointssuport=[Ptower(1) Ptower(2) Ptower(3);Pos2(1)
Pos2(2) Pos2(3);Pos2(1) Pos2(2) Pos2(3);Pos2(1) Pos2(2)
Pos2(3);Pos2(1) Pos2(2) Pos2(3)]
Vectorsuport=[MainTower_height(1) MainTower_height(2)
MainTower_height(3) ;Pulley1(1) Pulley1(2)
Pulley1(3);Pulley2(1) Pulley2(2) Pulley2(3);Pulley3(1)
Pulley3(2) Pulley3(3);Pulley4(1) Pulley4(2) Pulley4(3)]
Plotsuport=drawVector(Pointssuport, Vectorsuport, 'red');
hold on
%drawing of the axes
xlabel('X')
ylabel('Y')
zlabel('Z')
daspect([1 1 1])
%calculation of the distances between upper and lower
pulleys
Prodesc1= dot(Pf1,Pf2)
Prodesc2= dot(Pf1,Vectoraxis2)
Endpointpf1=[(Endpointaxis2(1)+Pf1(1))
(Endpointaxis2(2)+Pf1(2)) (Endpointaxis2(3)+Pf1(3))]
Endpointpf2=[(Endpointaxis2(1)+Pf2(1))
(Endpointaxis2(2)+Pf2(2)) (Endpointaxis2(3)+Pf2(3))]
```

```
Endpointpf3=[(Endpointaxis2(1)+Pf3(1))
(Endpointaxis2(2)+Pf3(2)) (Endpointaxis2(3)+Pf3(3))]
Endpointpf4=[(Endpointaxis2(1)+Pf4(1))
(Endpointaxis2(2)+Pf4(2)) (Endpointaxis2(3)+Pf4(3))]
Distance1=((Endpointpf1(1)-Pulley1(1))^2 +(Endpointpf1(2)-
Pulley1(2))^2+(Endpointpf1(3)-Pulley1(3))^2)^0.5
Distance2=((Endpointpf2(1)-Pulley2(1))^2 +(Endpointpf2(2)-
Pulley2(2))^2+(Endpointpf2(3)-Pulley2(3))^2)^0.5
Distance3=((Endpointpf3(1)-Pulley3(1))^2 +(Endpointpf3(2)-
Pulley3(2))^2+(Endpointpf3(3)-Pulley3(3))^2)^0.5
Distance4=((Endpointpf4(1)-Pulley4(1))^2 +(Endpointpf4(2)-
Pulley4(2))^2+(Endpointpf4(3)-Pulley4(3))^2)^0.5
%Calculation and drawing of the cables in charge of the X
turn
Pointsdistance=[Endpointpf1(1) Endpointpf1(2)
Endpointpf1(3);Endpointpf4(1) Endpointpf4(2)
Endpointpf4(3)]
Vectordistance=[(Endpointpf1(1)-Pulley1(1)-Pos2(1))
(Endpointpf1(2)-Pulley1(2)-Pos2(2)) (Endpointpf1(3)-
Pulley1(3)-Pos2(3));(Endpointpf4(1)-Pulley4(1)-Pos2(1))
(Endpointpf4(2)-Pulley4(2)-Pos2(2)) (Endpointpf4(3)-
Pulley4(3)-Pos2(3))]
TransVectordistance=transpose(Vectordistance);
Plotdistace=drawVector(Pointsdistance, -Vectordistance,
'g');
%Calculation and drawing of the cables in charge of the X
turn
Pointsdistance2=[Endpointpf2(1) Endpointpf2(2)
Endpointpf2(3);Endpointpf3(1) Endpointpf3(2)
Endpointpf3(3)]
Vectordistance2=[(Endpointpf2(1)-Pulley2(1)-Pos2(1))
(Endpointpf2(2)-Pulley2(2)-Pos2(2)) (Endpointpf2(3)-
Pulley2(3)-Pos2(3));(Endpointpf3(1)-Pulley3(1)-Pos2(1))
(Endpointpf3(2)-Pulley3(2)-Pos2(2)) (Endpointpf3(3)-
Pulley3(3)-Pos2(3))]
TransVectordistance2=transpose(Vectordistance2);
Plotdistace=drawVector(Puntosdistance2, -Vectordistance2,
'y');
hold off
Distance23=Distance2+Distance3
Distance14=Distance1+Distance4
%calculation of moments and forces for static model
Tension1=100
Tension2=100
Vmoment1=transpose(TransVectordistance(:,1))
Vmoment2=transpose(TransVectordistance2(:,1))
Vmoment3=transpose(TransVectordistance2(:,2))
```

```
Vmoment4=transpose(TransVectordistance(:,2))
Nmoment1=norm(transpose(TransVectordistance(:,1)))
Nmoment2=norm(transpose(TransVectordistance2(:,1)))
Nmoment3=norm(transpose(TransVectordistance2(:,2)))
Nmoment4=norm(transpose(TransVectordistance(:,2)))
Vfmoment1=Vmoment1/Nmoment1
Vfmoment2=Vmoment2/Nmoment2
Vfmoment3=Vmoment3/Nmoment3
Vfmoment4=Vmoment4/Nmoment4
Moment1=Tension1*Vfmoment1*[0 ;(Endpointpf1(2)-
MainTower_height(2)) ;(Endpointpf1(3)-
MainTower_height(3))]
Moment2=Tension1*Vfmoment2*[0 ;(Endpointpf2(2)-
MainTower_height(2)) ;(Endpointpf2(3)-
MainTower_height(3))]
Moment3=Tension1*Vfmoment3*[0 ;(Endpointpf3(2)-
MainTower_height(2)) ;(Endpointpf3(3)-
MainTower_height(3))]
Moment4=Tension1*Vfmoment4*[0 ;(Endpointpf4(2)-
MainTower_height(2)) ;(Endpointpf4(3)-
MainTower_height(3))]
TorsorX=Moment1+Moment3
TorsorY=Moment2+Moment4
```

11.2.2 ELECTIRC NOISE TEST CODE

```
// declare the variables
int sensorPin1 = A2;
int sensorPin2 = A1;
float cur;
float vol;
float ene;
int i;
float currecord;
float volrecord;
void setup() {
Serial.begin(9600);
}

void loop() {

for(i=0;i<100;i++){
// read the value from the sensor:
cur = analogRead(sensorPin1);
Serial.println(cur);
// incorporate the calculated sensor characterization and apply analog value conversion :
cur=((cur*5)/(1023)-0.314)/0.7955;
currecord=cur+currecord;
// read the value from the sensor:
vol = analogRead(sensorPin2);
```

```
// apply analog value conversion:
vol=vol*5/1023;
volrecord=vol+volrecord;
}
//
cur=currecord/100;
vol=volrecord/100;
ene=cur*vol;
// print the values into the serial port:
Serial.println("energy");
Serial.println(ene);
Serial.println("voltage");
Serial.println(vol);
Serial.println("current");
Serial.println(cur);
delay(2000);
// reseting the values for starting the next measurement:
i=0;
currecord=0;
volrecord=0;
}
```

11.2.3 ARDUINO MEGA CODE

11.2.3.1.1 MAIN LOOP

```
//Include the needed library, we will use softer serial communication with
the Serial2
#include <SoftwareSerial.h>
//#include <avr/power.h>
#include <Stepper.h>
#include <ArduinoJson.h>
//SoftwareSerial Arduino_SoftSerial (10,11);
//LCD config
//#include <Wire.h>
//#include <LiquidCrystal_I2C.h>
//LiquidCrystal_I2C lcd(0x3f,20,4);  //sometimes the LCD adress is not
0x3f. Change to 0x27 if it dosn't work.
//Initialise Arduino to NodeMCU (5=Rx & 6=Tx)
//SoftwareSerial nodemcu(5, 6);
int pinA1= A1;
int pinA2= A2;
//Initialisation of DHT11 Sensor
float poten1;
float poten2;
float aa;
//Define the used
#define Serial2_RX 10  //Connect the TX pin from the ESP to this RX pin of
the Arduino
#define Serial2_TX 11  //Connect the TX pin from the Arduino to the RX pin
of ESP

int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
```

```
int LED5 = 6;
int Potentiometer_1 = A0;
int Potentiometer_2 = A1;
int Potentiometer_3 = A2;
int Potentiometer_4 = A3;
int switch1 = 7;
int switch2 = 8;
int switch3 = 9;


////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//////////////////////////////Variables you must change according to your values//////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
//Add your data: SSID + KEY + host + location + id + password
/////////////////////////////////////////////
const char SSID_ESP[] = "Fancy mesh-network";         //Give EXACT name of your WIFI
const char SSID_KEY[] = "Fjordgata7";              //Add the password of that WIFI connection
const char* host = "solartrackertrondheim.000webhostapp.com";  //Add the host without "www" Example: electronoobs.com
String NOOBIX_id = "99999";                        //This is the ID you have on your database, I've used 99999 becuase there is a maximum of 5 characters
String NOOBIX_password = "12345";                  //Add the password from the database, also maximum 5 characters and only numerical values
String location_url = "/TX.php?id=";               //location of your PHP file on the server. In this case the TX.php is directly on the first folder of the server
                                                   //If you have the files in a different folder, add thas as well, Example: "/ESP/TX.php?id="     Where the folder is ESP
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////


  char c;
  String dataIn;
String String1;
String String2;
String send1mn;
String send2mn;
String send3mn;
String send4mn;
String send5mn;
String recive1nm;
String recive2nm;
String recive3nm;
String recive4nm;
String recive5nm;
String aux;
```

```
float ElevationGoogle ; float AzimutGoogle ; float HourGoogle;

//Used variables in the code
String url = "";
String URL_withPacket = "";
unsigned long multiplier[] =
{1,10,100,1000,10000,100000,1000000,10000000,100000000,1000000000};
//MODES for the ESP
const char CWMODE = '1';//CWMODE 1=STATION, 2=APMODE, 3=BOTH
const char CIPMUX = '1';//CWMODE 0=Single Connection, 1=Multiple
Connections


//Define the used functions later in the code, thanks to Kevin Darrah, YT
channel:  https://www.youtube.com/user/kdarrah1234
boolean setup_ESP();
boolean read_until_ESP(const char keyword1[], int key_size, int
timeout_val, byte mode);
void timeout_start();
boolean timeout_check(int timeout_ms);
void serial_dump_ESP();
boolean connect_ESP();
void connect_webhost();
unsigned long timeout_start_val;
char scratch_data_from_ESP[20];//first byte is the length of bytes
char payload[200];
byte payload_size=0, counter=0;
char ip_address[16];


////////////////////////////////////////////////////////////////////////////
//////////////////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////////////
//Variable to SEND to the DATABASE
bool sent_bool_1 = 0;
bool sent_bool_2 = 0;
bool sent_bool_3 = 0;
int  sent_nr_1 = 0;
int  sent_nr_2 = 0;
int  sent_nr_3 = 0;
int  sent_nr_4 = 0;
int  sent_nr_5 = 0;

//Variable RECEIVED from the DATABASE
bool received_bool_1 = 0;
bool received_bool_2 = 0;
bool received_bool_3 = 0;
bool received_bool_4 = 0;
bool received_bool_5 = 0;
int  received_nr_1 = 0;
int  received_nr_2 = 0;
int  received_nr_3 = 0;
int  received_nr_4 = 0;
int  received_nr_5 = 0;
String received_text = "";
```

```
////////////////////////////////////////////////////////////////////////////
//////////////////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////////////



//Store received chars in this variables
char t1_from_ESP[5];  //For time from web
char d1_from_ESP[2];  //For received_bool_2
char d2_from_ESP[2];  //For received_bool_2
char d3_from_ESP[2];  //For received_bool_3
char d4_from_ESP[2];  //For received_bool_4
char d5_from_ESP[2];  //For received_bool_5
char d9_from_ESP[6];  //For received_nr_1
char d10_from_ESP[6]; //For received_nr_2
char d11_from_ESP[6]; //For received_nr_3
char d12_from_ESP[6]; //For received_nr_4
char d13_from_ESP[6]; //For received_nr_5
char d14_from_ESP[300]; //For received_text


//DEFINE KEYWORDS HERE
const char keyword_OK[] = "OK";
const char keyword_Ready[] = "Ready";
const char keyword_no_change[] = "no change";
const char keyword_blank[] = "#&";
const char keyword_ip[] = "192.";
const char keyword_rn[] = "\r\n";
const char keyword_quote[] = "\"";
const char keyword_carrot[] = ">";
const char keyword_sendok[] = "SEND OK";
const char keyword_linkdisc[] = "Unlink";

const char keyword_t1[] = "t1";
const char keyword_b1[] = "b1";
const char keyword_b2[] = "b2";
const char keyword_b3[] = "b3";
const char keyword_b4[] = "b4";
const char keyword_b5[] = "b5";
const char keyword_n1[] = "n1";
const char keyword_n2[] = "n2";
const char keyword_n3[] = "n3";
const char keyword_n4[] = "n4";
const char keyword_n5[] = "n5";
const char keyword_n6[] = "n6";
const char keyword_doublehash[] = "##";


//SoftwareSerial Serial2(10, 11);// rx tx
const int stepsPerRevolution = 10;
// change this to fit the number of steps per revolution
// for your motor

// initialize the stepper library on pins 8 through 11:
Stepper stepper(stepsPerRevolution, 4, 5, 6, 7);
```

```
void setup(){//        SETUP     START

//  lcd.init();                  //Init the LCD
  //lcd.backlight();             //Activate backlight

  //Pin Modes for ESP TX/RX
  pinMode(Serial2_RX, INPUT);
  pinMode(Serial2_TX, OUTPUT);

  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);

  pinMode(Potentiometer_1, INPUT);
  pinMode(Potentiometer_2, INPUT);
  pinMode(Potentiometer_3, INPUT);
  pinMode(Potentiometer_4, INPUT);

  pinMode(switch1, INPUT);
  pinMode(switch2, INPUT);
  pinMode(switch3, INPUT);


  digitalWrite(LED1,LOW);
  digitalWrite(LED2,LOW);
  digitalWrite(LED3,LOW);
  digitalWrite(LED4,LOW);
  digitalWrite(LED5,LOW);

Serial1.begin(9600);
  //Arduino_SoftSerial.begin(9600);
  Serial2.begin(9600);//default baudrate for ESP
//   nodemcu.begin(9600);
  //Serial2.listen();//not needed unless using other software serial
instances
   stepper.setSpeed(10);
  Serial.begin(9600); //for status and debug
   // set the speed at 60 rpm:
  digitalWrite(12,LOW);
  digitalWrite(13,HIGH);
  delay(2000);//delay before kicking things off
  setup_ESP();//go setup the ESP

}



void loop(){

  sent_nr_1 = analogRead(Potentiometer_1);
  send1mn=sent_nr_1;
  sent_nr_2 = analogRead(Potentiometer_2);
  send2mn=sent_nr_2;
```

```
  sent_nr_3 = analogRead(Potentiometer_3);
  send3mn=sent_nr_3;
  sent_nr_4 = analogRead(Potentiometer_4);
  send4mn=sent_nr_4;
  sent_bool_1 = digitalRead(switch1);
  sent_bool_2 = digitalRead(switch2);
  sent_bool_3 = digitalRead(switch3);
  delay(500);
  MEGADRIVE( );
Serial.println("Elevation : ")+Serial.println(ElevationGoogle);
    Serial.println("Azimut : ");
    Serial.println(AzimutGoogle);
    Serial.println("Hour : ");
    Serial.println(HourGoogle);

  aa=0;
  send_to_server_1();
  send_to_server_2();
  send_to_server_3();
  send_to_server_4();
  send_to_server_5();
  digitalWrite(LED1,received_bool_1);
  digitalWrite(LED2,received_bool_2);
  digitalWrite(LED3,received_bool_3);
  digitalWrite(LED4,received_bool_4);
  digitalWrite(LED5,received_bool_5);
}
//End of the main loop
```

11.2.3.1.2 NODEMCU SERIAL CMUNICATION CODE

```
void MEGADRIVE( ) {while(aa==0){

  Serial1.print(send1mn+"!"+send2mn+"@"+send3mn+"#"+send4mn+"\n");
   delay(500);
   while(Serial1.available()>0){
    c =Serial1.read();
    if(c=='\n'){break;}
    else {dataIn+=c;}
   }
   if(c=='\n'){
    Serial.println(dataIn);
    c=0;


   }if(dataIn!=""){

    dataIn.toLowerCase();
    dataIn.trim();
    recive1nm=dataIn.substring(0,dataIn.indexOf('!'));
    recive2nm=dataIn.substring(dataIn.indexOf('!')+1,dataIn.indexOf('@'));
    recive3nm=dataIn.substring(dataIn.indexOf('@')+1,dataIn.indexOf('#'));
    //String4=payload.substring(payload.indexOf('#')+1,payload.length())
ElevationGoogle= recive1nm.toFloat(); AzimutGoogle= recive2nm.toFloat() ;
HourGoogle= recive3nm.toFloat();

     aa=1;}
    //Serial.println("Relay3 : "+String3);
    //Serial.println("Relay4 : "+String4);
```

```
     dataIn=""; delay(500);

}}
```

11.2.3.1.3 ESP SETUP CODE

```
boolean connect_ESP(){//returns 1 if successful or 0 if not


  Serial.println("CONNECTING");
  //or 443 para HTTPS
  //enshare.000webhostapp.co

Serial2.print("AT+CIPSTART=0,\"TCP\",\"http://solartrackertrondheim.000webhostapp.com\",80\r\n");//connect to your web server


  //read_until_ESP(keyword,size of the keyword,timeout in ms, data save 0-no 1-yes 'more on this later')
  if(read_until_ESP(keyword_OK,sizeof(keyword_OK),5000,0)){//go look for 'OK' and come back
  serial_dump_ESP();//get rid of whatever else is coming
  Serial.println("CONNECTED");//yay, connected
  Serial2.print("AT+CIPSEND=0,");//send AT+CIPSEND=0, size of payload
  Serial2.print(payload_size);//the payload size
  serial_dump_ESP();//everything is echoed back, so get rid of it
  Serial2.print("\r\n\r\n");//cap off that command with a carriage return and new line feed

  if(read_until_ESP(keyword_carrot,sizeof(keyword_carrot),5000,0)){//go wait for the '>' character, ESP ready for the payload
    Serial.println("READY TO SEND");

    Serial.println(payload_size);

    for(int i=0; i<payload_size; i++)
    {//print the payload to the ESP
    Serial2.print(payload[i]);



    Serial.print(payload[i]);

    }

    /*
    Serial.println("");
    Serial.println(payload_size);
    Serial.println("");
    */

    if(read_until_ESP(keyword_sendok,sizeof(keyword_sendok),5000,0)){//go wait for 'SEND OK'
    Serial.println("SENT");//yay, it was sent
    return 1;//get out of here, data is about to fly out of the ESP
  }// got the SEND OK
    else// SEND OK
  Serial.println("FAILED TO SEND");
```

```
  }//got the back carrot >
  else
  Serial.println("FAILED TO GET >");

  }//First OK
  else{
  Serial.println("FAILED TO CONNECT");//something went wrong
  setup_ESP();//optional, this will go setup the module and attempt to
repair itself - connect to SSID, set the CIPMUX, etc...
  }

}// VOID CONNECT FUNCTION
```

11.2.3.1.4 SENT TO WEB SEVER CODE

```
boolean connect_ESP(){//returns 1 if successful or 0 if not


  Serial.println("CONNECTING");
  //or 443 para HTTPS
  //enshare.000webhostapp.co

Serial2.print("AT+CIPSTART=0,\"TCP\",\"http://solartrackertrondheim.000webh
ostapp.com\",80\r\n");//connect to your web server


  //read_until_ESP(keyword,size of the keyword,timeout in ms, data save 0-
no 1-yes 'more on this later')
  if(read_until_ESP(keyword_OK,sizeof(keyword_OK),5000,0)){//go look for
'OK' and come back
  serial_dump_ESP();//get rid of whatever else is coming
  Serial.println("CONNECTED");//yay, connected
  Serial2.print("AT+CIPSEND=0,");//send AT+CIPSEND=0, size of payload
  Serial2.print(payload_size);//the payload size
  serial_dump_ESP();//everything is echoed back, so get rid of it
  Serial2.print("\r\n\r\n");//cap off that command with a carriage return
and new line feed

  if(read_until_ESP(keyword_carrot,sizeof(keyword_carrot),5000,0)){//go
wait for the '>' character, ESP ready for the payload
    Serial.println("READY TO SEND");

    Serial.println(payload_size);

    for(int i=0; i<payload_size; i++)
    {//print the payload to the ESP
    Serial2.print(payload[i]);



    Serial.print(payload[i]);

    }

    /*
    Serial.println("");
    Serial.println(payload_size);
    Serial.println("");
    */
```

```
    if(read_until_ESP(keyword_sendok,sizeof(keyword_sendok),5000,0)){//go
wait for 'SEND OK'
    Serial.println("SENT");//yay, it was sent
    return 1;//get out of here, data is about to fly out of the ESP
  }// got the SEND OK
    else// SEND OK
  Serial.println("FAILED TO SEND");

  }//got the back carrot >
  else
  Serial.println("FAILED TO GET >");

  }//First OK
  else{
  Serial.println("FAILED TO CONNECT");//something went wrong
  setup_ESP();//optional, this will go setup the module and attempt to
repair itself - connect to SSID, set the CIPMUX, etc...
  }

}// VOID CONNECT FUNCTION
```

11.2.3.1.5 SERIAL COMUNICATION WITH ESP01

```
//pretty simple function - read everything out of the serial buffer and
whats coming and get rid of it
void serial_dump_ESP(){
  char temp;
  while(Serial2.available()){
    temp =Serial2.read();
    delay(1);//could play around with this value if buffer overflows are
occuring
  }//while
  //Serial.println("DUMPED");


}//serial dump
```

11.2.3.1.6 NODEMCU CODE

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
 //D6 = Rx & D5 = Tx
SoftwareSerial nodemcu(D6, D5);
#define ON_Board_LED 2  //--> Defining an On Board LED, used for indicators
when the process of connecting to a wifi router

const char* ssid = "Gang_wifi"; //--> Your wifi name or SSID.
const char* password = "wifihouse"; //--> Your wifi password.


//-----------------------------------------Host & httpsPort
const char* host = "script.google.com";
const int httpsPort = 443;
//-----------------------------------------

WiFiClientSecure client; //--> Create a WiFiClientSecure object.
```

```
String GAS_ID =
"AKfycbwn5B1t0ftFi9F3p47XpgsUnk6XL8nl3NoZYiEnucrOO7ty7oxO7Pv_9kxFPx336kmP";
//--> spreadsheet script ID

void setup() {
  /// Initialize Serial port
  Serial.begin(9600);
  nodemcu.begin(9600);
  while (!Serial) continue;
    WiFi.begin(ssid, password); //--> Connect to your WiFi router
  Serial.println("");

  pinMode(ON_Board_LED,OUTPUT); //--> On Board LED port Direction output
  digitalWrite(ON_Board_LED, HIGH); //--> Turn off Led On Board

  //----------------------------------------Wait for connection
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    //----------------------------------------Make the On Board Flashing
LED on the process of connecting to the wifi router.
    digitalWrite(ON_Board_LED, LOW);
    delay(250);
    digitalWrite(ON_Board_LED, HIGH);
    delay(250);
    //----------------------------------------
  }
  //----------------------------------------
  digitalWrite(ON_Board_LED, HIGH); //--> Turn off the On Board LED when it
is connected to the wifi router.
  Serial.println("");
  Serial.print("Successfully connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  //----------------------------------------

  client.setInsecure();
}

void loop() {
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow
sensor)
  float potenacu1=0;
  float potenacu2=0;
  float poten1med=0;
  float poten2med=0;
  //for(int i=0;i<=1;i++)
    //{
      StaticJsonBuffer<1000> jsonBuffer;
  JsonObject& data = jsonBuffer.parseObject(nodemcu);

 //if (data == JsonObject::invalid()) {
    //Serial.println("Invalid Json Object");
   // jsonBuffer.clear();
```

```
    //return;
 // }

  Serial.println("JSON Object Recieved");
  Serial.print("Recieved potenciometro1:  ");
  float poten1 = data["potenciometro1"];
  Serial.println(poten1);
  Serial.print("Recieved potenciometro2:  ");
  float poten2 = data["potenciometro2"];
  Serial.println(poten2);
  Serial.println("-----------------------------------------");
  potenacu1=poten1+potenacu1;
   potenacu2=poten2+potenacu2;
   delay(1000);
   // }
  poten1med=potenacu1/1;
  poten2med=potenacu2/1;
  sendData(poten1med, poten2med);
}
// Subroutine for sending data to Google Sheets
void sendData(float Poten1 , float Poten2) {

  Serial.println("==========");
  Serial.print("connecting to ");
  Serial.println(host);

  //----------------------------------------Connect to Google host
  if (!client.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }
  //----------------------------------------

  //----------------------------------------Processing data and sending
data

  String url = "/macros/s/" + GAS_ID + "/exec?value1=" + String(Poten1) +
"&value2=" + String(Poten2)+ "&value3=" + 34;
  Serial.print("requesting URL: ");
  Serial.println(url);

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
         "Host: " + host + "\r\n" +
         "User-Agent: BuildFailureDetectorESP8266\r\n" +
         "Connection: close\r\n\r\n");

  Serial.println("request sent");
  //----------------------------------------

  //----------------------------------------Checking whether the data was
sent successfully or not
  while (client.connected()) {
    String line = client.readStringUntil('\n');
    if (line == "\r") {
      Serial.println("headers received");
      break;
    }
```

```
  }
  String line = client.readStringUntil('\n');
  if (line.startsWith("{\"state\":\"success\"")) {
    Serial.println("esp8266/Arduino CI successfull!");
  } else {
    Serial.println("esp8266/Arduino CI has failed");
  }
  Serial.print("reply was : ");
  Serial.println(line);
  Serial.println("closing connection");
  Serial.println("==========");
  Serial.println();
  //----------------------------------------
}
```

A

11.2.4 WEB CODES

11.2.4.1 MAIN WEB PAGE CODE

11.2.4.1.1 DATA BASE CONNECTION CODE

```
<?php
$con=mysqli_connect("localhost","id18417627_solartracker","SoporteSol@r2010
","id18417627_esp8266");// server, user, password, database
?>
```

11.2.4.1.2 WEB CODE

```
<?php
//This line will make the page auto-refresh each 15 seconds
$page = $_SERVER['PHP_SELF'];
$sec = "15";
?>


<html>
<head>
<!--//I've used bootstrap for the tables, so I inport the CSS files for
taht as well...-->
<meta http-equiv="refresh" content="<?php echo $sec?>;URL='<?php echo
$page?>'">
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css
">
<!-- jQuery library -->
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></sc
ript>
<!-- Latest compiled JavaScript -->
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"><
/script>
</head>
```

```
<body>
<?php
include("database_connect.php"); //We include the database_connect.php
which has the data for the connection to the database


// Check the connection
if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
//Again, we grab the table out of the database, name is ESPtable2 in this
case
$result = mysqli_query($con,"SELECT * FROM ESPtable2");//table select



//Now we create the table with all the values from the database
echo "<table class='table' style='font-size: 30px;'>
        </thead>

    <tbody>
      <tr class='active'>

        <td>Movement Capability</td>
        <td>Safe Mode </td>
               <td>Automatic mode (ON) \Manual mode (OFF)</td>
               <td>Photoresistors (ON) \ Sun position calculation
(OFF)</td>
        <td>Step tracking mode (ON)\ Potenciometre mode (OFF)</td>

      </tr>
               ";

//loop through the table and print the data into the table
while($row = mysqli_fetch_array($result)) {

   echo "<tr class='success'>";
    $unit_id = $row['id'];

    $column1 = "RECEIVED_BOOL1";
    $column2 = "RECEIVED_BOOL2";
    $column3 = "RECEIVED_BOOL3";
    $column4 = "RECEIVED_BOOL4";
    $column5 = "RECEIVED_BOOL5";

    $current_bool_1 = $row['RECEIVED_BOOL1'];
        $current_bool_2 = $row['RECEIVED_BOOL2'];
        $current_bool_3 = $row['RECEIVED_BOOL3'];
        $current_bool_4 = $row['RECEIVED_BOOL4'];
        $current_bool_5 = $row['RECEIVED_BOOL5'];

        if($current_bool_1 == 1){
```

```
$inv_current_bool_1 = 0;
    $text_current_bool_1 = "ON";
    $color_current_bool_1 = "#6ed829";
    }
    else{
$inv_current_bool_1 = 1;
    $text_current_bool_1 = "OFF";
    $color_current_bool_1 = "#e04141";
    }


    if($current_bool_2 == 1){
$inv_current_bool_2 = 0;
    $text_current_bool_2 = "ON";
    $color_current_bool_2 = "#6ed829";
    }
    else{
$inv_current_bool_2 = 1;
    $text_current_bool_2 = "OFF";
    $color_current_bool_2 = "#e04141";
    }


    if($current_bool_3 == 1){
$inv_current_bool_3 = 0;
    $text_current_bool_3 = "ON";
    $color_current_bool_3 = "#6ed829";
    }
    else{
$inv_current_bool_3 = 1;
    $text_current_bool_3 = "OFF";
    $color_current_bool_3 = "#e04141";
    }


    if($current_bool_4 == 1){
$inv_current_bool_4 = 0;
    $text_current_bool_4 = "ON";
    $color_current_bool_4 = "#6ed829";
    }
    else{
$inv_current_bool_4 = 1;
    $text_current_bool_4 = "OFF";
    $color_current_bool_4 = "#e04141";
    }


    if($current_bool_5 == 1){
$inv_current_bool_5 = 0;
    $text_current_bool_5 = "ON";
    $color_current_bool_5 = "#6ed829";
    }
    else{
$inv_current_bool_5 = 1;
    $text_current_bool_5 = "OFF";
    $color_current_bool_5 = "#e04141";
    }
```

```
        echo "<td><form action= update_values.php method= 'post'>
        <input type='hidden' name='value2' value=$current_bool_1
size='15' >
        <input type='hidden' name='value' value=$inv_current_bool_1
size='15' >
        <input type='hidden' name='unit' value=$unit_id >
        <input type='hidden' name='column' value=$column1 >
        <input type= 'submit' name= 'change_but' style=' margin-left: 25%;
margin-top: 10%; font-size: 30px; text-align:center; background-color:
$color_current_bool_1' value=$text_current_bool_1></form></td>";


        echo "<td><form action= update_values.php method= 'post'>
        <input type='hidden' name='value2' value=$current_bool_2
size='15' >
        <input type='hidden' name='value' value=$inv_current_bool_2
size='15' >
        <input type='hidden' name='unit' value=$unit_id >
        <input type='hidden' name='column' value=$column2 >
        <input type= 'submit' name= 'change_but' style=' margin-left: 25%;
margin-top: 10%; font-size: 30px; text-align:center; background-color:
$color_current_bool_2' value=$text_current_bool_2></form></td>";

        echo "<td><form action= update_values.php method= 'post'>
        <input type='hidden' name='value2' value=$current_bool_3
size='15' >
        <input type='hidden' name='value' value=$inv_current_bool_3
size='15' >
        <input type='hidden' name='unit' value=$unit_id >
        <input type='hidden' name='column' value=$column3 >
        <input type= 'submit' name= 'change_but' style=' margin-left: 25%;
margin-top: 10%; font-size: 30px; text-align:center; background-color:
$color_current_bool_3' value=$text_current_bool_3></form></td>";

        echo "<td><form action= update_values.php method= 'post'>
        <input type='hidden' name='value2' value=$current_bool_4
size='15' >
        <input type='hidden' name='value' value=$inv_current_bool_4
size='15' >
        <input type='hidden' name='unit' value=$unit_id >
        <input type='hidden' name='column' value=$column4 >
        <input type= 'submit' name= 'change_but' style=' margin-left: 25%;
margin-top: 10%; font-size: 30px; text-align:center; background-color:
$color_current_bool_4' value=$text_current_bool_4></form></td>";

        echo "<td><form action= update_values.php method= 'post'>
        <input type='hidden' name='value2' value=$current_bool_5
size='15' >
        <input type='hidden' name='value' value=$inv_current_bool_5
size='15' >
        <input type='hidden' name='unit' value=$unit_id >
```

```
        <input type='hidden' name='column' value=$column5 >
        <input type= 'submit' name= 'change_but' style=' margin-left: 25%;
margin-top: 10%; font-size: 30px; text-align:center; background-color:
$color_current_bool_5' value=$text_current_bool_5></form></td>";

        echo "</tr>
          </tbody>";

}
echo "</table>
<br>
";
?>


<?php
include("database_connect.php");

if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$result = mysqli_query($con,"SELECT * FROM ESPtable2");//table select

echo "<table class='table' style='font-size: 30px;'>
        </thead>

    <tbody>
      <tr class='active'>
        <td>Manual Azumuth input</td>
        <td>Manual Elevation input</td>
      </tr>
                ";

while($row = mysqli_fetch_array($result)) {

        echo "<tr class='success'>";

    $column6 = "RECEIVED_NUM1";
    $column7 = "RECEIVED_NUM2";
    $column8 = "RECEIVED_NUM3";
    $column9 = "RECEIVED_NUM4";
    $column10 = "RECEIVED_NUM5";

    $current_num_1 = $row['RECEIVED_NUM1'];
        $current_num_2 = $row['RECEIVED_NUM2'];
        $current_num_3 = $row['RECEIVED_NUM3'];
        $current_num_4 = $row['RECEIVED_NUM4'];
        $current_num_5 = $row['RECEIVED_NUM5'];


        echo "<td><form action= update_values.php method= 'post'>
        <input type='text' name='value' style='width: 120px;'
value=$current_num_1  size='15' >
        <input type='hidden' name='unit' style='width: 120px;'
value=$unit_id >
        <input type='hidden' name='column' style='width: 120px;'
value=$column6 >
```

```
        <input type= 'submit' name= 'change_but' style='width: 120px; text-
align:center;' value='change'></form></td>";



        echo "<td><form action= update_values.php method= 'post'>
        <input type='text' name='value' style='width: 120px;'
value=$current_num_2  size='15' >
        <input type='hidden' name='unit' style='width: 120px;'
value=$unit_id >
        <input type='hidden' name='column' style='width: 120px;'
value=$column7 >
        <input type= 'submit' name= 'change_but' style='text-align:center'
value='change'></form></td>";


        echo "</tr>
          </tbody>";

}
echo "</table>
<br>
";
?>
<?php

include("database_connect.php");

if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM ESPtable2");//table select


echo "<table class='table' style='font-size: 30px;'>
        </thead>

    <tbody>
      <tr class='active'>
        <td>Azimuth Angle</td>
        <td>Elevation Angle</td>
        <td>Voltage </td>
                <td>Current</td>
      </tr>
                ";


while($row = mysqli_fetch_array($result)) {

        echo "<tr class='info'>";

        echo "<td>" . $row['SENT_NUMBER_1'] . "</td>";
        echo "<td>" . $row['SENT_NUMBER_2'] . "</td>";
        echo "<td>" . $row['SENT_NUMBER_3'] . "</td>";
        echo "<td>" . $row['SENT_NUMBER_4'] . "</td>";
```

```
        echo "</tr>
        </tbody>";


}
echo "</table>
<br>
";
?>
```

11.2.4.1.3 CODE 2

```
<?php
If you paste that link to your browser, it should update b1 value with this
TX.php file. Read more details below.
The ESP will send a link like the one above but with more than just b1. It
will have b1, b2, etc...
*/

//We loop through and grab variables from the received the URL
foreach($_REQUEST as $key => $value)  //Save the received value to the hey
variable. Save each cahracter after the "&"
{
        //Now we detect if we recheive the id, the password, unit, or a
value to update
if($key =="id"){
$unit = $value;
}
if($key =="pw"){
$pass = $value;
}
if($key =="un"){
$update_number = $value;
}

if($update_number == 1)
{
        if($key =="n1"){
                $sent_nr_1 = $value;
        }
}
else if($update_number == 2)
{
        if($key =="n2"){
        $sent_nr_2 = $value;
        }
}
else if($update_number == 3)
{
        if($key =="n3"){
        $sent_nr_3 = $value;
        }
}
else if($update_number == 4)
{
        if($key =="n4"){
        $sent_nr_4 = $value;
```

```
        }
}

else if($update_number == 5)
        {
        if($key =="b6"){
        $sent_bool_1 = $value;
        }
        if($key =="b7"){
        $sent_bool_2 = $value;
        }
        if($key =="b8"){
        $sent_bool_3 = $value;
        }
}
}//End of foreach


include("database_connect.php");       //We include the database_connect.php
which has the data for the connection to the database


// Check  the connection
if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//Now we update the values in database
if($update_number == 1)        //If the received data is for SENT_NUMBER_1,
we update that value
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_1 =
$sent_nr_1 WHERE id=$unit AND PASSWORD=$pass");
        }
else if($update_number == 2)   //The same and so on...
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_2 =
$sent_nr_2 WHERE id=$unit AND PASSWORD=$pass");       ;
        }
else if($update_number == 3)
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_3 =
$sent_nr_3 WHERE id=$unit AND PASSWORD=$pass");       ;
        }
else if($update_number == 4)
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_4 =
$sent_nr_4 WHERE id=$unit AND PASSWORD=$pass");       ;
        }

else if($update_number == 5)
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_BOOL_1 =
$sent_bool_1, SENT_BOOL_2 = $sent_bool_2, SENT_BOOL_3 = $sent_bool_3
                WHERE id=$unit AND PASSWORD=$pass");   ;
        }
```

```
//In case that you need the time from the internet, use this line
date_default_timezone_set('UTC');
$t1 = date("gi");      //This will return 1:23 as 123

//Get all the values form the table on the database
$result = mysqli_query($con,"SELECT * FROM ESPtable2");      //table select
is ESPtable2, must be the same on yor database

//Loop through the table and filter out data for this unit id equal to the
one taht we've received.
while($row = mysqli_fetch_array($result)) {
if($row['id'] == $unit){

                //We update the values for the boolean and numebers we
receive from the Arduino, then we echo the boolean
                //and numbers and the text from the database back to the
Arduino
                $b1 = $row['RECEIVED_BOOL1'];
                $b2 = $row['RECEIVED_BOOL2'];
                $b3 = $row['RECEIVED_BOOL3'];
                $b4 = $row['RECEIVED_BOOL4'];
                $b5 = $row['RECEIVED_BOOL5'];

                $n1 = $row['RECEIVED_NUM1'];
                $n2 = $row['RECEIVED_NUM2'];
                $n3 = $row['RECEIVED_NUM3'];
                $n4 = $row['RECEIVED_NUM4'];
                $n5 = $row['RECEIVED_NUM5'];

                $n6 = $row['TEXT_1'];

                //Next line will echo the data back to the Arduino
                echo "
_t1$t1##_b1$b1##_b2$b2##_b3$b3##_b4$b4##_b5$b5##_n1$n1##_n2$n2##_n3$n3##_n4
$n4##_n5$n5##_n6$n6##";

}



}// End of the while loop
?>

<?php
If you paste that link to your browser, it should update b1 value with this
TX.php file. Read more details below.
The ESP will send a link like the one above but with more than just b1. It
will have b1, b2, etc...
*/

//We loop through and grab variables from the received the URL
foreach($_REQUEST as $key => $value)  //Save the received value to the hey
variable. Save each cahracter after the "&"
{
```

```
        //Now we detect if we recheive the id, the password, unit, or a
value to update
if($key =="id"){
$unit = $value;
}
if($key =="pw"){
$pass = $value;
}
if($key =="un"){
$update_number = $value;
}

if($update_number == 1)
{
        if($key =="n1"){
                $sent_nr_1 = $value;
        }
}
else if($update_number == 2)
{
        if($key =="n2"){
        $sent_nr_2 = $value;
        }
}
else if($update_number == 3)
{
        if($key =="n3"){
        $sent_nr_3 = $value;
        }
}
else if($update_number == 4)
{
        if($key =="n4"){
        $sent_nr_4 = $value;
        }
}

else if($update_number == 5)
        {
        if($key =="b6"){
        $sent_bool_1 = $value;
        }
        if($key =="b7"){
        $sent_bool_2 = $value;
        }
        if($key =="b8"){
        $sent_bool_3 = $value;
        }
}
}//End of foreach


include("database_connect.php");       //We include the database_connect.php
which has the data for the connection to the database


// Check  the connection
```

```
if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//Now we update the values in database
if($update_number == 1)        //If the received data is for SENT_NUMBER_1,
we update that value
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_1 =
$sent_nr_1 WHERE id=$unit AND PASSWORD=$pass");
        }
else if($update_number == 2)   //The same and so on...
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_2 =
$sent_nr_2 WHERE id=$unit AND PASSWORD=$pass");       ;
        }
else if($update_number == 3)
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_3 =
$sent_nr_3 WHERE id=$unit AND PASSWORD=$pass");       ;
        }
else if($update_number == 4)
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_NUMBER_4 =
$sent_nr_4 WHERE id=$unit AND PASSWORD=$pass");       ;
        }

else if($update_number == 5)
        {
                mysqli_query($con,"UPDATE ESPtable2 SET SENT_BOOL_1 =
$sent_bool_1, SENT_BOOL_2 = $sent_bool_2, SENT_BOOL_3 = $sent_bool_3
                WHERE id=$unit AND PASSWORD=$pass");  ;
        }



//In case that you need the time from the internet, use this line
date_default_timezone_set('UTC');
$t1 = date("gi");      //This will return 1:23 as 123

//Get all the values form the table on the database
$result = mysqli_query($con,"SELECT * FROM ESPtable2");      //table select
is ESPtable2, must be the same on yor database

//Loop through the table and filter out data for this unit id equal to the
one taht we've received.
while($row = mysqli_fetch_array($result)) {
if($row['id'] == $unit){

                //We update the values for the boolean and numebers we
receive from the Arduino, then we echo the boolean
                //and numbers and the text from the database back to the
Arduino
                $b1 = $row['RECEIVED_BOOL1'];
                $b2 = $row['RECEIVED_BOOL2'];
                $b3 = $row['RECEIVED_BOOL3'];
                $b4 = $row['RECEIVED_BOOL4'];
```

```
                $b5 = $row['RECEIVED_BOOL5'];

                $n1 = $row['RECEIVED_NUM1'];
                $n2 = $row['RECEIVED_NUM2'];
                $n3 = $row['RECEIVED_NUM3'];
                $n4 = $row['RECEIVED_NUM4'];
                $n5 = $row['RECEIVED_NUM5'];

                $n6 = $row['TEXT_1'];

                //Next line will echo the data back to the Arduino
                echo "
_t1$t1##_b1$b1##_b2$b2##_b3$b3##_b4$b4##_b5$b5##_n1$n1##_n2$n2##_n3$n3##_n4
$n4##_n5$n5##_n6$n6##";

}



}// End of the while loop
?>
```

11.2.4.1.4 CODE 3

```
<?php
//This file will get the values when you click any of the ON/OFF buttons or
change buttons on the index.php file
//We get that value and send it to the datapase table and by that update
the values
$value = $_POST['value'];              //Get the value
$unit = $_POST['unit'];                        //Get the id if the unit
where we want to update the value
$column = $_POST['column'];            //Which coulumn of the database,
could be the RECEIVED_BOOL1, etc...

//connect to the database
include("database_connect.php"); //We include the database_connect.php
which has the data for the connection to the database

// Check the connection
if (mysqli_connect_errno()) {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

//Now update the value sent from the post (ON/OFF, change or send button)
mysqli_query($con,"UPDATE ESPtable2 SET $column = '{$value}'
WHERE id=$unit");

//go back to the interface
header("location: index.php");
?>
```

11.2.4.2 SPREADSHEET FOR DATA COLLECTION CODE

```
function doGet(e) {
  Logger.log( JSON.stringify(e) );  // view parameters
  var result = 'Ok'; // assume success
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
```

```
  }
  else {
    var sheet_id = '1fAW-WAu-lRaT28DQSrIhMrFxW3-LFiHu42EQR5wUw1s';
        // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet();
        // get Active sheet
    var newRow = sheet.getLastRow() + 1;

    var rowData = [];
    d=new Date();
    rowData[0] = d; // Timestamp in column A
    rowData[1] = d.toLocaleTimeString(); // Timestamp in column A

    for (var param in e.parameter) {
      Logger.log('In for loop, param=' + param);
      var value = stripQuotes(e.parameter[param]);
      Logger.log(param + ':' + e.parameter[param]);
      switch (param) {
        case 'value1': //Parameter 1, It has to be updated in Column in
Sheets in the code, orderwise
          rowData[2] = value; //Value in column A
          result = 'Written on column A';
          break;
        case 'value2': //Parameter 2, It has to be updated in Column in
Sheets in the code, orderwise
          rowData[3] = value; //Value in column B
          result += ' Written on column B';
          break;
        case 'value3': //Parameter 3, It has to be updated in Column in
Sheets in the code, orderwise
          rowData[4] = value; //Value in column C
          result += ' Written on column C';
          break;
          case 'value4': //Parameter 4, It has to be updated in Column in
Sheets in the code, orderwise
          rowData[5] = value; //Value in column E
          result += ' Written on column D';
          break;
          case 'value5': //Parameter 5, It has to be updated in Column in
Sheets in the code, orderwise
          rowData[6] = value; //Value in column E
          result += ' Written on column E';
          break;
        default:
          result = "unsupported parameter";
      }
    }
    Logger.log(JSON.stringify(rowData));
    // Write new row below
    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
    newRange.setValues([rowData]);
  }
  // Return result of operation
  return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
  return value.replace(/^["']|['"]$/g, "");
```

```
}
```

11.2.4.3 SPREADSHEET FOR THE CALCULATION OF THE POSITION OF THE SUN CODE

```
var ss =
SpreadsheetApp.openById('1g75Y8BZ386HefE1u6Tbkg5Fr1V0LoHuQvh8PYUYIn2Y');
var sheet = ss.getSheetByName('output');

function doGet(e){

  var read = e.parameter.read;

  if (read !== undefined){
    return
ContentService.createTextOutput(sheet.getRange('C1').getValue());
  }
  }
```

11.3 SPECIFICATION SHEET OF ST2408PH

Teknisk datablad

Solfølgertårn for nordiske forhold

Suntrack Nordic AS leverer solfølgertårn tilpasset nordiske forhold. Solid stålkonstruksjon, få bevegelige deler og store utslagsvinkler gir optimal funksjonalitet.

I norden er det store forskjeller på solvinkel og solposisjon gjennom dagen og året. Om vinteren er solen bare så vidt oppe over horisonten mens på sommeren kan den være oppe neste hele døgnet (midnattsol). Denne variasjonen registreres av det GPS baserte styresystemet og to-akse motorer sørger for at solcellepanelene er rettet direkte mot solen hele dagen for best mulig utnyttelse av tilgjengelig solenergi. Totalrotasjon er opptil 300 grader. Medfølgende vindsensor sørger for at panelene legges horisontalt ved vindstyrke over 8m/s.

Solfølgertårn krever lite vedlikehold, kun kontroll av bolter og smøring av bevegelige deler, samt normalt vedlikehold i forhold til utvendige stålkonstruksjoner.

Suntrack Nordic AS kan levere solfølgertårn for paneler fra 1kW til 12kW.

Suntrack artikkel nummer	ST2204PH	ST2408PH	ST2612PH	ST3618PH
Plass for solcellepaneler (antall)	4	8	12	18
Solcellepanel størrelse (ca, mm) - 320Wp pr. panel	1600x1000	1600x1000	1600x1000	1600x1000
Panel rader	2	2	2	3
Panel kolonner	2	4	6	6
Solfølgertårn total høyde med paneler (mm)	4060	4060	4060	5500
Solfølgertårn total bredde med paneler (mm)	2100	4100	6100	6100
Solfølgertårn, total vekt (kg)	290	370	550	850
Fundament innfesting - boltsirkel (mm)	6 x 20 x 276	6 x 20 x 276	6 x 20 x 276	8 x 20 x 400
Fundamentrør for innstøping – lengde (mm)	1700	2000	2500	Boltsirkel
Horisontal solfølger vinkel (grader fra sør)	+/- 150	+/- 150	+/- 150	+/- 150
Vertikal solfølger vinkel (grader)	20 - 90	20 - 90	20 - 90	20 - 90
Sikkerhetshastighet for vind – panel legges flat (m/s)	8	8	8	8
Nødstopp	Ja	Ja	Ja	Ja
Vind sensor	Ja	Ja	Ja	Ja
Helningssensor	Ja	Ja	Ja	Ja
GPS posisjonskontroll	Ja	Ja	Ja	Ja
Styrepanel, separat tilførsel	230V/10A	230V/10A	230V/10A	230V/10A
Samsvarserklæring / CE (EN60204/55014)	Ja	Ja	Ja	Ja

Ta kontakt i dag for tilbud og løsning tilpasset dine forhold og behov.

Suntrack Nordic AS
Marumveien 70
3231 Sandefjord, Norway

Org. nr.: NO 922 183 627 MVA
Bank: 6010.05.18468

Telefon: +47 90 10 19 20
www.suntracknordic.com
sales@suntracknordic.com