This Master's Thesis is intended to develop a characterization equipment for the characterization of Aerospace Photovoltaic Panels in order to obtain higher TRL levels. Throughout this thesis, the analysis, design and implementation of the client's requirements are detailed. The system includes a Photovoltaic measurement system as well as the design and manufacturing of a low cost Thermal and Vacuum Chamber able to recreate the harshest conditions in the space environment.

Juan Manuel López Torralba is a Telecommunication engineer from Granada, Spain. He was born in January 22, 1994. This work completes the Master in Telecommunication Enginering from the University of Granada after have succesfully completed the Bachelor's Degree in 2016, with a specialization in Telecommunication System.

Andrés María Roldán Aranda is the academic head of the present project, and the student's tutor. He is professor in the Departament of Electronics and Computers Technologies

Copy for the student /Copia para el alumno

MASTER'S THESIS

Characterization Equipment of Aerospace Photovoltaic Panels

Juan Manuel López Torralba

TELECOMMUNICATION ENGINEERING

17/18

UNIVERSITY OF GRANADA

Master of Telecommunication Engineering

Master's Thesis

Characterization Equipment of Aerospace Photovoltaic Panels

Juan Manuel López Torralba

2017/2018

Tutor: Andrés María Roldán Aranda

MASTER OF
TELECOMMUNICATION ENGINEERING

MASTER THESIS

"Characterization Equipment of Aerospace Photovoltaic Panels"

ACADEMIC COURSE: 2017/2018

Juan Manuel López Torralba

MASTER OF TELECOMMUNICATION ENGINEERING

"Characterization Equipment of Aerospace Photovoltaic Panels"

AUTHOR:

Juan Manuel López Torralba

SUPERVISED BY:

Andrés María Roldán Aranda

DEPARTMENT:

Electronics and Computers Technologies

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Máster de D. Juan Manuel López Torralba,

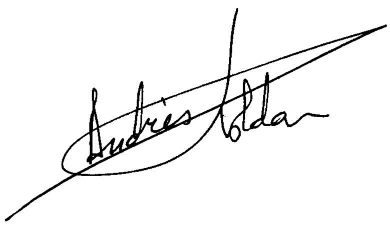
Informa:

que el presente trabajo, titulado:

“Characterization Equipment of Aerospace Photovoltaic Panels”

ha sido realizado y redactado por el mencionado alumno bajo nuestra dirección, y con esta fecha autorizo a su presentación.

Granada, a 12 de Septiembre de 2018



Fdo. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 12 de Septiembre de 2018

Fdo. Juan Manuel López Torralba

Fdo. Andrés María Roldán Aranda

Equipo de Caracterización de Paneles Fotovoltaicos Aerospaciales

Juan Manuel López Torralba

PALABRAS CLAVE:

Paneles Fotovoltaicos, Equipos de Instrumentación, Curvas de caracterización I-V, Cámara Térmica de Vacío, Alto Vacío, Bajo Coste, Componentes validados para uso espacial, TRL, Solidworks, MATLAB, GUIDE, SCPI, electrónica.

RESUMEN:

El propósito principal de este trabajo es desarollar un sistema de caracterización de paneles solares aerospaciales que permita obtener un mayor TRL y afrontar con mayores garantías las certificaciones necesarias para poner en orbita un componente electrónico. Estas certificaciones se realizan para asegurar que dicho componente soportará las adversas condiciones de vacío y temperatura que tienen lugar en el espacio. El sistema desarollado esta compuesto por un subsistema de medida de paneles fotovoltaicos y una cámara TVAC de bajo coste diseñada y manufacturada de forma casera. Además, el sistema es controlado por medio de una interfaz de usuario encargada tanto de la configuración de los equipos como de la representación y el almacenamiento de los datos obtenidos por medio de las pruebas realizadas.

Desde el comienzo del proyecto se sigue una metodología orientada tanto al producto como también a la situación del mercado, lo cual implica que cada decisión es tomada tras analizar todos los factores influyentes. Esos factores pueden ser tanto técnicos o ingenieriles como económicos o disponibilidad comercial. De esta forma, el proyecto se desarrolla a partir de los requrimientos técnicos facilitados por un cliente hipotético, a partir de los cuales, se obtienen los requerimientos funcionales de ingeniería. Posteriormente, se realiza un exhaustivo proceso de análisis, diseño e implementación del producto.

Este proyecto se presenta como Trabajo Fin de Máster de la titulación de Máster de Ingeniería de Telecomunicación de la Universidad de Granada. Una idea interesante es afrontar el proyecto con un enfoque multidisciplinar en el cual el alumno es capaz de demostrar no solo los conceptos aprendidos durante esta etapa académica, sino también, los aprendidos en su especialidad, durante el Grado de Telecomunicación, y los desarrollados durante la realización de este mismo proyecto.

Varias y distintas tecnologías han sido empleadas durante este proyecto con el fin de

aportar un valor añadido al mismo. Algunas de estas tecnologías son el control y configuracion de equipos electónicos, el uso de software de diseño CAD/CAM y otras herramientas de software y programación. Además, el sistema de control se presenta por medio de una GUI la cual se centra en la funcionalidad, su facilidad de uso y la experienca final del usuario. El objetivo es que este sistema se consolide como una herramienta crucial para los futuros y exitosos retos que aguardan al equipo GranaSAT.

Este proyecto culmina con la obtención de un sistema de caracterización completo, el cual garantiza su cumplimiento de los requerimientos técnicos del cliente y los requisitos funcionales de ingeniería derivados de estos.

Characterization Equipment of Aerospace Photovoltaic Panels

Juan Manuel López Torralba

KEYWORDS:

PV Panels, Instrumentation Equipment, I-V Characterization Curve,TVAC Chamber, High Vacuum, Low cost, Space-worthy components, Technology Readiness level, Solidworks, MATLAB, GUIDE, SCPI, electronics.

ABSTRACT:

The main purpose of this project is the development of a system able to characterize solar panels in order to obtain higher TRL levels. This system consist of a photovoltaic measurement system and a low cost home-made Thermal and Vacuum Chamber. Furthermore, the system is managed by a graphical user interface where the obtained curves and data can also be visualized.

From the beginning of the project, the methodology sought a product and market oriented philosophy. Each decision concerning the project's development, without exception, is taken considering all the criteria involves, that is, engineering criteria as well as the pricing and the market availability. In this manner, the project arises from requirements which might be given by an assumed and imaginary client. Technical requirements emerge then from those previous requirements. Afterwards, a complete process involving analysis, design and implementation is carried out.

This project is presented as a Master Thesis within the Master in Telecommunication Engineering of the University of Granada. Therefore, it is captivating to not merely focus on a specialised but also a multidisciplinary approach in which the student uses the competences acquired during the academic period as well as the concepts learn by the realization of this work.

In order to enhance the significance of the project, several and distinct technologies are used. Some of the technologies used during the project are instrumentation and control, CAD/CAM design and software design. Moreover, the management system is presented via GUI which focus on the user experience and the ease of use. This system is intended to be a crucial tool for the new projects and challenges the GranaSAT Team will face.

The result of this work concludes obtaining a comprehensive measurement system which meets the Functional and the Client's Technical Requirements previously specified.

"We can be heroes
just for one day"
David Bowie

Agradecimientos

En primer lugar quiero agradecer a mi familia, por todas las cosas que han hecho por mí durante todos estos años, por quererme siempre como soy y por ayudarme siempre que lo he necesitado. A HuiWen, por estar siempre a mi lado y apoyarme en buenos y malos momentos.

A mis compañeros y amigos, por hacer más llevadera esta estapa.

Finalmente, no puedo más que agradecer a mis compañeros de GranaSAT su colaboración durante estos dos años y, especialmente, a Andrés María Roldán Aranda, mi tutor, por sus enseñanzas y el tiempo invertido en hacernos mejores ingenieros.

INDEX

FIGURE INDEX

TABLE INDEX

GLOSARY

Parque Tecnológico de la Salud Located in Granada, it is a space of teaching, research and business excellence.

AM0 The spectrum outside the atmosphere, approximated by the 5,800 K black body, meaning "zero atmospheres".

AM1.5D The direct standard spectrum includes radiation coming from singular small surroundings of the sun (5.8° aperture angle) and it is projected orthogonally onto the cell. Calculations with the same atmospheric conditions as for the 1.5g spectrum then yield a total irradiance of 90 mW/cm^2 [16].

AM1.5G The global standard spectrum. It is calculated from the reference AM0 Spectrum. The AM is 1.5, the normal vector formed by the sun and the cell is 11.2°.

CubeSAT Miniaturized satellite for space research made up of multiples of 10×10×10 cm cubic units with no more than 1.33 kg per unit..

Datalogger Electronic device that records data over time from sensors or instruments.

GranaSAT GranaSAT is an academic project from the University of Granada consisting of the design and development of a picosatellite CubeSAT. Coordinated by the Professor Andrés María Roldán Aranda, GranaSAT is a multidisciplinary project with students from different degrees, where they can acquire and enlarge the necessary knowledge to front a real aerospace project. `http://granasat.ugr.es/`.

High Vacuum Pressure ranges from 10^{-6} to 10^{-8} Torr[21].

Kenny Purpose Interface Bus It is a framework for operating laboratory instruments that are controlled by GPIB or serial port connections to a computer. KPIB provides a unified interface for communicating with different instruments of the same type from different manufacturers.

Medium Vacuum Pressure ranges from 10^{-3} to 10^{-5} Torr[21].

NW It is an ISO-specified vacuum flange fitting. It is one of the industry standard for vacuum components and range..

O-ring Also known as toric joint. It is a ring-shaped seal or washer of rubber, plastic or metal, designed to avoid leaks during assembly between two or more parts.

Rough Vacuum Also called Low Vacuum, its pressure ranges from 1 to 10^{-3} Torr[21].

RS-232 Recommended Standard 232. It was introduced in 1960 for serial communication transmission of data.

Solidworks It is a solid modelling computer-aided design and computer-aided engineering software published by *Dassault Systèmes*.

ACRONYMS

AISI American Iron and Steel Institute.

AM Air Mass.

API Application Programming Interface.

ASCII American Standard Code for Information Interchange.

ASTM American Society of Testing and Materials.

CAD Computer Aided Design.

CAE Computer Aided Engineering.

CAM Computer Aided Manufacturing.

CC Constant Current.

CIC Controller In Charge.

COTS Commercial off the shell.

CR Constant Resistance.

CV Constant Voltage.

DAV Data Valid.

DC Direct Current.

DUT Device Under Test.

eload Electronic Load.

EPS Electric Power System.

ESA European Space Agency.

GaAs Gallium Arsenide.

Ge Germanium.

GPIB General Purpose Interface Bus.

GUI Graphical User Inteface.

GUIDE Graphical User Interface Development Environment.

HMI Metal Halide Arc Lamps.

HP-IB Hewlett Packard Interface Bus.

I-V Current - Voltage.

I/O Input/Output.

IEC International Electrotechnical Commission.

IOD In Orbit Demonstration.

IR Infra-Red.

Isc Short Circuit Current.

ISO International Organization for Standardization.

JIS Japanese Industrial Standard.

KF Klein Flange.

LAN Local Area Network.

LED Light-Emitting Diode.

Liquid Nitrogen.

LTI Long-Term Instability.

MATLAB MATrix LABoratory.

MLI Multi Layer Insulation.

MPP Maximum Power Point.

NDAC No Data Accepted.

NI National Instruments.

NRFD Not Ready For Data.

P-V Power - Voltage.

PC Personal Computer.

PV Photovoltaic.

QTH Quartz Tungsten Halogen Lamps.

SCPI Standard Commands for Programmable Instruments.

Si Silicio.

SMU Source Meter Unit.

SS Stainless Steel.

STI Short-Term Instability.

TRL Technology Readiness Level.

TVAC Thermal and Vacuum.

USB Universal Serial Bus.

Voc Open Circuit Voltage.

CHAPTER

1 INTRODUCTION

This Master Thesis is presented as a final compilation of the entire knowledge acquired throughout the Master's Degree in Telecommunication Engineering and, specially, all along the project fulfilment. The main purpose of the project is to develop a measurement system of aerospace PV panels in a low cost home-made TVAC Chamber in order to obtain higher TRL levels.

The project arises as a solution to the previous need for exhaustively testing the satellite GranaSAT-I in the space environment before launching. However, this task is arduous enough for just one engineer and goes beyond the scopes of the project in terms of time. Therefore, this project focus on the development of the PV measurement system. The PV measurement system consists of all the electronic instrumentation control algorithms needed to measure and characterize aerospace photovoltaic panels. Furthermore, for Mechanical test purposes, the design and manufacturing of the TVAC Chamber is undertaken. This TVAC Chamber is able to reach extremely low temperatures and to deal with High Vacuum conditions.

Before performing aerospace test the use of a previously calibrated solar simulator is required in order to check the proper functioning of the PV panels. The calibration procedure of the solar simulator will be described in the following chapters. In the future, all the features and functionalities developed in this project must be incorporated in a whole measurement system which is being expanded by the GranaSAT Aerospace Team.

The PV measurement system will allow the engineering team to obtain crucial data from the tested PV panels and interact with them for further analysis according to their needs.

1

Besides, this data will be useful to prevent anomalous behaviours of the panels or even avoid critical failures before the lift off.

In summary, this project makes use of both software and hardware technologies to solve a specific problem, the measurement and characterization of photovoltaic solar panels for aerospace purposes. In addition, the technology here developed will act as the fundamental basis of the future aerospace tests performed in the GranaSAT Aerospace Group.

The System Development Process consists of Functional Requirements Definition, System Analysis, System Design, Subsystems Testing and Integrated System Verification stages. The block diagram of the figure 1.1 shows the flow design from the beginning, the client's proposal, to the obtaining of the final PV Measurements System. The plan of action defined in the figure 1.4 arose as consequence of the previously mentioned design process.

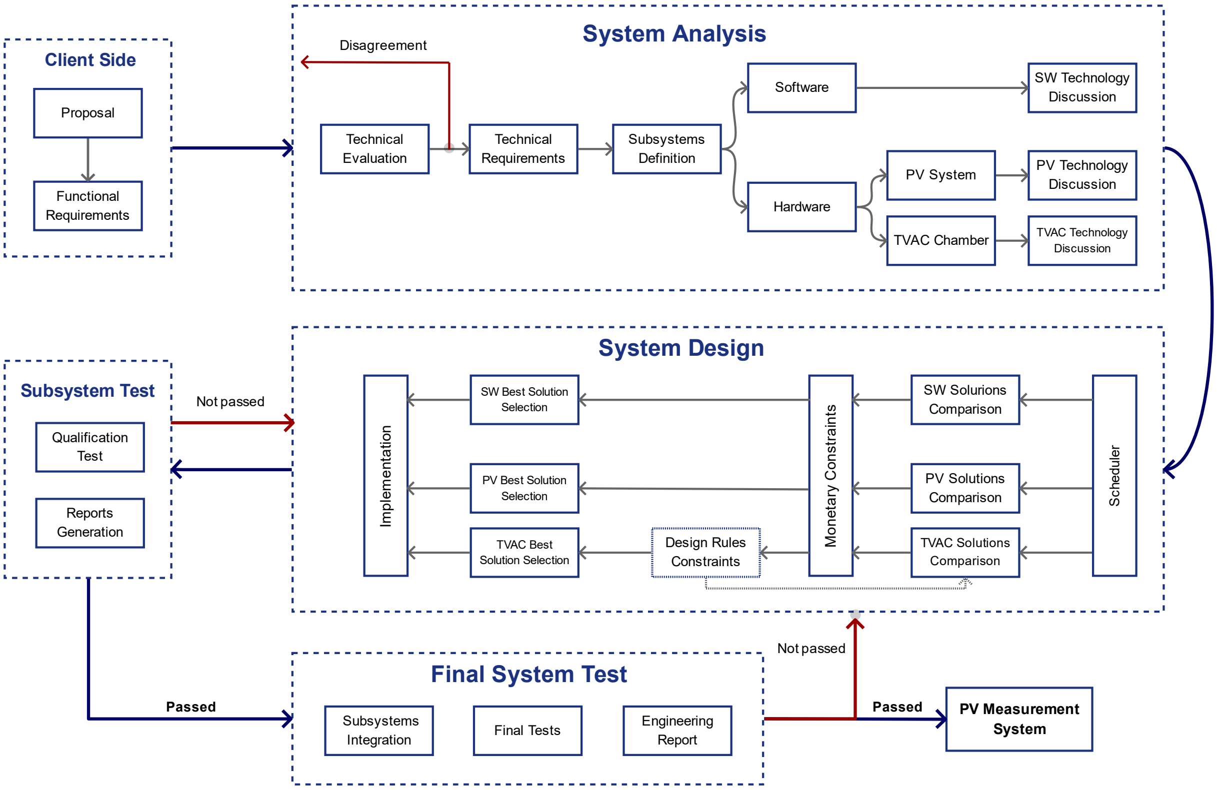


Figure 1.1 – *Design Process*

This Master Thesis has been developed with the cooperation of the GranaSAT Team. The GranaSAT Project arises as a multidisciplinary team built on people who are committed to acquire knowledge from the Aerospace and Electronics fields. The head of GranaSAT is the professor Andrés María Roldán Aranda. In the very beginning the project efforts focused on the building of a CubeSAT, today its ambition goes beyond what was expected. Nowadays, the GranaSAT Team can be located at *Parque Tecnológico de la Salud* surrounded by a lot of companies forming the a great technological space for development and learning. In the figure 1.2 the current logo can be observed.

Figure 1.2 – *GranaSAT logo*

1

1.1 State of art

All the spacecraft subsystems must be exhaustively tested before launch in order to obtain space-worthy solutions. These tests are carried out in dedicated equipment known as TVAC Chambers. A TVAC aims to recreate as closely as possible the space conditions which a satellite will be exposed to. An environmental chamber is the closest equipment than can be found in most electronic laboratories. Nevertheless, those climatic chambers are not able to perform test in vacuum, nor below room temperature.

ESA classifies the technical maturity of spacecraft components on a scale ranging from 1 to 9, denominated as TRL. In figure 1.3 the NASA TRL definition can be found. Notice that the ESA's definition of the TRL is quite similar.

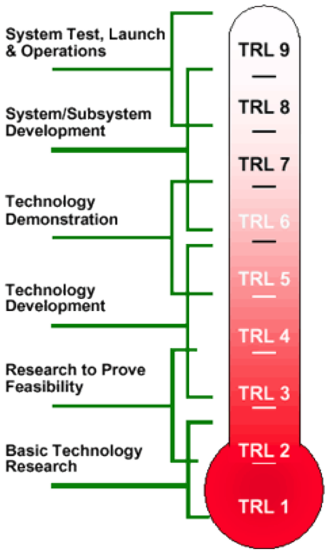


Figure 1.3 – *NASA Technology Readiness Level*

The reason why we decided to address this project was the technical difficulty and the high price involved in the realization of these tests. It exists in the market several solutions for the TVAC Chamber from different companies such as *Abbess*, *The Vacuum Projects SLU* or *Pfeiffer Vacuum Components & Solutions GmbH*, but at a costly price. Designing and manufacturing a low-cost TVAC Chamber from scratch allows us to save enough money and give us room for the acquisition of the electronic equipment needed for accomplishing the PV Panel qualification tests. This testing facility system will successfully characterize PV Panels and improve its TRL levels.

At the end of the project, a fully low-cost PV Panel measurement system will have been developed. The use of this system in the GranaSAT Aerospace Team will introduce students in electronics instrumentation and vacuum technologies, improving their skills in aerospace measuring procedures.

1.1.1 Cubesats

CubeSats are nano satellites with a mass below 10 kilograms which are commonly developed for the IOD of miniaturised technologies. CubeSats have already proved their worth as an excellent educational tool. Therefore, the CubeSats are experiencing a positive trend within the engineering companies product catalogue.

These pico-satellites follow the well-known CubeSat Standard, which specifies the configuration and the outer dimension of 1U CubeSat as a cubic unit of 10x10x10 cm. The maximum weight allowed for 1U Cubesat is 1.33 kg[37]. As many units as needed can be combined, depending on the mission requirements, promoting a high modularity and system integration. The CubeSat standard also encourages the use of COTS products as satellite subsystems available from different suppliers. This implies a reduction of the time expended in the design and implementation processes, which means low cost development.

A CubeSat is composed of different subsystems based on their range of capabilities [5]:

- Structure: It is the mechanical structure of the satellite and provides physical place to systems.
- Attitude Determination and Control System: It contains gyroscopes, magnetometers and sun sensors for determining and modifying the spacecraft's attitude and orbit if needed.
- Communication System: It provides bidirectional communication between Earth and satellite.
- Command and Data handling System: It is the computer of the satellite.
- Thermal System: It controls the working temperature of the satellite in an optimal way.
- Electrical Power System: It is responsible for power harvesting, storage, conditioning, control and distribution to the rest of systems. It must satisfy the energy needs of the whole satellite in an uninterrupted manner, both in light conditions and in eclipses[5].
- Payload: It is composed of IOD technologies.

Designing devices for their use in the space environment is not a straightforward task. A satellite must deal with a hostile environment in which the heavy conditions leads to a progressive deterioration of the electrical and mechanical systems [5]. A widely known fact is the high probability of the in-orbit mission failures for CubeSats. This is accepted by the CubeSat community due to the soaring expenses reduction contrasting with the bigger and better commercial satellites.

Some studies determines that after 90 days in the space environment, the EPS is the responsible of causing a critical failure of the spacecraft in a 36 % of the cases[4].

1

1.1.2 PV Cell Degradation

PV panels are composed of a set of cells in parallel, which also consist of multiple cells in series. These cells are actually semiconductors that are responsible for converting incident solar energy into electric current. These incident photons generate minority carriers on the front surface of the cell which diffuses into the cell junction to produce a photo current at the output. In space, the systems that make up the satellite are continuously exposed to radiation, causing imperfections that prevent these carriers from reaching the junction between cells and thus causing a decrease in the generated current. This lessening in the maximum power extracted from the solar panel is due to the decrease in the short-circuit current and the degradation of the junction between cells[28]. The main parameters used for monitoring the performance and indicating the deterioration of PV panels are the Isc and the MPP.

Solar panels are one of less shielded systems of a satellite because of its functions and operating principles. There are several ways to mitigate the effects of high intensity radiation on solar cells. The simplest is to use a thin, transparent glass film (coverglass) to protect the front surface of the cells and, to protect the back face, a substrate and the structure of the panel itself are used. The coverglass is coated with anti-reflective and conductive coatings, and bonded to the solar cell using a transparent adhesive. Low energy protons and electrons are absorbed by this coverglass as it causes its deceleration, preventing it from reaching the active part of the cell and causing some kind of deterioration[28][8]. The thickness of this coverglass is a crucial parameter in the design of the power stage of the missions. Thickness around 3 mills have the capacity to reject all the protons that strike with an energy lower than 2.8 MeV and electrons with energies lower than 200 keV. However, this protection causes a decrease in the specific power provided by the array of solar cells. One way to reduce this effect is to use arrays of concentration photovoltaic panels, assuming that these elements can withstand the high radiation in the environment. Concentration photovoltaic arrays have been proposed for outdoor planetary missions, electric solar propulsion missions and missions that operate in high radiation environments. These arrays are attractive for these missions because they have the potential to provide high specific power, greater radiation tolerance, and improved performance in LILT environments. A different approach is to develop cells that are more resistant to radiation[8].

The Isc and Voc degradation depends on the type of material used in the cell and its structure. As the incident flow increases, the degradation observed in Isc, Voc and MPP increases. A few decades ago, silicon cells were in the state of the art for space missions. Nowadays they are still operative in many satellites. However, today another type of configuration is used, due to the achievements of the GaAs systems and the use of semiconductors III-IV. The new III-V multi-junction cells have allowed a reduction in the size and mass of arrays of solar panels, in comparison with the Si cells previously used, maintaining comparable power levels[8].

1.2 Ambition

Telecommunication Engineering covers a wide range of subjects in which focus on and develop the acquired knowledge throughout the academic period. One topic I have always been interested is Aerospace electronics. Joining to the GranaSAT Team I wanted to getting up with the current state of the art and going a step beyond the education I had previously received. This work represents a major challenge given the combination of subjects such as instrumentation and control engineering, CAD/CAM software knowledge, programming languages as well as the experimental nature of the project.

Performing this Master Thesis is the icing on the cake of two absorbing and exciting years. In those years I have been involved in real issues which includes topics so far unexplored in the Master's Degree ranging from CAD/CAM design and manufacturing from scratch, requirement analysis, drawings generation, architecture design, product revisions, system integration as well as performing aerospace acceptance tests at laboratory environment.

Furthermore, being part of a project from zero to its culmination, going through all the stages of development, is an exhilarating experience that must be experienced.

The measurement system which arise from this Master Thesis is the result of a meticulous and passionate work which will be essential for the development of the GranaSAT Project. This Master Thesis project will conform the basis of the future Aerospace projects developed in GranaSAT, which is really stimulating.

1

1.3 Projects Goals and Objectives

The main objectives of this Master Thesis are the following:

- Studying and analysing the needs of the PV measurement system required.
- Deducing the main and secondary requirements of the system as well as the subsystem by which it is composed of, studying its viability, providing improvements and better possibilities.
- Performing an analysis of the current technologies and products on the market which might be used to solve our needs.
- Designing and manufacturing of a Thermal Vacuum Chamber for Thermal-Vacuum Testing.
- Generating several concept version of the TVAC Chamber, reasonably choosing the best one.
- Generating drawings for the TVAC Chamber to be checked before manufacturing.
- Performing validation and qualification test to the TVAC Chamber in order to determine its grade of success and functionality.
- Measuring the Solar Cell I-V Characteristic Curve of PV Panels.
- Implement the communication and control algorithms between the PV Panels and the electronic measurement equipment.
- Designing a GUI in MATLAB for controlling the electronic equipment and managing the measured data.
- Integration of the PV Panels measurement system with the TVAC Chamber.
- Performing a global validation test to check the global behaviour of the whole system.
- Acquiring mechanical design knowledge.
- Acquiring electronic instrumentation knowledge.
- Acquiring knowledge about Solar cell qualification tests.
- Approach the real engineering work and processes to the student.
- Demonstrate the knowledge acquired by the student all along the Master's Degree, strengthening the Bachelor's Degree acquired ones.
- Successfully overcome the Final Master Thesis subject.

1.4 Project Structure

This project is structured on 6 chapters and an addendum, each one forming a separate, yet complimentary description of the project. It is intended to describe not only the project, but also each one of the design and development stages of the system, strengthening the best meeting of the set goals and objectives.

These chapters are:

- Chapter 1: This chapter is an introduction of the objectives of the project and its motivation.
- Chapter 2: Summary of the system requirements definition from a client side point of view. An estimation of the project needs and scopes are obtained.
- Chapter 3: This project manage for the very first time crucial point such as the analysis of the client's requirement and the generation of technical requirements and constraints. The initial system block diagram can be obtained and its blocks can subsequently be analysed. Also, some concepts are presented in order to gain a major understanding of the systems.
- Chapter 4: This chapter deals with the system design. The distinct solutions analysed in the previous chapter are now implemented and adapted to current devices. It also specifies relevant design details in order to gain a major comprehension of the whole project.
- Chapter 5: This chapter introduce some validation test which will certify the correct functioning of the system. It will also prove the fulfilment of the client's requirement specified in Chapter 2.
- Chapter 6: Finally, the main conclusions extracted from the project and some future lines of research are presented.
- Addendum: The budget of the project ans derived costs are included as well as a picture documentation about the manufactured TVAC Chamber.

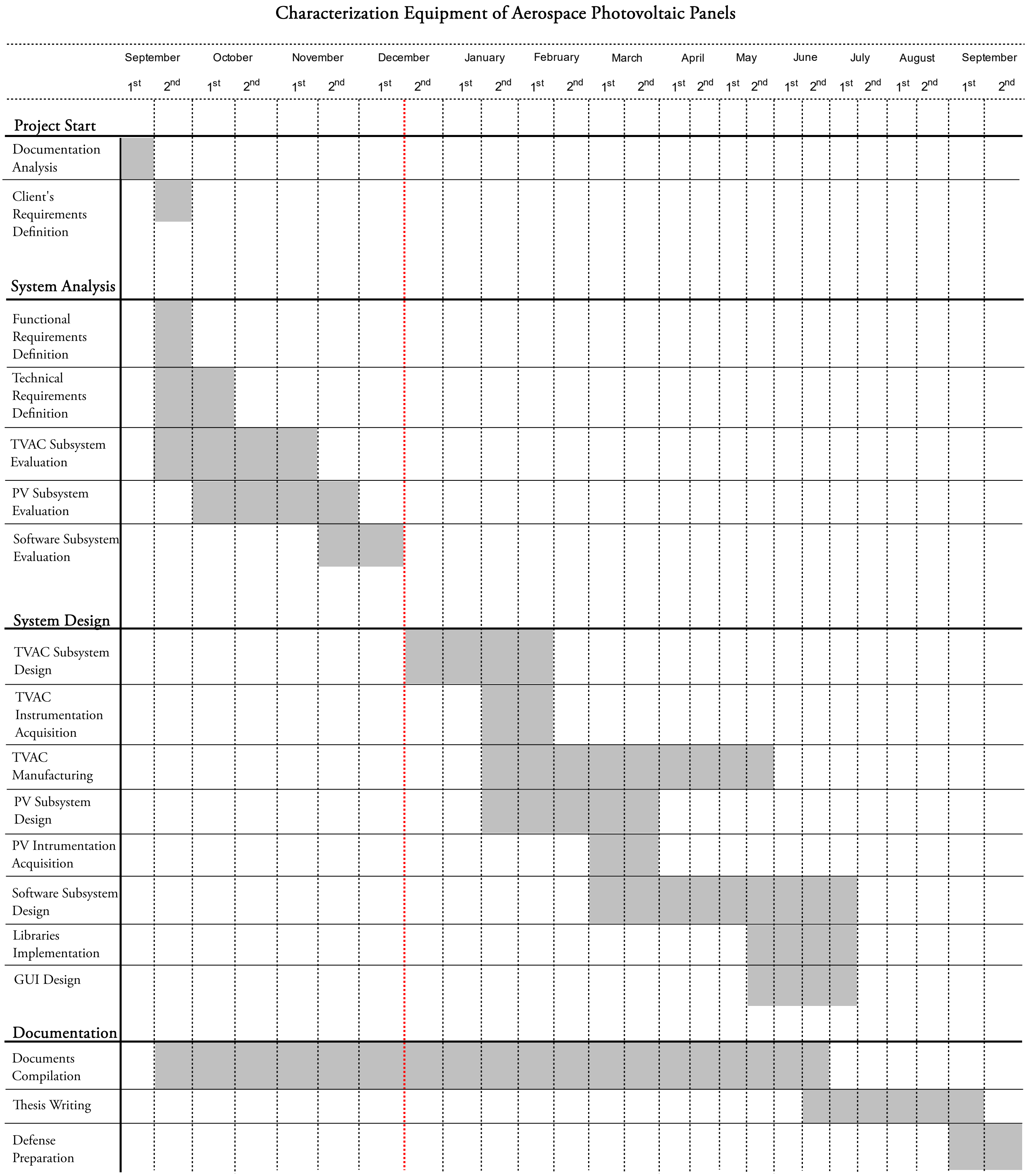


Figure 1.4 – *Gantt Chart of the Project*

CHAPTER

2

SYSTEM REQUIREMENTS DEFINITION

Making a terrific definition of the system requirements is essential for successfully develop any complex system. A bunch of system requirements will be defined in order to obtain an excellent product design. The technical requirements of the system will be extracted from it. In addition, monetary requirements will be stated.

2.1 Functional Requirement Specification

The functional requirements defines the implementation of the system from a client-side point of view. It states what the system is expected to do and how do it. Since the functional requirements are established from the client side, neither overly technical nor precise definition are required.

2.1.1 Hardware Requirements

1. To implement a system able to measure electrical data from solar panels.

2. To determine the I-V Characteristic Curve from solar panels.

3. To be able of simulating the Sun conditions in the tests.

4. To measure the light irradiance.
5. Designing and manufacturing of a Thermal Vacuum Chamber for Thermal-Vacuum Testing.
6. The TVAC must allow observing the aerospace test.
7. The TVAC must have two doors made from different materials.
8. Designing a Vacuum System for the TVAC Chamber.
9. The Vacuum System shall be able to decrease the inner pressure from ambient to Medium Vacuum and High Vacuum conditions.
10. The vacuum conditions must be maintained during the whole tests.
11. The pressure within the TVAC chamber will be measured with a vacuum transducer and controlled via RS-232.
12. Designing a Thermal System to recreate the extremely low temperatures in space.
13. The Thermal System must have total control of the temperatures inside the chamber during the thermal tests.
14. The TVAC Chamber shall have enough feed-through ports for data acquisition and instrumentation purposes.

2.1.2 Software Requirements

1. Designing a GUI in MATLAB for controlling the electronic equipment and managing the measured data.
2. To develop the necessary control algorithms for instrumentation equipment.
3. The GUI must be able to save the measured data.
4. The GUI shall storage the current session.
5. The GUI must save the input configuration parameters.
6. The GUI shall load the previous configuration parameters.

2.2 Monetary Requirements Specification

Concerning to the monetary requirement a low-cost philosophy is followed. The total cost of the design and manufacturing of the TVAC Chamber is expected to be less than 4000 € which is an affordable cost compared with the available market solutions. The complete system, including all the equipment, is required to be less than 10000 €.

CHAPTER

3 SYSTEM ANALYSIS

In this chapter, the technical requirements and constraints of the system will be set for the overall project. A top-down approach will be followed for analysing the system and gaining insights into its compositional subsystems. With a stepwise design a progressively reduction of the system abstraction level is sought. Starting with the big picture, the system's technical requirements will be defined based on the functional requirements given in the previous chapter.

The system analysis continues with the top level system's structure block diagram definition and discussion. Each subsystem is then refined in yet greater detail, breaking down from there into smaller segments. Thereafter, different technologies will be analysed and compared through which solve these requirements in order to select the best option for each subsystem. The Analysis is completed with the discussion of the possible solutions in the Chapter 4, System Design.

The System Design process consist of the implementation of the optimal solution for each subsystem according to the planning, monetary and mechanical constraints discussed in the Analysis Stage. This stage ends with a PV measurement system prototype and the manufacturing of the TVAC Chamber. Both subsystems must be tested and verified before and after their integration.

Once Testing and Verification phases have been carried out, if approved, our PV Measurement system will have been obtained and the project will be finished. If not approved, returning to the analysis stage is a must.

3.1 System Technical Requirements

The analysis of the Client's Technical Requirements will allow us to learn about the technologies available in the market, from which to develop the product. During the analysis stage it will be determined which tools are the most suitable or interesting, according to different points to be evaluated. Those point could be such as their fitting to the client's specifications, cost and awareness in the knowledge field.

According to the Client's Technical Requirements analysis, the main function of the desired product is to be able of measuring electrical characteristic of a solar panel inside a low-cost chamber in which perform aerospace test. Furthermore, the data obtained must be represented in the Graphical User Interface or GUI which will also allow the data management.

In order to meet those requirements, the system will consist of the subsystems shown in the figure 3.5.

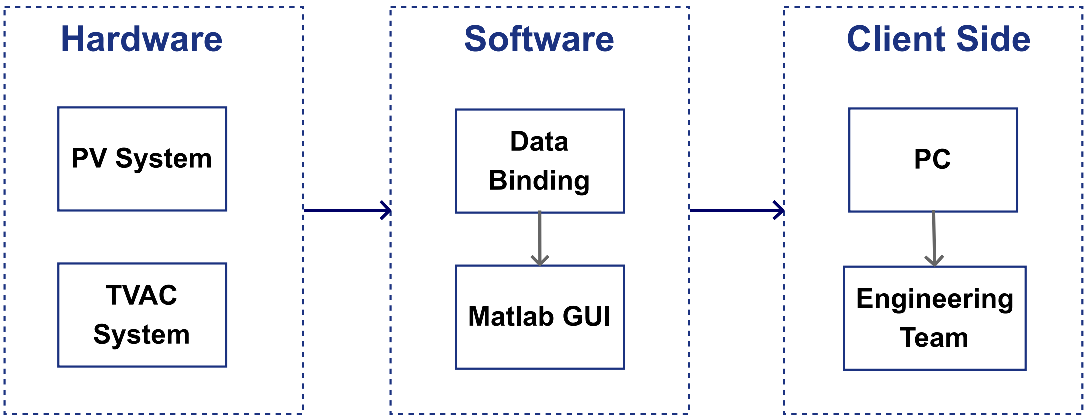


Figure 3.1 – *Top Level System Block Diagram*

As shown in the above figure, the system may consists on two disparate subsystems: Hardware & Software. The first one, the Hardware subsystem, is then divided into two new differentiated elements known as Photovoltaic and TVAC subsystems. As for the Software Subsystem, it might consists on the GUI and the instrumentation control algorithms. Finally, a specialist engineering team will be responsible for managing the entire system.

3.1.1 *Hardware*

The main requirement of the system is to include and develop the necessary equipment to be able of characterize aerospace photovoltaic panels of nano-satellites.

As said before, the hardware subsystem consists on the Photovoltaic and TVAC subsystems. These subsystem are clearly differentiated and are going to be analysed individually.

3.1.1.1 *Photovoltaic System*

As shown in the figure 3.2, the Photovoltaic System must consists on the following five components:

1. A **Solar Simulator** for testing the appropriate functioning of the solar PV cells in the space environment. It must to approximately simulate the Sun radiation flux in spectrum, intensity and uniformity [15]. It should ensure a trustworthy operation and straightforward maintenance. This equipment must be previously characterized. This characterization will consists on studying its intensity distribution, its solar irradiance and the stability of its light beam.

2. A **pyranometer** must be used to determine the proper operation of the solar panels. The pyranometer is used to effortlessly analysing the solar irradiance produced by the solar simulator. The data acquired from the pyranometer will be integrated in the GUI side.

3. The **Solar Panel** to be tested. In this project a 1-U CubeSAT photovoltaic panel will be used.

4. A **DC Programmable electronic** load for testing purposes. Using a electronic load instead of fixed resistor banks allows simpler and faster simulations. It must have communication interfaces such as GPIB, RS-232 or USB.

5. A **Power Supply** for performing the test and to supply the electrical energy to the system. It will also provide a boost voltage for the solar panel. Communication interfaces such as GPIB are required.

A cross-cutting requirement is to focus the development of the hardware subsystem in a low cost philosophy.

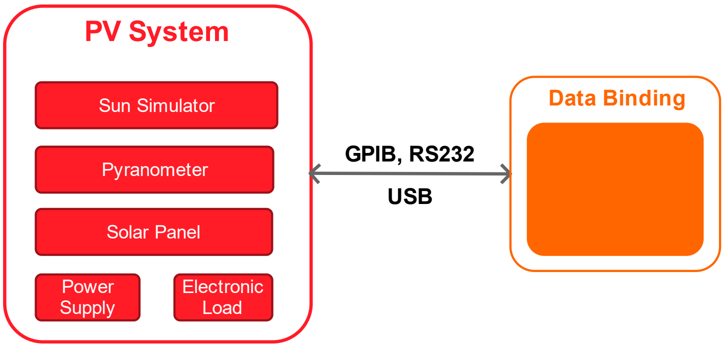


Figure 3.2 – *PV Subsystem Block Diagram*

3.1.1.2 *TVAC System*

As shown in the figure 3.3, the TVAC System consists on the following elements:

1. An **External Structure** able to deal with high vacuum conditions and extreme low temperatures. The structure must be accessible for manipulating several DUT and favours its maintenance.

2. An **Internal Structure** for cooling purposes. It should help to quickly decrease the inner temperature of the TVAC Chamber.

3. A **Vacuum System** for recreating the vacuum conditions available in the space environment. This system will be able to deal at least with conditions. Communication interfaces will be appreciated.

4. A **Thermal System** for recreating the harsh temperature conditions available in the space environment. The system must have total control of the temperatures inside the chamber during the thermal tests. The use of LN_2 for swiftly reduce the inner temperature is mandatory.

5. All the **Instrumentation Equipment** for monitoring the aerospace test. It must include multiple feed-through ports for instrumentation purposes and for the normal chamber operation. In addition, it is essential avoiding any pressure leakage or thermal change on the running tests. Finally, The data obtained must provide an adequate amount of information to the engineers for supervising the processes or even intervene in the ongoing operation if needed.

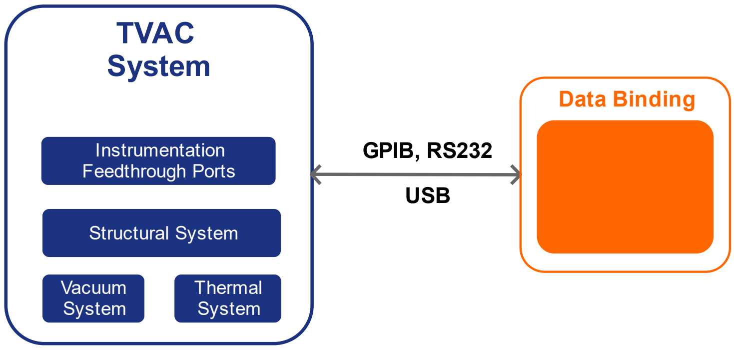


Figure 3.3 – *TVAC Subsystem Block Diagram*

3

3.1.2 Software

The essential requirement of the Software system is to be able of controlling the instrumentation equipment as well as managing the measured data from the performed aerospace tests. These data points must be presented to the user in a friendly and understandable manner.

The Software requirements are enumerated below:

1. To allow the connection between the instrumentation equipment of the Photovoltaic System and one PC. A GPIB interface will be used for the Electronic Load and the Power Supply. The pyranometer will be controlled via RS-232.
2. To develop the necessary API's for controlling the instrumentation equipment and integrate it in the same program.
3. To develop a MATLAB GUI for controlling all the processes.
4. The GUI must be able to receive and process the measured data for extracting the main parameters and save it in a computer.
5. The GUI shall storage the current session.
6. The GUI must save the input configuration parameters.
7. The GUI shall load the previous configuration parameters.

The software Subsystem is shown in the figure 3.4.

In the figure 3.5, a detailed block diagram of the system, drawn from the technical requirements mentioned in section 3.1, is shown.

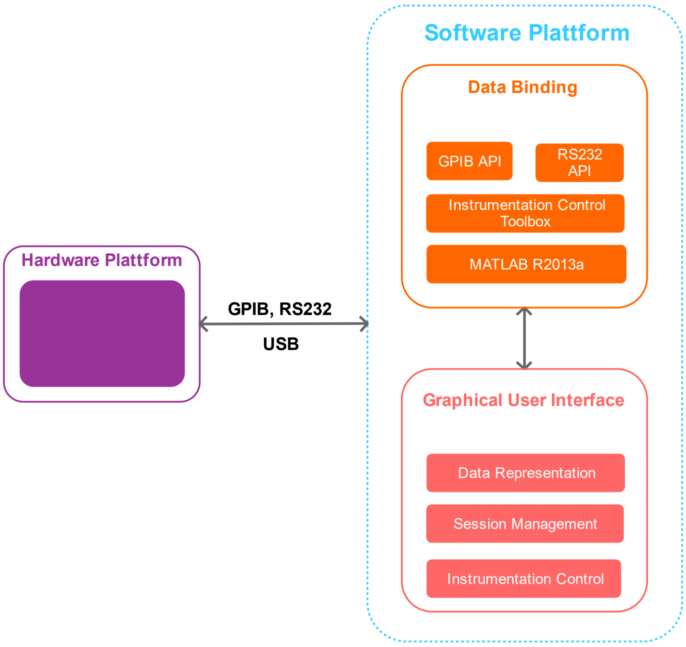


Figure 3.4 – *Software Subsystem Block Diagram*

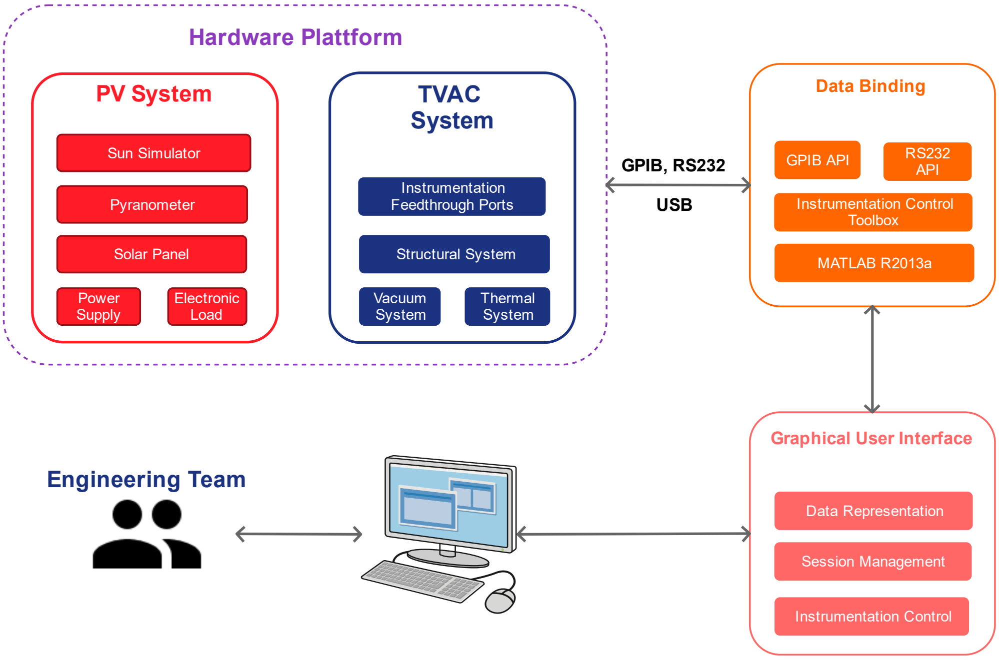


Figure 3.5 – *Top Level System Block Diagram*

3.2 Hardware Subsysem Analysis

As said in the section 3.1.1, the Hardware Subsystem is composed of PV and TVAC Systems. In this section, both subsystems are going to be examined. Therefore, different possibilities for each one will be studied and contrasted.

This subsystem must provide an optimal solution to the performing tests, recreating the harsh space environment conditions.

3.2.1 Photovoltaic System

The PV system is one of the mainstays of the Project since it will characterize the Solar Panels. As said in section 3.1.1.1, the Photovoltaic subsystem must provide the necessary equipment for performing successfully aerospace test. This subsystem is composed of the five elements that will be analysed on the following sections.

3.2.1.1 Solar Simulator

To be able to put a satellite in orbit, several meticulous tests must be done to ensure its adequate functioning in space conditions. Solar Simulators reproduce the conditions to which an in-orbit satellite will be exposed to. It is the equipment used for simulating the solar irradiance and the solar spectrum. It is one of the most acclaimed equipment for performing indoor solar PV qualification testing and other devices sensitive to sunlight under laboratory conditions. A solar simulator mostly contains a light source and the optics and filters for modifying the light beam [20][38].

The light emmitted from a solar simulator is characterized by three points: **spectral match** to sunlight, **temporal instability** of irradiance and **non-uniformity** of spectral irradiance[10]. Each dimension is classified by A, B or C levels. The solar simulator is then rated with the three letters which define each dimension in order of spectral match, spatial non-uniformity and temporal instability (for example: Class ABA).

The class of a solar simulator is referred to as the lowest level achieved across those three dimensions. Therefore, a solar simulator with a C level in spectral match, a B level in spatial uniformity and temporal instability would have a rating of C/B/B and could be considered as Class B. This could also lead the user to figure out that the solar simulator has class B ratings across all the dimensions. For that reason the manufacturer must clearly specify the class performance for each category [33].

The specifications for the various standards required for each class are defined in Table 3.1, 3.2, 3.3. Observe that there are key differences between the standards. LTI and STI are provided for the IEC standards while the ASTM Standard differentiates simulator classifications for different areas of illumination. Finally, the JIS Standard requires a lower temporal stability for class A and B than both the ASTM and IEC Standards [33].

Classification	Spectral Match (each interval)	Irradiance Spatial Non-Uniformity Non-Uniformity	Temporal Instability
Class A	0.75 - 1.25	2%	2%
Class B	0.60 - 1.25	5%	5%
Class C	0.40 - 2.00	10%	10%

Table 3.1 – *ASTM class specifications*

Classification	Spectral Match (each interval)	Irradiance Spatial Non-Uniformity	Short-Term Temporal Instability	Long-Term Temporal Instability
Class A	0.75 - 1.25	2%	0.5%	2%
Class B	0.60 - 1.25	5%	2%	5%
Class C	0.40 - 2.00	10%	10%	10%

Table 3.2 – *IEC class specifications*

3

Classification	Spectral Match (each interval)	Irradiance Spatial Non-Uniformity Non-Uniformity	Temporal Instability
Class A	0.75 - 1.25	2%	1%
Class B	0.60 - 1.25	3%	3%
Class C	0.40 - 2.00	10%	10%

Table 3.3 – *JIS class specifications*

A xenon light source is available in the laboratory. Similar equipment to this has been previously used in [26] for the characterization of solar panels in CubeSats. In this work, a simulator of low cost is compared with another one of greater cost and performance from ORIEL brand, concluding that there are no appreciable differences in terms of performance, radiated spectrum and beam homogeneity. Furthermore, our xenon light source it will be used for the PV qualification test. Before that, some concepts will be presented.

3.2.1.1.1 Light Sources

The most commonly used light sources are:

- Xenon Arc Lamps: These Lamps have been commonly used since 1960's due to its output achieving a close spectral match to the Spectrum except for the large infra-red spikes which must be attenuated. In the figure 3.6 its output irradiance against the wavelength curve is shown.

- HMI Lamps: Metal halide arc lamps provide a more stable alternative to xenon arc lamps. If compared, the HMI lamp give better temporal stability than Xenon Arc lamps. Furthermore, the HMI is low-priced and have a straightforward maintenance. Finally, the spectral perturbations in the infra-red wavelengths are much diminished from xenon, which means that these lamps are excellent for Class "A" solar simulators. For reference, an unfiltered HMI arc lamp will produce a B class spectral match. The output irradiance against the wavelength curve is shown in the figure 3.7.

3

- LED: High-power LED have been developed thanks to the great progresses made in the LED Technology domestic market. The LED solar simulator have longer lifetime and lower power consumption than arc lamps. However. they are only available in discrete wavelengths and the commonly available wavelength LED's don't cover the spectrum required for the more advanced multi-junction devices, which spectral response is over 1000nm.

- QTH Lamps: These lamps provide an excellent black-body match in the IR but deficient across the visible range. The output irradiance against the wavelength curve is shown in the figure 3.8.

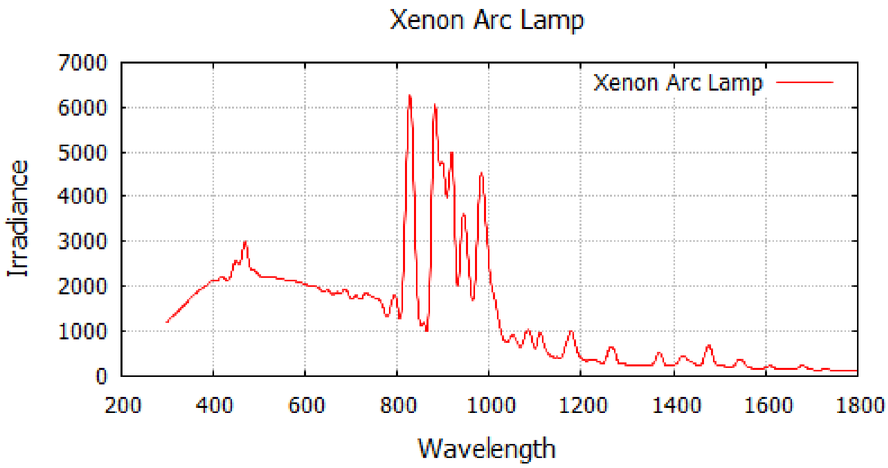


Figure 3.6 – *Xenon Arc Lamp Irradiance*

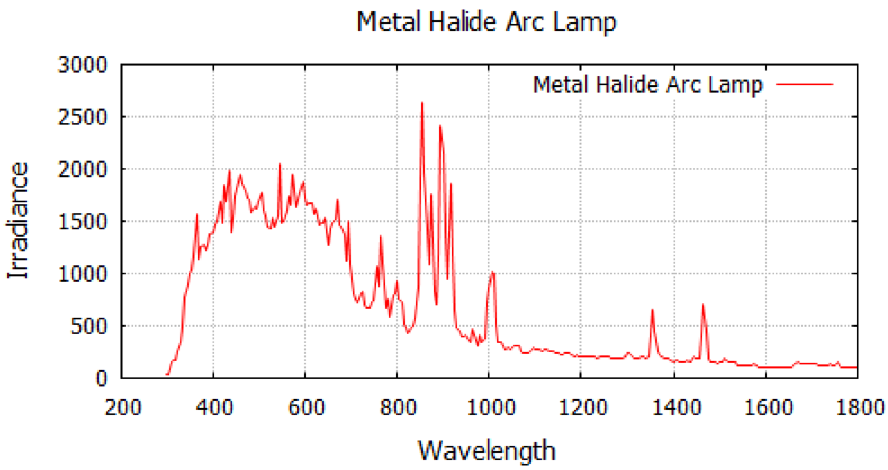


Figure 3.7 – *Metal Hallide Arc Lamp Irradiance*

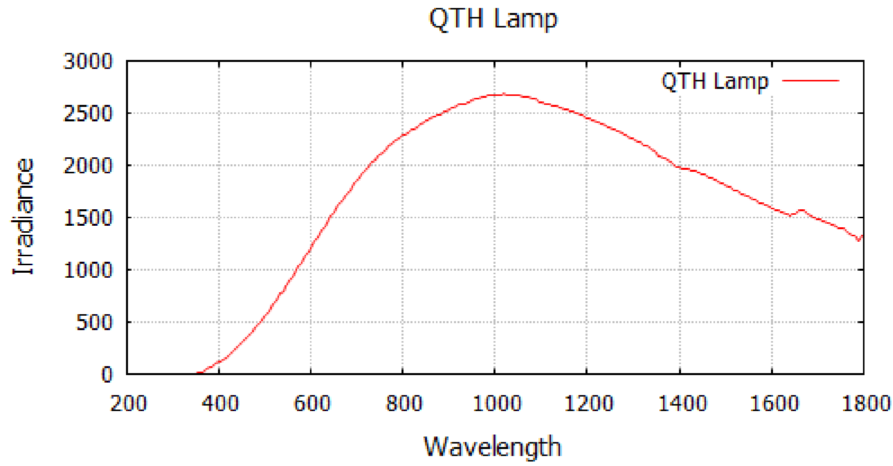


Figure 3.8 – *Quartz Tungsten Halogen Lamp Irradiance*

3

3.2.1.1.2 Spectral Match

Factors such as altitude, longitude, latitude, daytime or season might influence in the quantity and type of the received solar radiation on earth. Those elements interfere in the air mass through which solar radiation pass to arrive at Earth [10]. In order to measure the spectral match of a solar simulator, a comparison between the solar simulator spectra and the reference spectra is made. The spectral match is calculated by measuring the irradiance distribution as a percentage of the total irradiance of the solar simulator across specific wavebands [33].

The sun spectra have been standardized in several classes according to their Air Mass (AM) coefficient. The AM coefficient is defined in the equation 3.2.1, where L_O is the zenith path length at sea level and z is the zenith angle in degrees[39][16]. Solar PV cells used for space applications are generally characterized using AM0. In the Table 3.4 the reference spectra for AM0, AM1.5D and AM1.5G defined by the ASTM Standard is shown.

$$AM = \frac{L}{L_O} = \frac{1}{\cos z} \qquad (3.2.1)$$

Note that the equation 3.2.1 is just valid at h > 10°, therefore for most photovoltaic applications.

In the figure 3.9 the Standard Solar Spectrum for several AM coefficients and wavelengths is shown. According to the Table 3.1, for a solar simulator with a Class A rating in spectral match, its percentage irradiance distribution for each waveband must be must be within +/-25% of the reference spectrum for each waveband given in Table 3.4.

A solar simulator is usually considered as a xenon light source whose tolerances falls into at least class C of the ASTM standard, previously define in Table 3.1. This system is mainly considered as a xenon light source instead of a solar simulator if no tolerance for spectrum

Wavelength Interval [nm]	AM1.5D	AM1.5G	AM0
300-400	no spec	no spec	8%
400-500	16.9%	18.4%	16.4%
500-600	19.7%	19.9%	16.3%
600-700	18.5%	18.4%	13.9%
700-800	15.2%	14.9%	11.2%
800-900	12.9%	12.5%	9%
900-1100	16.8%	15.9%	13.1%
1100-1400	no spec	no spec	12.2%

Table 3.4 – *ASTM Spectral Distribution of Irradiance Performance Requirements[33]*

match is supplied. In most cases, the more accurate the solar simulator spectrum matches the sun spectrum, the more costly the system could be[10].

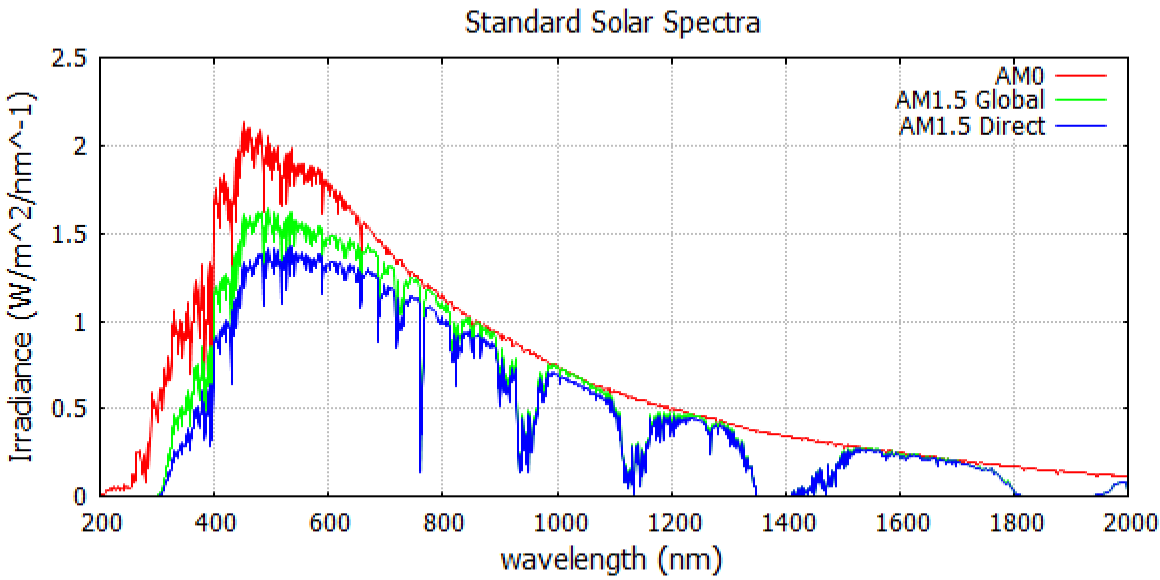


Figure 3.9 – *Standard Solar Spectrum[33]*

3.2.1.1.3 Spatial Non-Uniformity

One of the most challenging characteristics to meet in solar simulators is the uniformity of the light over the area under illumination. A mapping of the irradiance produced by a solar simulator beam over the area under test must be done in order to measure its non-uniformity [33]. By dividing this area in a grid in which the measurement points are located, the percentage of non-uniformity can be calculated with the following equation:

$$Non - Uniformity(\%) = \frac{max_{irradiance} - min_{irradiance}}{max_{irradiance} + min_{irradiance}} \times 100\% \tag{3.2.2}$$

The non-uniformity tolerance for each class is defined in Tables 3.1, 3.2 and 3.3. The quantity of measurement point and its positions depends on the test area size and the standard being followed. A summary for each of the three standards is given below.

ASTM[20].

- "Divide the defined test area into at least 36 equally sized (by area) test positions. Using the uniformity device, determine the irradiance in each of the test positions."
- "The uniformity device shall be no larger than the area of the individual test positions."
- "The uniformity device shall be at least large enough that the area of the device times the number of test positions is greater than 25% of the total defined test area."

3

IEC[11].

- "Divide the designated test area into at least 64 equally sized (by area) test positions (blocks). The maximum uniformity detector size shall be the minimum of":
  - A: The designated test area divided by 64.
  - B: 400 cm^2
- "The area covered by the detector measurements should be 100% of the designated test area. The measurement positions should be distributed uniformly over the designated test area."
- "Example: Large-area solar simulator A designated test area of 240 cm x 160 cm gives a maximum area of uniformity detector size of 600 cm^2 if divided by 64. As this value is greater than 400 cm^2 the maximum uniformity detector size is 400 cm^2 leading to 76 test positions."

JIS[32]

- "The light receiving area of light receiver to be used for measurement shall be a circle with a diameter not exceeding 2 cm or a square with a side not exceeding 2 cm and 4% or under of the effective irradiated plane by the solar simulator. Carry out the measurement with a detector capable of receiving at least the light with incident angle +/-15% without any trouble."
- "The uniformity device shall be no larger than the area of the individual test positions."
- "The uniformity device shall be at least large enough that the area of the device times the number of test positions is greater than 25% of the total defined test area."

The manufacturer should provide a document to demonstrate how the non-uniformity properties can be accomplished through a detailed report. To not matching at least Class C means the system being considered as a xenon light source.

3.2.1.1.4 Temporal Instability

This parameter represent the system fluctuation during the interval required to obtain an I-V curve. In other word, it is a measure of the light beam stability over a short interval using the equation **??**. In the Tables 3.1, 3.2 and 3.3 the temporal stability of the light beam defined for each class can be observed. To not matching at least Class C means the system being considered as a xenon light source.

$$TemporalInstability(\%) = \frac{max_{irradiance} - min_{irradiance}}{max_{irradiance} + min_{irradiance}} \times 100\% \quad (3.2.3)$$

For obtaining the temporal instability, the irradiance produced by the solar simulator is measured using an IV measurement system consisting of a PV cell connected to a SMU. For the STI the Isc is measured many times over the required time period. For IV applications the LTI refers to the length of time required to conduct an IV sweep[11].

3

3.2.1.2 Pyranometer

The solar spectrum is the radiation emitted by the sun over wavelengths ranging from 0.15 to 4.00 μm. Besides, the solar spectrum can be divided into several groups as function of its wavelength. In the Table 3.5 the solar spectrum group classification is shown. The measurement of the solar radiation on the earth is referred to as short-wave radiation. In the figure 3.10 the theoretical solar spectrum is shown.

Name	Wavelength [nm]	Photon Energy [eV]	Notes
UVC	100 - 280	4.43 - 12.4	Short-wave, germicidal, completely absorbed by the atmosphere.
UVB	280 - 315	3.94 - 4.43	Medium-wave, mostly absorbed by the ozone layer .
UVA	315 - 400	3.10 - 3.94	Long-wave, black light, not absorbed by the ozone layer.
Visible light	400 - 700	1.65 - 3.26	Electromagnetic spectrum that is visible to the human eye.
Infrared	700 - 1000	1.24m - 1.7	It can be also divided in A,B,C groups as function of its wavelength.

Table 3.5 – *Theoretical Solar Spectrum Classification*

According to the requirement 2 from section 3.1.1 a pyranometer must be used to determine the adequate functioning of the PV system. It will be needed for analyzing the solar spectrum produced by the Solar Simulator. It converts the incident solar radiation into an electrical signal that can be measured. The solar spectrum than can be obtained is just a portion of the whole. Therefore, this instrument does not perceive long-wave radiation (4 to 100 μm). Another important factor concerning pyranometers is the zenith angle. It is also known as the solar radiation angle or the polar angle ϕ. Another consideration to take into account is that the pyranometer should be connected to a digital multimeter or Datalogger in order to be able of measuring the output current[2][1].

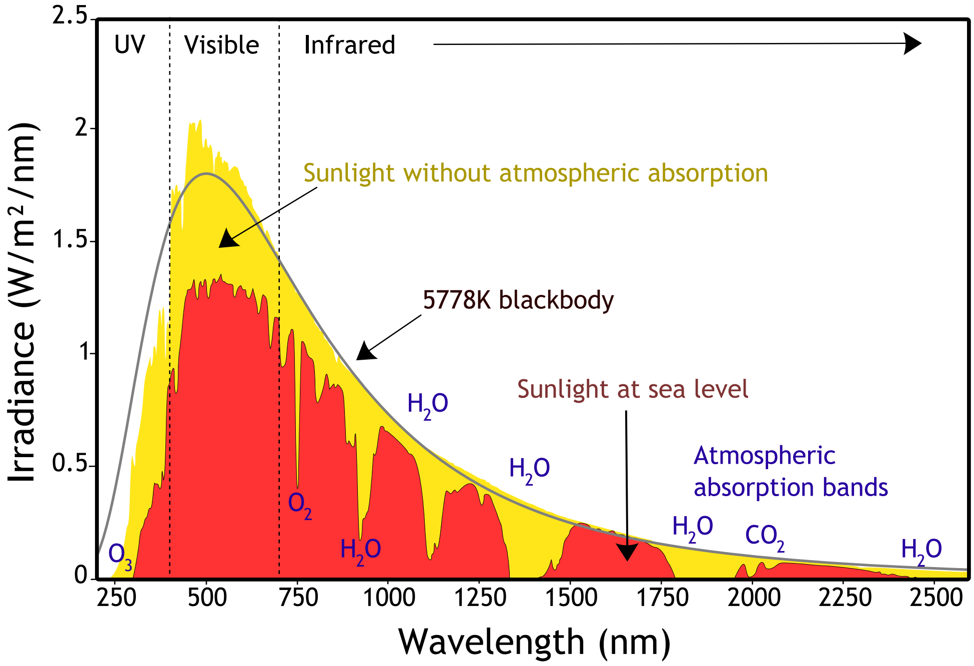


Figure 3.10 – *Theoretical Solar Spectrum[3].*

3.2.1.2.1 Pyranometer Classification

The spectral response of the pyranometer depend on the technology used. In the Figure 3.11 it is shown the solar reference spectrum at sea level with coefficient in comparison with different types of pyranometers. According to the ISO-9060 Standard, there are two particular technologies a pyranometer can be classified: Thermopile Technology & Silicon Semiconductor Technology [14].

- **Thermopile Pyranometers**

The thermopile pyranometer is mostly used when data integrity and high accuracy are required in scientific research or laboratory environments. These pyranometers have an flat spectral response ranging from 300 to 3000 nm, as can be observed in Figure 3.11. It uses the thermocouple principle to generate a voltage proportional to the temperature difference between a black absorbing surface and a reference which can be either a white reflective surface or the internal portion of the sensor base. This temperature rise is proportional to the incident irradiance reaching the black surface. The thermopile pyranometer's black surface uniformly absorbs solar radiation across the solar spectrum [2] [31]. To protect the black absorbing surface from environmental conditions, which would introduce some errors in the irradiance detection, a glass dome is usually needed. This dome is made from single or double layers of ground and polished optical glass covering the thermopile[13]. These glass domes uniformly pass the irradiance to the black surface. It also contains a small cartridge of desiccant to absorbs any dew. In the Figure 3.12 a diagram of a thermopile pyranometer is shown.

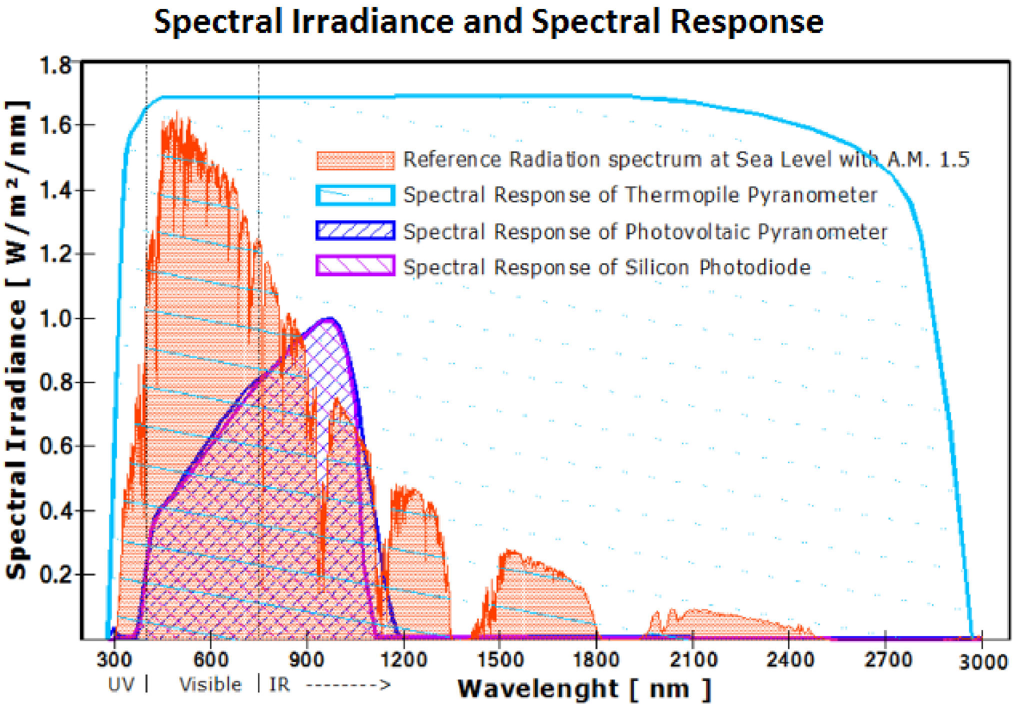


Figure 3.11 – *Theoretical Solar Spectrum & Pyranometer Spectral Response [7].*

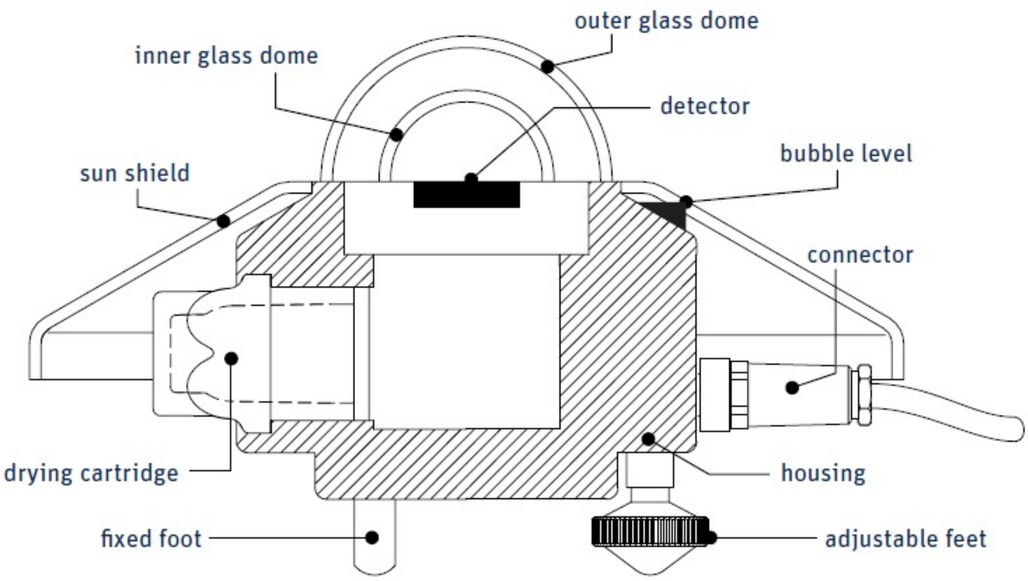


Figure 3.12 – *Thermopile Pyranometer Diagram [19].*

- **Silicon Photocell Pyranometers**

The Silicon Photocell Pyranometer is mostly used when high accuracy is not required and for lower budgets. It is a more cost effective solution than a thermopile pyranometer with a faster time of response. However, lower precision under cloudy conditions are obtained. The silicon photocell pyranometers are based on based on light-sensitive semiconductor chips. Its working principle is similar to how a solar panel works. A current passing through a shunt resistor produces an output voltage in the range of $\mu V/W/m^2$. The main drawback is that the spectral response of these pyranometers is limited to a portion of the whole solar spectrum approximately ranging from 350 nm to 1100 nm with a peak in the IR region. The best performance of these pyranometers occur when hey are used to measure the global solar radiation under the same clear sky conditions used to calibrate them. In these conditions they provides a response similar to thermopile pyranometer.

In the Figure 3.13 a comparison between the measured output produced under clear and overcast conditions for both a secondary standard thermopile and low-cost silicon pyranometers is shown.

3

There is on the market a lot of different solutions available including the both mentioned technologies, thermopile and Silicon Photocell. To sum up, the thermopile pyranometer gives more accurate measurements over a wider range of wavelengths in all atmospheric conditions than the Silicon Photocell Pyranometer. However, the last one offers a similar response under clear sky conditions and a faster response time for a lower price. Taking into account the pros and cons of both types of pyranometer, it is concluded that the Silicon Photocell Pyranometer fits better into the system requirements.

In the Table 3.6 a comparison between the specifications of several silicon photocell pyranometer is shown.

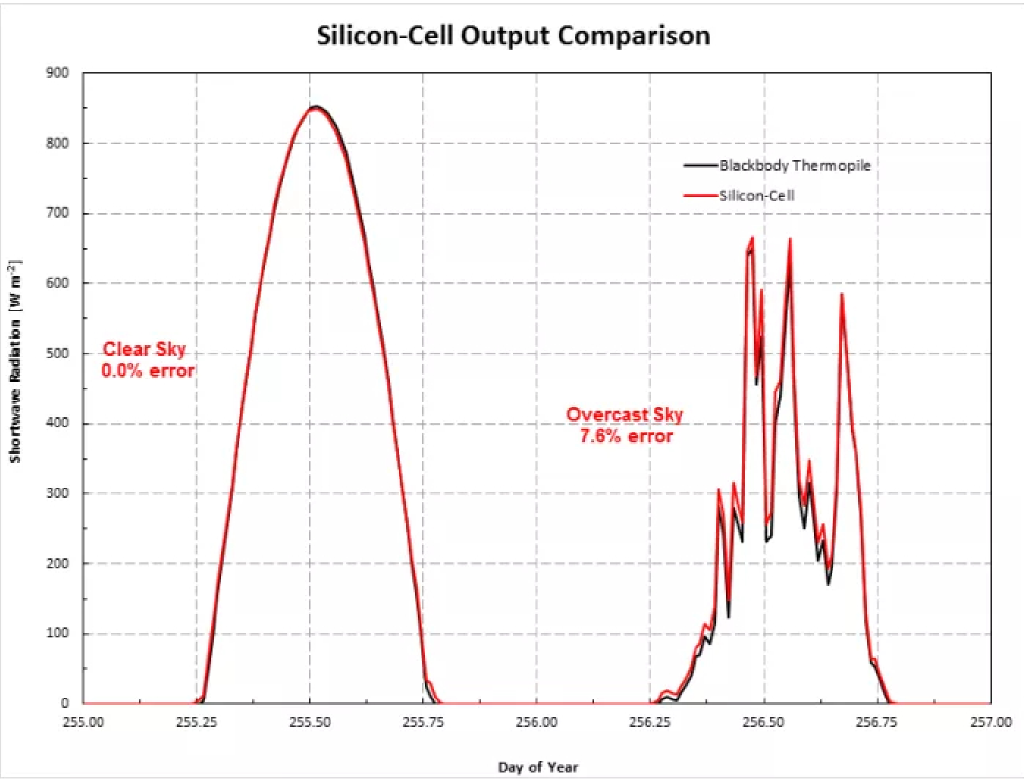


Figure 3.13 – *Comparison between silicon pyranometer and thermopile pyranometer output [2].*

Product	**SP-110-SS**	**SP-212-SS**	**SP-215-SS**	**LI-200R**	**SP Lite2**
Manufacturer	Apogee Instruments	Apogee Instruments	Apogee Instruments	LI-COR	Kip&zonen
Sensitivity	0.2 mV/W/m^2	2 mV/W/m^2	4 mV/W/m^2	75 /W/m^2	60 - 100 /W/m^2
Non-Linearity	< 1 % up to 2000 W/m^2	< 1 % up to 1250 W/m^2	< 1 % up to 1250 W/m^2	< 1 % up to 3000 W/m^2	-
Response Time	< 1 ms	< 1 ms	< 1 ms	< 500 ns	< 1
Spectral Range	360 - 1120 nm	360 - 1120 nm	360 - 1120 nm	400 - 1100 nm	400 - 1100 nm
Cosine Response	$\pm 5\% at 75°$ zenith angle	$\pm 5\% at 75°$ zenith angle	$\pm 5\% at 75°$ zenith angle	Cosine corrected up to 82° zenith angle	< 10 W/m^2
Temperature Response	< 0.04 $\pm 0.04\% perC$	< 0.04 $\pm 0.04\% perC$	< 0.04 $\pm 0.04\% perC$	-	< 0.15 %
Operating Relative Humidity Range	0 to 100 % 0 to 100 %	0 to 100 %	0 to 100 %	0 to 95 %	-
Operating Temperature Environment	-40 to 70 °C	-40 to 70 °C	-40 to 70 °C	-40 to 65 °C	-40 to 80 °C
Price	275 €	285	300	330 €	366 €
Choice	✓	×	×	×	×

Table 3.6 – *Silicon photocell pyranometer specification comparison*

3.2.1.3 Solar Panels

A solar array is a combination of several PV cells connected in series-parallel configurations in order to obtain the required voltage and current for energy harvesting. Those PV cells converts incident sunlight into DC electricity. Since 1958, it has been a valuable source if power for satellites due to its high power output per unit mass.

The PV effect consists on two diverse materials producing an electrical potential with their common junction being illuminated by photons. Those photons transfer energy to the system culminating with the creation of electron–hole pairs in a solid semiconducting material separated at the junction. Finally they are accelerated under the electric field created by a potential gradient and, if the circuit is closed, they circulate as electrical current.

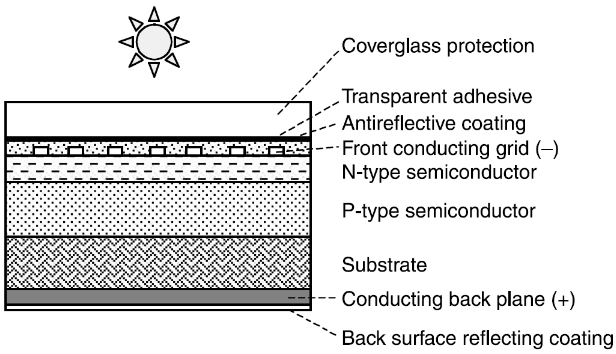


Figure 3.14 – *Basic construction of PV cell with performance enhancing features. [29].*

In the Figure 3.14 a basic construction of PV cell is shown. It can be observed a p–n junction of two semiconductors with metallic contacts on both sides of it. In addition to the basics, several enhancements such as reflective coatings and transparent coverglass protection have been included to maximize the light absorption and protect the surface against particles hitting it, respectively[29].

$$\mu = \frac{electrical\ power\ output}{solar\ power\ trespassing\ the\ cell} \tag{3.2.4}$$

The photoelectric energy conversion efficiency is a crucial factor of the PV cell performance. Theoretical maximum energy conversion efficiency for Ge is over 16%, for Si is 24%, 29% for GaAs and 38% for Three-junction cells[29]. This factor is defined in the equation 3.2.4. In the figure 3.15 a global overview of the solar cell efficiency conversion for each technology can be observed.

Finally, there are some concepts relative to solar panels that must be taken into account before the system design. Those concepts are relative to the I-V Curve that must be obtained for characterized the Solar Panel and the type of solar panel used in space applications.

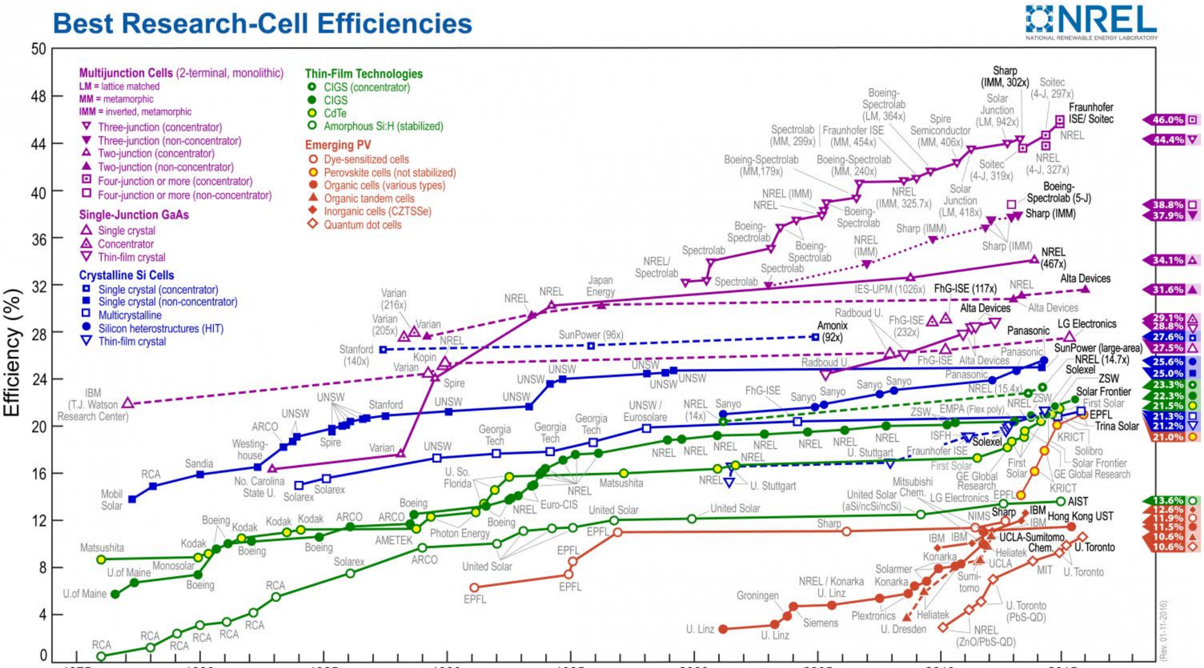


Figure 3.15 – *PV Solar Cell efficiency. Credits: NREL*

3.2.1.3.1 Multi Junction Solar Panels

Multi-junction GaAs cell have rose in popularity for space application. They consist of two tandem GaAs cells separated by thin tunnel junction of GaInP, followed by a third tandem GaInP cell, separated by an AlInP tunnel junction. This junction is used for voltage drop mitigation of the otherwise forward-biased p–n junction [29]. The GaInP/GaAs cells also capture IR photons. Many spacecraft builders have increasingly used the GaInP/GaAs cells on Ge substrate seeking higher efficiencies.

The structure of a Multi-Junction solar cell and the spectral irradiance absorption for every junction as a function of the wavelength are shown in the Figure 3.16.

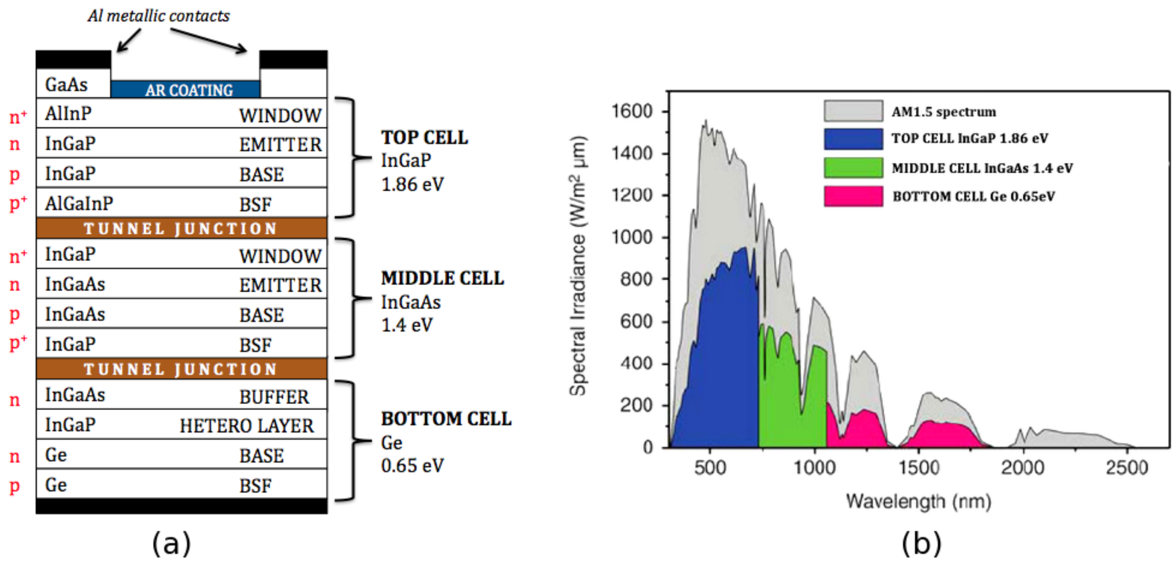


Figure 3.16 – *a) GaAs cells on Ge Multi-Junction Solar cell structure and b) Spectral irradiance against wavelength for each junction. Credits: Fraunhofer ISE*

3.2.1.3.2 I-V Characteristic Curve

The first quadrant I-V Curve of PV cell in sunlight represent its electrical characteristic. The Isc and the Voc are the primary parameters for describing the cell electrical performance. The Isc is mesured by shorting the output terminal of the circuit and measuring the current under full illumination. It represent the maximum current the cell can deliver at the working illumination and temperature conditions[29]. The Voc is measured by opening the output terminal of the circuit. Under this condition, since the circuit is open, the current is zero and the resistance is infinitely high. In the Figure 3.17 (a) the I-V Curve and 3.17 (b) P-V Curve of a cell is shown.

The output power of the panel at any point along the curve is the product of the voltage and the current at that point. Note that the cell produce no power at Isc or Voc conditions since the voltage or current is zero, respectively. Furthermore, the maximum power is produced at voltage corresponding to the knee point of the I-V curve. For that reason, the PV Power circuits design encourage the panel operating slightly of the left hand side of the knee point. In this side, the cell works like a constant current source, generating voltage to match with the load resistance. The cell works like a constant voltage source with an internal resistance on the right side of the knee point, where the current drops rapidly with small rise in voltage[29].

Finally, an increase of temperature produces an increase in the Isc, whereas the Voc decreases. As for the power, the maximum power available is higher in lower temperatures, hence the cold temperatures the better for the PV cell, as it generates more power[29].

In the Figure 3.18 the temperature effects on solar cells I-V Curve are shown.

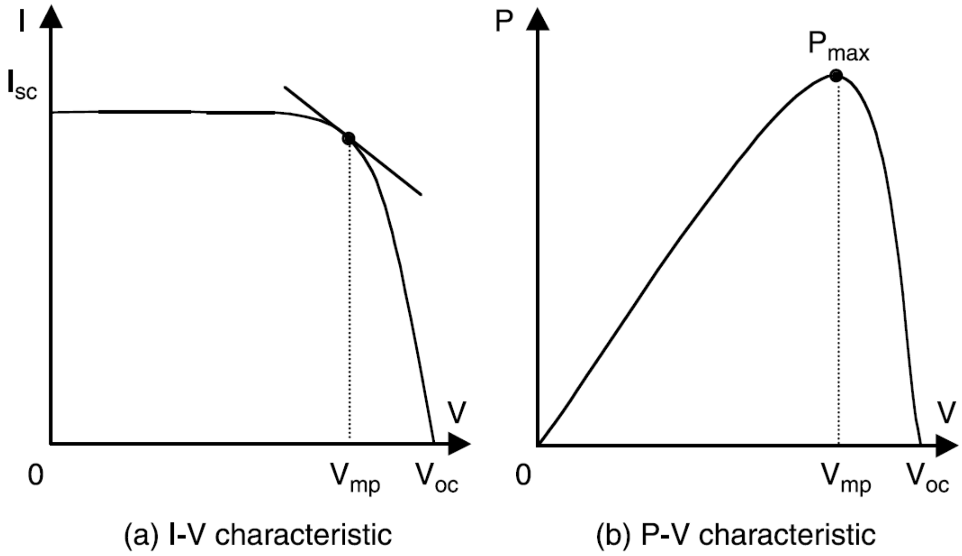


Figure 3.17 – *Cell characteristics in sunlight and the maximum power point[29].*

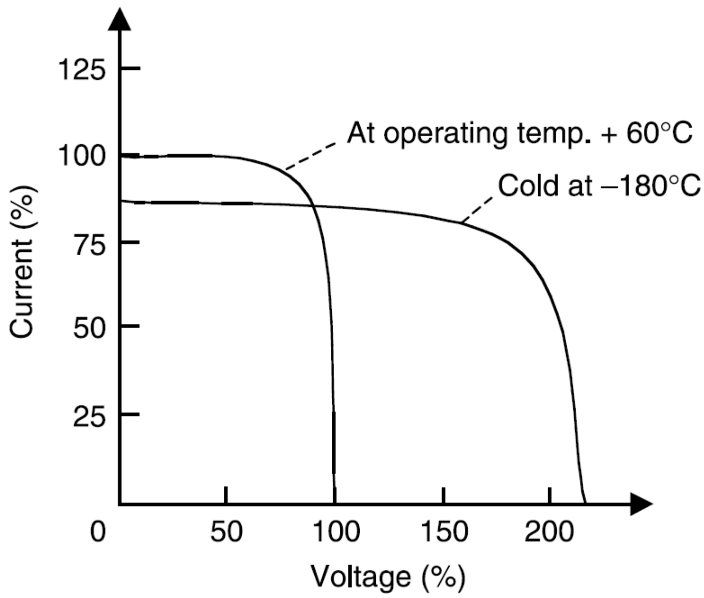


Figure 3.18 – *Temperature effects on solar cell I-V Curve[29].*

3.2.1.4 DC Electronic Load

For solar cell and module testing, an appropriate and adjustable load is indispensable. A DC electronic load (eload) is a device with an adjustable load with a resistance value ranging from nearly zero to infinity which can measure the output power of a DUT. It is composed of a number of power transistor which output current is converted into heat that is then dissipated[27][35].

The eload is able to measure both the voltage that the DUT is dropping across the electronic load and the output current. The electronic loads may typically operate in either the constant current (CC) mode, the constant voltage (CV) mode or the constant resistance (CR) mode. In each operation mode the set variable is a constant while the other parameters vary according to the source unit and the eload configuration, see Figure 3.19. For example, in the CC mode, the current continue constant irrespective of the applied voltage[35][9].

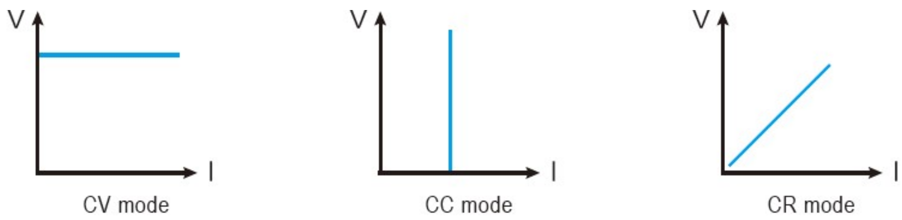


Figure 3.19 – *Electronic load operation modes.*

In the figure 3.20, a circuit diagram of an electronic load is shown.

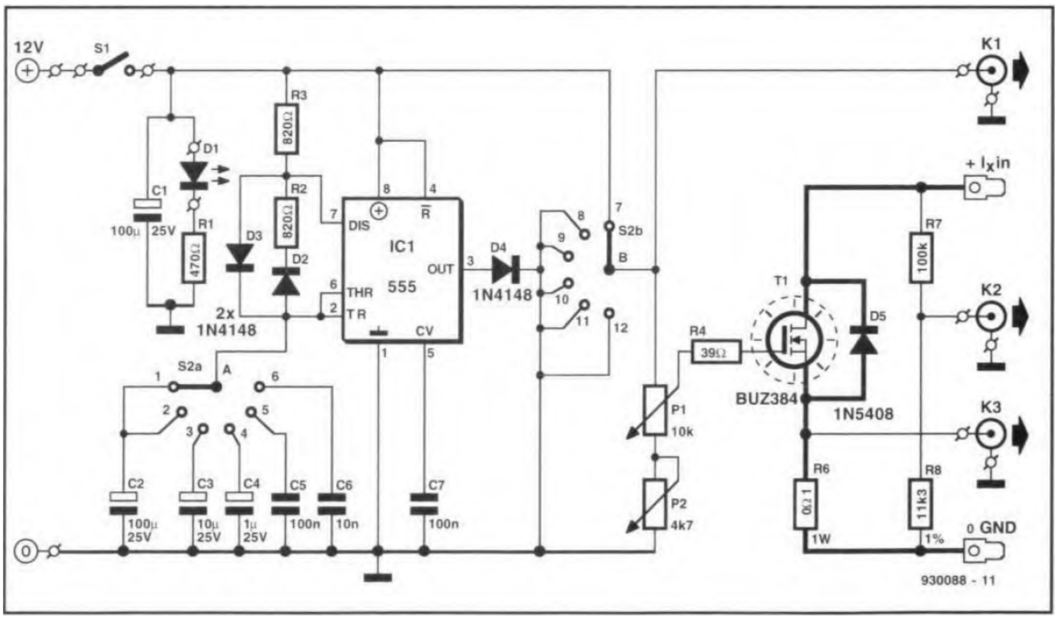


Figure 3.20 – *Circuit diagram of an electronic load[27].*

3.2.1.4.1 I-V Curve Characterization

For solar cell testing, the eload is used in CV mode to catch the I-V curve by sweeping the voltage and measuring the output current or vice-versa.

A zero voltage potential across an illuminated cell is required in order to begin the I-V curve test. As said in Section 3.2.1.3.2, the Isc is measured at 0 V. However, a drawback in the use of an eload in solar cells test is that its performance begins to reduce at voltages lower than a threshold voltage (usually 3 V). The eload current handling ability also get worse in those conditions. A solution to this downside is to configure a power supply to boost the potential across the cell above the threshold voltage of the electronic load[35]. This configuration is shown in the Figure 3.21

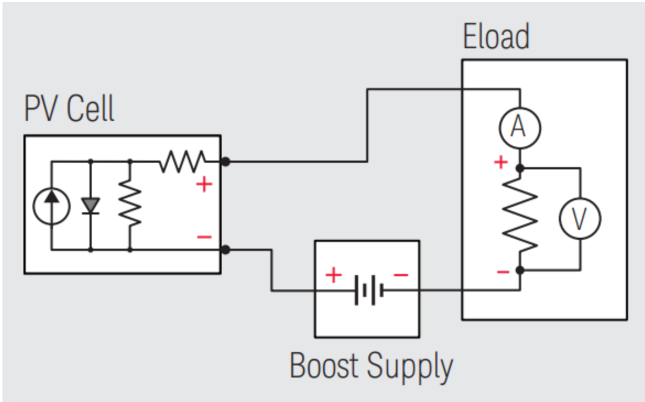


Figure 3.21 – *Electronic Load with a boost power supply[35]*

Notice that the more the cell potential is boosted above the eload threshold the lower the eload's available voltage range is. This could be seen as an obstacle, however, it introduces some benefits such as the reverse bias measurement capability. Finally, to capture the forward bias I-V curve you must add the boost voltage to each voltage step of the eload's sweep.

There are three different approach to obtain the I-V Curve of a solar cell DUT[35]:

- **Stepping both the eload and the boost supply**: Measuring the current by performing a DC sweeping (eload in CV mode) from the Voc voltage down to Isc conditions. When the eload is at V_{boost} voltage, the boost power supply's voltage is swept up in order to obtain the reverse bias region.
- **Stepping only one eload or the boost supply**: In this approach just one of the instrument's voltage is swept. However, the voltage ranbge of the test is limited to the instrument being swept and a larger load is needed.
- **Short circuiting the electronic load**: Similar to the first method until reaching the 0 V potential across the solar DUT. From that point, the reverse bias is measured by shorting the eload. This is done by placing the the output of the boost power supply directly across the solar cell.

According to the advantages and disadvantages of the different methods above describes, the first approach has been chosen to obtain the I-V Curve. Nevertheless, we are not interested in the reverse bias region of the solar cell and, therefore, is not going to be measured. Notice that the same procedures can be done by starting in the reverse bias region and then finishing in Voc conditions.

3.2.1.4.2 Market Analysis

A programmable DC electronic load is needed for the system. In this section a comparison between the characteristic of some programmable eload will be performed in order to chose the one which best fit for the project fulfilment. This comparison is shown in Table 3.7.

Model	B&K Precision 8601	B&K Precision 8602	B&K Precision 8610	B&K Precision 8614	KeySight 6063B
Power [W]	250	200	750	1500	250
Operating Voltage [V]	0 - 120	0 - 500	0 - 120	0 - 120	3 - 240
Rated Current [A]	0 - 60	0 - 15	0 - 120	0 - 240	0 - 10
CC Mode					
Range	0 to 6 A, 0 to 60 A	0 to 3 A, 0 to 15 A	0 to 12 A, 0 to 120 A	0 to 24 A, 0 to 240 A	0 to 1 A, 0 to 10 A
Accuracy	$\pm(0.05\% + 0.05\%FS)$	$\pm(0.05\% + 0.05\%FS)$	±(0.05%+0.1% FS)	±(0.05%+0.1% FS)	0.15% $\pm 10mA$
Resolution	0.1 mA, 1 mA	0.1 mA, 1 mA	1 mA, 10 mA	1 mA, 10 mA	8 mA
CV Mode					
Range	0 to 18 V, 0 to 120 V	0 to 50 V, 0 to 500 V	0 to 18 V, 0 to 120 V	0 to 18 V, 0 to 120 V	-
Accuracy	$\pm(0.05\% + 0.02\%FS)$	$\pm(0.025\% + 0.05\%FS)$	$\pm(0.025\% + 0.05\%FS)$	$\pm(0.025\% + 0.025\%FS)$	0.12% ± 120 mV
Resolution	1 mV, 10 mV	1 mV, 10 mV	0.1 mV, 1 mV	0.1 mV, 1 mV	10 mV
Comm Interfaces	USB, RS-232 & GPIB	USB, RS-232 & GPIB	USB, RS-232 & GPIB	USB, RS-232 & GPIB	GPIB
Price	1.378,22 €	1.378,22 €	2.494,88 €	3.692,02 €	1.100,00 €
Choice	×	×	×	×	✓

Table 3.7 – *Programmable DC electronic load comparison.*

Although some electronic loads models have higher powers or entry ranges, according to the low cost philosophy of the project and considering the system's needs, the KeySight 6063B has been finally selected.

3.2.1.5 Power Supply

As previously said in Section 3.2.1.4, a power supply will be needed in order to lift the solar DUT $\geq$ 3 V. The power supply available in the electronics laboratory is the Keysight E3631A 80 W Triple Output DC Power Supply which has GPIB and RS-232 communication interfaces and can be programmed using SCPI commands. The GPIB interface is used to send control commands to the equipment via PC.

For the better understating and handling of the system, some characteristic and basics of this Power Supply will be explained in this section. In the Figure 3.22, the front panel of the E3631A is shown.

1. Meter and adjust selection keys
2. Tracking enable/disable key
3. Display limit key
4. Recall operating state key
5. Store operating state/Local key
6. Error/Calibrate key
7. I/O Configuration / Secure key
8. Output On/Off key
9. Control knob
10. Resolution selection keys
11. Voltage/current adjust selection key

3

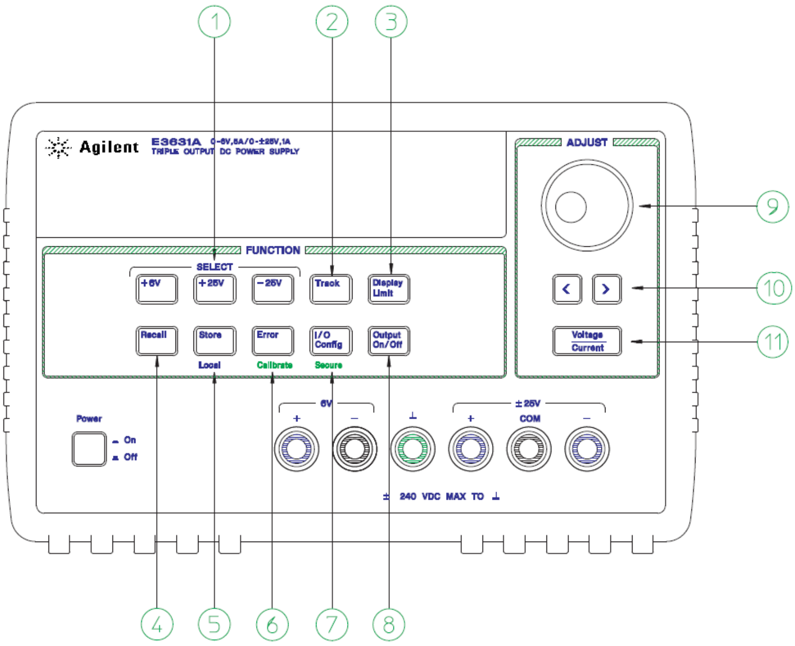


Figure 3.22 – *E3631A Power Supply Front Panel.*

The functions of each key is of the frontal panel is described below:

1. **Meter and adjust selection keys**: Select the output voltage and current of any one supply (+6V, +25V, or -25V output) to be monitored on the display and allow knob adjustment of that supply.

2. **Tracking enable/disable key**: Enables / disables the track mode of ±25V supplies.

3. **Display limit key**: Shows the voltage and current limit values on the display and allows knob adjustment for setting limit values.

4. **Recall operating state key**: Recalls a previously stored operating state from location "1", "2", or "3".

5. **Store operating state/Local key**: Stores an operating state in location "1", "2", or "3" / or returns the power supply to local mode from remote interface mode.

6. **Error/Calibrate key**: Displays error codes generated during operations, self-test and calibration / or enables calibration mode (the power supply must be unsecured before performing calibration).

7. **I/O Configuration / Secure key**: Configures the power supply for remote interfaces / or secure and insecure the power supply for calibration.

8. **Output On/Off key**: Enables or disables all three power supply outputs. This key toggles between two states.

9. **Control knob**: Increases or decreases the value of the blinking digit by turning clockwise or counter clockwise.

10. **Resolution selection keys**: Move the flashing digit to the right or left.

11. **Voltage/current adjust selection key**: Selects the knob function to voltage control or current control.

In the Table 3.8 some specification of the power supply are given.

	Channel 1	**Channel 2**	**Channel 3**
Output Rating	0 to 6 V, 0 to 5 A	0 to 25 V, 0 - 1 A	0 to -25 V, 0 - 1 A
Programming Accuracy at 25° C	0.05% + 20 mV 0.15% + 4 mA	0.05% + 20 mV 0.15% + 4 mA	0.1% + 5 mV 0.2% + 10 mA
Ripple & Noise 20 Hz to 20 MHz	<350 μV$_{rms}$/ 2 mV p-p <500 μA$_{rms}$,	<350 μV$_{rms}$/2 mV p-p <500 μA$_{rms}$	< 350 μV$_{rms}$/2 mV p-p <2 mA_{rms}
Readback Accuracy at 25° C	0.05% + 10 mV 0.15% + 4 mA	0.05% + 10 mV 0.15% + 4 mA	0.1% + 5 mV 0.2% + 10 mA

Table 3.8 – *Keysight E3631A Triple Output DC Power Supply Specifications[36]*

According to the characteristic ans specification above given for the Keysight E3631A Power Supply, it is concluded that it perfectly fits with the system requirements.

3.2.2 TVAC System

Secondly, as stated in the functional requirements 2.1, a TVAC Chamber must be designed and manufactured to be able to perform thermal and vacuum tests on PV cells. The TVAC system is composed of the elements stated in Section 3.1.1.2. Those elements will be further analysed on the following sections.

3.2.2.1 Structure

The final structure the system will have is a critical issue of the project since it must withstand very adverse pressure and temperature conditions for performing aerospace test. Important issues such as the camera shape, the amount of feed-though ports to be considered, the material used as well as its final weight and dimensions must be taken into account for accomplishing a good design for less cost.

Controlling deflections allow reliable sealing and endurance the structure of the chamber. Stiffeners can be added for strengthen weaker structures. There are a wide range of structural shapes for TVAC chambers ranging from simple boxes and cylinders to spheres and geodesic balls [12]. The different shapes of a TVAC chamber and their rigidity level is shown in Figure 3.23

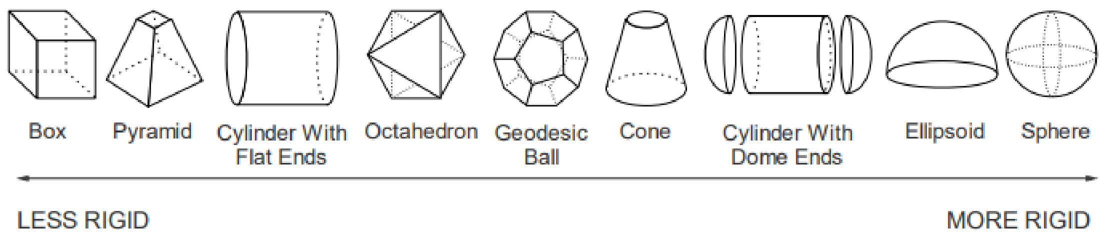


Figure 3.23 – *TVAC shapes possibilities and rigidity levels[12].*

According to the above figure, the sphere present the best structural rigidity and material utilization. On the other hand a box shaped chamber is cheaper and can be easier designed. Furthermore, it can be strengthened if needed. Making good choices could significantly impact the cost and reliability of the project.

3

Both designs, spherical and boxed, were selected for the TVAC Chamber. That mean designing and developing two different TVAC chambers for then select the best one. In the Chapter 4 both design will be shown. Finally the box shaped chamber were chosen considering the manufacturability, internal dimensions and the final expenses.

The integrity of the chamber is not an easy task to maintenance due to sections removal for the allocating of doors ports and feed-throughs. For that reason the use of flanges and stiffeners are necessary to maintain the rigidity. According to the requirement stated at 2.1.1, two configuration possibilities for the TVAC Chamber door will be designed. AISI 304 Stainless Steel (SS) and methacrylate are the material choice for the gates.

SS is the material chosen for many vacuum application. Other material such as aluminium, titanium or glass can also be used. Porous materials should be avoided. The cost compromise solution chosen for the project is the use of SS for the components exposed to vacuum and aluminium or other material for the component not exposed to vacuum[17].

For performing aerospace test, cleanliness is crucial which means to treat the surfaces exposed to vacuum with special treatments such as electro-polishing or graining.

The chamber also contains elements belonging to the thermal and vacuum systems as well as the the whole instrumentation equipment and feed-through ports used for instrumentation purposes.

3.2.2.2 Vacuum System

The Vacuum system shall decrease the pressure within the TVAC chamber to a requested level in a brief period of time on a clean atmosphere. The vacuum system of a TVAC Chamber usually consist on two interconnected pumping units working in two stages[23][12][22]. First, a rotary pump is used to reduce the inner pressure from ambient to Medium Vacuum conditions. Then a Turbo-molecular pump is used for releasing the pressure down to High Vacuum conditions. The Turbo molecular pump is only needed in cases in which the High Vacuum conditions are required. Notice that the use of two pumping units is the consequence of the limited performance which the available pumping units can achieve.

The rotary pump already available in the laboratory is a Telstar oil-sealed rotary vane pump. It consist of vanes mounted on an eccentric rotor inside of a circular housing. In the Figure 3.24 it is shown.

Furthermore, a vacuum transducer must be used for measuring the vacuum conditions. Communication interfaces will be appreciated. In the Tables 3.9 and 3.10 a comparison between some analysed vacuum transducer is shown. In those tables, a wide range of transducers have been analysed. Those with a broader vacuum range result in costly solutions which exceeds our budget. Considering that our test will run at a minimum of Medium Vacuum conditions, those whose measuring range reach 10^{-5} Torr are acceptable for the system. Therefore, the vacuum transducer which best fits with the requirements and the project's budget is the MKS 925 MicroPirani. Thus, it has been selected for the project.

Figure 3.24 – *Telstar rotary vane pump.*

	925 MicroPirani	**901P MicroPirani**	**910 DualTrans**
Measuring Range	1.0 x 10^{-5} Torr to Atmosphere	1.0 x 10^{-5} Torr to 1500 Torr	1.0 x 10^{-5} Torr to 1500 Torr
Operating Temperature Range	0° to 40° C	0° to 40° C	0° to 40° C
Communication Interfaces	RS485 / RS232	RS485 / RS232	RS485 / RS232
Analog Output Resolution	16 bit	16 bit	16 bit
Power Requirements	9 to 30 VDC < 1.5 watts max	9 to 30 VDC < 1.2 watts max	9 to 30 VDC < 1.2 watts max
Overpressure Limit	3000 Torr absolute	1500 Torr	2250 Torr
Materials Exposed to Vacuum	304 stainless steel, Silicon, SiO_2, Si_3N_4, Gold, Viton, Low out gassing epoxy resin	304 stainless steel, Silicon, SiO_2, Si_3N_4 Gold, Viton, Low out gassing epoxy resin	304 stainless steel, Silicon, SiO_2, Si_3N_4, Gold, Viton, Low out gassing epoxy resin
Price	**680 €**	460 €	700 €
Choice	✓	×	×

Table 3.9 – *Vacuum Transducer Comparison [Part 1]*

	275 Mini-Convectron	390 Micro-Ion ATM
Measuring Range	1.0 x 10^{-4} to 1000 Torr	1.0 x 10^{-8} to 100 mTorr: ±5% of Reading; 100 mTorr to 150 Torr: ±2.5% of Reading; 150 to 1000 Torr: ±1.0%*of Reading*
Operating Temperature Range	0° to 40° C	10° to 40° C
Communication Interfaces	RS485, 2 set point relays	RS485
Analog Output Resolution	16 bit	-
Power Requirements	11.5 to 26.5 VDC, 0.12 A at 11.5 VDC, 2 W max	24 VDC +10% to -15%, 1 A, 22W nominal;
Materials Exposed to Vacuum	304 stainless steel, borosilicate glass, Kovar, alumina, NiFe alloy, polyimide	304 stainless steel, tantalum, tungsten, yttria-coated iridium, Kovar, borosilicate glass, gold
Price	480 €	2.200,00 €
Choice	×	×

Table 3.10 – *Vacuum Transducer Comparison [Part 2]*

	Artic A10	**Isotemp 6200 R35**	**Isotemp 4100 R28**
Temperature Range [°]	-100 to 100	-35 to 200	-28 to 100
Bath Volume [**L**]	4.0 - 6.0	6.8 - 8-6	6.8 - 8-6
Cooling Capacity [**W**]	240	800	500
Pressure [**mbar**]	200	750	310
Refrigerant	R134a	R404a	16 bit
Net Weight [**kg**]	27.5	54.9	35.8
Price	2,680.00 €	**3,295.00 €**	2,875.00 €
Choice	×	✓	×

Table 3.11 – *Refrigerated/Heated Bath Circulator Comparison*

3.2.2.3 Thermal System

The thermal system shall recreate the harshly low temperatures in space environment. The system requires a total control of the inner temperatures during the thermal tests. Those temperatures are usually collected using thermocouples. Most TVAC chambers implement thermal modes consisting of cooling and heating stages [23][12]. The cooling stage usually consists on LN_2 injection to swiftly cool the inner temperature down to $-180°$. This liquid is injected through the walls of the chamber using a metallic pipe[23][12]. The heating stage uses the recirculating chiller for heating up the inside of the chamber. In both stages an insulated piping system covered by MLI film is used for the fluid's distribution[23][12].

In the Table 3.11 a comparison between several chiller is shown. In that table, it can be appreciated that the Isotemp 6200 R35 has the broader temperature range. It has a lower limit of -35 ° C which in combination with the LN_2 injection might conform our Cooling Stage. The Isotemp 6200 R35 could also be used for the Heating Stage due to its upper temperature range, which is the highest in the table. Although it is the most expensive chiller in the comparison, it perfectly fits in the project requirement. Thus, it has been selected for the project.

3.2.2.4 Instrumentation Equipment

The Instrumentation Equipment is crucial for monitoring the aerospace tests. The data obtained provides an valuable information to the engineering team for supervising or even intervention on the ongoing test. Multiple feed-through port are included for this purpose. In the next figures, a collection of feed-through port are shown in order to make an approach to this kind of equipment.

Finally, a 12 multiplexed channel *Siglent SDM3065X 6 ½ Digits Dual-Display Digital Multimeter* is used. It has an USB interface which allows the user to measure temperatures in different internal points, currents and voltages. This equipment is already available in the laboratory.

Figure 3.25 – *Electrical Feedthrough examples.*

Figure 3.26 – *Thermocouple Feedthrough example.*

3.3 Software Subsystem Analysis

As stated in the software requirements 3.1.2, the aim of this subsystem is to control the instrumentation equipment for performing the aerospace test as well as managing the data acquired. Those data records must be not only handled but also presented to the user in a friendly way. Furthermore, along with those records, the software should show as much technical information as possible about the equipment used, the project, etc. The software will be design with MATLAB and will be all integrated in a GUI.

For a better understanding of the design that will be carried out in the next chapter, details about the GPIB Communication and the SCPI standards will be given in section3.3.1 and 3.3.2, respectively.

3

3.3.1 SCPI Overview

SCPI is a standard specifically designed for controlling programmable instruments. This Standard aims to promote a common syntax and language for all of them. In SCPI the commands are structured in a command tree. Most manufacturers of programmable instruments such as Keithley (Agilent, HP), Tektronix, Fluke, and Racal support SCPI. Furthermore, SCPI is the most common standard used to control instruments over interfaces such as LAN, GPIB, , and USB which are supported by Agilent VISA or NI VISA I/O Libraries. For additional information, refer to the IEEE 488.1-2003 and IEEE 488.2-1992.

A typical command is made up of keywords prefixed with colons (:). Those keywords are then followed by parameters. The common syntax and term are explained as follows.

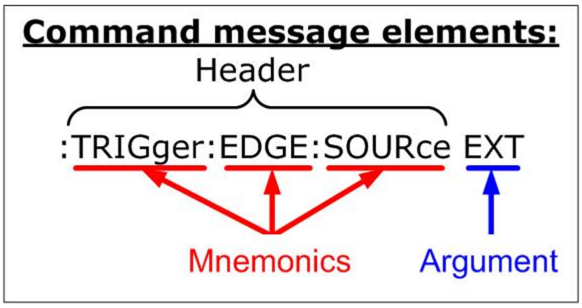


Figure 3.27 – *SCPI Command Example.*

Common Terms

- **Command**: Instruction in SCPI consisting of mnemonics, arguments, and punctuation which combined form messages that control instruments.
- **Controller**: Any device used to control the instrument.
- **Event Command**: Events commands that cannot be queried.

- **Program Message**: Combination of one or more properly formatted commands.
- **Query**: Special type of command used to instruct the instrument to make response data available. It ends with a question mark.
- **Response Message**: Data in specific SCPI formats sent from the instrument to the controller.

Command Syntax

<table>
<tr><th>Character</th><th>Meaning</th></tr>
<tr><td>|</td><td>It indicates alternative choices.</td></tr>
<tr><td>[]</td><td>It indicates that the enclosed parameters are optional</td></tr>
<tr><td><></td><td>It represent the a needed item</td></tr>
</table>

Table 3.12 – *SCPI Special Characters*

3.3.2 GPIB Interface Analysis

Originally GPIB was known as HP-IB that was originally introduced by Hewlett Packard for controlling their test equipment. It not only allows data to flow between any of the instrument on the bus but also at reasonable speed suitable for the worst case instrument. A maximum of 15 instruments might be connected in on the bus with with a maximum bus length not exceeding 20 m[34]. Computers that does not have a GPIB interface might be upgraded with a GPIB card.

A total of 16 active lines are used in the bus of which, 8 are used for data transfer, three are used for handshaking, and the remaining 5 are used for status and control information.

Some **Advantages**:

- Standard hardware interface.
- Simplicity.
- Interface present on many bench instruments even from different manufacturers.
- A single controller can control multiple instruments.

Disadvantages:

- Bulky connector.
- Low bandwidth compared to modern interfaces.
- It does not includes a common command language.

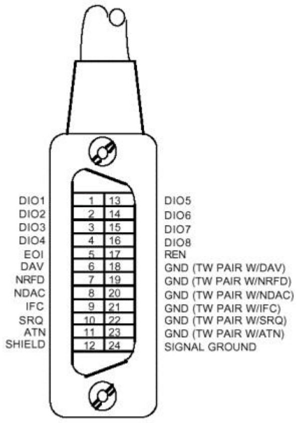


Figure 3.28 – *GPIB connector pinout. Source [NI]*

3

3.3.2.0.1 Functional Concepts

A least three functional elements will be needed for communications with data exchange within the bus:

- **Controller**: It is the device that manages and control the data flow. For more than one controller a CIC is mandatory. No communication is possible without controllers.
- **Listener**: One device must be listening to the instructions or data coming from the bus. It is possible to have several listeners at the same time.
- **Talker**: One device sending information to the bus. More than one talker at the same time is not allowed.

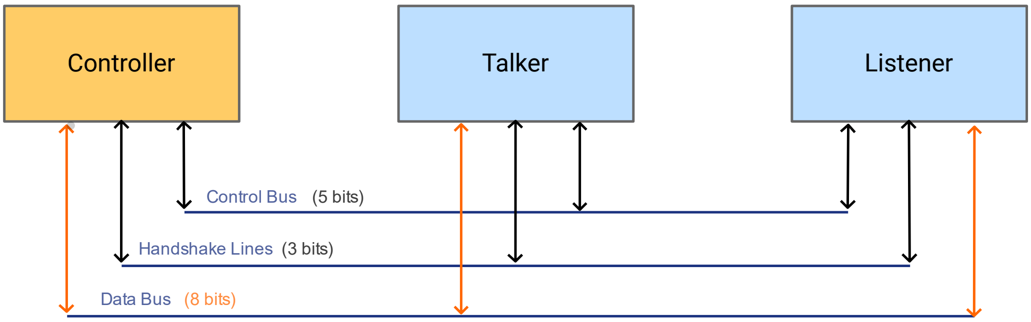


Figure 3.29 – *GPIB System.*

3.3.2.0.2 GPIB Operation

The GPIB operation is build around it handshaking protocol controlled by the DAV, NDAC, and the NRFD lines. The NRFD line is used by the listeners to indicate their state of readiness to accept data by holding low the line. If the line is low no data transfer take place. If the NRFD line is high it means that all the instruments are ready and data transmission can be initiated. The talker is who place the data on line and pull low the DAV line, which signal the listeners they are able to acquired the data on the bus. Those listeners instructed to receive the data will hold low the NDAC line. When the listeners have read the data each device stop to hold the line NDAC low. This line will rise when the last device removes its hold, which means all data being accepted. Then the talker knows it and the next byte can be transferred[34].

3

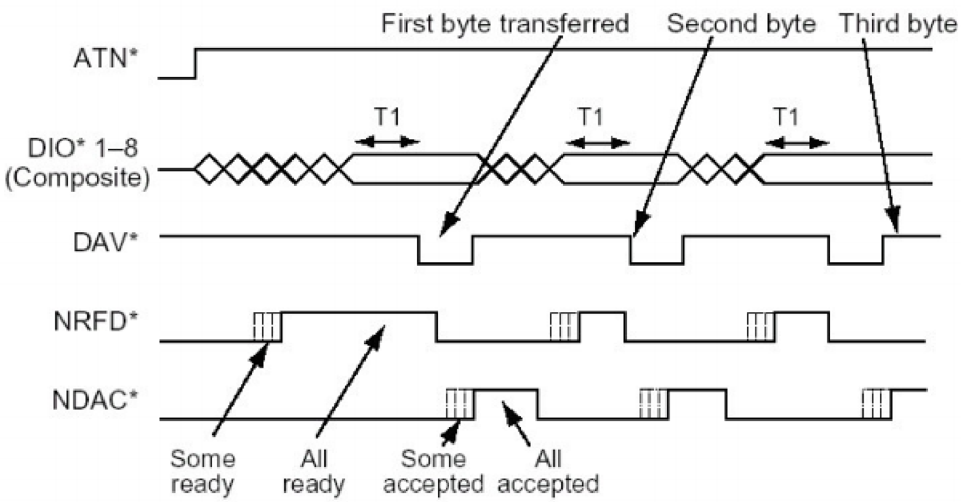


Figure 3.30 – *GPIB Handshake.*

More details can be found on the IEEE-488.2 Standard.

3.3.3 Agilent GPIB Driver

In this section the Agilent GPIB libraries installation and the connection between the equipment will be explained. Firstly, the *IO Agilent Libraries Suite v16.3* must be installed. This library controls the I/O ports of the workstation and enables communication between disparate instrumentation equipment over GPIB, RS-232, USB, LAN, etc. Once the suite has been installed, the equipment is connected. Connection cables such as the Keysight 10833A GPIB cable are used. Some restrictions about the length of the cables and the number of connected element are stated by Agilent in order to maintain certain data rates in the bus. For transfer rates higher than 500$kbps$ less than 1 meter cable for 3 interconnected equipments.

Figure 3.31 – *Agilent 82357A GPIB/USB connector.*

The interface communication between the GPIB buses and the workstation is done by the *Agilent 82357A GPIB/USB* connector which allows up to 14 connected equipments. In the Figure 3.32 the connection scheme between the PC and the equipment shown.

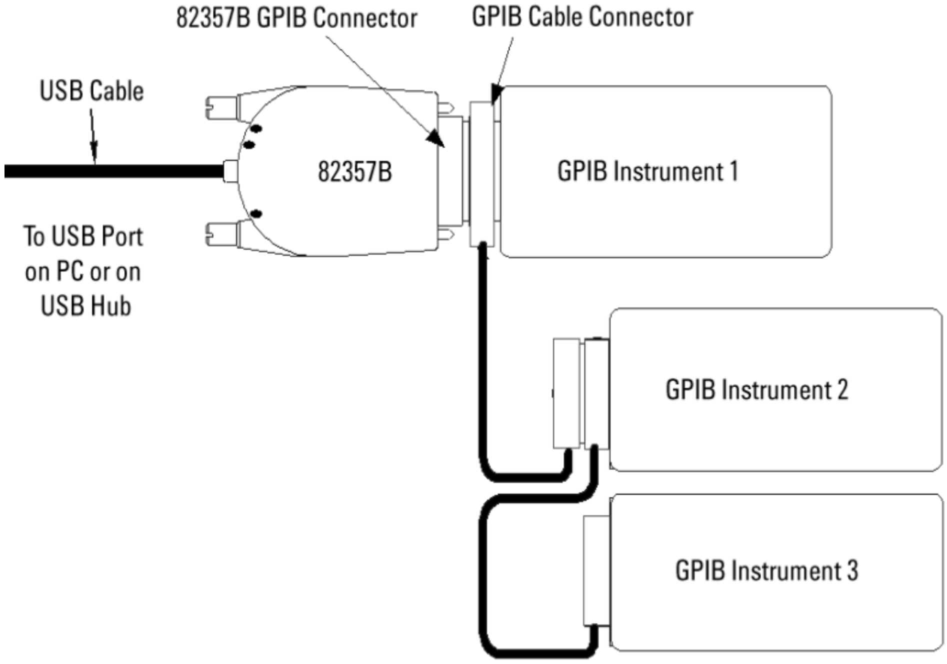


Figure 3.32 – *GPIB-USB controller to connect multiple instruments[34].*

CHAPTER

4 SYSTEM DESIGN

In this chapter, each system and subsystem previously analysed in Chapter 3 will be designed and implemented. Furthermore, both, the constraints and requirements extracted in former sections, will be taken into account in the design. This includes sections 3.1.1.1, 3.1.1.2 and 3.1.2 as well as the design analysis included in sections 3.2 and 3.3.

The TVAC subsystem will be firstly detailed starting with its structural shape, the vacuum and thermal system designs and the instrumentation equipment chosen for data acquisition. Drawings and 3D model will be attached as a part of the design process.

Then the PV system solution will be explained and detailed, including the solar simulator calibration as well as the software tools and libraries used to control the equipments and characterize the panels. Thus, the structures of the GPIB communication libraries, as well as the MATLAB Instrument Control Toolbox will be also described.

Finally, the developed GUI software tool which controls the equipment, represents the information and plot the I-V and PV curves will be defined and explained.

4.1 TVAC System Design

First, the TVAC system will be designed in accordance with the analysis performed in Section 3.2.2, including the internal and external structures mentioned in 3.1.1.2.

The mechanical structure of the TVAC Chamber will be designed by using the CAD/CAE solid modelling computer program known as Solidworks. This CAD tool will allow the mechanical design of the chamber providing 3D models and sketches of the product to be then manufactured.

Before the final drawings of the TVAC Chamber were generated, an important decision involving the chamber's structure was made. This is the aforementioned shape of the chamber. A bad decision might result in an unexpectedly large cost overrun in the program. In the Figure 4.1 the isometric views of both branches, the spherical and the box shaped, are shown.

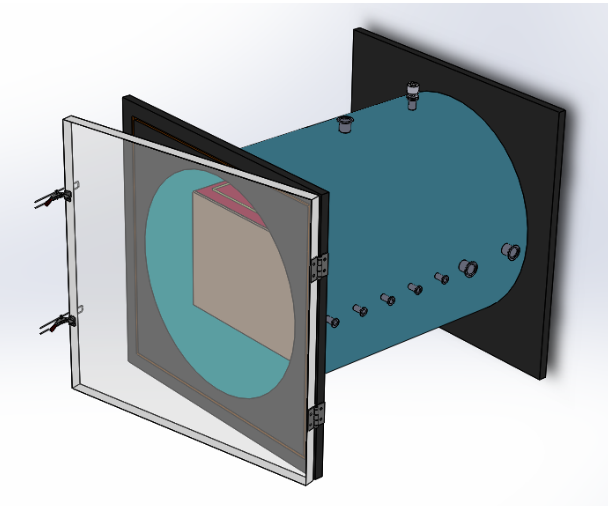


Figure 4.1 – *Spherical TVAC branch model. Right Side.*

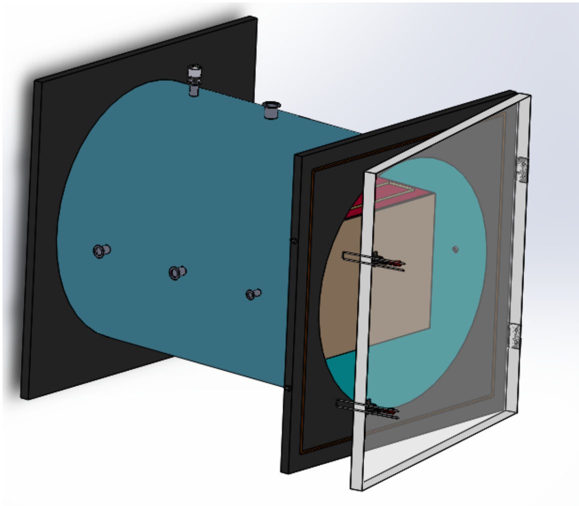


Figure 4.2 – *Spherical TVAC branch model. Left Side*

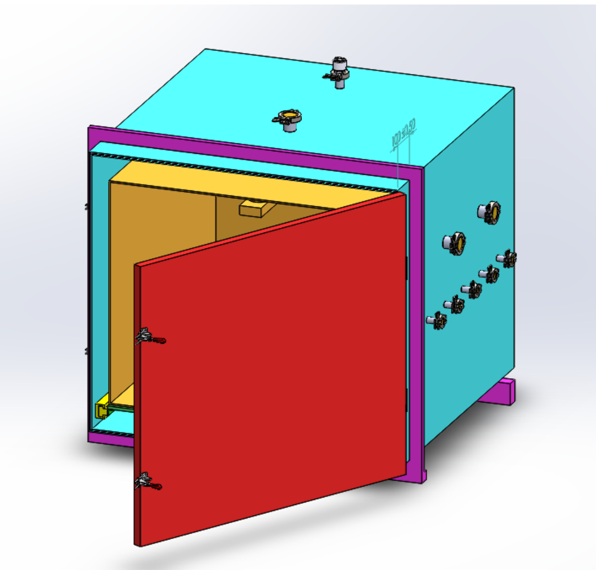


Figure 4.3 – *Box shaped TVAC branch model. Right Side.*

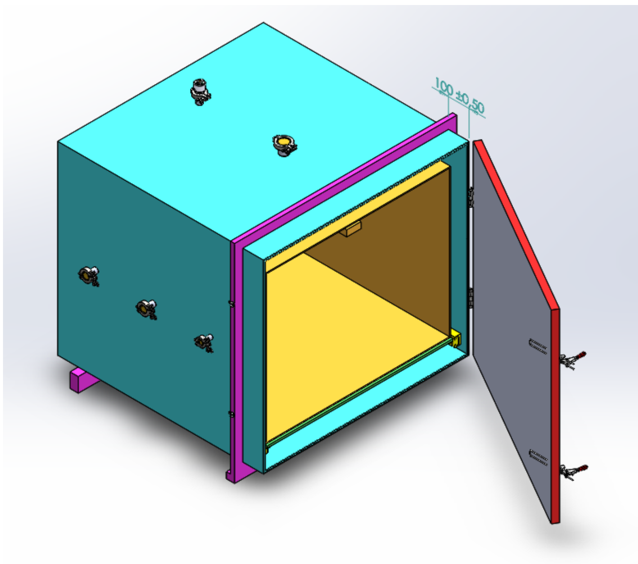
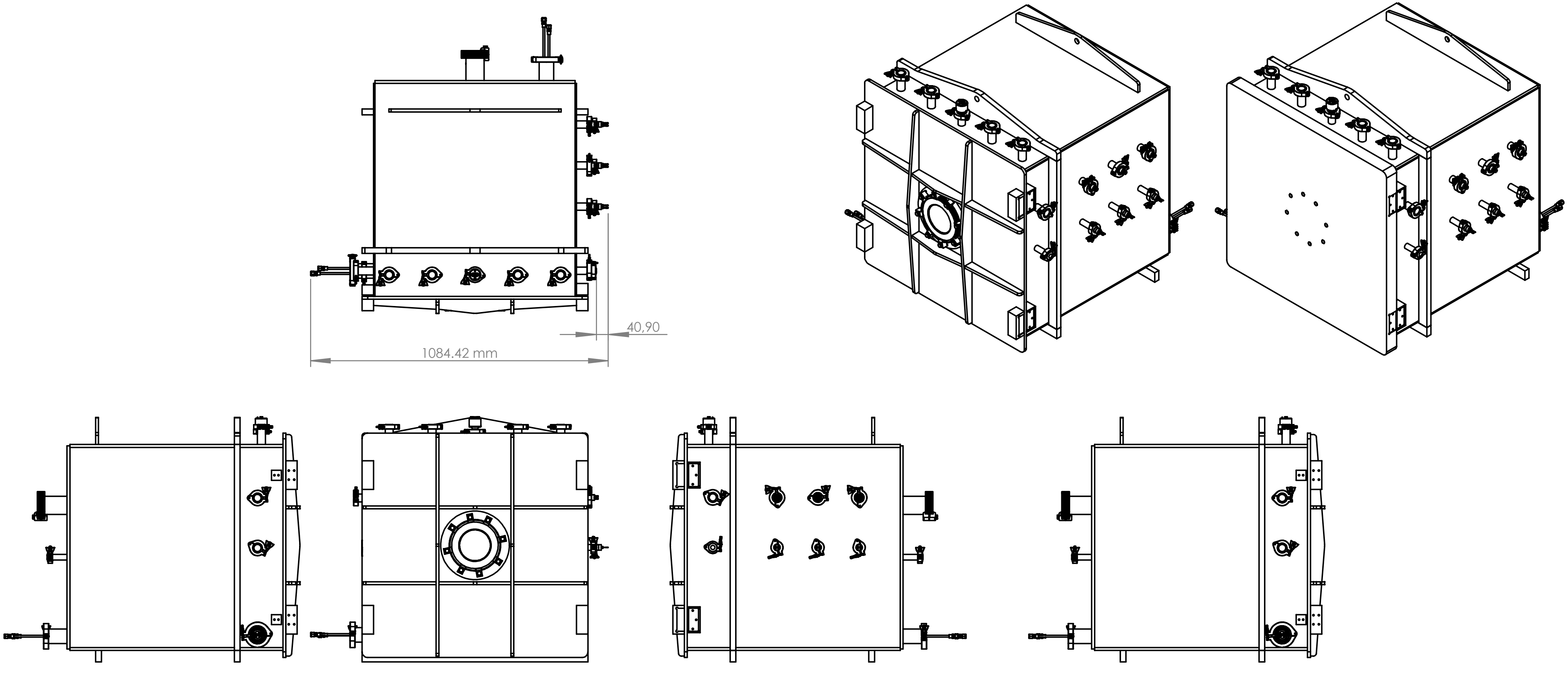


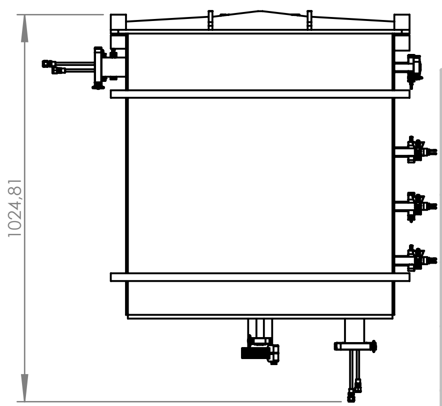
Figure 4.4 – *Box shaped TVAC branch model. Left Side*

Finally, after some discussions, it was concluded that the box shaped TVAC Chamber fits better with the project. Therefore, the box shaped chamber shown in the Figure 4.3 and 4.4 has been be the basis on which the final designs have been be developed.

Notice that the final model of the Chamber has been redesigned and improved up to a total of 6 times. Thus, up to 6 versions have been developed as result of the communication with *Mecanizados Granada SL* and always following its advices and the design and mechanical constraints. Each design decision has been made taken into account the several trade-off involved.

In the following pages the mechanical design of the TVAC Chamber is shown.





SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

TÍTULO:

CÁMARA TVAC
MEDIDAS GENERALES

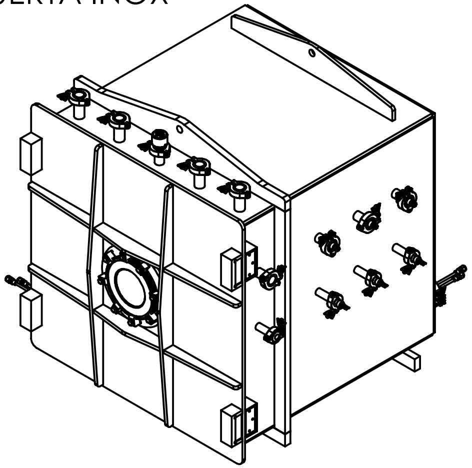
N.º DE DIBUJO
TVAC_drawings

A4

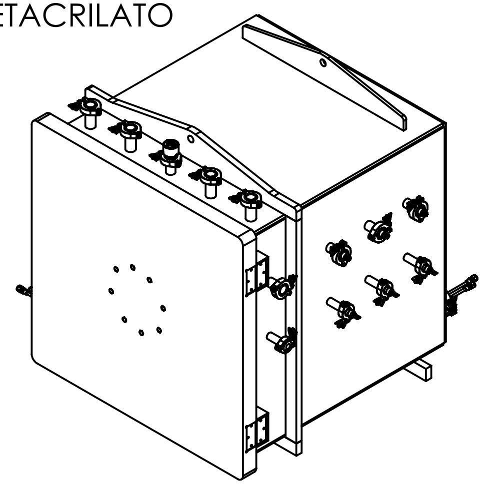
ESCALA:1:20

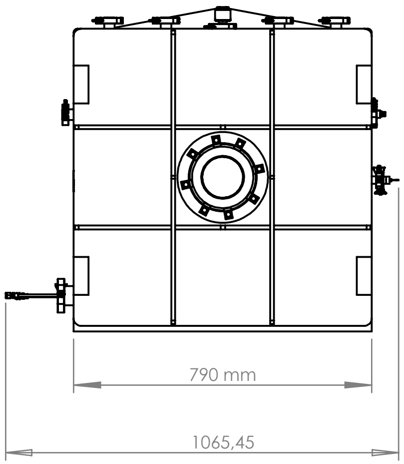
HOJA 1 DE 19

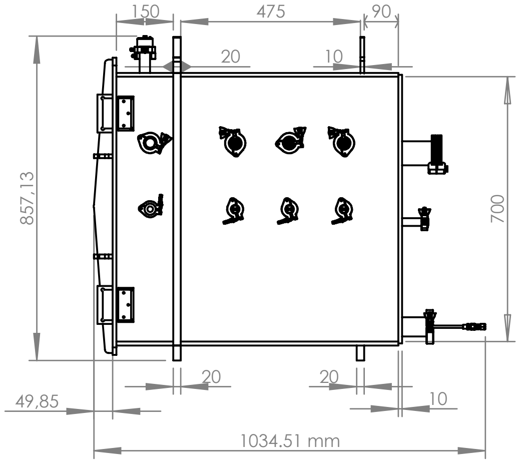
CONFIGURACIÓN PUERTA INOX

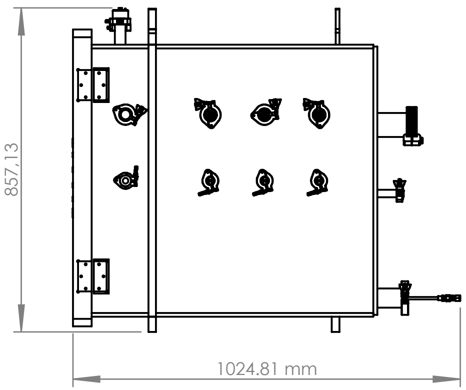


CONFIGURACIÓN PUERTA METACRILATO

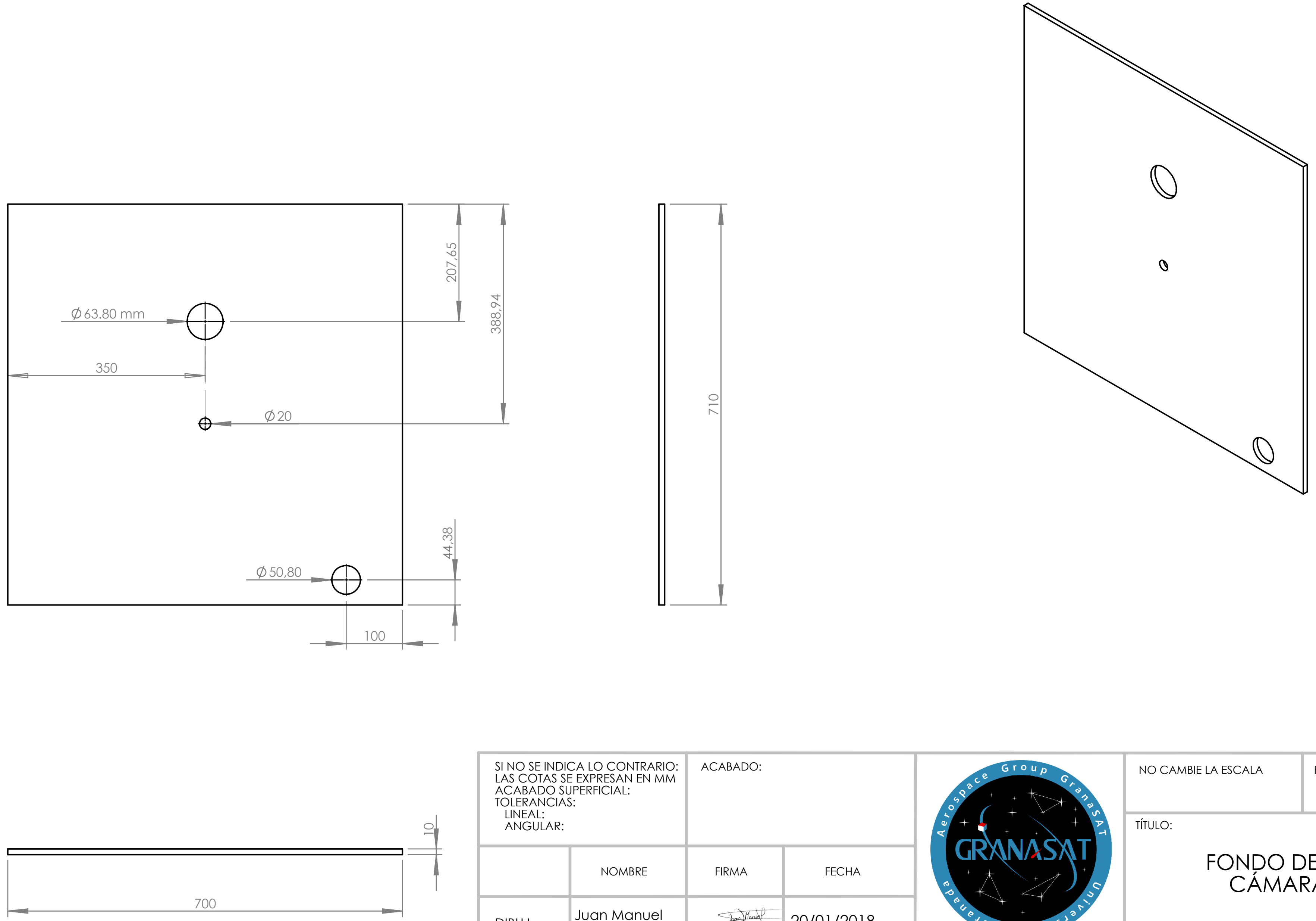








SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			NO CAMBIE LA ESCALA	REVISIÓN: 5
	NOMBRE	FIRMA	FECHA		TÍTULO: CÁMARA TVAC DETALLE DE CONFIGURACIONES	
DIBUJ.	Juan Manuel López Torralba		20/01/2018			
VERIF.	Andrés María Roldán Aranda		20/01/2018	Dpto. Electrónica y Tecnología de Computadores.	N.º DE DIBUJO TVAC_drawings	A4
APROB.	Andrés María Roldán Aranda		20/01/2018	University of Granada C/ Fuente Nueva s/n 18001 Granada, Spain	ESCALA:1:20	HOJA 2 DE 19



SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

TÍTULO:

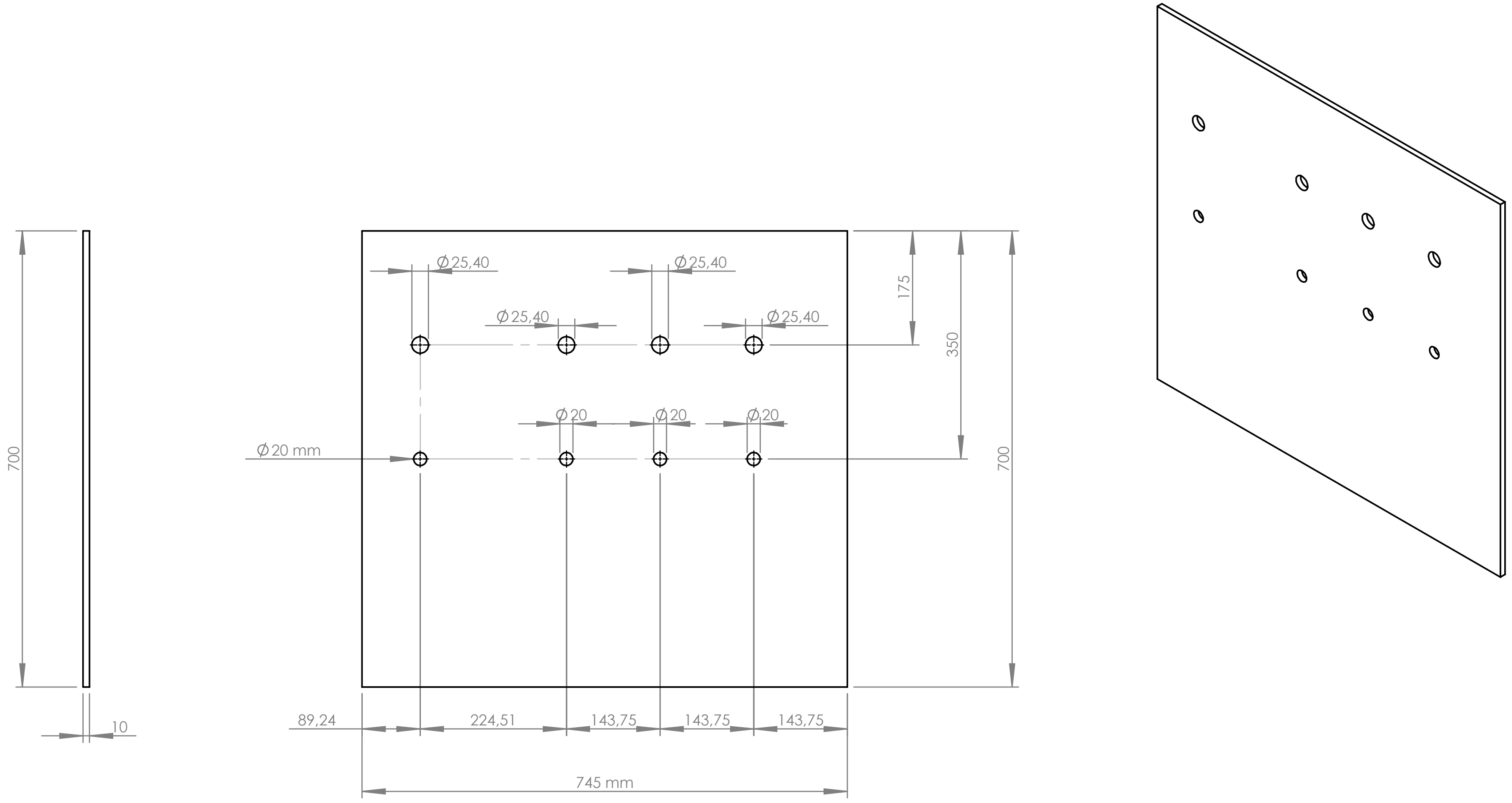
FONDO DE LA CÁMARA

N.º DE DIBUJO
TVAC_drawings

A4

ESCALA:1:10

HOJA 3 DE 19



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		
	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

TÍTULO:

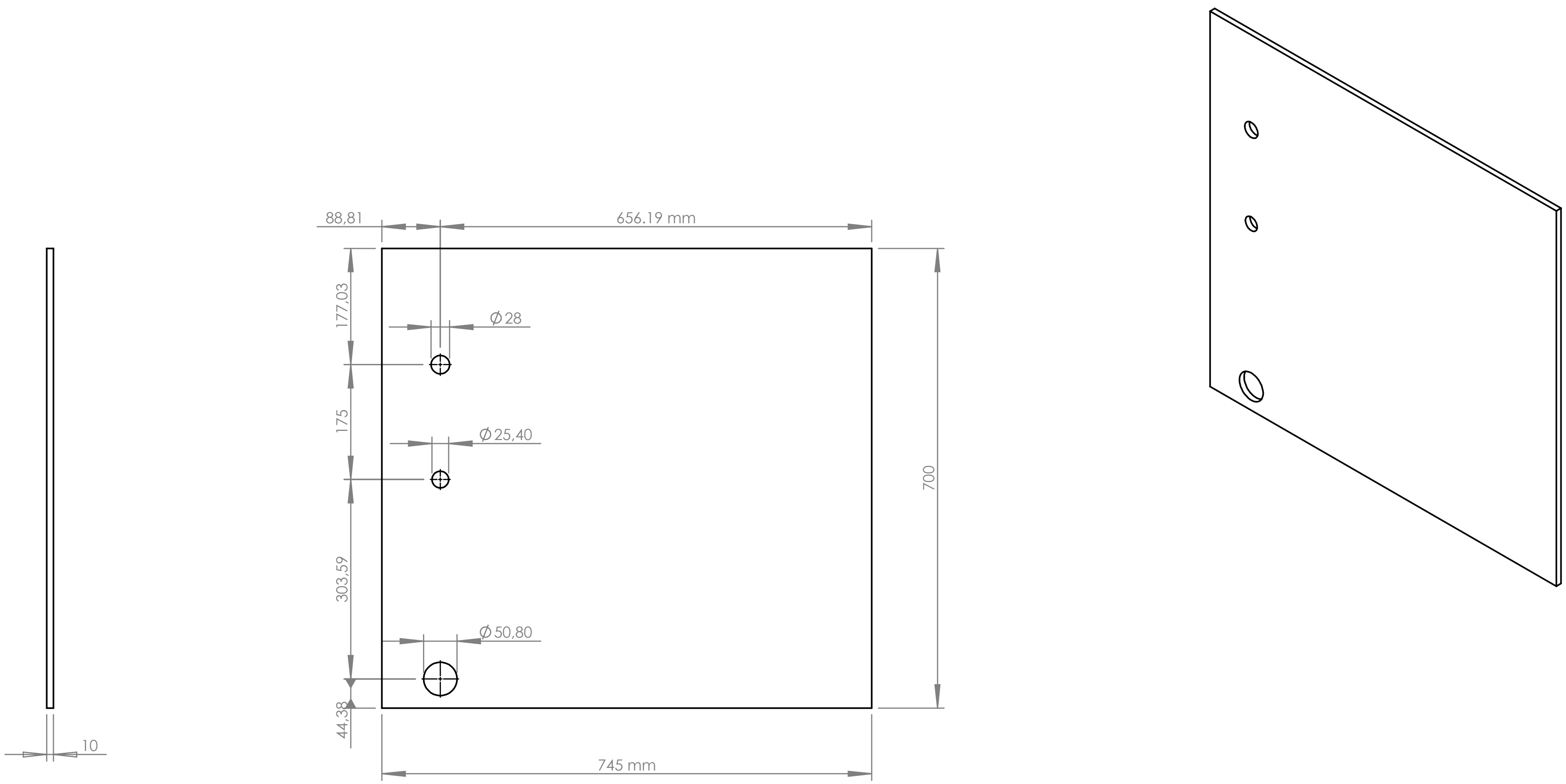
LATERAL DERECHO DE LA CÁMARA

N.º DE DIBUJO: TVAC_drawings

A4

ESCALA:1:10

HOJA 4 DE 19

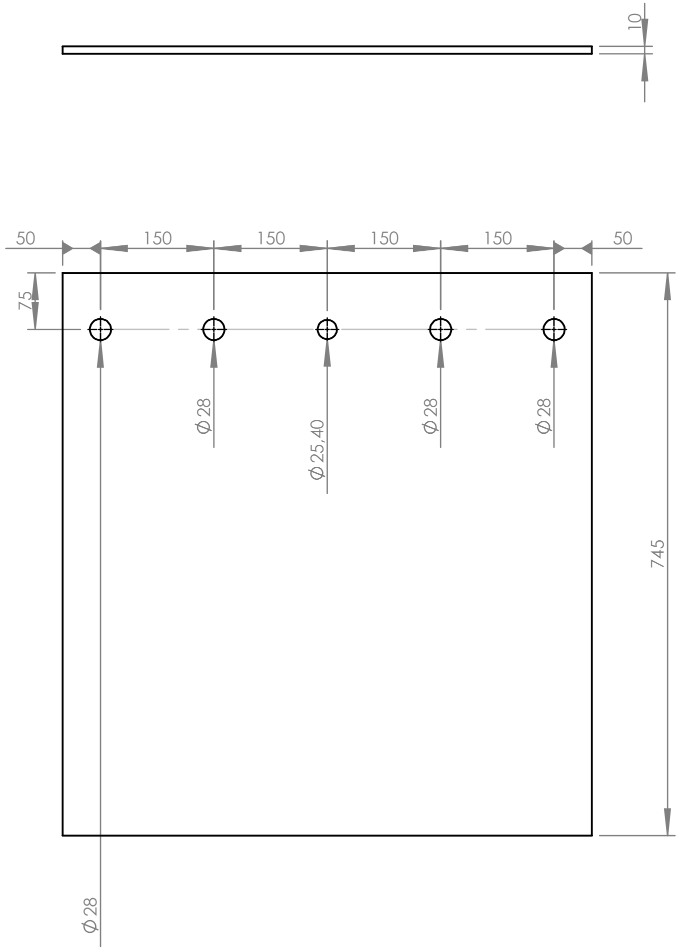


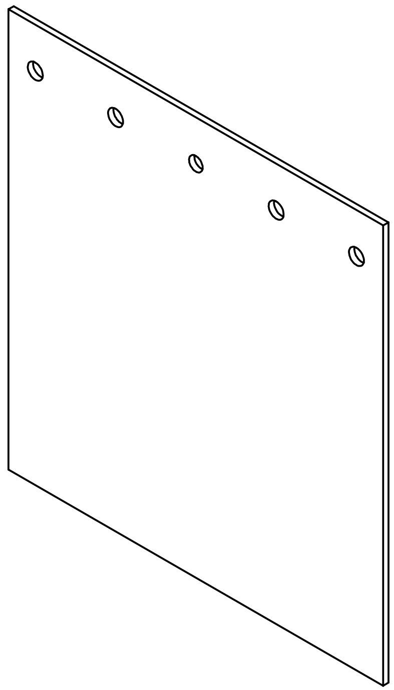
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		
	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA	REVISIÓN: 5
TÍTULO: LATERAL IZQUIERDO DE LA CÁMARA	
N.º DE DIBUJO TVAC_drawings	A4
ESCALA:1:10	HOJA 5 DE 19



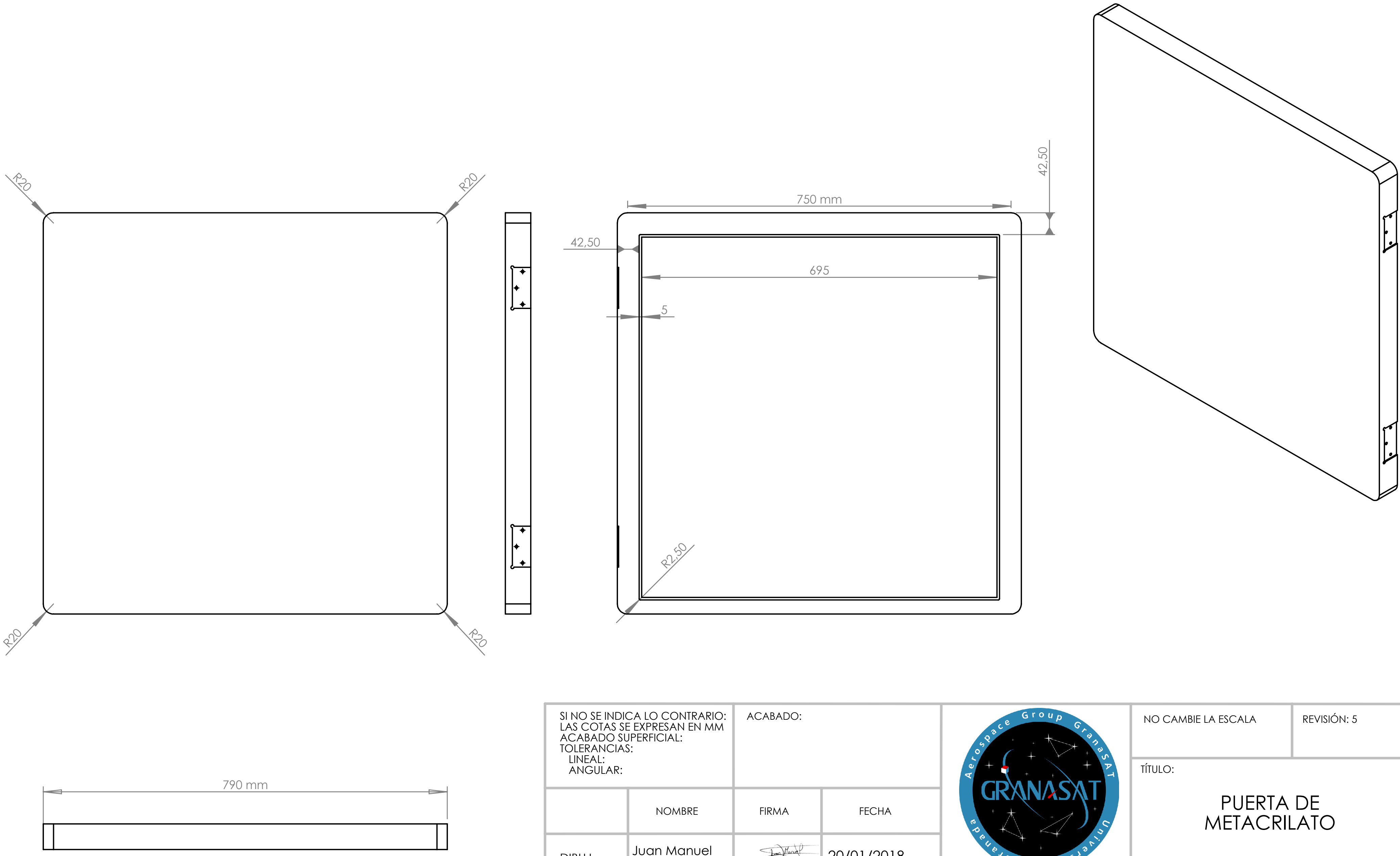


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:	
	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA	REVISIÓN: 5
TÍTULO: TAPA SUPERIOR	
N.º DE DIBUJO TVAC_drawings	A4
ESCALA:1:10	HOJA 6 DE 19



SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Aerospace Group GranaSAT University of Granada

GRANASAT

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

TÍTULO:

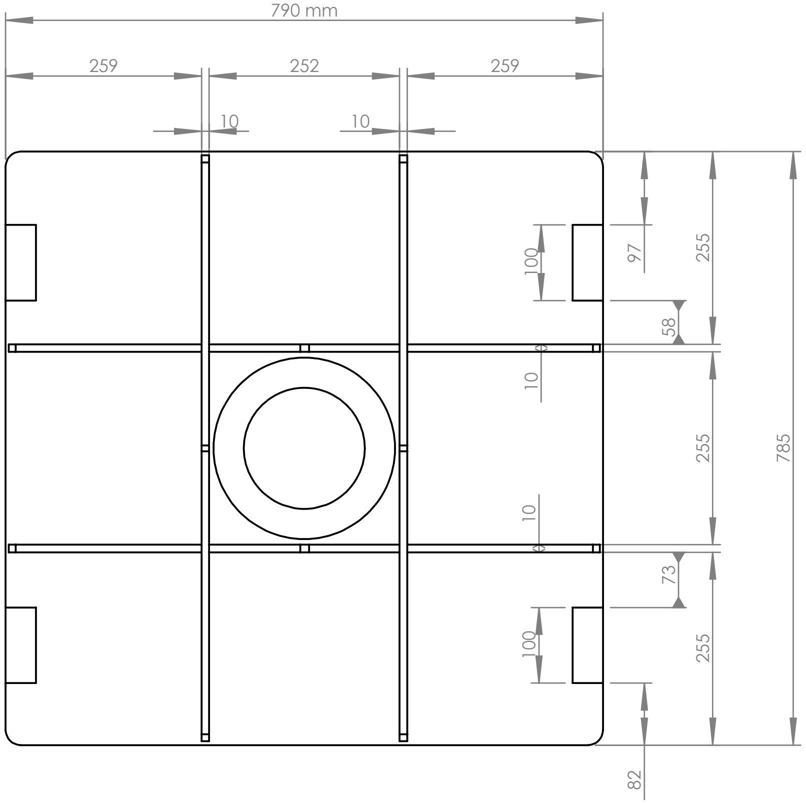
PUERTA DE METACRILATO

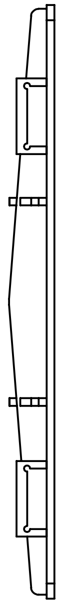
N.º DE DIBUJO
TVAC_drawings

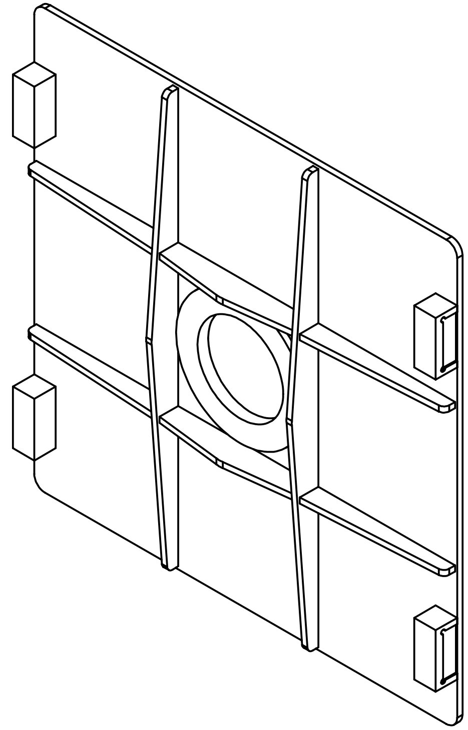
A4

ESCALA:1:10

HOJA 7 DE 19





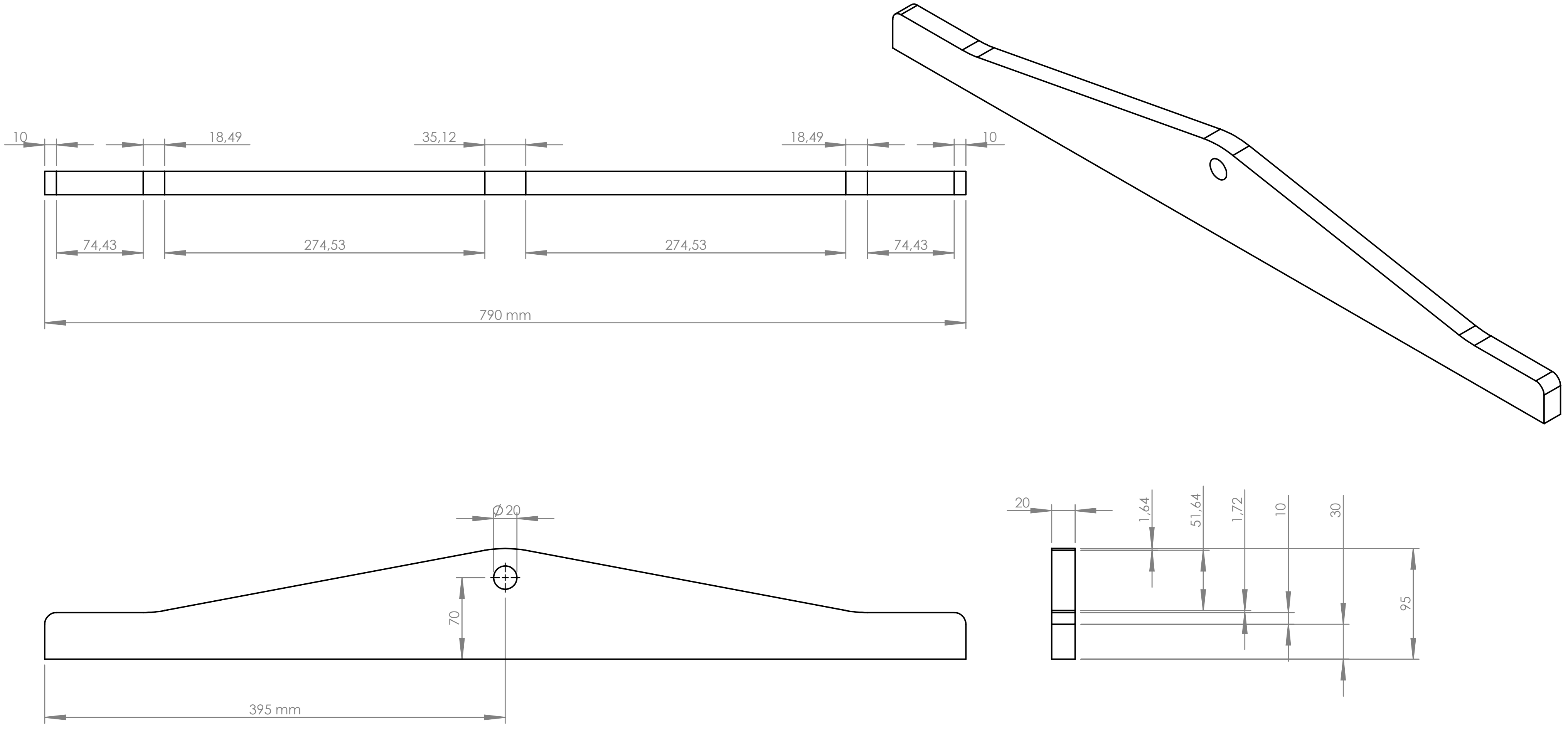


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		
	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA	REVISIÓN: 5
TÍTULO: PUERTA DE INOX	
N.º DE DIBUJO: TVAC_drawings	A4
ESCALA:1:10	HOJA 8 DE 19



SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.
University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

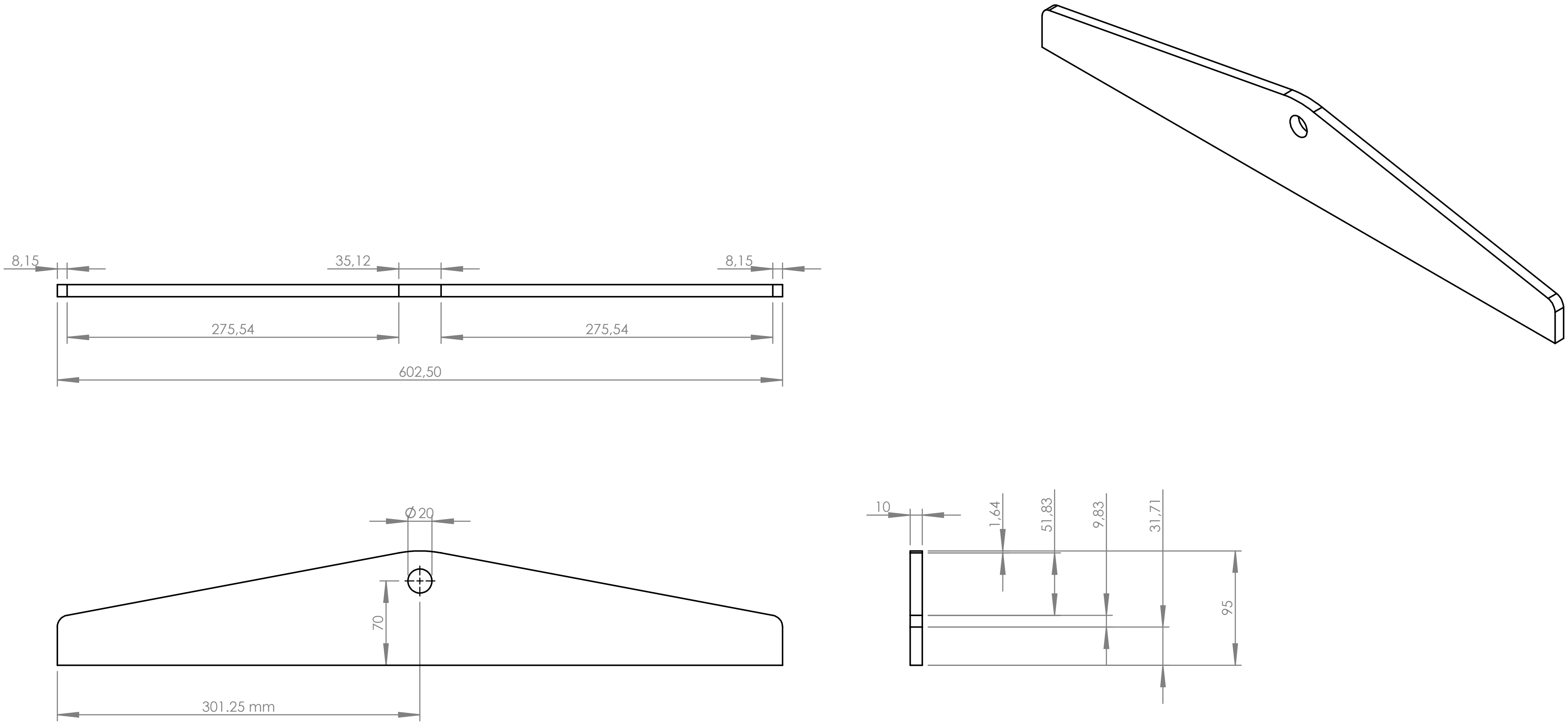
TÍTULO:

LARGUERO SUPERIOR

N.º DE DIBUJO
TVAC_drawings

A4

ESCALA:1:5



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:	
	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

TÍTULO:

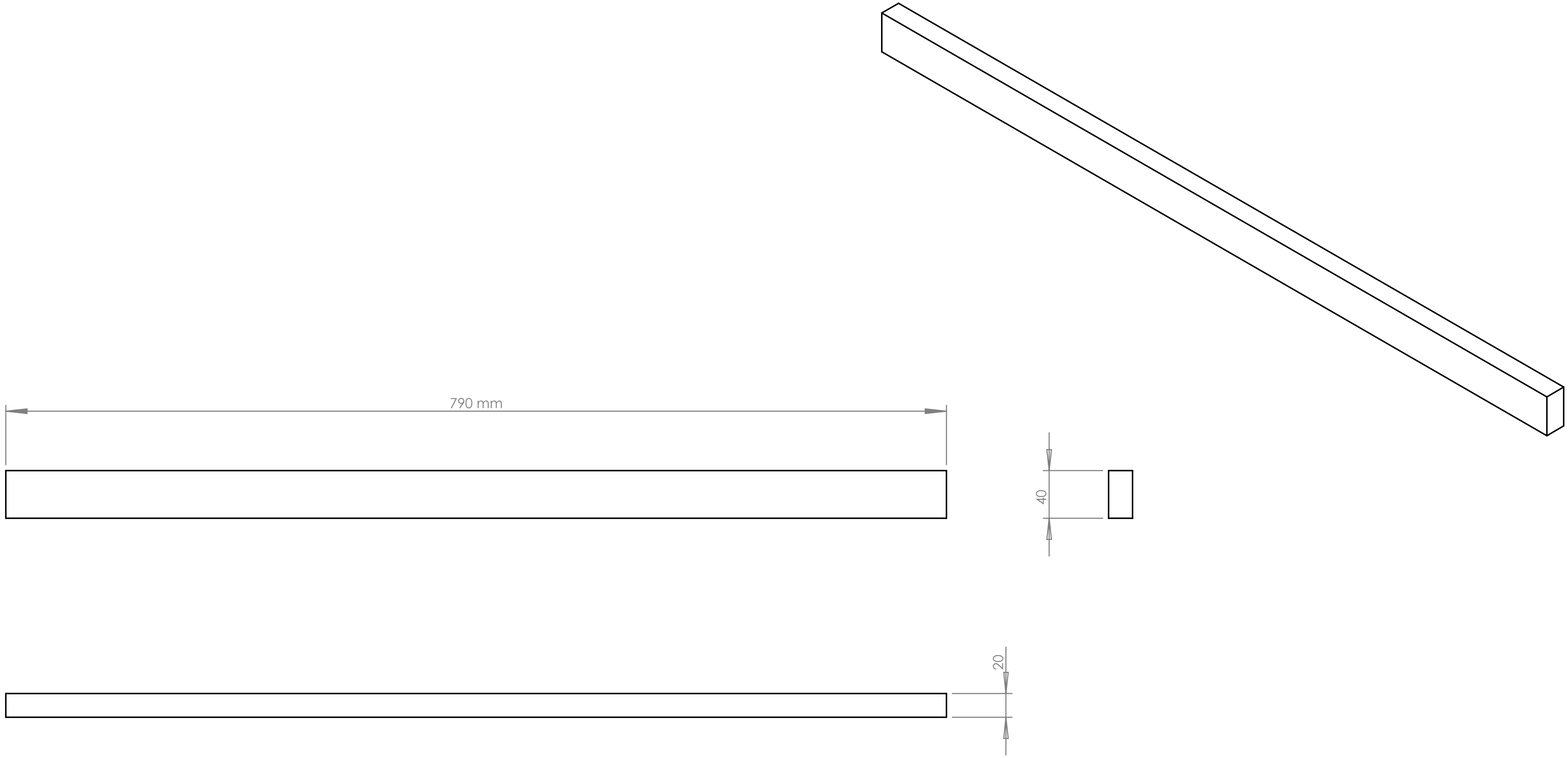
SOPORTE TRASERO ELEVACIÓN

N.º DE DIBUJO: TVAC_drawings

A4

ESCALA:1:5

HOJA 10 DE 19



2 unidades inox de 20

SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

TÍTULO:

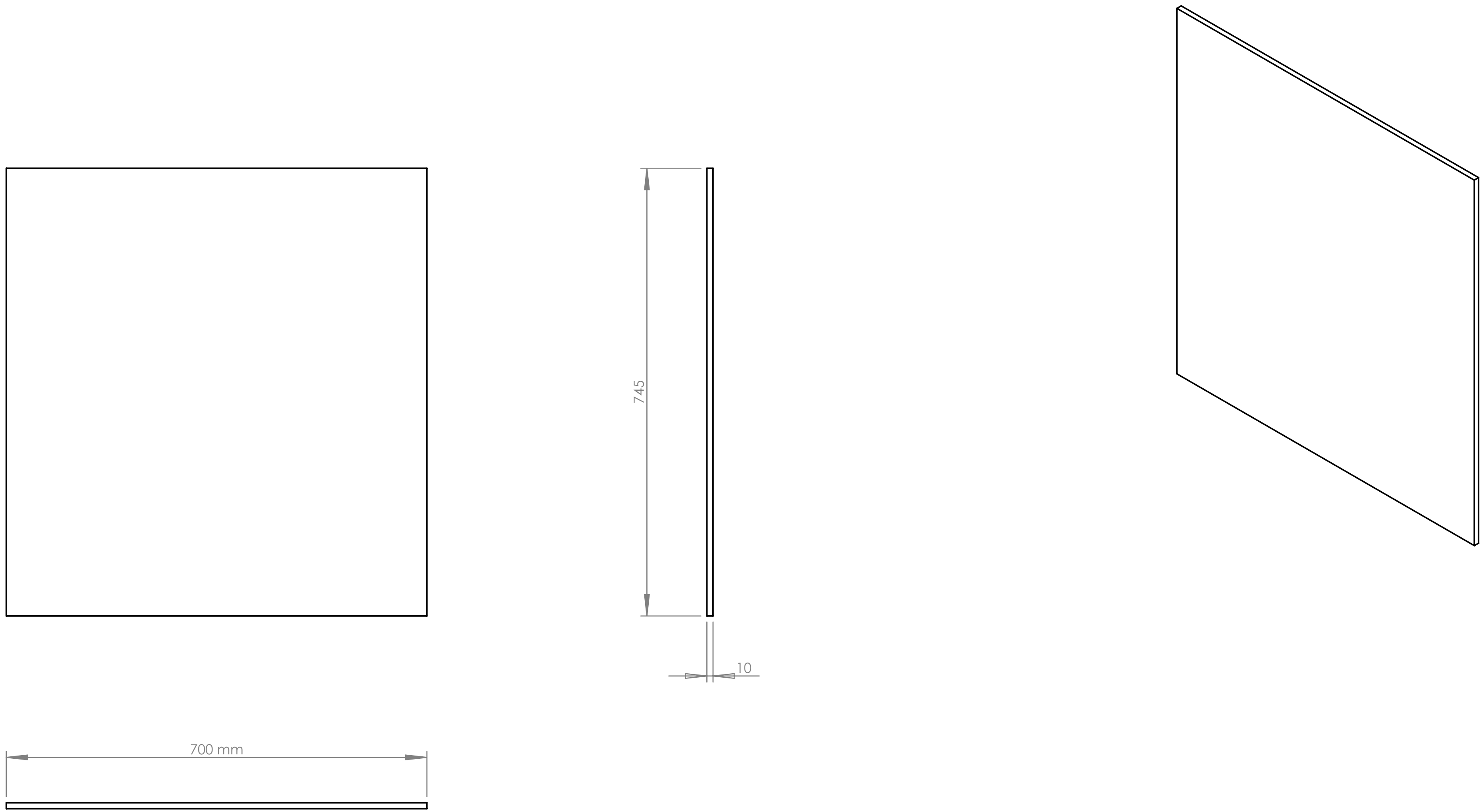
BASE MARCO

N.º DE DIBUJO
TVAC_drawings

A4

ESCALA:1:5

HOJA 11 DE 19

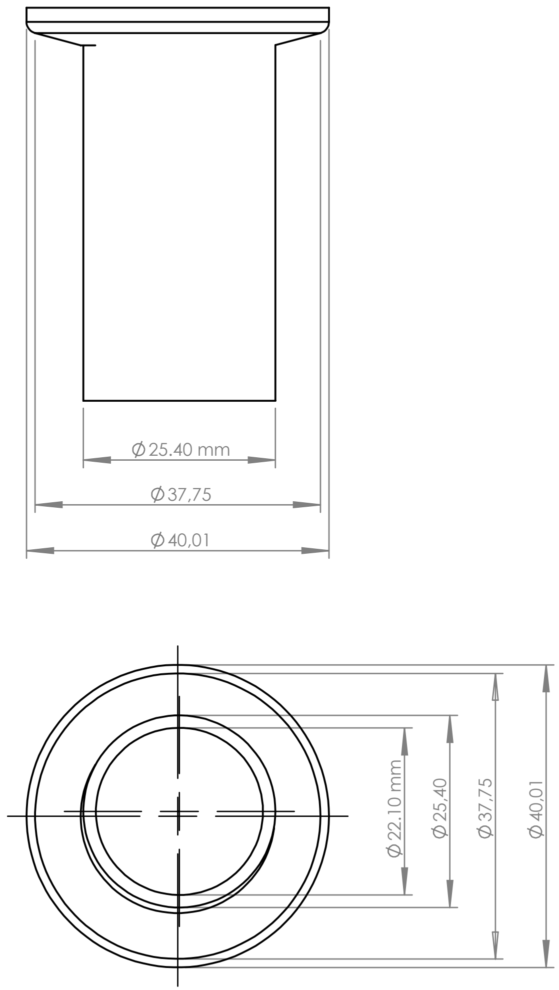


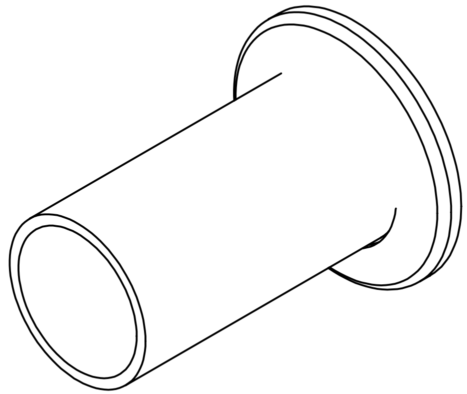
5 UNIDADES INOX DE 10

SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:			NO CAMBIE LA ESCALA	REVISIÓN: 5
	NOMBRE	FIRMA	FECHA	TÍTULO: BASE	
DIBUJ.	Juan Manuel López Torralba		20/01/2018		
VERIF.	Andrés María Roldán Aranda		20/01/2018	N.º DE DIBUJO TVAC_drawings	A4
APROB.	Andrés María Roldán Aranda		20/01/2018	ESCALA:1:10	HOJA 12 DE 19

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain



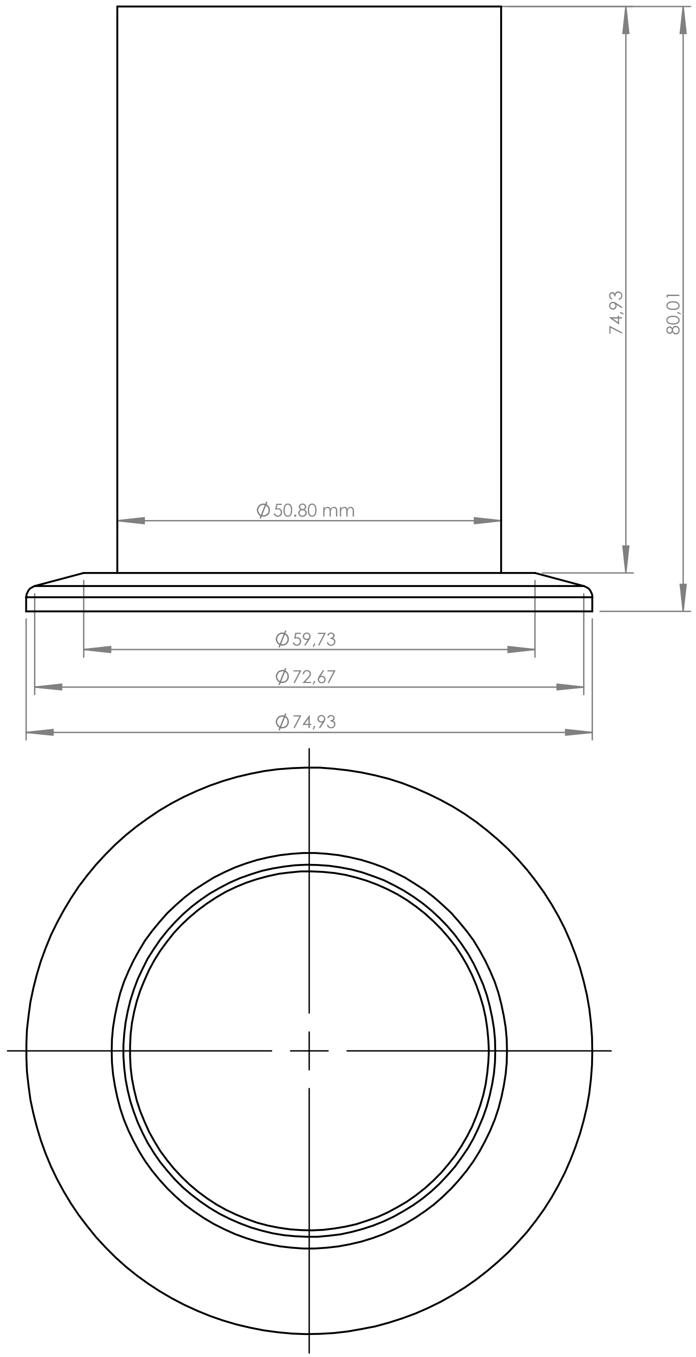


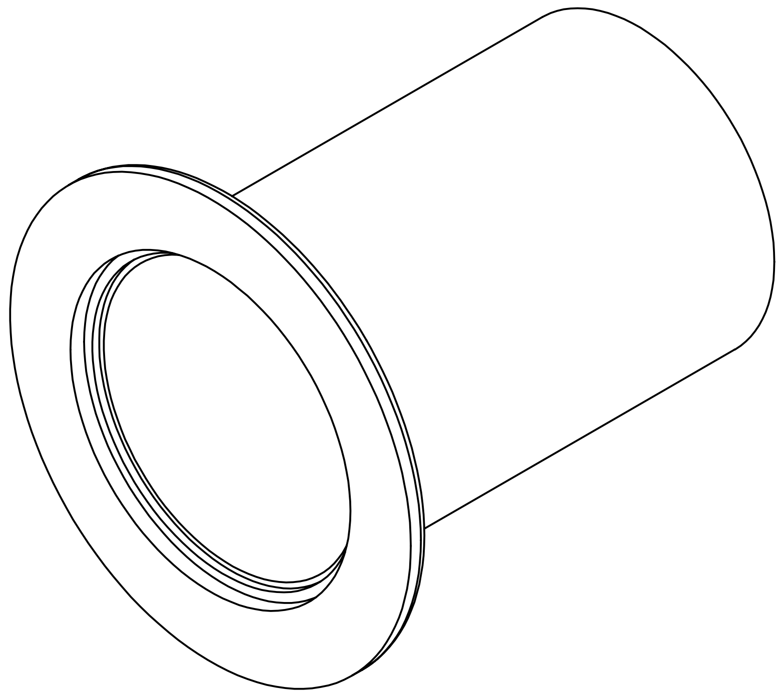
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:	
	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA	REVISIÓN: 5
TÍTULO: 1N-NW-25B	
N.º DE DIBUJO TVAC_drawings	A4
ESCALA:1:1	HOJA 13 DE 19



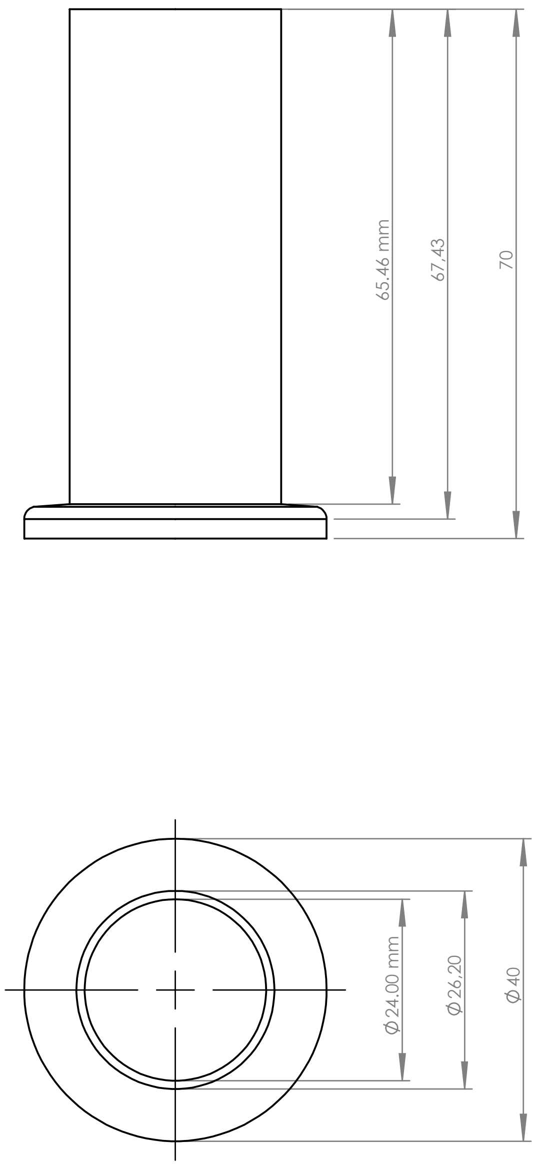


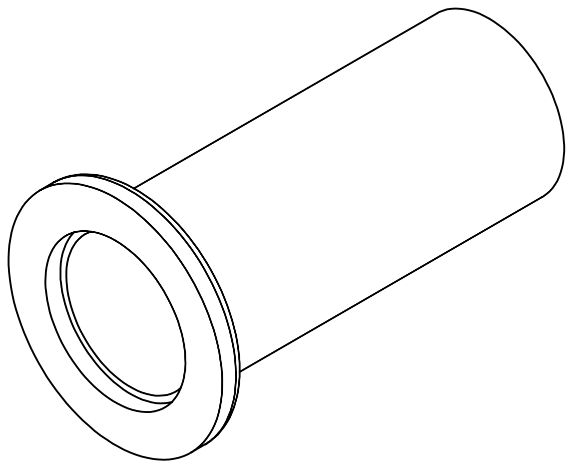
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		
	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.

University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA	REVISIÓN: 5
TÍTULO: 1N-NW-50B	
N.º DE DIBUJO: TVAC_drawings	A4
ESCALA:1:1	HOJA 14 DE 19





SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

	NOMBRE	FIRMA	FECHA
DIBUJ.	Juan Manuel López Torralba		20/01/2018
VERIF.	Andrés María Roldán Aranda		20/01/2018
APROB.	Andrés María Roldán Aranda		20/01/2018

Dpto. Electrónica y Tecnología de Computadores.
University of Granada
C/ Fuente Nueva s/n 18001
Granada, Spain

NO CAMBIE LA ESCALA

REVISIÓN: 5

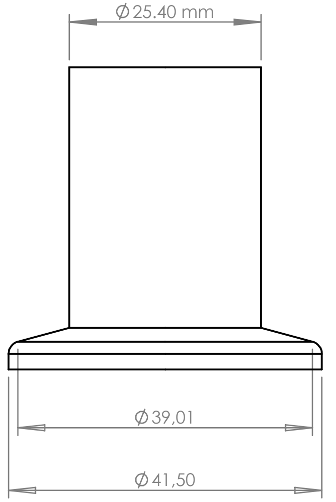
TÍTULO:

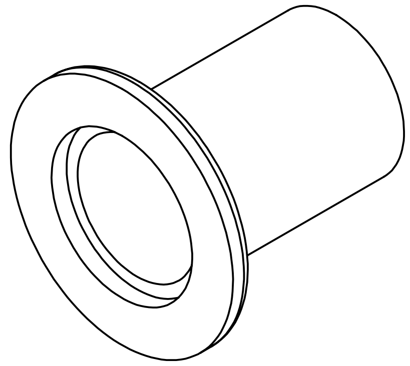
NW-25FL-M316L

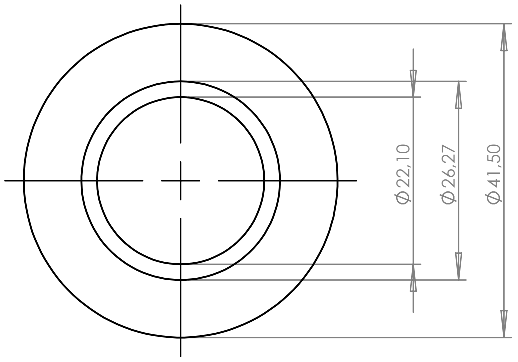
N.º DE DIBUJO
TVAC_drawings

A4

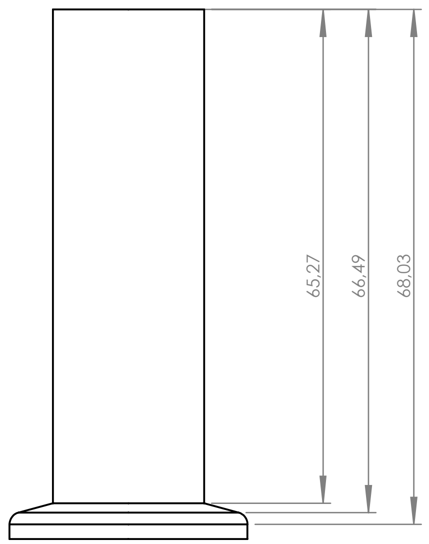
ESCALA:1:1

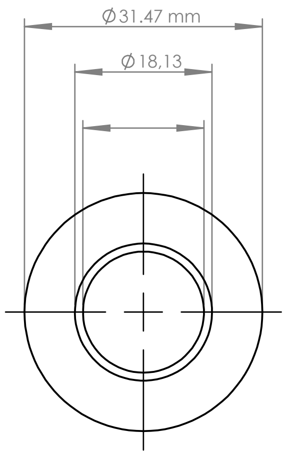


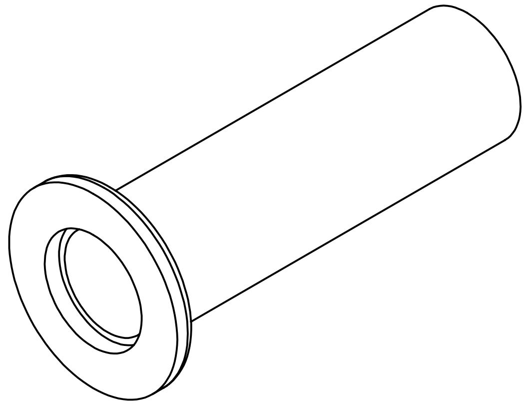




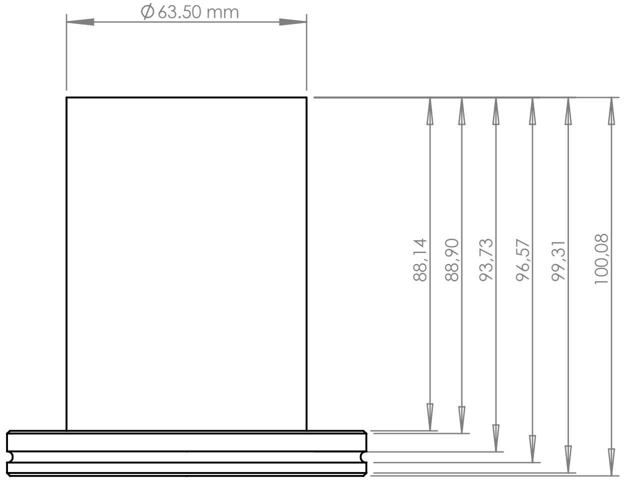
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			NO CAMBIE LA ESCALA	REVISIÓN: 5
	NOMBRE	FIRMA	FECHA		TÍTULO: NW-25FL-316L	
DIBUJ.	Juan Manuel López Torralba		20/01/2018			
VERIF.	Andrés María Roldán Aranda		20/01/2018	Dpto. Electrónica y Tecnología de Computadores. University of Granada C/ Fuente Nueva s/n 18001 Granada, Spain	N.º DE DIBUJO TVAC_drawings	A4
APROB.	Andrés María Roldán Aranda		20/01/2018		ESCALA:1:1	HOJA 16 DE 19

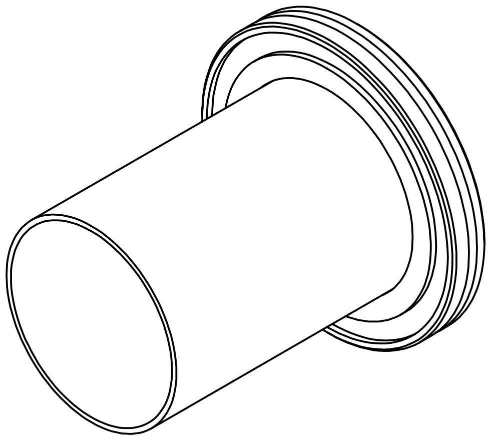


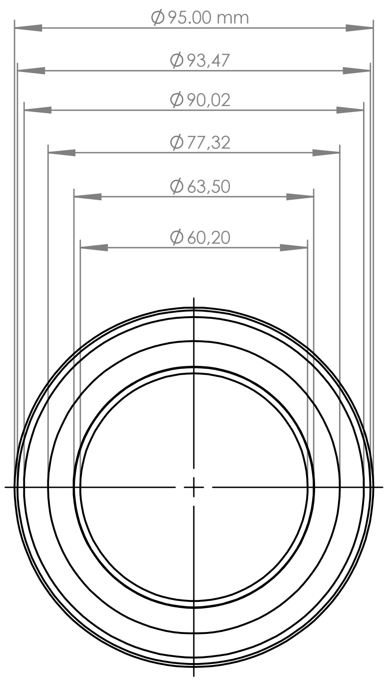




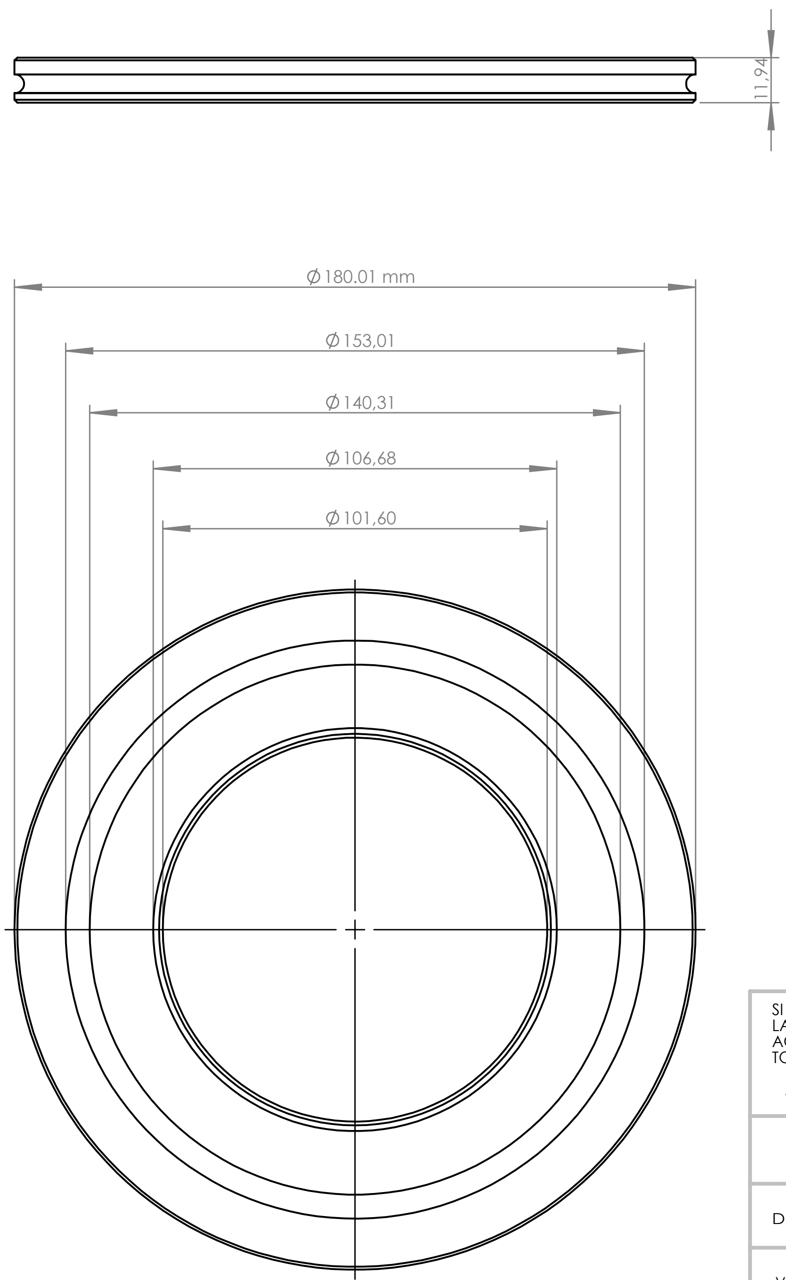
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			NO CAMBIE LA ESCALA	REVISIÓN: 5
	NOMBRE	FIRMA	FECHA		TÍTULO: NW-16FL-M316L	
DIBUJ.	Juan Manuel López Torralba		20/01/2018			
VERIF.	Andrés María Roldán Aranda		20/01/2018	Dpto. Electrónica y Tecnología de Computadores.	N.º DE DIBUJO TVAC_drawings	A4
APROB.	Andrés María Roldán Aranda		20/01/2018	University of Granada C/ Fuente Nueva s/n 18001 Granada, Spain	ESCALA:1:2	HOJA 17 DE 19

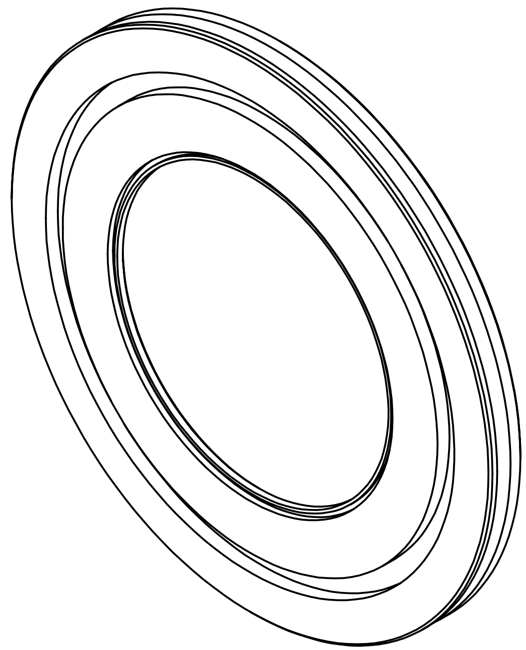






SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			NO CAMBIE LA ESCALA	REVISIÓN: 5
	NOMBRE	FIRMA	FECHA		TÍTULO: 1-N-ISO-63-OF-V	
DIBUJ.	Juan Manuel López Torralba		20/01/2018			
VERIF.	Andrés María Roldán Aranda		20/01/2018	Dpto. Electrónica y Tecnología de Computadores. University of Granada C/ Fuente Nueva s/n 18001 Granada, Spain	N.º DE DIBUJO TVAC_drawings	A4
APROB.	Andrés María Roldán Aranda		20/01/2018		ESCALA:1:2	HOJA 18 DE 19





SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			NO CAMBIE LA ESCALA	REVISIÓN: 5
	NOMBRE	FIRMA	FECHA		TÍTULO: LVP-160 Viewport	
DIBUJ.	Juan Manuel López Torralba		20/01/2018			
VERIF.	Andrés María Roldán Aranda		20/01/2018	Dpto. Electrónica y Tecnología de Computadores. University of Granada C/ Fuente Nueva s/n 18001 Granada, Spain	N.º DE DIBUJO TVAC_drawings	A4
APROB.	Andrés María Roldán Aranda		20/01/2018		ESCALA:1:2	HOJA 19 DE 19

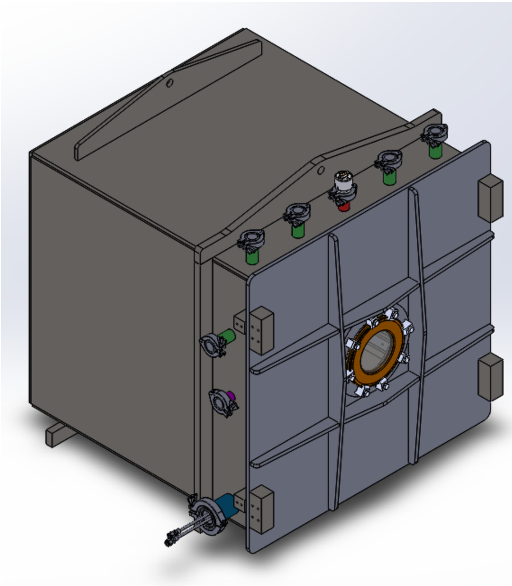


Figure 4.5 – *TVAC Chamber with the methacrylate door.*

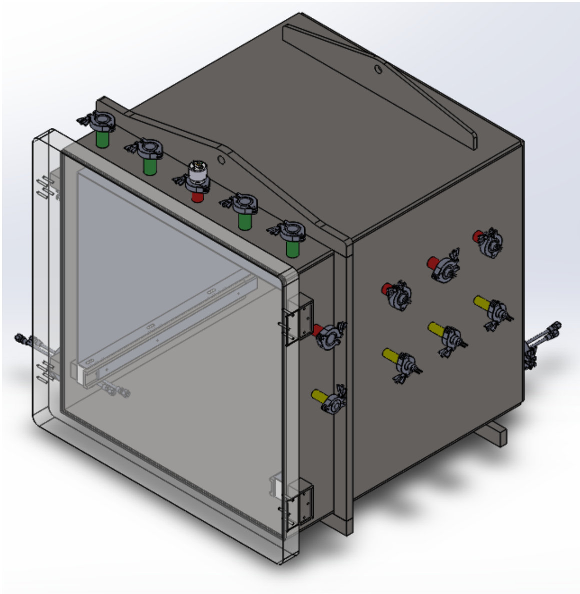


Figure 4.6 – *TVAC Chamber with the 304 stainless steel door*

4

Figure 4.7 – *TVAC Chamber exploded view*

4.1.1 Structure

The designed TVAC Chamber weighs a total of 260.4 kg and has a capacity of 372.7 l. The parts of the chamber which are exposed to vacuum such as chamber walls, shrouds, gates as well as the feed-though ports are made of AISI 304 SS[24]. This choice have been made considering the out-gassing rates and high strength of the AISI 304 SS[23].The same SS have been used for the assembly of the chamber walls[22]. As said in 3.2.2.1, the chamber includes two gates configuration capabilities, 10 mm thickness SS and 40 mm in thickness methacrylate. The first one contains a 90 mm diameter view-port in its centre attached. To attach it, a washer shape supplementary structure has been added to the SS gate in order to gain thickness and strength in the assembly. The view-port is attached to the assembly by using claw clamps, which bolts go into tapped holes on the supplementary structure. Then a total of eight claw clamps are used to avoid leakages in the view-port. The use of less clamps results in a likelihood of leakage[24]. In the Figures 4.5 and 4.6 both configuration are shown.

The internal structure of the chamber consist of a shroud covered by MLI film, which purpose is to reduce the cooling time[23], as well as a piping system between them. The internal chamber can be observed in Figure 4.8. An squared structured AISI 304 SS 2 mm in thickness has been used for the shroud. This shroud, whose dimensions are 650x650x525 mm, also contains a rail guide for accessibility purposes[24]. Additionally, in order to increase the structural rigidity of the TVAC Chamber, a set of crossbars has been added to outer structure[17]. The footing structure acts as a support base for preventing vibrations originated in the vacuum and thermal systems[12].

For instrumentation purposes a set of feed-through port have been added. NW Wing Nut Clamps and O-ring sealed off those ports. A total of 17 feed-through ports such as three NW-25 8 pin Electrical Instrumentation feed-through ports, which are able to transmit signal voltages and currents up to 1 kV and 7 A, respectively, as well as three NW-16 Type-K Thermocouple have been added. In addition, two LN_2 KF-50 feed-through ports have been added to the chamber for the injection and extraction (as gas) of the dual liquid nitrogen. Moreover, several welding sockets of different sizes are added to meet future needs. Those ports which are not being used are ended with blank flanges.

To sum up, in the Table 4.1 a list of characteristic of the chamber is shown.

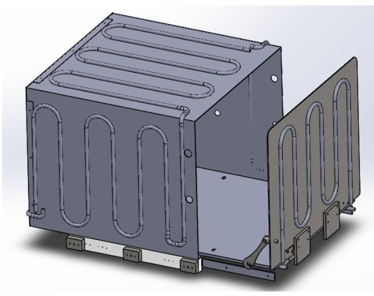


Figure 4.8 – *Internal Chamber. Shroud and LN_2 circuit.[24]*

Shape	Box
Inner Dimensions	700x745x700 mm
External Dimensions	1065x1024x857 mm
Internal Volume	372.75 l
Shroud Dimensions	650x650x525 mm
Structural Material	AISI 304 SS
Flanges Material	Stainless Steel
Flange Type	ISO KF-16/25/50/63
Gate Configuration	Methacrylate & AISI 304 SS
View-port Type	Kodial Type
View-port Diameter	90 mm
Weight	260.39 kg

Table 4.1 – *Features of the designed TVAC Chamber*

4.1.2 Vacuum System

The block diagram of the Vacuum System is shown in the Figure 4.9. Notice that the Turbo molecular pump is only needed when High Vacuum conditions, so it can be detached from the system for most tests. For that reason, the system does not currently have Turbo molecular pump.

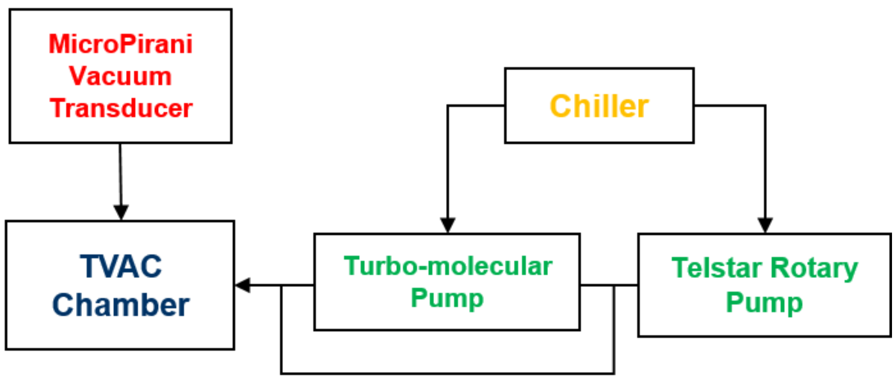


Figure 4.9 – *Vacuum System block diagram[24]*

The *925 MicroPirani Vacuum Transducer* will be controlled by using a RS-232 interface. The pumps have not include any programmable interface.

4.1.3 Thermal System

As stated in 3.2.2.3 the thermal system consist of two stages for cooling and heating the chamber. Also, as said in the aforementioned section, the internal temperatures of the chamber are collected using three NW-16 Type K thermocouples from which precious data shall be obtained in the future. In the cooling stage the injected LN_2 is the discharged through a Dual LN_2 feed-through located on the chamber's lateral. The Dual LN_2 feed-through ports used are built with dual coaxial tubes which neatly reduces condensation ensuring the seal.

The block diagram of the Thermal System is shown in the Figure 4.10.

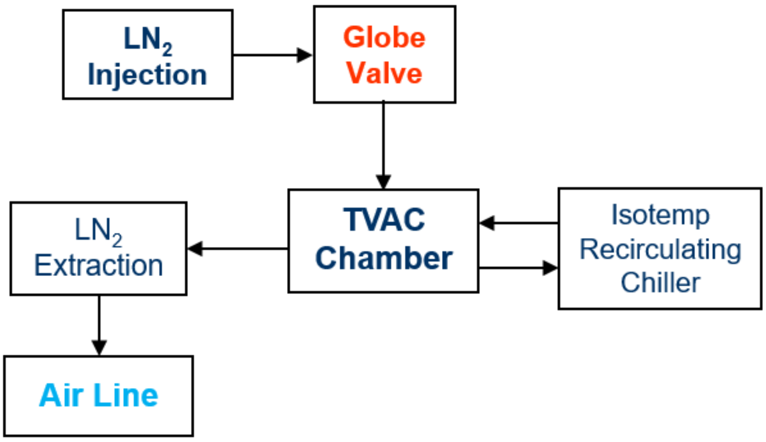


Figure 4.10 – *Thermal System block diagram[24]*

4

The *Isotemp 6200 R35 Recirculating Chiller* includes a RS-232 programmable interface. Its library, which has been implemented in MATLAB, will be shown in section 4.3.2.

4.1.4 Intrumentation Equipment

As stated in 4.1.1, the TVAC Chamber has been designed with has a total of seventeen feed-through ports for instrumentation purposes as well as two port for the LN_2 dealing. Not all the ports are currently used. Those unused port are able for expansion purposes.

In Table 4.2 a list of the functionalities of the feed-through ports can be found.

Finally, as stated in 3.2.2.4, *Siglent SDM3065X Digital Multimeter* is used through USB to measure temperatures, current and voltages selected by the engineer. In section 4.3.1.3, the implemented library for controlling it will be shown.

Feed-through port	Quantity	Functionality
KF-50 Dual LN_2	2	1x LN_2 Injection 1x LN_2 Extraction
KF-25 Half Nipple	5	1x MicroPirani Vacuum Transducer 3x Electrical Instrumentation 1x Future Needs
KF-25 Long Nipple	1	Recirculating Chiller
KF-16 Metric Long Nipple	5	3x Type K Thermocouples 1x Telstar Rotary pump 1x Recirculating Chiller
KF-25 Metric Long Nipple	5	Future Needs
ISO-63 Nipple	1	Turbo-molecular pump

Table 4.2 – *List of the instrumentation feed-through ports attached to the TVAC Chamber*

4.2 PV System Design

The Photovoltaic System consist of instruments such as the laboratory solar simulator, the *Apogee SP-110* pyranometer, *Keysight 6063B* DC programmable Electronic Load, the *HP-E3631A* Power Supply and the solar panel to be tested. The objective of the PV system is to characterize the solar panel by obtaining its I-V and P-V Characterization Curve.

In the Figure 4.11 a diagram of the system configuration is shown.

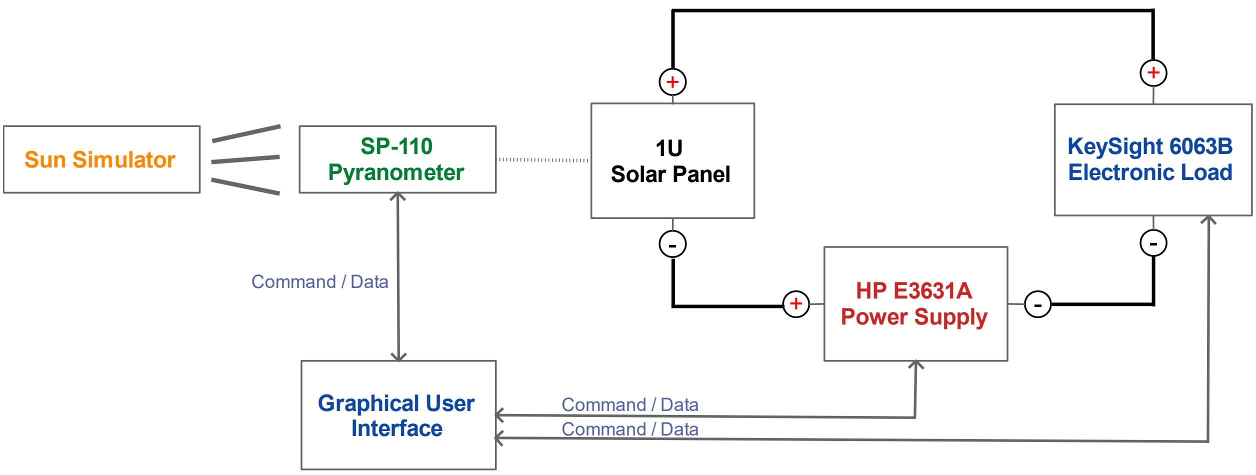


Figure 4.11 – *PV System Block Diagram.*

In the above figure the block diagram of the PV system can be observed. In this diagram a solar simulator irradiates a beam light over a solar panel. The solar panel is connected to an electronic load and a power supply in order to obtain its I-V and P-V curves. The *HP-E3631A Power Supply* is used to boost the cell potential above the eload's threshold as mentioned in section 3.2.1.4.1. The light emitted from the solar simulator is captured by using the pyranometer *Apogee SP-110*. This pyranometer must be located in a position in which no shadow on the solar panel is produced. It is also used for characterize the solar simulator spatial uniformity, which will be further explained in section 4.2.1.

The instrumentation equipment used in the aforementioned block diagram is controlled by using a personal computer through GPIB to USB interface and the *Keithley 3116 USB Data Acquisition*. Those instruments controlled through GPIB are the *HP-E3631A* Power Supply and the *Keysight 6063B* Electronic Load. As for the *Apogee SP-110* pyranometer, the datalogger *Keithley 3116* is used for data acquisition. The solar simulator has not includes a programmable interface.

4.2.1 Solar Simulator

In this section the spatial uniformity of the solar simulator beam over a gridded area, as shown in the Figure 4.12, is measured. The area is divided in 100 test point, however, just the 55 test positions which extends the size of one 1U solar panel are considered. The centre of each point is separated 2 cm from the next one. The measured values are obtained in mV range by using the pyranometer *Apogee SP-110*. The solar simulator is located at a distance of 1.70 m from the light receiving area which irradiates with a power of 2 kW. The beam light emitted by the solar simulator is 44 cm of diameter as shown in Figure 4.14.

In the Figure 4.13, the measured spatial uniformity is shown. In those figures the non uniformity of the distribution can be shown. It is observed that the the beam is slightly shifted to the left of the gridded area. This may be due to a non optimal configuration of the solar simulator lenses. The same issue is reported in [26].

4

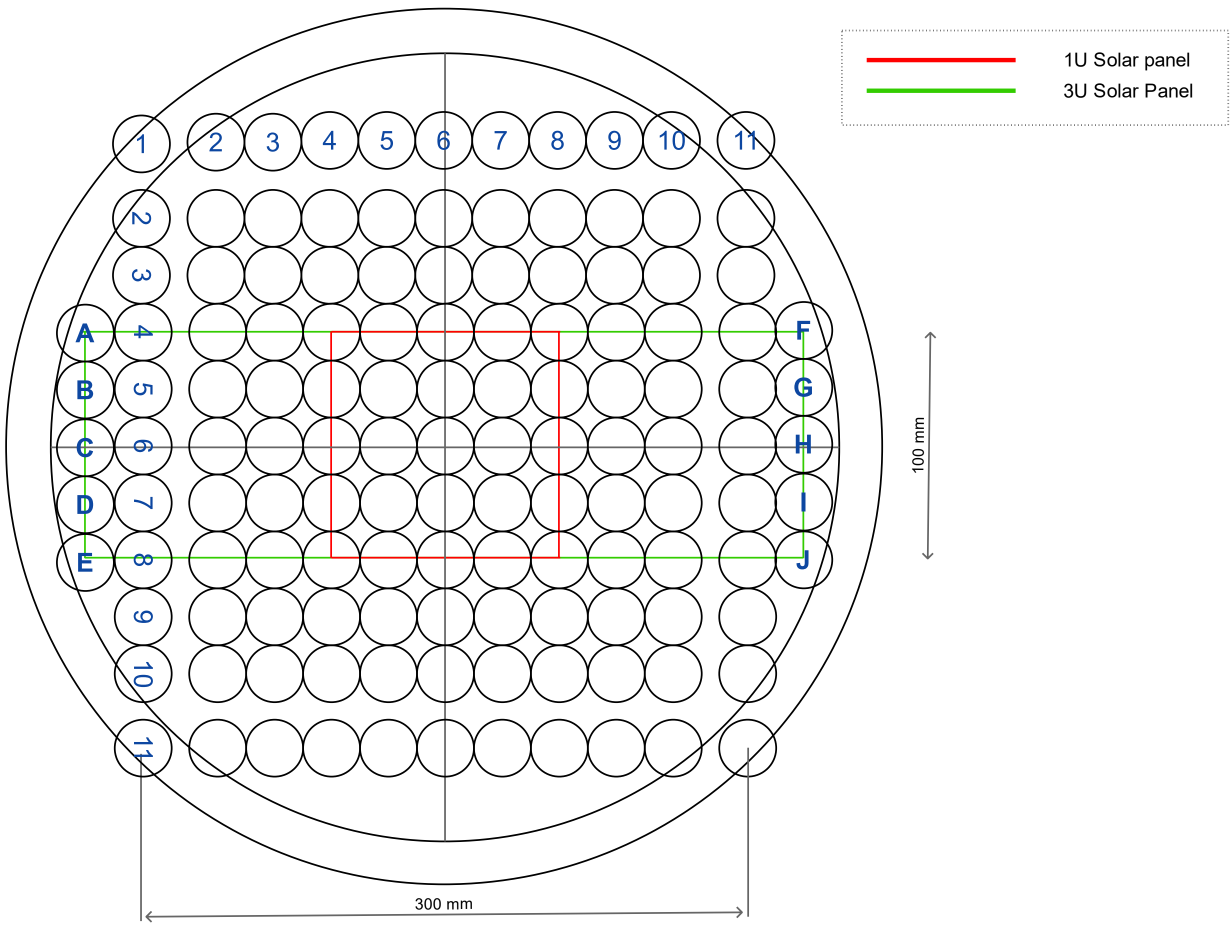


Figure 4.12 – *Template used for the spatial uniformity measurement.*

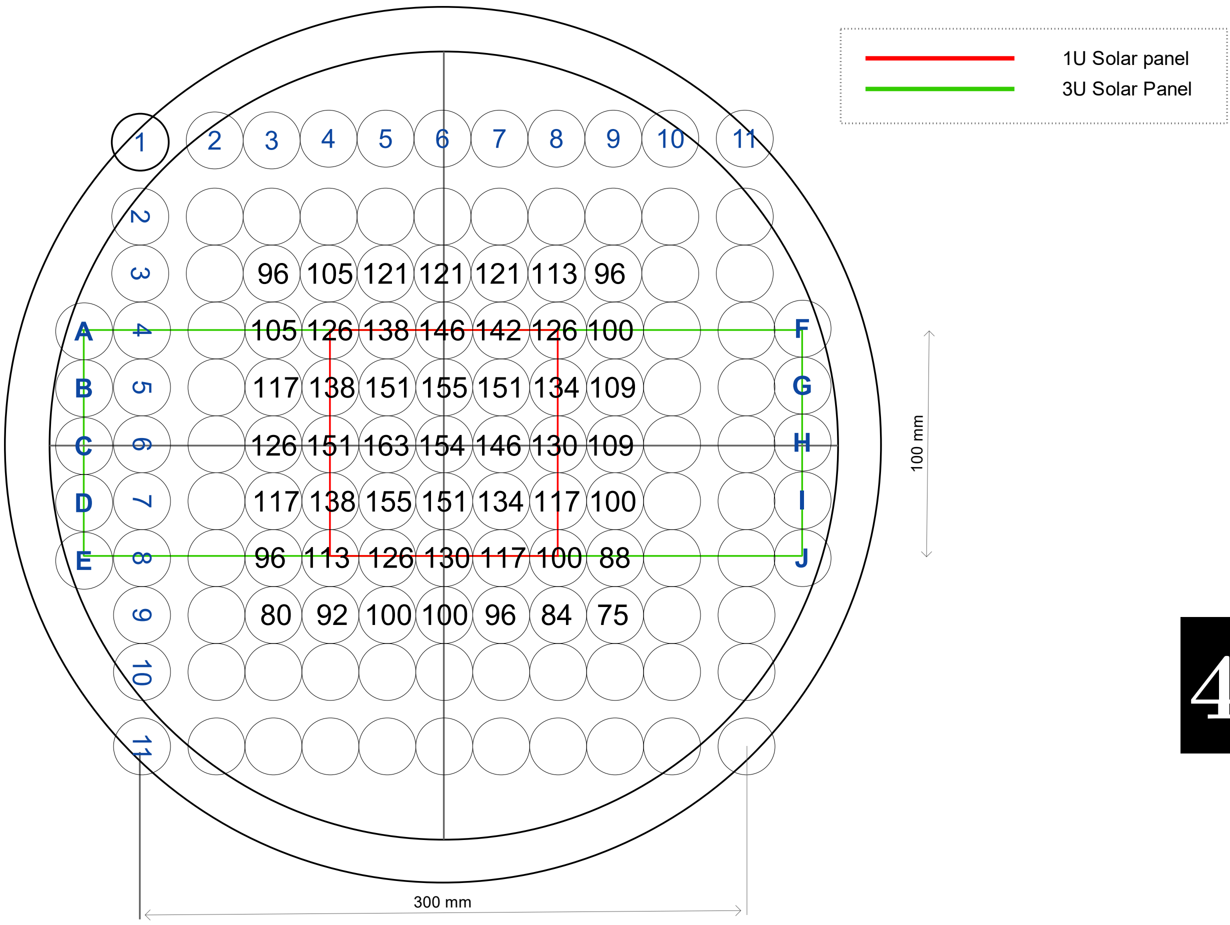


Figure 4.13 – *Spatial non uniformity measurement.*

4.3 Software Design

Once the whole hardware subsystem has been designed, the software subsystem must be implemented in order to control those subsystem, to perform the aerospace tests as well as to receive crucial data from them. In this section all the developed libraries and implemented codes will be shown and detailed. Finally, the final design of the GUI will be elucidated in section 4.3.4.

The MATLAB tool is the keynote of the software system due to the GUI is designed by using this tool. Notice that the GUI controls all the electronic equipment. Both GPIB and RS-232 communication interfaces are needed for controlling the several electronics equipments which are part of the hardware subsystem. The GPIB is used for controlling the *HP-E3631A* Power Supply and the *Keysight 6063B* DC programmable Electronic Load

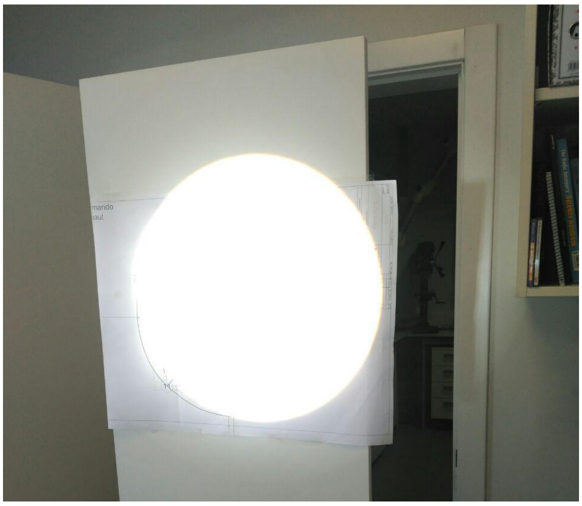


Figure 4.14 – *Solar Simulator emitted beam.*

which belong to the PV subsystem's equipment and the *Siglent SDM3065X Digital Multimeter*, which is part of the TVAC subsystem. As for the the RS-232 interface, it is used to control the *925 MicroPirani Vacuum Transducer* and the *Isotemp Recirculating Chiller*, both of them belonging to the TVAC subsystem's equipment.

4

A distinction must be made between the PV and TVAC subsystem related to the software design. The GUI has been designed integrating all the tools and libraries needed to control the PV subsystem and to characterize the DUT. The designed GUI does not yet control the electronic equipment that are part of the TVAC subsystem. This is because of the temporal planning of the project. However, the necessary libraries to control its equipment have been implemented with the aim that in the future the thermal and vacuum test related with the TVAC subsystem could be integrated within the GUI in order to improve the aerospace test and the solar panel characterization.

In the sections 4.3.1 and 4.3.2 the GPIB and RS-232 libraries are shown.

4.3.1 GPIB libraries

In this section, the GPIB libraries used for controlling the PV System will be further discussed and detailed. The communication between the electronic equipment and the computer is linked by using the MATLAB Instrument Control Toolbox and the Kenny Purpose Interface Bus operating framework. First of all, a serial communication between the electronic devices and MATLAB must be created. In this communication parameters such as the name of the equipment, the logical board index, the address of the device or the instruction commands are sent. As mentioned in section 3.3.3, due to the lack of GPIB interfaces in the laboratory computers, the *Agilent 82357B GPIB/USB* connector is used for the communication between the instrumentation equipment and the computer.

Once the communication between the computer and the instrumentation equipment is established, the several SCPI commands from each instruments will be used for controlling its functionalities. Each command sent to the instrument needs its object reference and the *fprintf()* function. According to the command used, a response must be received or not. The response messages remains in a buffer until they are read. The same structure used for sending commands is used for obtaining a response from the equipment but using the *fscanf()* function. Those commands are sent as ASCII codes. In the next code section, a command example is shown.

```
fprintf(io, 'INST:SEL P6V'); % where 'io' is the GPIB object
retval=fscanf(io,'%s'); % where 'retval' is the response message from the device
  and '%s' specifies a string response.
```

Code 4.1 – *SCPI command example in the Kenny Purpose Interface Bus framework*

Finally, the libraries for controlling the *HP-E3631A* Power Supply and the *Keysight 6063B* DC programmable Electronic Load will be implemented in order to abstracting the underlying implementation from the GUI. Those libraries are then integrated in the Kenny Purpose Interface Bus framework. Therefore, the function that the GUI will used for controlling the equipment is as shown in Code 4.2.

```
RETVAL = KPIB(INSTRUMENT, GPIB, COMMAND, VALUE, CHANNEL, AUX, VERBOSE)
```

Code 4.2 – *Kenny Purpose Interface Bus function*

4

4.3.1.1 *HP-E3631A* Power Supply library structure

The library developed for the power supply is shown in the Code 4.3. For calling this library several parameters are passed as arguments. Those parameters are the following:

- command: It uses the SCPI syntax.
- value: It is used for writing purposes. For example setting the voltage to a specified value.
- channel: It selects the channel of the electronic equipment.
- aux: auxiliary arguments.
- verbose: It refers to a mode that displays extended information.

This library first analyses the GPIB object status passed as an argument. Then the syntax is built by evaluating the command through the nested switch blocks. When the command match a block, the value could be use to define the final syntax. In this library, a set of valid commands are implemented:

- *init*: It is used to reset the instrument and clear registers
- *read*: Reads the output levels of the specified Output CHANNEL.

- *setV*: Sets the output voltage to VALUE in Volts.
- *setI*: Sets the output current to VALUE in Amps.
- *off*: Turns off both outputs.
- *on*: Turns on both outputs.


```
%% HP Power Supplies ('HP_POWER')
%
% HP_power is the list of HP Power supplies which have a common syntax
% use INSTRUMENT = 'HP_POWER' for generic code
HP_power={'HP_POWER','HP_POWERM','HP_6614C','HP_E3631A','HP_E3632A','HP_E3633A',
 'HP_E3634A','HP_E3641A','HP_E3647A'};
HP_power_M={'HP_POWERM','HP_E3631A','HP_E3647A'}; % These instruments have
 multiple output channels
%
%

%RETVAL = KPIB('INSTRUMENT', GPIB, 'COMMAND', VALUE, CHANNEL, AUX, VERBOSE)

if (any(strcmpi(instrument, HP_power)) || strcmpi(instrument, 'all'))
    io = port(GPIB, instrument, 0, verbose,gpib_interface_BOARDINDEX);

    if (io ~=0) && (strcmp(get(io,'Status'),'open') ~=0)

        if any(strcmpi(instrument, HP_power_M))

            if ~(any(channel == [1 2 3]))
                channel=1;
            end
            if strcmpi(instrument,'HP_E3631A')
                switch channel
                    case 1
                        fprintf(io, 'INST:SEL P6V');
                    case 2
                        fprintf(io, 'INST:SEL P25V');
                    case 3
                        fprintf(io, 'INST:SEL N25V');
                end
            else
                fprintf(io, 'INST:SEL OUT%d', channel); % Selects the output
            end
            if verbose >= 2, fprintf(1, 'kpib/%s(%d): Output %d ',instrument,
             GPIB,channel); end
        end

        switch command
            case 'init'
                fprintf(io,'*RST');
                if verbose >=2,
                 fprintf('kpib/HP_POWER: RESET');
                end
                retval=0;

            case 'instrument_id'
                fprintf(io,'*IDN?');
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib/HP_POWER:
                 Identification:',retval); end


            case 'read'
                switch value % return a single value or both V & I?
```

```
                    case {'volt','volts','V','v'}
                        % read the voltage
                        fprintf(io, 'MEAS:VOLT?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'reads %f Volts\n',retval); end
                    case {'curr','I','A','current'}
                        % read the current
                        fprintf(io, 'MEAS:CURR?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'reads %f Amps\n',retval); end
                    otherwise
                        % read the output
                        fprintf(io, 'MEAS:VOLT?');
                        retval.volt = fscanf(io,'%f');
                        fprintf(io, 'MEAS:CURR?');
                        retval.curr = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'reads %f Volts & %f Amps\n',retval.volt,retval.curr); end
                end

            case {'setV','volt','voltage','set'}
                % set the voltage
                fprintf(io, 'VOLT %f',value); % Sets voltage.
                if verbose >= 2, fprintf('Output Voltage set to %g Volts\n',value); end

            case {'setI','curr','current'}
                % set the current
                fprintf(io, 'CURR %f',value); % Sets current.
                if verbose >= 2, fprintf('Output Current set to %g Amps\n',value); end

            case 'off'
                fprintf(io, 'OUTP OFF'); % Disables all outputs.
                if verbose >= 2, fprintf(1, 'Outputs off.\n'); end
            case 'on'
                fprintf(io, 'OUTP ON'); % Enables all outputs.
                if verbose >= 2, fprintf(1, ' Outputs on.\n'); end

            otherwise
                if verbose >= 1, fprintf('Error, command not supported. ["%s"]\n',command); end
        end

    else % catch incorrect address errors
        if verbose >= 1, fprintf('kpib/%s: ERROR: No instrument at GPIB %d\n',instrument,GPIB); end
        retval=0;
    end

    validInst = 1;
end % end HP_POWER
```

Code 4.3 – *HP-E3631A Power Supply library*

In the Figure 4.15 a flow diagram of the current library is shown in order to get a better understanding.

Finally, some of the calls to this function that take place in the GUI are the followings:

```

kpib(HP_E3631A_PowerSupply, 'init',0,0,0,Verbose);
```

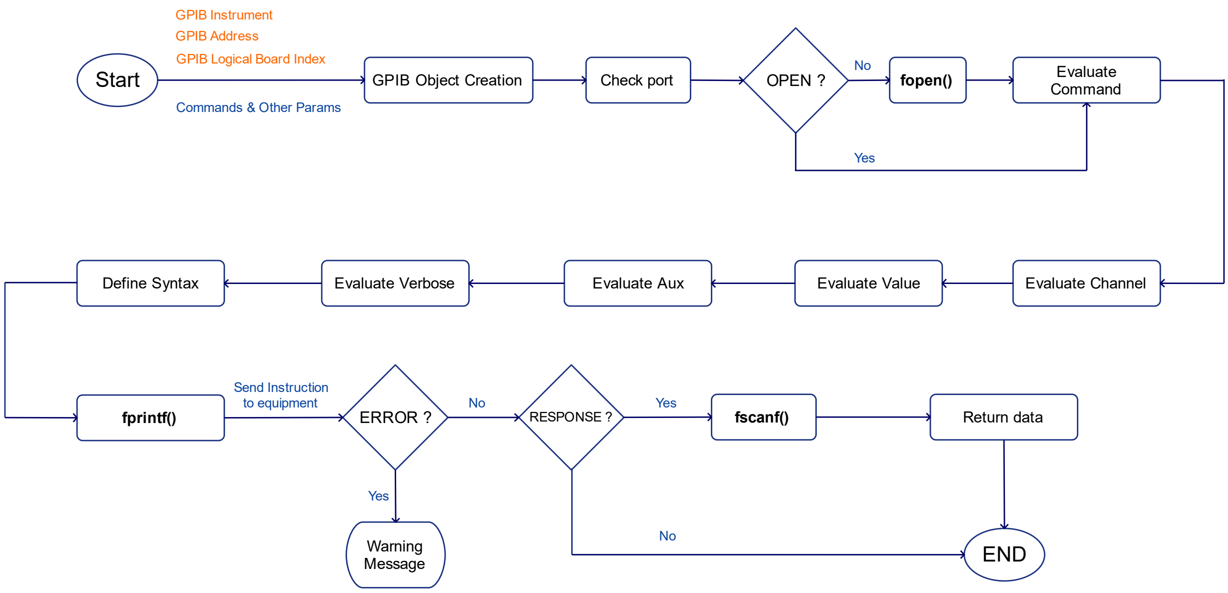


Figure 4.15 – *HP GPIB/KPIB Library Flow Diagram*


```
kpib(HP_E3631A_PowerSupply, 'instrument_id',0,0,0,Verbose);
kpib(HP_E3631A_PowerSupply, 'on',0,HP_E3631A_PowerSupply.Channel,0,Verbose);
kpib(HP_E3631A_PowerSupply, 'setV',HP_E3631A_PowerSupply.Boost_Voltage,
 HP_E3631A_PowerSupply.Channel,0,Verbose);      % set the voltage
```

Code 4.4 – *Kenny Purpose Interface Bus function*

4.3.1.2 *Keysight 6063B* DC programmable Electronic Load library structure

The library developed for the electronic load is shown in the Code 4.5.

```
%% KEYSIGHT 6063B ELECTRONIC LOAD
% Added by Juan Manuel Lopez Torralba

%RETVAL = KPIB('INSTRUMENT', GPIB, 'COMMAND', VALUE, CHANNEL, AUX, VERBOSE)

if (any(strcmpi(instrument, 'KEYSIGHT_6063B')) || strcmpi(instrument, 'all'))

    io = port(GPIB, instrument, 0, verbose,gpib_interface_BOARDINDEX);
    %io = port(GPIB, instrument, 0, verbose);
    if (io ~=0) && (strcmp(get(io,'Status'),'open') ~=0)



        switch command
            case 'init'
                fprintf(io,'*RST');
                retval=0;

            case 'error'
                fprintf(io,'SYST:ERR?');
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: Error:',retval); end
```

```

        case 'instrument_id'
            fprintf(io,'*IDN?'); % Typical Response:
            retval = fscanf(io,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','kpib: Identification:',
             retval); end


        case {'MODE','mode'}
            switch value % select between CV,CC,CR modes
                case {'CV','cv'}
                    % constant voltage
                    fprintf(io, 'MODE:VOLT:DC\n');
                    if verbose >= 2, fprintf('ELOAD mode sets to %g \n',
                     value); end
                    retval=0;

                case {'CR','cr'}
                    % constant resistance
                    fprintf(io, 'MODE:RES');
                    if verbose >= 2, fprintf('ELOAD mode sets to %g \n',
                     value); end
                    retval=0;

                case {'CC','cc'}
                    % constant current
                    fprintf(io, 'MODE:CURR:DC');
                    if verbose >= 2, fprintf('ELOAD mode sets to %g \n',
                     value); end
                    retval=0;

            end
        case 'read'
            switch value % return a single value or both V & I?
                case {'volt','volts','V','v'}
                    % read the voltage
                    fprintf(io, 'MEAS:VOLT?');
                    retval = fscanf(io,'%f');
                    if verbose >= 2, fprintf(1, 'Keysight 6063B output
                     voltage: %f Volts\n',retval); end
                case {'curr','I','A','current'}
                    % read the current
                    fprintf(io, 'MEAS:CURR?');
                    retval = fscanf(io,'%f');
                    if verbose >= 2, fprintf(1, 'Keysight 6063B output
                     current: %f Amps\n',retval); end
                otherwise
                    % read the output
                    fprintf(io, 'MEAS:VOLT?');
                    retval.volt = fscanf(io,'%f');
                    fprintf(io, 'MEAS:CURR?');
                    retval.curr = fscanf(io,'%f');
                    if verbose >= 2, fprintf(1, 'reads %f Volts & %f Amps\n'
                     ,retval.volt,retval.curr); end
            end


        case {'setV','volt','voltage','set'}
            % set the voltage
            fprintf(io, 'VOLT:LEV:IMM %f',value); % Sets voltage.
            if verbose >= 2, fprintf('Output Voltage set to %g Volts\n',
             value); end
            retval=0;

        case {'setI','curr','current'}
            set the current
```

```
                fprintf(io, 'CURR %f',value); % Sets current.
                if verbose >= 2, fprintf('Output Current set to %g Amps\n',value
                 ); end

            case 'off'
                fprintf(io, 'OUTP OFF'); % Disables all outputs.
                if verbose >= 2, fprintf(1, 'Outputs off.\n'); end
                retval=0;

            case 'on'
                fprintf(io, 'OUTP ON'); % Enables all outputs.
                if verbose >= 2, fprintf(1, ' Outputs on.\n'); end
                retval=0;

            case 'input_off'
                fprintf(io, 'INP:STAT OFF'); % Disables all outputs.
                if verbose >= 2, fprintf(1, 'Outputs off.\n'); end
                retval=0;

            case 'input_on'
                fprintf(io, 'INP:STAT ON'); % Enables all outputs.
                if verbose >= 2, fprintf(1, ' Outputs on.\n'); end
                retval=0;

            otherwise
                if verbose >= 1, fprintf('Error, command not supported. ["%s"]\n
                 ',command); end
                retval=0;
        end


    else % catch incorrect address errors
        if verbose >= 1, fprintf('kpib/%s: ERROR: No instrument at GPIB %d\n',
         instrument,GPIB); end
        retval=0;
    end

    validInst = 1;
end
```

Code 4.5 – *HP-E3631A Power Supply library*

Finally, some of the calls to this function that take place in the GUI are the followings:

```
kpib(KEYSIGHT_6063B_ElecLoad 'init',0,0,0,Verbose);
kpib(KEYSIGHT_6063B_ElecLoad, 'instrument_id',0,0,0,Verbose);
kpib(KEYSIGHT_6063B_ElecLoad, 'on',0,0,0,Verbose);
kpib(KEYSIGHT_6063B_ElecLoad, 'mode','cv',0,0,Verbose);
```

Code 4.6 – *Kenny Purpose Interface Bus function*

In the Figure 4.16 the electronic load's tree diagram is shown. Notice that this tree diagram has been taken as a reference for the implementation of this library. The flow diagram of this library is similar to the one shown in Figure 4.15

4.3.1.3 *Siglent SDM3065X* library

The SDM3065X is a digital multimeter used through USB to measure the different temperatures, voltages and current from TVAC Chamber by using the aforementioned

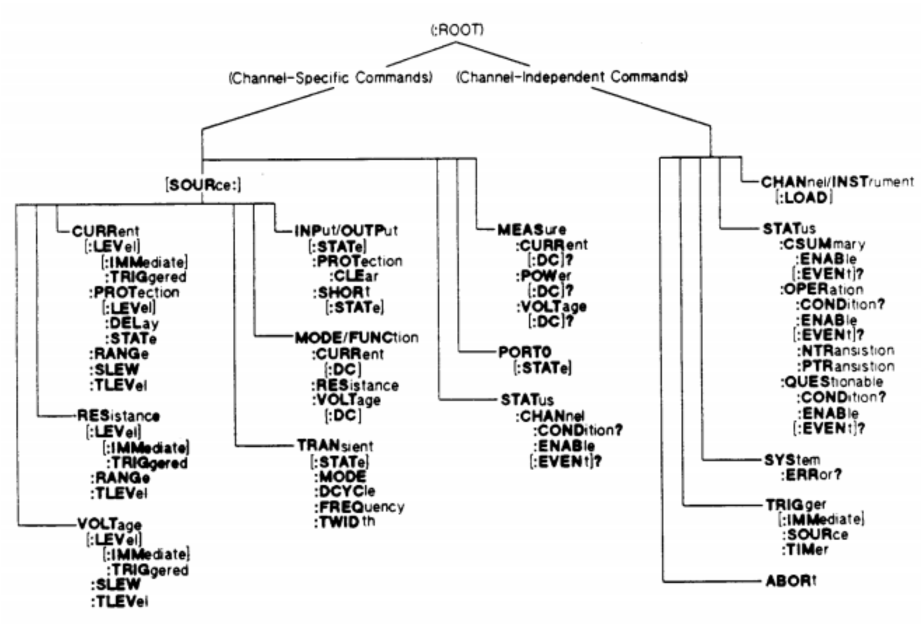


Figure 4.16 – *Electronic Loads Tree Diagram. [Credits: Keysight]*

feed-through ports. In this section the GPIB library which has been developed to control this equipment will be shown. It will be also integrated in the Kenny Purpose Interface Bus framework as the other libraries previously mentioned. In the Code 4.7, the library is shown. The manual of this instrument [30] has been used as reference for the implementation of the library.

```
%% SDM3065X Digital Multimeter
% Added by Juan Manuel Lopez Torralba

%RETVAL = KPIB('INSTRUMENT', GPIB, 'COMMAND', VALUE, CHANNEL, AUX, VERBOSE)

if (any(strcmpi(instrument, 'SDM3065X')) || strcmpi(instrument, 'all'))

    io = port(GPIB, instrument, 0, verbose,gpib_interface_BOARDINDEX);
    %io = port(GPIB, instrument, 0, verbose);
    if (io ~=0) && (strcmp(get(io,'Status'),'open') ~=0)

        switch command

            case 'abort'
                fprintf(io,'ABOR');
                retval=0;

            case 'fetch'
                fprintf(io,'FETCh?');
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X FETCH:',
                 retval); end

            case 'init'
                fprintf(io,'INIT:IMM');
                retval=0;

            case 'output_triger_slope'
                % Selects the slope of the voltmeter complete output
```

```
                % signal on the rear-panel VM Comp BNC connector.
                fprintf(io,'OUTPut:TRIGger:SLOPe?');
                retval=0;

            case 'erase_read'
                fprintf(io,'R? %f', value);
                retval=0;

            case 'error'
                fprintf(io,'SYST:ERR?');
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X Error:',
                 retval); end

            case 'instrument_id'
                fprintf(io,'*IDN?'); % Typical Response:
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X
                 Identification:',retval); end

          case 'set_sample_count'
                fprintf(io,'SAMPle:COUNt? %f', value);
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X number of
                 measurements:',retval); end

            case 'get_sample_count'
                fprintf(io,'SAMPle:COUNt?');
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X number of
                 measurements:',retval); end

            case 'get_unit_temperature'
                fprintf(io,'UNIT:TEMPerature?');
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X unit of
                 temperature:',retval); end


            case 'set_unit_temperature'
                fprintf(io,'UNIT:TEMPerature? %c'+ value);
                retval = fscanf(io,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X unit of
                 temperature:',retval); end

            case 'calculate_clear'
                % Clears all limits, histogram data, statistics and measurements
                fprintf(io,'CALC:CLE:IMM');
                retval=0;

            case 'calculate_limit'
                % This subsystem specifies measurements and indicates when a
                 limit has been exceeded
                switch value
                    case 'clear'
                        fprintf(io,'CALC:LIM:CLE');
                        retval=0;

                    case 'get_lower' % sets a lower limit
                        fprintf(io,'CALC:LIM:LOW?');
                      retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X
                         lower limit: ',retval); end

                    case 'get_upper' % sets an upper limit
                        fprintf(io,'CALC:LIM:UPP?');
```

```
                  retval = fscanf(io,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X upper limit: ',retval); end

                case 'set_lower' % sets a lower limit
                    fprintf(io,'CALC:LIM:LOW %f', value);
                    retval=0;

                case 'set_upper' % sets an upper limit
                    fprintf(io,'CALC:LIM:UPP %f', value);
                    retval=0;

                case 'stat_on'
                    fprintf(io,'CALC:LIM:STAT ON');
                    retval=0;

                case 'stat_off'
                    fprintf(io,'CALC:LIM:STAT OFF');
                    retval = fscanf(io,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','kpib SDM3065X :', retval); end
            end

        case 'CALCulate_TRANsform_HISTogram_Subsystem'
            disp('not implemented yet')
            retval=0;

        case 'CALCulate_SCALe_Subsystem'
            disp('not implemented yet')
            retval=0;

        case 'calculate_average'
            switch value
                case 'stat_on'
                    fprintf(io,'CALCulate:AVERage:STATe ON');
                    retval = fscanf(io,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X statistic computation enabled: ',retval); end

                case 'stat_off'
                    fprintf(io,'CALCulate:AVERage:STATe OFF');
                    retval = fscanf(io,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X statistic computation disabled: ',retval); end

                case 'clear'
                    fprintf(io,'CALC:AVER:CLE:IMM');
                    retval=0;

                case 'average_all' % sets an upper limit
                    fprintf(io,'CALCulate:AVERage:ALL?');
                    retval = fscanf(io,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X average all: ',retval); end

                case 'average'
                    fprintf(io,'CALCulate:AVERage:AVERage?');
                  retval = fscanf(io,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X mean : ',retval); end

                case 'count'
                    fprintf(io,'CALCulate:AVERage:COUNt?');
                    retval = fscanf(io,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X count: ',retval); end
```

```
                    case 'max'
                        fprintf(io,'CALCulate:AVERage:MAXimum?');
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X maximum: ',retval); end

                    case 'min'
                        fprintf(io,'CALCulate:AVERage:MINimum?');
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X minimum: ',retval); end

                    case {'peak','ptpeak'}
                        fprintf(io,'CALCulate:AVERage:PTPeak?');
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X peaks: ',retval); end

                    case {'sdeviation','deviation','sdev'}
                        fprintf(io,'CALCulate:AVERage:SDEViation?');
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X standard deviation: ',retval); end
                end




            case 'Configure'
                switch value
                    case 'configure'
                        fprintf(io,'CONF?');
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X present function, range, and resolution: ',retval); end

                    case 'continuity'
                        % Configure the instrument for continuity measurements
                        fprintf(io,'CONFigure:CONTinuity');
                        retval=0;

                    case 'current_dc'
                        fprintf(io,'CONFigure:CURRent:DC');
                        retval=0;

                    case 'current_ac'
                        fprintf(io,'CONFigure:CURRent:AC');
                        retval=0;

                    case 'diode'
                        fprintf(io,'CONFigure:DIODe');
                        retval=0;

                    case 'freq'
                        fprintf(io,'CONF:FREQ}');
                        retval=0;

                    case 'period'
                        fprintf(io,'CONF:PER}');
                        retval=0;

                    case 'resistance'
                        fprintf(io,'CONF:RES');
                        retval=0;

```

```
                    case 'fresistance'
                        fprintf(io,'CONF:FRES');
                        retval=0;

                    case 'temperature'
                        fprintf(io,'CONFigure:TEMPerature');
                        retval=0;

                    case 'dc_volt'
                        fprintf(io,'CONFigure:VOLT:DC');
                        retval=0;

                    case {'ac_volt'}
                        fprintf(io,'CONF:VOLT:AC');
                        retval=0;

                    case {'capacitance'}
                        fprintf(io,'CONF:CAP');
                        retval=0;
                end
            case 'data'
                switch value
                    case 'last'
                        fprintf(io,'DATA:LAST?');
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X last
                          measurement taken: ',retval); end
                    case 'points'
                        %Returns the total number of measurements currently in
                         reading memory.
                        fprintf(io,'DATA: POINts?');
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X
                         number of measurements in reading memory: ',retval);
                         end
                    case 'remove'
                        fprintf(io,'DATA:REMove? %f', value2);
                        retval = fscanf(io,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','kpib: SDM3065X Read
                          and erase the oldest readings from reading memory: ',
                         retval); end
                end
            case 'read'
                switch value
                    case 'read'
                        fprintf(io, 'READ?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X measurement: %f \n
                         ',retval); end

                    case {'Cont','cont','Continuity','continuity'}
                        % Sets all measurement parameters and trigger parameters
                          to their default values for
                        % continuity test and immediately triggers a measurement
                         . The results are sent directly to
                        % the instrument's output buffer.
                        fprintf(io, 'MEAS:CONT?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X measured
                         resistance: %f \n',retval); end

                    case {'volt','volts','V','v'}
                        % read the DC voltage
                        fprintf(io, 'MEAS:VOLT?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X output DC voltage:
```

```
%f Volts\n',retval); end

        case {'VAC','vac'}
            % read the AC voltage
            fprintf(io, 'MEAS:VOLT:AC?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X output AC voltage: %f Volts\n',retval); end

        case {'curr','I','A','current','Curr'}
            % Sets all measurement parameters and trigger parameters to their default values for AC or
            % DC current measurements and immediately triggers a measurement. Also specifies the
            % stalls through the incoming parameters
            fprintf(io, 'MEAS:CURR?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X output DC current: %f A\n',retval); end

        case {'DIODE','diode','D','d','diod'}
            % Sets all measurement parameters and trigger parameters to their default values for diode
            % test measurements and immediately triggers a measurement. The results are sent directly
            % to the instrument's output buffer.
            fprintf(io, 'MEAS:DIOD?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X output DC voltage: %f V\n',retval); end

        case {'Frequency','frequency','Freq','freq'}
            % Sets all measurement parameters and trigger parameters to their default values for
            % frequency or period measurements and immediately triggers a measurement. The results
            % are sent directly to the instrument's output buffer
            fprintf(io, 'MEAS:FREQ?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X default frequency measurements: %f Hz\n',retval); end


        case {'Resistance','resistance','Res','res'}
            % Sets all measurement and trigger parameters to their 2-wire (RESistance) measurements,
            % and immediately triggers a measurement. The results are sent directly to the instrument's
            % output buffer. Also specifies the stalls through the incoming parameters.
            fprintf(io, 'MEAS:RES?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X default frequency measurements: %f ohm\n',retval); end

        case {'FResistance','fresistance','fres'}
            % Sets all measurement and trigger parameters to their 4-wire (RESistance) measurements,
            % and immediately triggers a measurement. The results are sent directly to the instrument's
            % output buffer. Also specifies the stalls through the incoming parameters.
            fprintf(io, 'MEAS:FRES?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X default frequency measurements: %f ohm\n',retval); end
```

```
                    case {'Temp','temp','temperature','t'}
                        % read the temperature
                        fprintf(io, 'MEAS:TEMP?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X output temperature : %f \n',retval); end

                    case {'Capacitance','capacitance','Cap','cap'}
                        % read capacitance
                        fprintf(io, 'MEAS:CAP?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X output capacitance : %f F \n',retval); end

                end

            case 'sense'
                switch value
                    case 'function_cont'
                        fprintf(io, 'FUNC "CONT"');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                  case 'function_curr'
                        fprintf(io, 'FUNC "CURR"');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


                    case 'function_curr_ac'
                        fprintf(io, 'FUNC "CURR:AC"');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


                    case 'function_diod'
                        fprintf(io, 'FUNC "DIOD"');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


                    case 'function_freq'
                        fprintf(io, 'FUNC "FREQ"');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


                    case 'function_fres'
                        fprintf(io, 'FUNC "FRES"');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


                    case 'function_period'
                        fprintf(io, 'FUNC "PER"');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

```

```
            case 'function_res'
                fprintf(io, 'FUNC "RES"');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


            case 'function_temp'
                fprintf(io, 'FUNC "TEMP"');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

           case 'function_volt'
                fprintf(io, 'FUNC "VOLT"');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


            case 'function_volt_ac'
                fprintf(io, 'FUNC "VOLT:AC"');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


            case 'function_cap'
                fprintf(io, 'FUNC "CAP"');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


            case 'function_default'
                fprintf(io, 'FUNC');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

            case 'set_current_ac_null'
                fprintf(io, 'CURRent:AC:NULL:STAT %s', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


            case 'set_current_dc_null'
                fprintf(io, 'CURRent:DC:NULL:STAT %s', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

            case 'get_current_ac_null'
                fprintf(io, 'CURRent:AC:NULL:STAT?');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

            case 'get_current_dc_null'
                fprintf(io, 'CURRent:DC:NULL:STAT?');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

```

```
                    case 'set_current_ac_null_val'
                        fprintf(io, 'CURRent:AC:NULL:VAL %s', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'set_current_dc_null_val'
                        fprintf(io, 'CURRent:DC:NULL:VAL %s', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'get_current_ac_null_val'
                        fprintf(io, 'CURRent:AC:NULL:VAL?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'get_current_dc_null_val'
                        fprintf(io, 'CURRent:DC:NULL:VAL?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'get_current_ac_null_val_auto'
                        fprintf(io, 'CURRent:AC:NULL:VALue:AUTO?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'get_current_dc_null_val_auto'
                        fprintf(io, 'CURRent:DC:NULL:VALue:AUTO?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'set_current_ac_null_val_auto'
                        fprintf(io, 'CURRent:AC:NULL:VALue:AUTO %s', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'set_current_dc_null_val_auto'
                        fprintf(io, 'CURRent:DC:NULL:VALue:AUTO %s', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'current_ac_range'
                        fprintf(io, 'CURR:AC:RANGe');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                  case 'current_dc_range'
                        fprintf(io, 'CURR:DC:RANGe');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                  case 'get_current_ac_range_auto'
                        fprintf(io, 'CURR:AC:RANGe:AUTO?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

```

4

```
                    case 'get_current_dc_range_auto'
                        fprintf(io, 'CURR:DC:RANGe:AUTO?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                    case 'set_current_ac_range_auto'
                        fprintf(io, 'CURR:AC:RANGe:AUTO %s', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                    case 'set_current_dc_range_auto'
                        fprintf(io, 'CURR:DC:RANGe:AUTO %s', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                     case 'get_current_nlpc'
                        fprintf(io, 'CURR:NLPC?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                     case 'set_current_nlpc'
                        fprintf(io, 'CURR:NLPC %f', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                     case 'get_current_bandw'
                        fprintf(io, 'CURR:BAND?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                     case 'set_current_bandw'
                        fprintf(io, 'CURR:BAND %f', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                     case 'get_current_az'
                        fprintf(io, 'CURR:AZ?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                    case 'set_current_az'
                        fprintf(io, 'CURR:AZ %s', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                     case 'temp_null'
                        fprintf(io, 'SENSe:TEMPerature:NULL:STATe');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

                    case 'get_temp_null_val'
                        fprintf(io, 'TEMPerature:NULL:VALue?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
                        retval); end

```

```
            case 'set_temp_null_val'
                fprintf(io, 'TEMPerature:NULL:VALue %f', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

            case 'temp_null_auto'
                fprintf(io, 'TEMPerature:NULL:VALue:AUTO');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

            case 'temp_trans'
                fprintf(io, 'TEMPerature:TRANsducer?');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

              case 'current_trans_ude_list'
                fprintf(io, 'TEMPerature:UDEF:THER:TRANsducer:LIST?');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

              case 'current_trans_udef_ther'
                fprintf(io, 'TEMPerature:UDEF:THER:TRAN %s', value2);
                retval=0;

              case 'current_trans_mdef_ther'
                fprintf(io, 'TEMPerature:MDEF:THER:TRAN %s', value2);
                retval=0;

              case 'current_trans_udef_rtd'
                fprintf(io, 'TEMPerature:UDEF:RTD:TRAN %s', value2);
                retval=0;

              case 'current_trans_mdef_rtd'
                fprintf(io, 'TEMPerature:MDEF:RTD:TRAN %s', value2);
                retval=0;

              case 'current_trans_udef_ther_point'
                fprintf(io, 'TEMPerature:UDEF:THER:TRAN:POINt %s', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

              case 'current_trans_mdef_ther_point'
                fprintf(io, 'TEMPerature:MDEF:THER:TRAN:POINt %s', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

              case 'current_trans_udef_rtd_point'
                fprintf(io, 'TEMPerature:UDEF:RTD:TRAN:POINt %s', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

              case 'current_trans_mdef_rtd_point'
                fprintf(io, 'TEMPerature:MDEF:RTD:TRAN:POINt %s', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n',
```

```
                     retval); end

                  case 'volt_ac_null'
                        fprintf(io, 'VOLT:AC:NULL:STAT');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_dc_null'
                        fprintf(io, 'VOLT:DC:NULL:STAT');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_ac_null_val'
                        fprintf(io, 'VOLT:AC:NULL:VAL');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'VOLT_dc_null_val'
                        fprintf(io, 'VOLT:DC:NULL:VAL');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_ac_null_val_auto'
                        fprintf(io, 'VOLT:AC:NULL:VALue:AUTO');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end


                    case 'volt_dc_null_val_auto'
                        fprintf(io, 'VOLT:DC:NULL:VALue:AUTO');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_ac_range'
                        fprintf(io, 'VOLT:AC:RANGe');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                  case 'volt_dc_range'
                        fprintf(io, 'VOLT:DC:RANGe');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                  case 'volt_ac_range_auto'
                        fprintf(io, 'VOLT:AC:RANGe:AUTO');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                  case 'volt_dc_range_auto'
                        fprintf(io, 'VOLT:DC:RANGe:AUTO');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_nlpc'
                        fprintf(io, 'VOLT:NLPC');
                        retval = fscanf(io,'%f');
```

```
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_impedance'
                        fprintf(io, 'VOLT:DC:IMP');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_bandw'
                        fprintf(io, 'VOLT:BAND %f', value2);
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_az_on'
                        fprintf(io, 'VOLT:AZ ON');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'volt_az_off'
                        fprintf(io, 'VOLT:AZ OFF');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'cap_null'
                        fprintf(io, 'CAPacitance:NULL:STAT');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                    case 'cap_null_value'
                        fprintf(io, 'CAP:NULL:VAL');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                   case 'cap_null_val_auto'
                        fprintf(io, 'CAPacitance:NULL:VALue:AUTO');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                   case 'cap_range'
                        fprintf(io, 'CAPacitance:RANGe');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                   case 'cap_range_auto'
                        fprintf(io, 'CAP:RANGe:AUTO');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X function: %f \n', retval); end

                end

            case 'route'
                switch value
                    case 'stat'
                        fprintf(io, 'ROUTe:STATe?');
                        retval = fscanf(io,'%f');
                        if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end
```

```

        case 'get_scan'
            fprintf(io, 'ROUTe:SCAN?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

        case 'scan_on'
            fprintf(io, 'ROUTe:SCAN ON');
            retval=0;

        case 'scan_off'
            fprintf(io, 'ROUTe:SCAN OFF');
            retval=0;

         case 'get_start'
            fprintf(io, 'ROUTe: STARt?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

         case 'start_on'
            fprintf(io, 'ROUTe: STARt ON');
            retval=0;

         case 'start_off'
            fprintf(io, 'ROUTe: STARt OFF');
            retval=0;

         case 'function'
           fprintf(io, 'ROUTe: FUNCtion?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

        case 'get_delay'
            fprintf(io, 'ROUTe: DELay?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

        case 'set_delay'
            fprintf(io, 'ROUTe: DELay %f', value2);
            retval=0;

        case 'get_count_auto'
            fprintf(io, 'ROUTe:COUNt:AUTO?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

        case 'set_count_auto'
            fprintf(io, 'ROUTe:COUNt:AUTO %s', value2);
            retval=0;

         case 'get_count'
            fprintf(io, 'ROUTe:COUNt?');
            retval = fscanf(io,'%f');
            if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

         case 'set_count'
            fprintf(io, 'ROUTe:COUNt %f', value2);
            retval=0;

         case 'get_limit_high'
```

```
                fprintf(io, 'ROUTe: LIMIt:HIGH?');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

            case 'get_limit_low'
                fprintf(io, 'ROUTe: LIMIt:LOW?');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

            case 'set_limit_high'
                fprintf(io, 'ROUTe: LIMIt:HIGH %f', value2);
                retval=0;

            case 'set_limit_low'
                fprintf(io, 'ROUTe: LIMIt:LOW %f', value2);
                retval=0;

          case 'data'
                fprintf(io, 'ROUTe:DATA? %f', value2);
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

          case 'channel'
                fprintf(io, 'ROUTe: CHANnel?');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

            case 'relative'
                fprintf(io, 'ROUTe: RELAtive %s', value2);
                retval=0;

            case 'impedance_10m'
                fprintf(io, 'ROUTe: IMPedance 10M');
                retval=0;

             case 'impedance_10g'
                fprintf(io, 'ROUTe: IMPedance 10G');
                retval=0;

          case 'temperature_rtd'
                fprintf(io, 'ROUTe: TEMPerature:RTD PT100');
                retval=0;

          case 'temperature_ther'
                fprintf(io, 'ROUTe:TEMPerature:THER %s', value2);
                retval=0;

            case 'set_temperature_unit'
                fprintf(io, ' ROUTe:TEMPerature:UNIT %c', value2);
                retval=0;

            case 'threshold_cont'
                fprintf(io, 'ROUTe:CONTinuity:THReshold:VALue');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end

          case 'threshold_diod'
                fprintf(io, 'ROUTe:DIODe:THReshold:VALue');
                retval = fscanf(io,'%f');
                if verbose >= 2, fprintf(1, 'SDM3065X route: %f \n', retval); end
```

```
                    case 'frequency'
                        fprintf(io, 'ROUTe:FREQ');
                        retval=0;

                  case 'period'
                        fprintf(io, 'ROUTe:PER');
                        retval=0;

                end

            otherwise
                if verbose >= 1, fprintf('Error, command not supported. ["%s"]\n',command); end
                retval=0;
        end


    else % catch incorrect address errors
        if verbose >= 1, fprintf('kpib/%s: ERROR: No instrument at GPIB %d\n', instrument,GPIB); end
        retval=0;
    end

    validInst = 1;
end %
```

Code 4.7 – *SDM 3065X Digital Multimeter library*

4.3.2 RS-232 Libraries

The RS-232 communication interface is used through an USB converter in order to control the recirculating chiller and the vacuum transducer, both belonging to the TVAC subsystem. The first step to remotely control the equipment using MATLAB and the RS-232 interface is to establish a serial communication between them. In this communication parameters such as the baud rate , the parity, the stop bit or the time-out must be defined. In the Code 4.8, the fragment of code responsible for setting up this communication is shown.

```
serialObject = serial(serialPort, 'baud', 19200,...
    'StopBits',1  ,...
    'DataBits', 8,...
    'Parity', 'none',...
    'Timeout', Tiempo,...
    'Terminator','CR',...
    'FlowControl', 'none');
%                 'ReadAsyncMode', 'Continuous',...
%                 'ReadAsyncMode', 'manual',...
%                 'Timeout', 1,...
%
fopen(serialObject);
```

Code 4.8 – *RS-232 serial communication establishment*

In Sections 4.3.2.1 the library for the recirculating chiller is shown.

4.3.2.1 Isotemp 6200 Recirculating Chiller Library

The Manual reference used for the implementation of the following library can be found on its manual [6].


```
function retval=CommPort_IsoTemp(Instrument,serialPort,Tiempo,verbose,RW_command,actionCommand,value)

% Versión 1: (1/1/2018 Juan Manuel López Torralba
%            AC Serial Communications Protocol IsoTemp 6200 R35
%            Driver


%Once the serial port reference is created, it will be accessible the next time the file is used.
global serialObject;

%%

if (ischar(Instrument))
    instrument=Instrument;
else
    instrument=Instrument.Name;
    serialPort=Instrument.serialPort;
end
%% Serial Object Definition

serialObject = serial(serialPort, 'baud', 19200,...
    'StopBits',1  ,...
    'DataBits', 8 ,...
    'Parity', 'none' ,...
    'Timeout', Tiempo,...
    'Terminator','CR' ,...
    'FlowControl', 'none');
%                'ReadAsyncMode', 'Continuous',...
%                'ReadAsyncMode', 'manual',...
%                'Timeout', 1,...
%
fopen(serialObject);
%A=fread(serialObject);

%% BEGIN CODE

if (strcmpi(instrument, 'isotemp') || strcmpi(instrument, 'all'))

    switch RW_command
        case 'read'
            switch actionCommand

                case 'temperature'                       % Read: Internal Temperature in  C /F/K (it depends on the Instrument internal configuration)
                    fprintf(serialObject,'RT');
                    retval = fscanf(serialObject,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Internal Temperature [ C /F/K]:',retval); end

                case 'temperature_2'                     % Read: External Temperature in  C /F/K (it depends on the Instrument internal configuration)
                    fprintf(serialObject,'RT2');
                    retval = fscanf(serialObject,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: External Temperature [ C /F/K]:',retval); end
```

```
                    case 'displayed_setpoint'            % Read: Displayed Setpoint Temperature in  C /F/K (it depends on the Instrument internal configuration)
                        fprintf(serialObject,'RS');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Displayed Setpoint [ C /F/K]:',retval); end

                    case 'internal_RTA1'                 % Read: Real (internal) Temperature Adjustments (RTA 1) in  C /F/K. The RTA can be set  10C  ( 18F  ).
                        fprintf(serialObject,'RIRTA1');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Internal RTA 1 [ C /F/K]:',retval); end

                    case 'internal_RTA2'                 % Read: Real (internal) Temperature Adjustments (RTA 2) in  C /F/K. The RTA can be set  10C  ( 18F  ).
                        fprintf(serialObject,'RIRTA2');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Internal RTA 2 [ C /F/K]:',retval); end

                    case 'internal_RTA3'                 % Read: Real (internal) Temperature Adjustments (RTA 3) in  C /F/K. The RTA can be set  10C  ( 18F  ).
                        fprintf(serialObject,'RIRTA3');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Internal RTA 3 [ C /F/K]:',retval); end

                    case 'internal_RTA4'                 % Read: Real (internal) Temperature Adjustments (RTA 4) in  C /F/K. The RTA can be set  10C  ( 18F  ).
                        fprintf(serialObject,'RIRTA4');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Internal RTA 4 [ C /F/K]:',retval); end

                    case 'internal_RTA5'                 % Read: Real (internal) Temperature Adjustments (RTA 5) in  C /F/K. The RTA can be set  10C  ( 18F  ).
                        fprintf(serialObject,'RIRTA5');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Internal RTA 5 [ C /F/K]:',retval); end

                    case 'external_RTA1'    % Read: Real (external) Temperature Adjustments (RTA 1) in  C /F/K. The RTA can be set  10C  ( 18F  ).
                        fprintf(serialObject,'RERTA1');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: External RTA 1 [ C /F/K]:',retval); end

                    case 'external_RTA2'    % Read: Real (external) Temperature Adjustments (RTA 2) in  C /F/K. The RTA can be set  10C  ( 18F  ).
                        fprintf(serialObject,'RERTA2');
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: External RTA 2 [ C /F/K]:',retval); end

                    case 'external_RTA3'    % Read: Real (external) Temperature
```

```
Adjustments (RTA 3) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'RERTA3');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: External RTA 3 [ C /F/K]:',retval); end

        case 'external_RTA4'     % Read: Real (external) Temperature Adjustments (RTA 4) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'RERTA4');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: External RTA 4 [ C /F/K]:',retval); end

        case 'external_RTA5'     % Read: Real (external) Temperature Adjustments (RTA 5) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'RERTA5');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: External RTA 5 [ C /F/K]:',retval); end

        case 'setpoint_1'                        % Read: Setpoint 1 temperature in C /F/K (The setpoint is the desired fluid temperature).
            fprintf(serialObject,'RS1');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Setpoint 1 [ C /F/K]:',retval); end

        case 'setpoint_2'                        % Read: Setpoint 2 temperature in C /F/K.
            fprintf(serialObject,'RS2');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Setpoint 2 [ C /F/K]:',retval); end

        case 'setpoint_3'                        % Read: Setpoint 3 temperature in C /F/K.
            fprintf(serialObject,'RS3');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Setpoint 3 [ C /F/K]:',retval); end

        case 'setpoint_4'                        % Read: Setpoint 4 temperature in C /F/K.
            fprintf(serialObject,'RS4');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Setpoint 4 [ C /F/K]:',retval); end

        case 'setpoint_5'                        % Read: Setpoint 5 temperature in C /F/K.
            fprintf(serialObject,'RS5');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Setpoint 5 [ C /F/K]:',retval); end

        case 'high_temperature_fault'            % Read: High temperature fault in C /F/K.
            fprintf(serialObject,'RHTF');
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: High temperature fault value [ C /F/K]:',retval); end

```

```
            case 'high_temperature_warn'          % Read: High temperature warning in  C /F/K.
                fprintf(serialObject,'RHTW');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: High temperature warning value [ C /F/K]:',retval); end

            case 'low_temperature_fault'          % Read: Low temperature fault in  C /F/K.
                fprintf(serialObject,'RLTF');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Low temperature fault value [ C /F/K]:',retval); end

            case 'low_temperature_warn'           % Read: Low temperature warning in  C /F/K.
                fprintf(serialObject,'RLTW');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Low temperature warning value [ C /F/K]:',retval); end

            case 'proportional_heat_band_setting'          % Read: Proportional heat band setting in %.
                fprintf(serialObject,'RPH');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Percentage of proportional HEAT band setting [%]:',retval); end

            case 'proportional_cool_band_setting'          % Read: Proportional cool band setting in %.
                fprintf(serialObject,'RPC');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Percentage of proportional COOL band setting:',retval); end

            case 'integral_heat_band_setting'              % Read: Integrall heat band setting in Repeats per minute.
                fprintf(serialObject,'RIH');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Integral HEAT band setting [RepeatsPerMinute]:',retval); end

            case 'integral_cool_band_setting'              % Read: Integral cool band setting in Repeats per minute.
                fprintf(serialObject,'RIC');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Integral COOL band setting [RepeatsPerMinute]:',retval); end

            case 'derivative_heat_band_setting'            % Read: Derivative heat band setting in minutes.
                fprintf(serialObject,'RDH');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Derivative HEAT band setting [Minutes]:',retval); end

            case 'derivative_cool_band_setting'            % Read: Derivative COOL band setting in minutes.
                fprintf(serialObject,'RDC');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Derivative COOL band setting [Minutes]:',retval); end

            case 'temperature_precision'                   % Read: Temperature precision.
```

```
                fprintf(serialObject,'RTP');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Temperature Precision:',retval); end

            case 'temperature_units'                    % Read: Temperature units [ C /F/K].
                fprintf(serialObject,'RTU');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Temperature units [ C /F/K]:',retval); end

            case 'unit_on'                              % Read: Units. True or False [1/0].
                fprintf(serialObject,'RO');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Show Temperature units [ C /F/K] status bit:',retval); end

            case 'external_probe_enabled'               % Read: External Probe enabled. Enabled/Disabled [1/0].
                fprintf(serialObject,'RE');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Reading External Probe status bit:',retval); end

            case 'auto_restart_enabled'                 % Read: Auto Restart enabled. Enabled/Disabled [1/0].
                fprintf(serialObject,'RAR');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Auto Restart status bit:',retval); end

            case 'energy_saving_mode'                   % Read: External Probe enabled. Enabled/Disabled [1/0].
                fprintf(serialObject,'REN');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200:  Energy saving mode status bit:',retval); end

            case 'time'                                 % Read: Time [hh:mm:ss].
                fprintf(serialObject,'RCK');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: The current time is:',retval); end

            case 'date'                                 % Read: Date [mm/dd/yy] or [dd/mm/yy].
                fprintf(serialObject,'RDT');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: The current date ([mm/dd/yy] or [dd/mm/yy]) is :',retval); end

            case 'date_format'                          % Read: Date Format [mm/dd/yy] or [dd/mm/yy].
                fprintf(serialObject,'RDF');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: The date format is:',retval); end

            case 'ramp_status'                          % Read: Ramp Status [Stopped/Running/Paused].
                fprintf(serialObject,'RRS');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: The Ramp
```

```
Status [Stopped/Running/Paused] is:',retval); end

            case 'firmware_version'                    % Read: Firmware Version.
                fprintf(serialObject,'RVER');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: The Firmware version is:',retval); end

            case 'firmware_checksum'                   % Read: Firmware Checksum.
                fprintf(serialObject,'RSUM');
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: The Firmware Checksum is:',retval); end

            case 'unit_fault_status'              % This command returns 5 values. These are decimal representations of hexadecimal values. Each individual bit of the value represents a different warning, fault or status.
                fprintf(serialObject,'RUFS');
                [V1,V2,V3,V4,V5] = fscanf(serialObject,'%s');
                retval = [V1;V2;V3;V4;V5];
                if verbose >=2, fprintf(1,'%s %s\n','isoTemp 6200: Read Unit Fault System:',retval); end

        end


    case 'set'
        switch 'actionCommand'

            case 'displayed_setpoint'             % Set: Displayed Setpoint Temperature in  C /F/K (it depends on the Instrument internal configuration)
                fprintf(serialObject,'SS %f',value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Set displayed setpoint to ',value,' C  :',retval); end

            case 'internal_RTA1'     % Set: Real (internal) Temperature Adjustments (RTA 1) in  C /F/K. The RTA can be set   10C   (   18F   ).
                fprintf(serialObject,'SIRTA1 %f',value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 1 set to ',value,'[ C /F/K] :',retval); end

            case 'internal_RTA2'     % Set: Real (internal) Temperature Adjustments (RTA 2) in  C /F/K. The RTA can be set   10C   (   18F   ).
                fprintf(serialObject,'SIRTA2 %f',value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 2 set to ',value,'[ C /F/K] :',retval); end

            case 'internal_RTA3'     % Set: Real (internal) Temperature Adjustments (RTA 3) in  C /F/K. The RTA can be set   10C   (   18F   ).
                fprintf(serialObject,'SIRTA3 %f',value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 3 set to ',value,'[ C /F/K] :',retval); end

            case 'internal_RTA4'     % Set: Real (internal) Temperature Adjustments (RTA 4) in  C /F/K. The RTA can be set   10C   (
```

```
18F ) .
            fprintf(serialObject,'SIRTA4 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 4 set to ',value,'[ C /F/K] :',retval); end

        case 'internal_RTA5'    % Set: Real (internal) Temperature Adjustments (RTA 5) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'SIRTA5 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 5 set to ',value,'[ C /F/K] :',retval); end

        case 'external_RTA1'    % Set: Real (external) Temperature Adjustments (RTA 1) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'SERTA1 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 1 set to ',value,'[ C /F/K] :',retval); end

        case 'external_RTA2'    % Set: Real (external) Temperature Adjustments (RTA 2) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'SERTA2 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 2 set to ',value,'[ C /F/K] :',retval); end

        case 'external_RTA3'    % Set: Real (external) Temperature Adjustments (RTA 3) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'SERTA3 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 3 set to ',value,'[ C /F/K] :',retval); end

        case 'external_RTA4'    % Set: Real (external) Temperature Adjustments (RTA 4) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'SERTA4 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 4 set to ',value,'[ C /F/K] :',retval); end

        case 'external_RTA5'    % Set: Real (external) Temperature Adjustments (RTA 5) in C /F/K. The RTA can be set 10C ( 18F ).
            fprintf(serialObject,'SERTA4 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Internal RTA 5 set to ',value,'[ C /F/K] :',retval); end

        case 'setpoint_1'               % Set: Setpoint 1 temperature in C /F/K (The setpoint is the desired fluid temperature).
            fprintf(serialObject,'SS1 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Setpoint 1 set to ',value,'[ C /F/K] :',retval); end

        case 'setpoint_2'               % Set: Setpoint 2 temperature in C /F/K.
            fprintf(serialObject,'SS2 %f',value);
            retval = fscanf(serialObject,'%s');
```

```
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Setpoint 2 set to ',value,'[ C /F/K] :',retval); end

        case 'setpoint_3'                 % Set: Setpoint 3 temperature in C /F/K.
            fprintf(serialObject,'SS3 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Setpoint 3 set to ',value,'[ C /F/K] :',retval); end

        case 'setpoint_4'                 % Set: Setpoint 4 temperature in C /F/K.
            fprintf(serialObject,'SS4 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Setpoint 4 set to ',value,'[ C /F/K] :',retval); end

        case 'setpoint_5'                 % Set: Setpoint 5 temperature in C /F/K.
            fprintf(serialObject,'SS5 %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Setpoint 5 set to ',value,'[ C /F/K] :',retval); end

        case 'high_temperature_fault'     % Set: High temperature fault in C /F/K.
            fprintf(serialObject,'SHTF %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: High temperature fault value set to ',value,'[ C /F/K] :',retval); end

        case 'high_temperature_warn'      % Set: High temperature warning in C /F/K.
            fprintf(serialObject,'SHTW %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: High temperature warning value set to ',value,'[ C /F/K] :',retval); end

        case 'low_temperature_fault'      % Set: Low temperature fault in C /F/K.
            fprintf(serialObject,'SLTF %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Low temperature fault value set to ',value,'[ C /F/K] :',retval); end

        case 'low_temperature_warn'       % Set: Low temperature warning in C /F/K.
            fprintf(serialObject,'SLTW %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Low temperature warning value set to ',value,'[ C /F/K] :',retval); end

        case 'proportional_heat_band_setting'      % Set: Proportional heat band setting in %.
            fprintf(serialObject,'SPH %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Percentage of proportional Heat band setting set to ',value,': ',retval); end

        case 'proportional_cool_band_setting'      % Set: Proportional cool band setting in %.
```

```
            fprintf(serialObject,'SPC %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Percentage of proportional Cool band setting set to ',value,': ',retval); end

        case 'integral_heat_band_setting'            % Set: Integrall heat band setting in Repeats per minute.
            fprintf(serialObject,'SIH %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Integral heat band setting set to ',value,'[RepeatPerMinute]: ',retval); end

        case 'integral_cool_band_setting'            % Set: Integral cool band setting in Repeats per minute.
            fprintf(serialObject,'SIC %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Integral cool band setting set to ',value,'[RepeatPerMinute]: ',retval); end

        case 'derivative_heat_band_setting'          % Set: Derivative heat band setting in minutes.
            fprintf(serialObject,'SDH %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Derivative heat band setting set to ',value,'[Minutes]: ',retval); end

        case 'derivative_cool_band_setting'          % Set: Derivative cool band setting in minutes.
            fprintf(serialObject,'SDC %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Derivative cool band setting set to ',value,'[Minutes]: ',retval); end

        case 'temperature_resolution'                 % Set: Temperature resolution.
            fprintf(serialObject,'STR %f',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Temperature Resolution set to ',value,'[ C ]: ',retval); end

        case 'temperature_units'                      % Set: Temperature units [ C ].
            fprintf(serialObject,'STU %s',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Temperature unit set to ',value,' ( C ): ',retval); end

        case 'unit_on_status'                         % Set: Unit Status. True or False [1/0].
            fprintf(serialObject,'SO %d',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Temperature unit status  bit set to ',value,' :',retval); end

        case 'external_probe_on_status'                % Set: External Probe ON status. Enabled/Disabled [1/0].
            fprintf(serialObject,'SE %d',value);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200:
```

```
                    External Probe status bit set to',value,' :',retval); end

            case 'auto_restart_enabled'                       % Set: Auto
             Restart bit status. Enabled/Disabled [1/0].
                fprintf(serialObject,'SAR %d',value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200:
                 Auto Restart status bit set to ',value,': ',retval); end

            case 'energy_saving_mode'                         % Set: Energy
             saving mode bit status. Enabled/Disabled [1/0].
                fprintf(serialObject,'SEN %f',value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200:
                 Energy saving mode status bit set to ',value,': ',retval);
                 end

            case 'pump_speed'                          % Set: Pump Speed [L/M/H
             ](Low/Medium/High).
                fprintf(serialObject,'SPS %s',value);
                retval = fscanf(serialObject,'%s');

                if strcmpi(value, 'L')
                    aux = 'Low';
                elseif strcmpi(value, 'H')
                    aux = 'High';
                else
                    aux = 'Medium';
                end

                if verbose >=2, fprintf(1,'%s %s %s %s\n','isoTemp 6200:
                 Pump speed set to ',aux,' :',retval); end
                clear aux;

            case 'ramp_number'                                % Set: Ramp
             Status [Stopped/Running/Paused].
                fprintf(serialObject,'SRN %f',value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200:
                 Ramp Status set to ',value,' :',retval); end

        end
    end

end

fclose(serialObject);
delete(serialObject);
clear serialObject;

end
```

Code 4.9 – *Recirculating Chiller Library*

4.3.2.2 MKS 925 MicroPirani Vacuum Transducer

This vacuum pressure transducer will be used when vacuum test take place in the TVAC Chamber. The query command list of this instrument can be found in its website [18]. The syntax of the command can be observed in the following code fragment:

```
Syntax:
 All commands are on the form:
   @<ADR><COM>!<PAR>;FF
 and all queries are on the form:
   @<ADR><QRY>?;FF
 where:
   <ADR>: RS address of transducer.
   <COM>: Command.
   <PAR>: Parameter.
   <QRY>: Query.
 The address is in the 001-253 range.
```

Code 4.10 – *MKS 925 Command Syntax[18]*

An example piece of code for the communication with the transducer is shown in Code 4.11.


```
# Transducer Information
@123DT?;FF #DT  (Query)   Device type          Typical Response = @123ACKMICROPIRANI;FF
@123FV?;FF #FV  (Query)   Firmware Version     Typical Response = @123ACK1.33;FF
@123MF?;FF #MF  (Query)   Manufacturer Name    Typical Response = @123ACKMKS;FF
@123SN?;FF #SN  (Query)   Serial Number        Typical Response = @123ACK1302856880;FF
@123TIM?;FF #TIM (Query)  Time on (hours of operation)      Typical Response = @123ACK277;FF
@123TEM?;FF #TIM (Query)  MicroPirani sensor temperature    Typical Response = @123ACK2.57E+1;FF

# Pressure Reading
@123PR1?;FF #PR  (Query)  Sensor pressure as 3 digit floating point value Example Response = @123ACK9.00E+2;FF
@123PR4?;FF #PR  (Query)  Sensor pressure as 4 digit floating point value Example Response = @123ACK9.000E+2;FF

# Calibration and adjustment information
@123U?;FF #U     (Query)  Pressure unit setup (Torr, mbar, Pascal)   Typical Response = @123ACKTORR;FF // TORR
@123U!<TORR,MBAR,PASCAL>;FF   # Set pressure unit setup (Torr. mbar. Pascal) Typical Response =@123ACKMBAR;FF
```

Code 4.11 – *MKS 925 Example Command List*

Finally, the implemented library for controlling the vacuum transducer is shown in 4.12.

```
function retval=CommPort_mks925(Instrument,serialPort,Tiempo,verbose,RW_command, actionCommand,value)

% Versión 1: (1/8/2018 Juan Manuel López Torralba
%            Serial Communications Protocol MKS 925 MNicroPirani
%            Driver


%Once the serial port reference is created, it will be accessible the next time
```

```
 the file is used.
global serialObject;

if (ischar(Instrument))
    instrument=Instrument;
else
    instrument=Instrument.Name;
    serialPort=Instrument.serialPort;
end

if serialPort
    if ischar(serialPort)
        serialPort = str2num(serialPort);
    end
else
    serialPort = 123;
end
%% Serial Object Definition

serialObject = serial(serialPort, 'baud', 19200,...
    'StopBits',1  ,...
    'DataBits', 8,...
    'Parity', 'none',...
    'Timeout', Tiempo,...
    'Terminator','CR',...
    'FlowControl', 'none');
%                 'ReadAsyncMode', 'Continuous',...
%                 'ReadAsyncMode', 'manual',...
%                 'Timeout', 1,...
%
fopen(serialObject);

%% BEGIN CODE

if (strcmpi(instrument, 'mks925') || strcmpi(instrument, 'all'))

    switch RW_command
        case 'query'
            switch actionCommand

                case 'baudrate'                    % Query: Baud Rate
                    fprintf(serialObject,'@%dBR?;FF',serialPort);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK9600;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: Communication Baud Rate:',retval); end

                case 'address'                     % Query: Transducer communication address (001 to 253)
                    fprintf(serialObject,'@%dAD?;FF',serialPort);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK253;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: Transducer communication address:',retval); end

                case 'delay'                       % Query: Communication delay between receive and transmit sequence.
                    fprintf(serialObject,'@%dRSD?;FF',serialPort);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACKON;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: Communication delay between receive and transmit sequence:',retval); end

                case 'pressure'                    % Query: Communication delay between receive and transmit sequence.
                    fprintf(serialObject,'@%dPR1?;FF',serialPort);
```

```
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACK9.00E+2;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani sensor pressure as 3 digit floating point value:',retval); end

        case 'pressure_accurate'        % Query: MicroPirani sensor pressure as 4 digit floating point value.
            fprintf(serialObject,'@%dPR4?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACKSET;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani sensor pressure as 4 digit floating point value:',retval); end

        case 'setpoint_relay_1'         % Query: Setpoint relay 1 status (SET=Relay energized / CLEAR=Relay deenergized)
            fprintf(serialObject,'@%dSS1?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACKSET;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relay 1 status:',retval); end

        case 'setpoint_relay_2'         % Query: Setpoint relay 2 status (SET=Relay energized / CLEAR=Relay deenergized)
            fprintf(serialObject,'@%dSS2?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relay 2 status:',retval); end

        case 'setpoint_relay_3'         % Query: Setpoint relay 3 status (SET=Relay energized / CLEAR=Relay deenergized)
            fprintf(serialObject,'@%dSS3?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relay 3 status:',retval); end

        case 'setpoint_switch_1'        % Query: Setpoint 1 switch value
            fprintf(serialObject,'@%dSP1?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACK1.00E-2;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint switch 1 value:',retval); end

        case 'setpoint_switch_2'        % Query: Setpoint 2 switch value
            fprintf(serialObject,'@%dSP2?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint switch 2 value:',retval); end

        case 'setpoint_switch_3'        % Query: Setpoint 3 switch value
            fprintf(serialObject,'@%dSP3?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint switch 3 value:',retval); end

        case 'setpoint_hysteresis_1'        % Query: Setpoint 1 hysteresis switch value
            fprintf(serialObject,'@%dSH1?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACK1.10E-2;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint hysteresis switch 1 value:',retval); end
```

```

                    case 'setpoint_hysteresis_2'        % Query: Setpoint 2 hysteresis switch value
                        fprintf(serialObject,'@%dSH2?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint hysteresis switch 2 value:',retval); end

                    case 'setpoint_hysteresis_3'        % Query: Setpoint 3 hysteresis switch value
                        fprintf(serialObject,'@%dSH3?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint hysteresis switch 3 value:',retval); end

                    case 'setpoint_enable_1'        % Query: Setpoint 1 enable status
                        fprintf(serialObject,'@%dEN1?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        % Example Response: @xxxACK1.10E-2;FF
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint enable status 1:',retval); end

                    case 'setpoint_enable_2'        % Query: Setpoint 2 enable status
                        fprintf(serialObject,'@%dEN2?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint enable status 2:',retval); end

                    case 'setpoint_enable_3'        % Query: Setpoint 3 enable status
                        fprintf(serialObject,'@%dEN3?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint enable status 3:',retval); end

                    case 'setpoint_relaydirection_1'        % Query: Setpoint relaydirection (ABOVE or BELOW)
                        fprintf(serialObject,'@%dEN1?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        % Example Response: @xxxACKBELOW;FF
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relaydirection 1:',retval); end

                    case 'setpoint_relaydirection_2'        % Query: Setpoint relaydirection (ABOVE or BELOW)
                        fprintf(serialObject,'@%dEN2?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relaydirection 2:',retval); end

                    case 'setpoint_relaydirection_3'        % Query: Setpoint relaydirection (ABOVE or BELOW)
                        fprintf(serialObject,'@%dEN3?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relaydirection 3:',retval); end

                    case 'setpoint_safety_relay'        % Query: Setpoint safety delay
                        fprintf(serialObject,'@%dSPD?;FF',serialPort);
                        retval = fscanf(serialObject,'%s');
                        % Example Response: @xxxACKON;FF
                        if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint safety delay:',retval); end
```

```
        case 'model_number'         % Query: model number (925)
            fprintf(serialObject,'@%dMD?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACK925;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Model Number:',retval); end

        case 'device_type'          % Query: Device type name (MicroPirani)
            fprintf(serialObject,'@%dDT?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACKMICROPIRANI;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Device Type:',retval); end

        case 'manufacturer'         % Query: Manufacturer Name
            fprintf(serialObject,'@%dMF?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACKMKS;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Manufacturer Name:',retval); end

        case 'hardware_version'         % Query: Hardware version
            fprintf(serialObject,'@%dHV?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACKA;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Hardware version:',retval); end

        case 'firmware_version'         % Query: Firmware version
            fprintf(serialObject,'@%dFV?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACK1.33;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Firmware version:',retval); end

        case 'serial_number'         % Query: Serial Number
            fprintf(serialObject,'@%dSN?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACK1302856880;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Serial Number:',retval); end

        case 'switch_enable'         % Query: switch enable
            fprintf(serialObject,'@%dSW?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACK1302856880;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani switch enable:',retval); end

        case 'time'         % Query: Time on ( hours of operation )
            fprintf(serialObject,'@%dTIM?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACK277;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Time on ( hours of operation ):',retval); end

        case 'temperature'          % Query: MicroPirani sensor temperature
            fprintf(serialObject,'@%dTEM?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACK2.57E+1;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani sensor temperature:',retval); end

```

```
        case 'text_string'          % Query: MicroPirani user programmed text_string
            fprintf(serialObject,'@%dUT?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACKVACUUM1;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani user programmed text_string:',retval); end

        case 'status_check'          % Query: MicroPirani Transducer status check
            fprintf(serialObject,'@%dT?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACKO;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Transducer status check:',retval); end

        case 'pressure_unit'          % Query: MicroPirani Pressure unit setup (Torr, mbar, Pascal)
            fprintf(serialObject,'@%dU?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @123ACKTORR;FF // TORR
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Pressure unit setup:',retval); end

        case 'calibration_gas'          % Query: MicroPirani sensor calibration gas (Nitrogen, Air, Argon, Helium, Hydrogen, H2O, Neon, CO2, Xenon)
            fprintf(serialObject,'@%dGT?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACKNITROGEN;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani sensor calibration gas:',retval); end

        case 'vacuum'                 % Query: Provides delta pressure value between current vacuum zero adjustment and factory calibration.
            fprintf(serialObject,'@%dVAC?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACK5.12E-5;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani delta pressure value:',retval); end

        case 'atmospheric'              % Query: Provides delta pressure value between current current atmospheric adjustment and factory calibration.
            fprintf(serialObject,'@%dATM?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACK1.22E+1;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani delta atmospheric value:',retval); end

        case 'analog_voltage_output_1'              % Query: Analog voltage output 1: Pressure assignment and calibration
            fprintf(serialObject,'@%dAO1?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACK10;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Analog voltage output 1:',retval); end

        case 'analog_voltage_output_2'              % Query: Analog voltage output 2: Pressure assignment and calibration
            fprintf(serialObject,'@%dAO2?;FF',serialPort);
            retval = fscanf(serialObject,'%s');
            % Example Response: @xxxACK10;FF
            if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Analog voltage output 2:',retval); end
```

```
            end


        case 'set'
            switch 'actionCommand'

                case 'displayed_setpoint'              % Set: Displayed Setpoint Temperature in  C /F/K (it depends on the Instrument internal configuration)
                    fprintf(serialObject,'SS %f',value);
                    retval = fscanf(serialObject,'%s');
                    if verbose >=2, fprintf(1,'%s %f %s %s\n','isoTemp 6200: Set displayed setpoint to ',value,' C  :',retval); end

                case 'setpoint_switch_1'           % Set: Setpoint 1 switch value
                    fprintf(serialObject,'@%dSP1!%f;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK2.00E+1;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint switch 1 value:',retval); end

                case 'setpoint_switch_2'           % Set: Setpoint 2 switch value
                    fprintf(serialObject,'@%dSP2!%f;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint switch 2 value:',retval); end

                case 'setpoint_switch_3'           % Set: Setpoint 3 switch value
                    fprintf(serialObject,'@%dSP3!%f;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint switch 3 value:',retval); end

                case 'setpoint_hysteresis_1'         % Set: Setpoint 1 hysteresis switch value
                    fprintf(serialObject,'@%dSH1!%f;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK1.10E-2;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint hysteresis switch 1 value:',retval); end

                case 'setpoint_hysteresis_2'         % Set: Setpoint 2 hysteresis switch value
                    fprintf(serialObject,'@%dSH2!%f;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint hysteresis switch 2 value:',retval); end

                case 'setpoint_hysteresis_3'         % Set: Setpoint 3 hysteresis switch value
                    fprintf(serialObject,'@%dSH3!%f;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint hysteresis switch 3 value:',retval); end

                case 'setpoint_enable_1'         % Set: Setpoint 1 enable status (ON/OFF)
                    fprintf(serialObject,'@%dEN1!%s;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK1.10E-2;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint enable status 1:',retval); end

                case 'setpoint_enable_2'         % Set: Setpoint 2 enable status (ON/OFF)
```

```
                fprintf(serialObject,'@%dEN2!%s;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint enable status 2:',retval); end

            case 'setpoint_enable_3'          % Set: Setpoint 3 enable status (ON/OFF)
                fprintf(serialObject,'@%dEN3!%s;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint enable status 3:',retval); end

            case 'setpoint_relaydirection_1'          % Set: Setpoint relaydirection (ABOVE or BELOW)
                fprintf(serialObject,'@%dEN1!%s;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                % Example Response: @xxxACKBELOW;FF
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relaydirection 1:',retval); end

            case 'setpoint_relaydirection_2'          % Set: Setpoint relaydirection (ABOVE or BELOW)
                fprintf(serialObject,'@%dEN2!%s;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relaydirection 2:',retval); end

            case 'setpoint_relaydirection_3'          % Set: Setpoint relaydirection (ABOVE or BELOW)
                fprintf(serialObject,'@%dEN3!%s;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint relaydirection 3:',retval); end

            case 'setpoint_safety_relay'          % Set: Setpoint safety delay
                fprintf(serialObject,'@%dSPD!ON;FF',serialPort);
                retval = fscanf(serialObject,'%s');
                % Example Response: @xxxACKON;FF
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Setpoint safety delay:',retval); end

            case 'baudrate'                   % Set: Baud Rate (4800, 9600, 19200, 38400,57600, 115200, 230400)
                fprintf(serialObject,'@%dBR!%d;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                % Example Response: @xxxACK19200;FF
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: Communication Baud Rate:',retval); end

            case 'address'                    % Set: Transducer communication address (001 to 253)
                fprintf(serialObject,'@%dAD!%d;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                % Example Response: @xxxACK253;FF
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: Transducer communication address:',retval); end

            case 'delay'                      % Set: Communication delay between receive and transmit sequence (ON/OFF)
                fprintf(serialObject,'@%dRSD!%s;FF',serialPort,value);
                retval = fscanf(serialObject,'%s');
                % Example Response: @xxxACKON;FF
                if verbose >=2, fprintf(1,'%s %s\n','MKS 925: Communication delay between receive and transmit sequence:',retval); end

```

```
                case 'pressure_unit'        % Set: MicroPirani Pressure unit setup (Torr, mbar, Pascal)
                    fprintf(serialObject,'@%dU!%s;FF',serialPort,value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACKMBAR;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Pressure unit setup:',retval); end

                case 'calibration_gas'        % Set: MicroPirani sensor calibration gas (Nitrogen, Air, Argon, Helium, Hydrogen, H2O, Neon, CO2, Xenon)
                    fprintf(serialObject,'@%dGT!%s;FF',serialPort, value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACKARGON;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani sensor calibration gas:',retval); end

                case 'vacuum'                 % Set: Executes MicroPirani zero adjustment
                    fprintf(serialObject,'@%dVAC!;FF',serialPort);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani delta pressure value:',retval); end

                case 'atmospheric'             % Set: Executes MicroPirani full scale atmospheric adjustment.
                    fprintf(serialObject,'@%dATM!%f;FF',serialPort, value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani delta atmospheric value:',retval); end

                case 'analog_voltage_output_1'             % Set: analog voltage output 1 calibration
                    fprintf(serialObject,'@%dAO1!%d;FF',serialPort, value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK10;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Analog voltage output 1:',retval); end

                case 'analog_voltage_output_2'             % Set: analog voltage output 2 calibration
                    fprintf(serialObject,'@%dAO2!%d;FF',serialPort, value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACK10;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Analog voltage output 2:',retval); end

                case 'user_tag'               % Set: transducer user tag
                    fprintf(serialObject,'@%dUT!%s;FF',serialPort, value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACKLOADLOCK;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Set transducer user tag:',retval); end

                case 'user_switch'             % Set: Enable / disable user switch (ON/OFF)
                    fprintf(serialObject,'@%dSW!%s;FF',serialPort, value);
                    retval = fscanf(serialObject,'%s');
                    % Example Response: @xxxACKON;FF
                    if verbose >=2, fprintf(1,'%s %s\n','MKS 925: MicroPirani Enable / disable user switch:',retval); end

            end
    end
```

```
end

fclose(serialObject);
delete(serialObject);
clear serialObject;

end
```

Code 4.12 – *MKS 925 RS-232 Library [18]*

4.3.3 Keithley KUSB-3116

The KUSB-3116 is a high-performance, multifunction data acquisition up to 16 analog input channel module for the USB bus [25]. This acquisition module is intended to be controlled with MATLAB and to transfer the acquired data directly to MATLAB for its analysis and representation. This module is used with to acquire data from the *SP-110* pyranometer.

Before the module could be used several drivers and adaptors must be installed. First of all, the device drivers, which includes the Open Layers API as well as some data acquisition and calibration applications, must be installed. In the Figure 4.17, the data flow model between the module and MATLAB is shown.

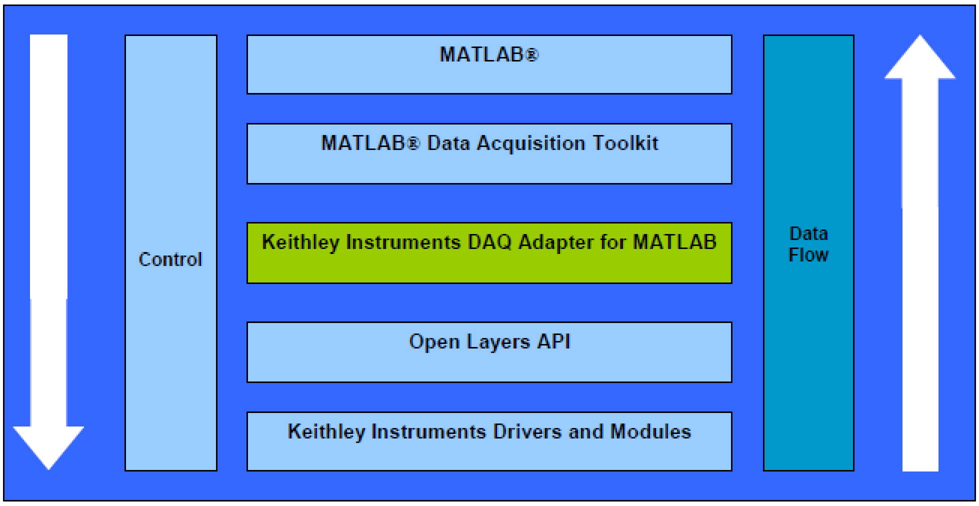


Figure 4.17 – *Keithley-MATLAB Data Flow Model[25].*

Some of the Requirements to use the Keithley module are the following:

- Java.
- Open Layers Version 3.0 or higher.
- MATLAB Version 7 (R14) Service Pack 3 or higher (32 bits).

- MATLAB Data Acquisition Toolbox Version 2.7 or higher.
- quickDAQ Software.
- DAQ Adaptor for MATLAB from Keithley instruments.

Once all the drivers and software tools have been installed, the DAQ Adaptor for MATLAB must be registered in MATLAB. Only the DAQ Adaptor for MATLAB from Keithley instruments works. For doing that, MATLAB must be started in administrator mode and the file *Dtoll.dll* must be referenced as follows:

```
%rehash toolboxcache
%daqregister ('Path del archivo Dtoll.dll )
rehash toolboxcache;
daqregister('c:\Program Files(x86)\Keithley Instruments\DAQ Adaptor for MATLAB\
 Dtol.dll')

%ans = %'Dtol.dll' successfully registered.
```

Code 4.13 – *Installing DAQ Adaptor for MATLAB.*

Finally, an example of how operate with the adaptor in MATLAB is shown in the following piece of code:

```
ai0 = analoginput (    dtol    , 0); % It creates an analog input object 'ai0'
 for a board with index 0
addchannel(ai0, 0);                 % It adds the hardware channel 0 to the 'ai0'
 object
sample = getsample(ai0);            % It gets a sample from that channel.
```

Code 4.14 – *Code Example for operating the KUSB.3116 module.*

4.3.4 Graphical User Interface

In this section the designed software, from which all the simulations are carried out, are going to be meticulously explained. The algorithms used as well as its file structure and flow diagrams will be introduced in order to gain a major understanding of its operation. This GUI have been developed by using the GUIDE MATLAB environment. This environment is based on Java and allows creation of interactive windows, dialog boxes, progress bars, etc.

The Graphical User Interface is the responsible for controlling the instrumentation equipment and managing the data acquired from the aerospace tests, as said in Section 3.1.2. This GUI uses the GPIB Libraries implemented for controlling the electronic load (4.3.1.2) and the power supply (4.3.1.1) as well as the Keithley *KUSB-3116 DAQ Adaptor*(4.3.3) which is used for controlling the pyranometer. Therefore, just the PV System have been include in the final version of this GUI tool due to the manufactured TVAC Chamber was obtained off-schedule. However, all the necessary libraries for controlling the TVAC System have been developed (4.3.2.1, 4.3.1.3 and 4.3.2.2) and tested, thus, those libraries are available for future improvements. Nevertheless, the PV System is good enough for performing aerospace test and characterize the solar panel.

4

In the Figure 4.18, the structure of the suite and items associated with this project are shown. It can be observed that several and distinct software are located under the folder *GranaSAT_Tracker_V15_3 \measure*.

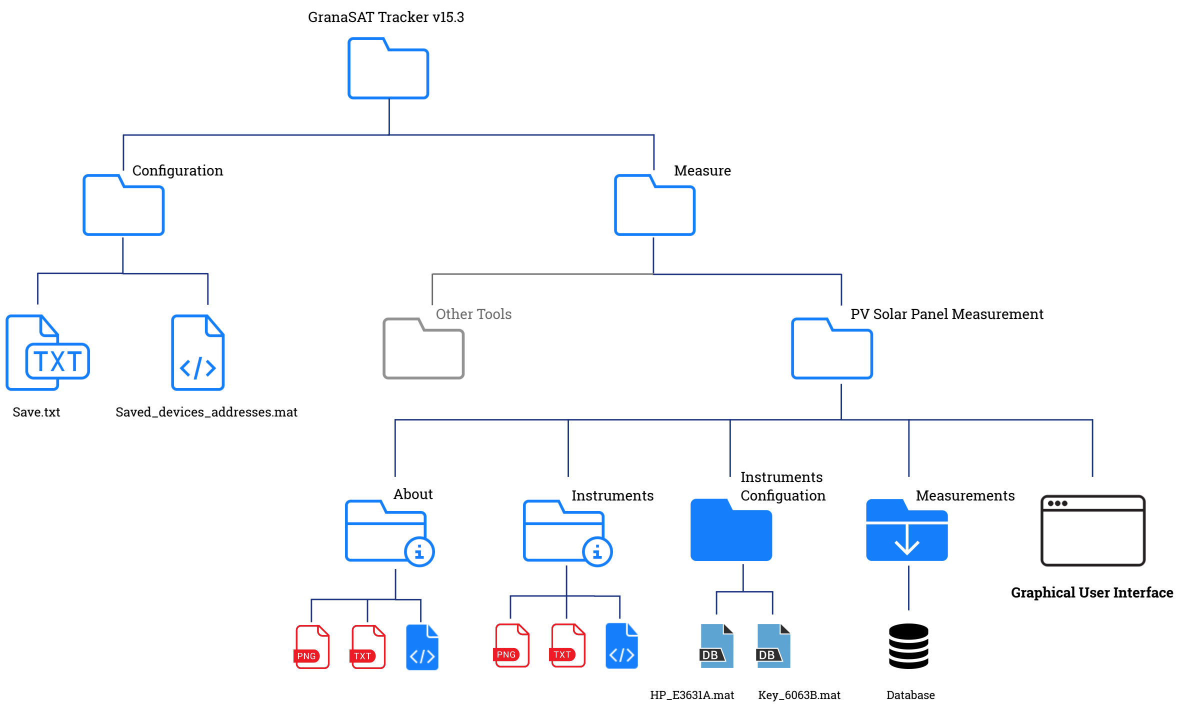


Figure 4.18 – *File Organization Diagram*

The configuration folder of the application contains saved parameters from previous sessions such as the number of samples taken, the threshold limits of the measurement, the maximum number of iterations, etc. Those parameters are needed to provide a better user's experience. When a measurement session is complete, new updated parameters are storage in the file *saved_devices_addresses.mat*. Then, under the Measure directory, the solar panel characterization tool can be found. It is also structured in more folder according to distinct functions of the tool. For example, under the *About* directory some pictures and codes associated with the *About* tab of the measurement panel are found. The *Instruments Configuration* directory contains files with the configuration of each device used in the measurement, this is, the *HP-E3631A Power Supply* and the *Keysight 6063B Electronic Load*. Those parameters are its GPIB addresses, its logical board index, its name and the channel output.

The GUI has been integrated inside a suite developed by the GranaSAT Team where another simulations related with the aerospace environment such as orbit trackers materialize. This suite has a launcher located under the *GranaSAT_Tracker_V15_3* root folder. This launcher allows the user to select different measurement options according to the needs of the test. In our case, this tool is the Solar Panel Measurement. In the Figure 4.19, a caption of this launcher is shown. This launcher is a MATLAB file ('.m') with an associated figure file ('.fig') containing some text and radio buttons for the selection of the measurement tab. Those radio buttons have been associated with the equivalent GUI option. This is shown in the Code 4.15.

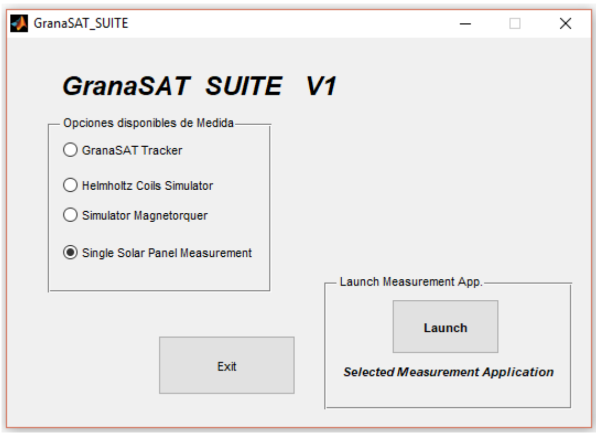


Figure 4.19 – *Granasat Suite v1.0*

```
if get(handles.radiobutton_Suite_Tracker,'Value')
    run('.\GranaSAT_Tracker');
elseif get(handles.radiobutton_Suite_CoilsSimulator,'Value')
    run('.\measure\coils_simulator\helmholtz_coils_simulator');
elseif get(handles.radiobutton_SimulatorMagnetorquer,'Value')
    run('.\measure\magnetorquer_simulator\magnetorquer_simulator');
elseif get(handles.radiobutton_Suite_SingleSolarPanelMeasurement,'Value')
    run('.\measure\solar_panels\PV_TEST_GUI');
end
```

Code 4.15 – *IF loop for choosing the PV Measurement Tool.*

In the Figure 4.20, the PV Measurement Tool application is shown. A total of five distinct areas can be observed. These zones are the following:

- Menus: File, Instruments and About.
  - File Menu: Load, Save and Exit Tabs.
    1. Load Tab: It allows you to load variables ('.mat') into the program.
    2. Save tab: It allows you to save important data into a var ('.mat'). The data types are saved as struct (including the graphs).
    3. Exit tab: It closes the window.
  - Instruments Menu: HP_E3631A, KeySight 6063B, and Keithley KUSB-3116 DAQ Module Tabs. General Information and the data-sheet of the equipment have been included.
  - About Menu: It contains a Credits Tab with information about the authors and the license.

4

- Input Data: Important Parameters needed for the measurement. This values can be given by the user or being loaded from previous sessions.
  1. V Max: The maximum voltage the solar panel can reach.
  2. I Max: The maximum current the solar panel can reach.
  3. V step: The increment of voltage of the eload.
  4. Max iterations: The maximum number of iterations in the test.
  5. V Min: The minimum voltage the solar panel can reach.
  6. I Min: The minimum current the solar panel can reach.
  7. Samples: The number of samples per iteration the current will be measured. Then it will be averaged.
  8. Delay: The delay between the samples the current will be measured per iteration.
  9. Samples / Voltage Increment: This box allows the user to select different samples options. Those options are: 1, 5, 10 and 50.
  10. Data File: The name of the file where the test data will be stored. It is a *.mat* file.
  11. DAQ Acquisition Channel:
- Measured Values: It displays in real time measured values from the test.
  1. Max Power Point: In Watts. In the I-V test, it represents the point that maximizes the power. It is represented at the end of the test.
  2. Voc: It is the open circuit voltage. It is represented at the end of the test in volts.

3. Fill Factor: It is a measure of the quality of a solar cell. It is the MPP divided by the Voc and the Isc.
4. Isc: It is the short circuit current. It is represented at the end of the test in amperes.
5. Solar Shortwave: In Watts. It is the amount of power (light) received by the pyranometer.
6. Current: In amperes. It is the current circulating across the electronic load in real time.

- Run/Stop Button: It has two functionalities: Run the program and stop it changing its Text field properties.

  1. The main program can now be stopped before the operation begins.
  2. The main program can now be stopped after the operation had finished and before the data representation.

- Plotting Area: Real Time plotting of the performing test.

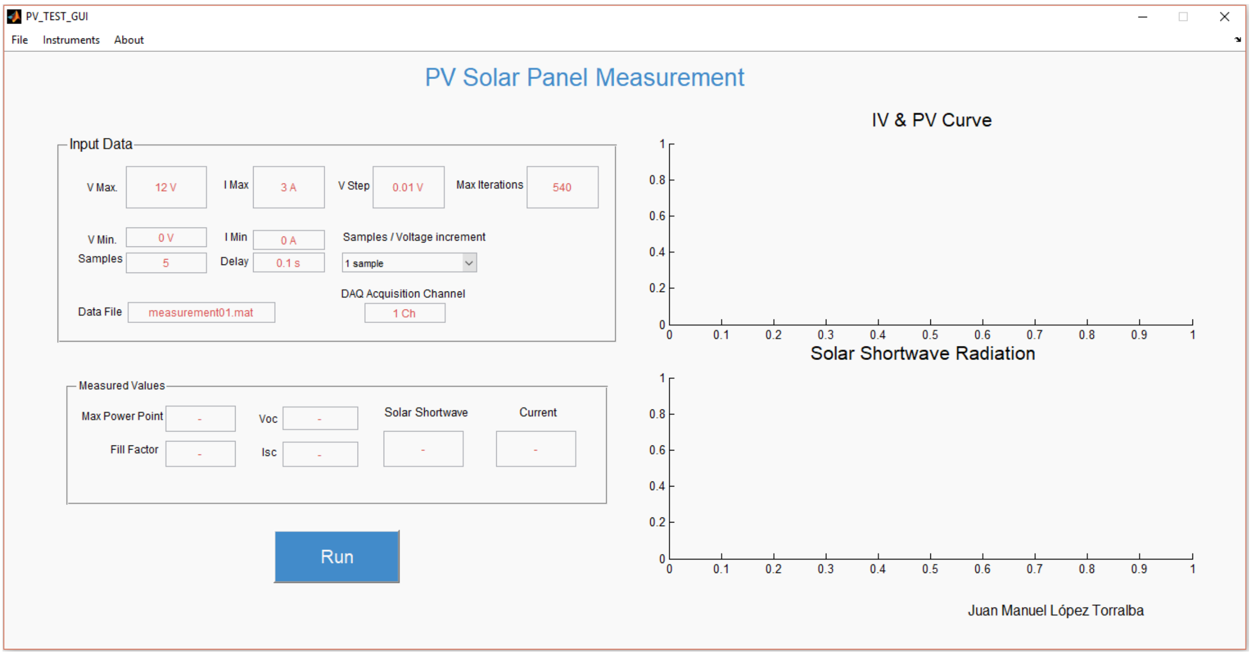


Figure 4.20 – *PV Measurement Tool application.*

Now, a graphical review of the aforementioned areas that comprise the tool is carried out. First, in the Figure 4.21 the drop-down File Menu is shown. The Load and Save tabs launch a file explorer menu for searching or creating the desired files. The Instruments tab is shown in expanded view in Figure 4.22. In Figure 4.23 an example of the window emerged from the selection of the HP E3631A option is shown. Finally, the Credits tab is shown in Figure 4.25.

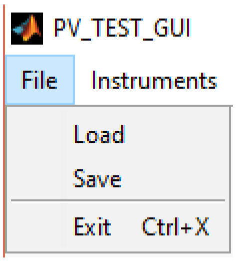


Figure 4.21 – *Application File Menu.*

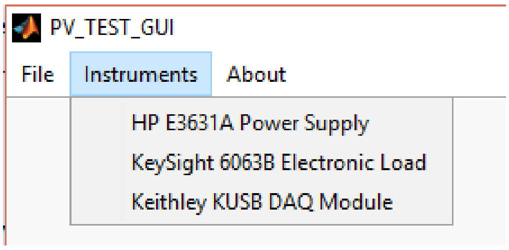


Figure 4.22 – *Application Instruments Menu*

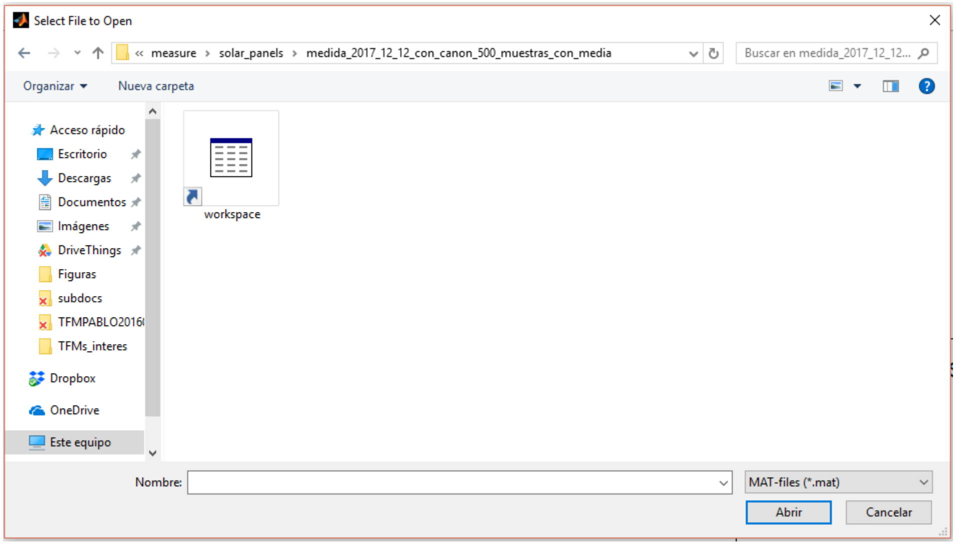


Figure 4.23 – *Application File Menu: Load Tab.*

Figure 4.24 – *Application Instruments Menu: HP E3631A.*

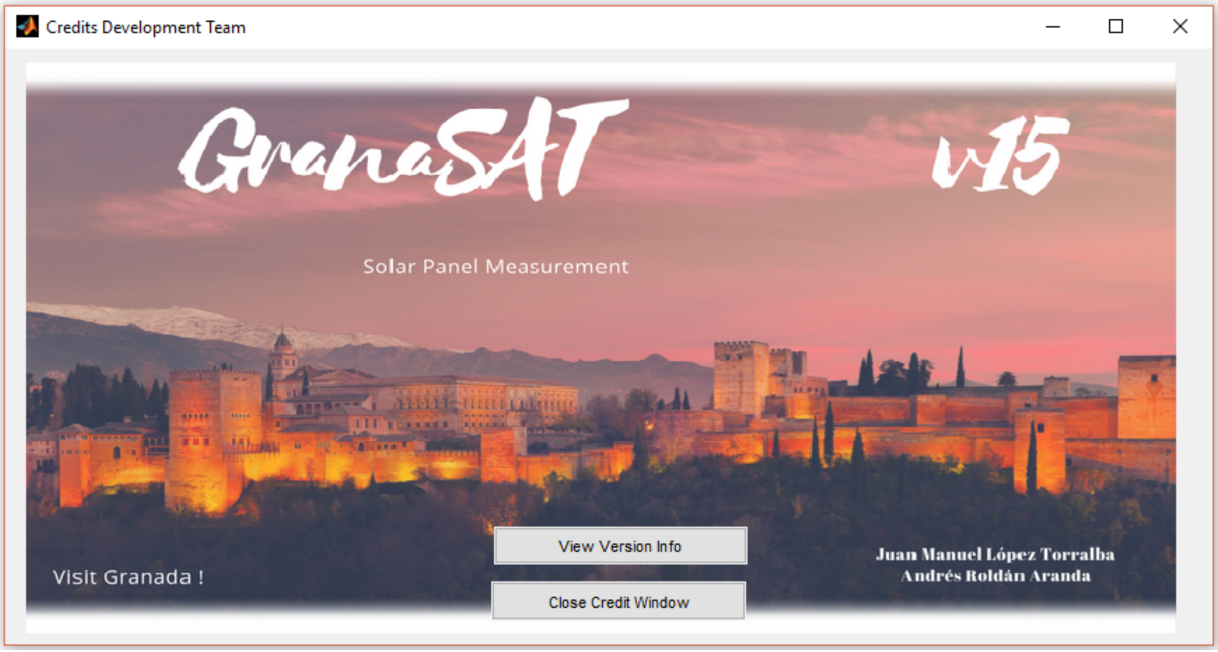


Figure 4.25 – *Application About Menu: Credits Tab.*

The general flow diagram of the Solar Panel Characterization Tool is presented in Figure 4.26. Once the Suite has been run and the Solar Panel Measurement Tool launched, the application firstly declare global variables for the Input Data area management and storage purposes. Then the configuration parameters of the instrumentation equipment are loaded from the Instruments Configuration folder. The connection status of the equipment in subsequently checked. If one instrument is disconnected, an error dialog box is displayed and the application must be restarted. If no errors, the main application interface is displayed. Notice that parameters from previous sessions are displayed in the Input Data area. Even so, these values can be replace by new ones as desired. When the **Run** button is pressed, the new introduced parameters are saved in internal variables of the software. This values are also stored in a matrix file for further analysis. Immediately after have pressed the **Run** button it is transformed in a **Stop** button changing its text field property and functionalities. The **Stop** button can be pressed twice, before the configuration of the equipment and after the I-V Characterization Test. If the **Stop** button is pressed during the test, no effect is expected. During the test, the I-V/P-V and the Solar Shortwave radiation Curves are plotted in real time. Furthermore, the current and solar shortwave single values are displayed in the Measured Values area. Once the test has finished, the remaining parameters are calculated and displayed in that area. Finally, some graphs and reports are generated.

The Characterization test shown in Figure 4.26 is further detailed in the flow diagram shown in Figure 4.27. The approach used in this algorithm is similar as the stated in Section 3.2.1.4.1. Once the Run button is pressed, the parameters previously defined in the Input Data area are now inputs for the algorithm. The first step is to initialize the power supply and then set the boost voltage to a level defined by the user, usually 3 V. Then the electronic load

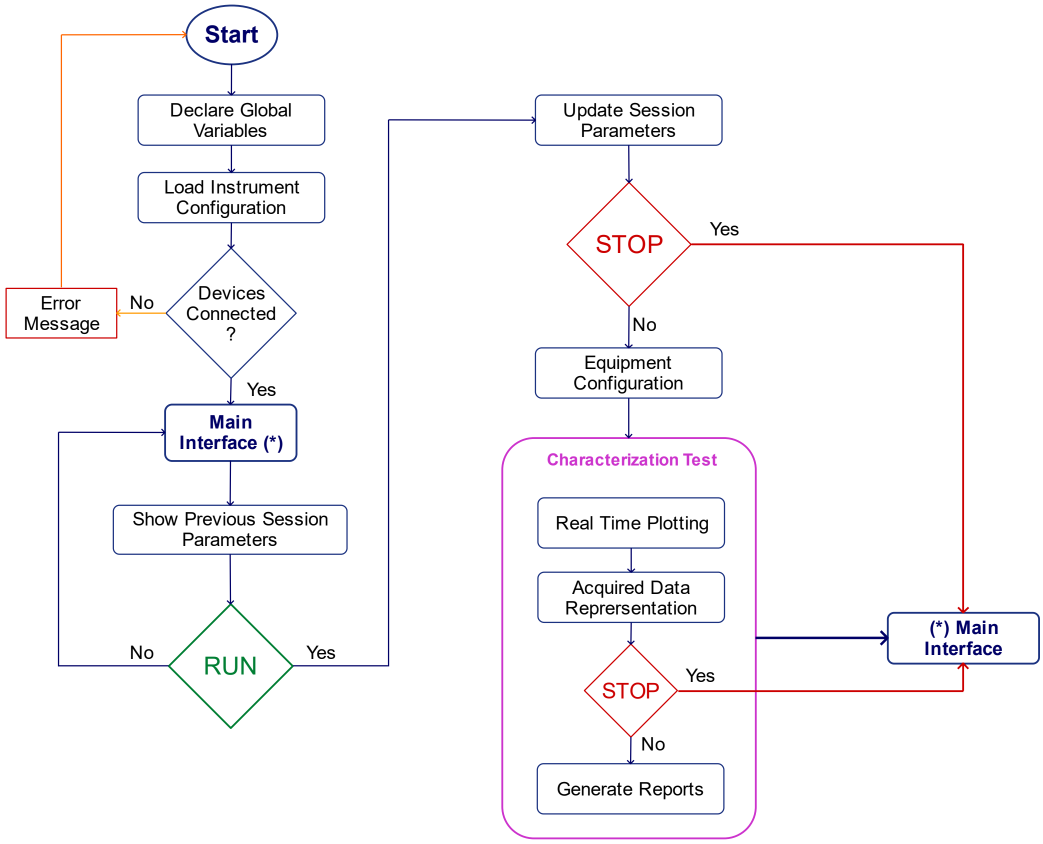


Figure 4.26 – *Solar Panel Characterization General Flow Diagram.*

is also initialised and the CV functioning mode is selected. Next, a while loop is executed before checking if the condition variable is true. The body of the while loop is the code that executes the measurement. The while condition is firstly initialised to true. When the execution is inside the loop, the first thing to do is to set the voltage of the electronic load zero voltage potential across the solar panel. Now the voltage is fixed and the current across the eload can be read after a pause of some milliseconds. The measurement of this current is repeated up to N times according to the user's preferences. This pause can already be fixed by the user in the appropriate area of the GUI. This pause is introduced in the algorithm due to an observed delay between the transfer of the command to the equipment and its establishment. Another relevant issue is the variance observed in the current measured by the electronic load. For that reason the current measurement is done inside a for loop. Then its average value is computed. Finally, the while condition is recomputed, after an increment of the loop counter. If true, the process is repeated but for an increment of voltage defined by the user.

The while condition is recomputed as follows:

```
condition = (numIterations < maxIterations) & (aux_V <
  PV_Solar_Panel_Measurement.Vmax);
```

Code 4.16 – *Recompute while condition.*

4

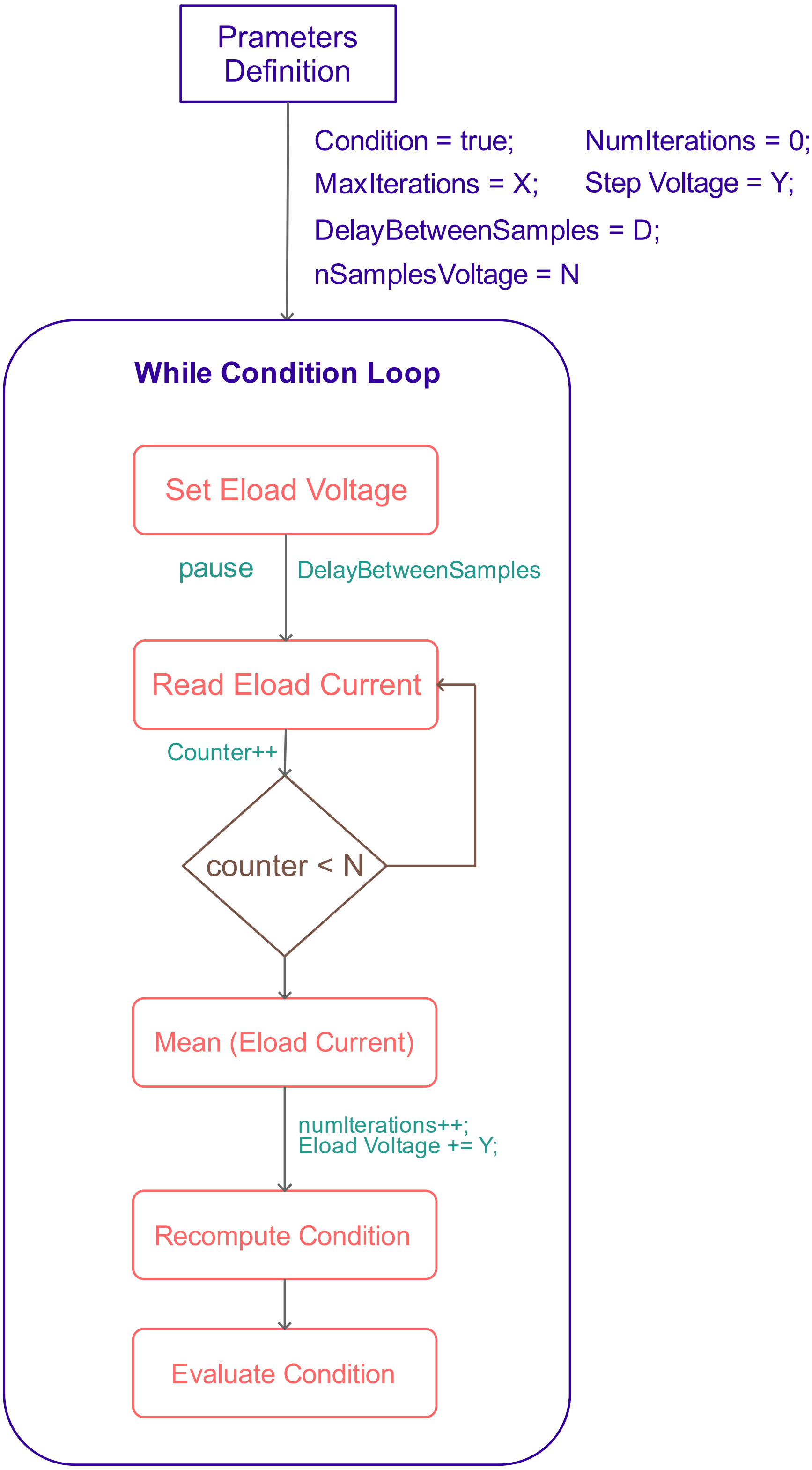


Figure 4.27 – *Characterization Test While Loop Flow Diagram.*

CHAPTER

5

VALIDATION & TEST

After system analysis, technology selection, systems design and the final implementation, the re-examination of the client's requirements and its fulfilment by the system is mandatory. In this Chapter some validation tests are carried out. Those test have been performed in order to separately certify the appropriate functioning of the PV and TVAC Systems as well as the Software Subsystem.

According to the client's requirement related with the PV System established in Section 2, the following results are drawn:

1. The system is able to obtain the I-V Characterization Curve from solar panels as well as to measure valuable electrical parameters from the performing test. In the Figure 5.1, the obtained I-V/PV Curve for a 1U Solar Panel is shown. In Table 5.1 some electrical parameters obtained in the test are shown. This test has been measured using an halogen projector just for testing the software, which explains the difference within the theoretical data measured under space conditions. In Figure 5.2 the I-V Curve is shown in more detail.

Electrical Parameter	Obtained Data	Theoretical Data
Voc	4.77 V	4.66 V
Isc	0.21 A	0.51 A
MPP	0.81 W	2.41 W

Table 5.1 – *Electrical Parameter obtained from 5.1.*

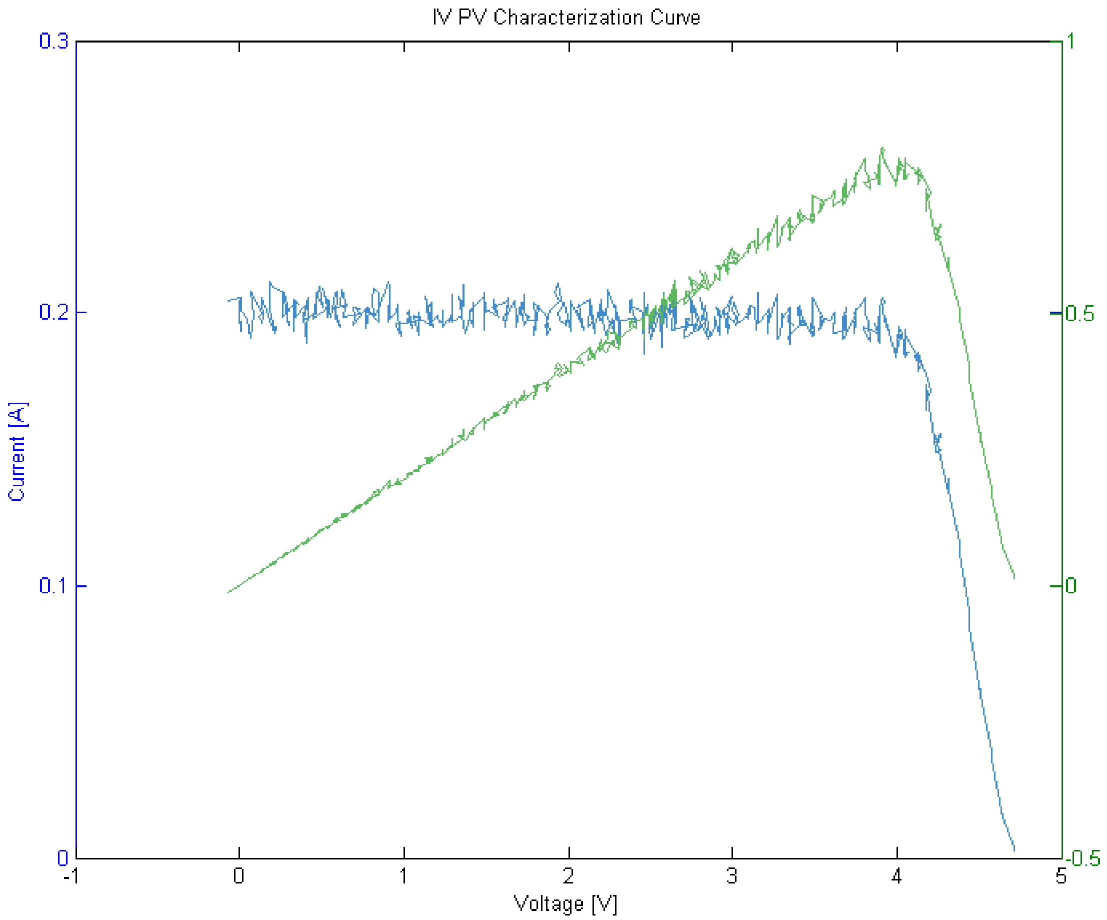


Figure 5.1 – *IV PV Characterization Curve Test for 1U Solar Panel.*

2. The system is able to simulate the sun conditions due to the *Triple Onda Xenon Solar Simulator* as shown in Section 4.2.1, as well as to measure the light irradiance by using the *SP-110* pyranometer and the *KUSB-3116 DAQ*.

3. The system is managed by using a GUI designed in MATLAB which controls all the equipment and data. In Sections 4.3.1 the libraries designed for the HP-E3631A Power Supply (4.3.1.1) and the Keysight 6063B Electronic Load (4.3.1.2) were shown.

4. The GUI implements the control algorithm for managing the equipment and to perform the tests. It was shown in Section 4.3.4 and more detailed in 4.26.

5. The designed GUI is capable of saving all the measured data in text and *.mat* files. Thus, if required, the data from each test would be available for the future.

6. The configuration parameters are automatically saved in the GUI. Those parameters can be found in the *Intruments Configuration* folder. Another session data is stored in the *saved_devices_addresses.mat* file 4.18.

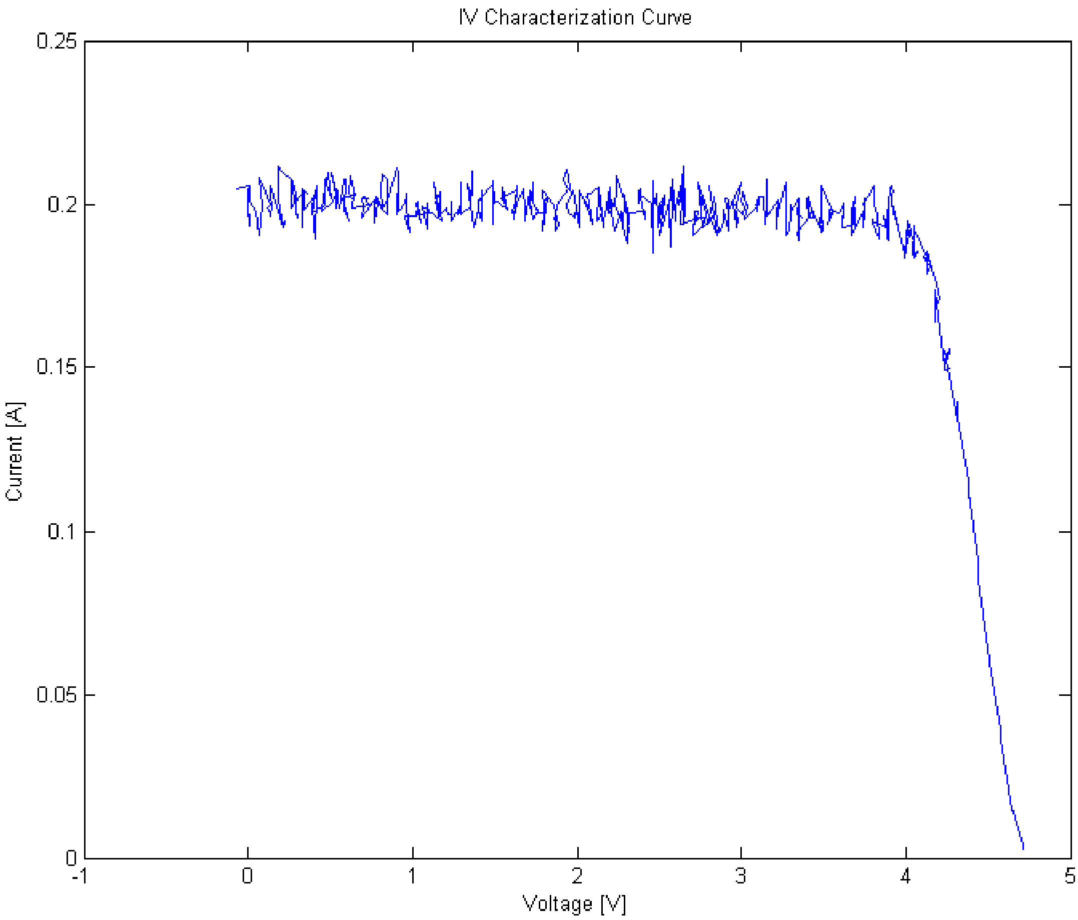


Figure 5.2 – *IV Characterization Curve Test for 1U Solar Panel.*

7. The system is able to plot the performing test in real-time (5.3)(5.4), displaying the measured current, the status of tool as well as an animated bar showing the performance status of the test.

As for the client's requirement related with the TVAC System established in Section 2, the following results are drawn:

1. In addition to the PV System and GUI design, the client established the design and manufacturing of a TVAC Chamber for performing thermal and vacuum tests. The final design was shown in 4.1. This system includes the following elements:

   - Two Gates Configuration to be able to observe the inside during tests, as required by the client in the requirements 6 and 7:

     (a) Methacrylathe Door: 40 mm in thickness provides a perfect visualization. It is perfect for Rough Vacuum and Medium Vacuum conditions.

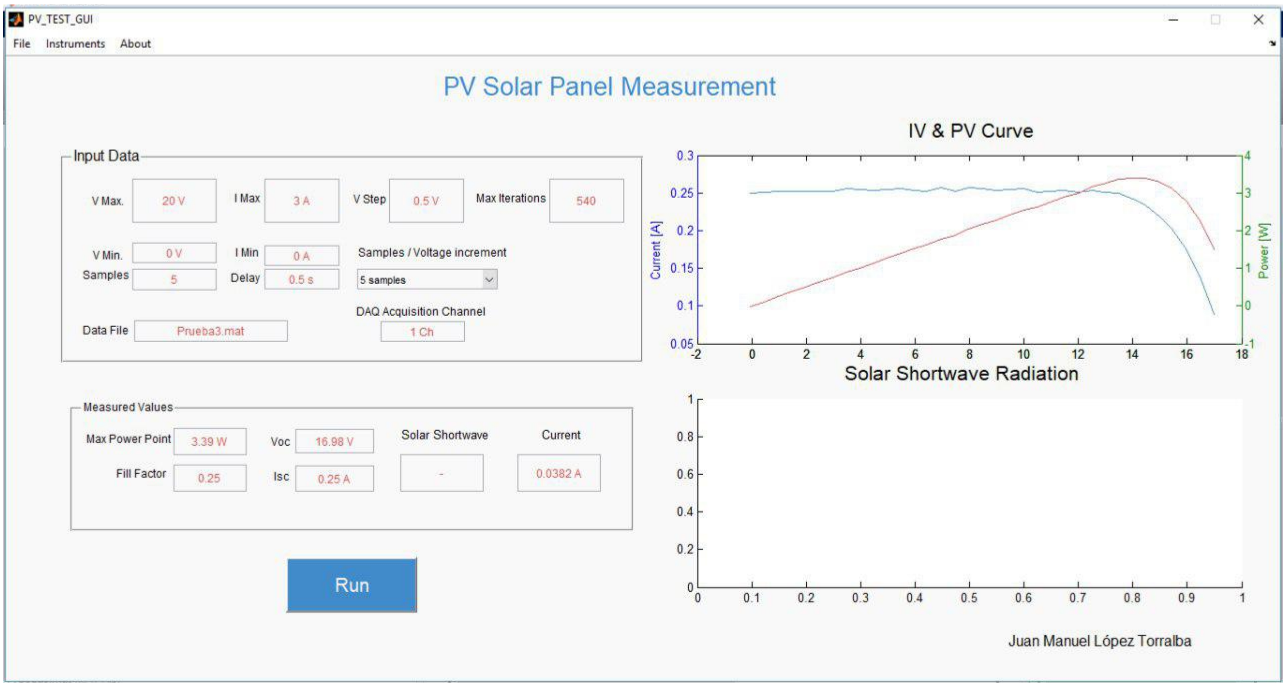


Figure 5.3 – *IV Characterization Curve Test.*

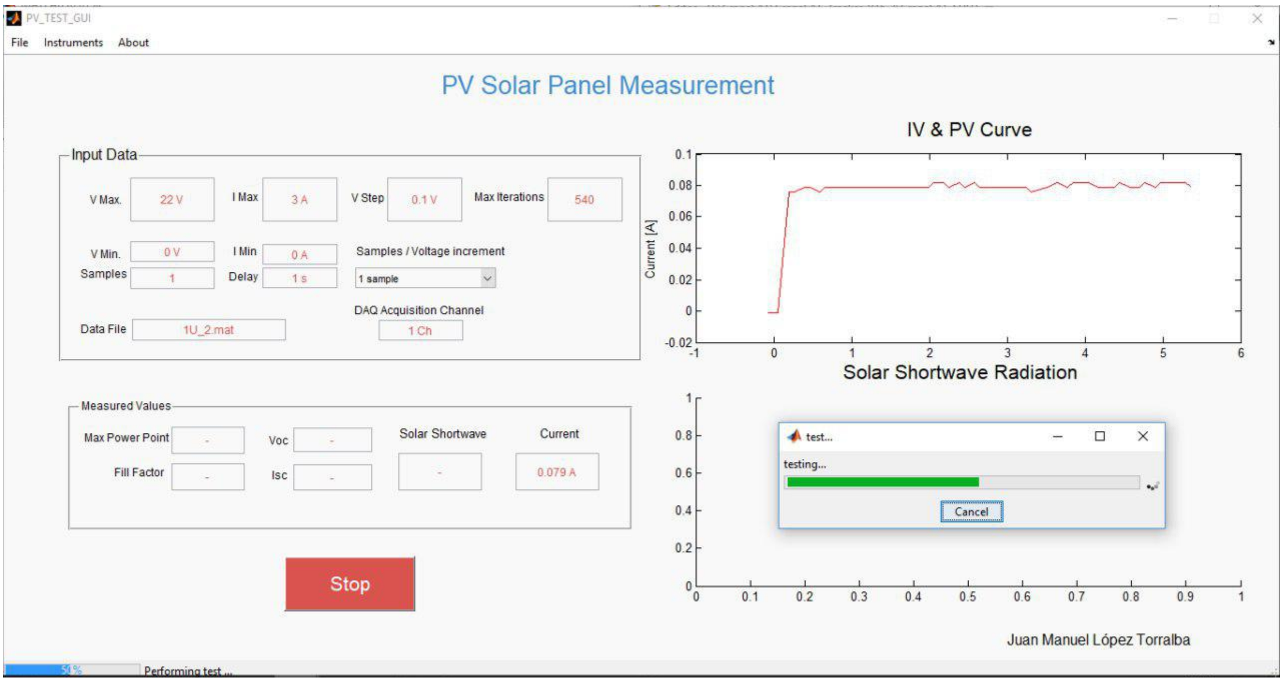


Figure 5.4 – *GUI performing Real-Time.*

(b) SS Door: It contains a 90 mm diameter view-port It will be used for more demanding test.

- Vacuum System: It would be able to decrease the pressure from ambient up to Medium Vacuum and High Vacuum using a rotary pump or adding turbo-molecular pump, respectively. The vacuum condition must be maintained during the test duration. Thus, a 24 hours-length vacuum test has been performed. In this test the TVAC Chamber has reached Medium Vacuum conditions by using the Telstar rotary pump directly. In Figure 5.5, the TVAC Chamber reaching Medium Vacuum conditions after 1 hour is shown. The results have been obtained by using the *MKS 925 Vacuum Transducer* controlled via MATLAB.

- Thermal System. Implements Cooling and Heating stages as stated in Section 4.1.3 which temperature could range from $-180°$, by using LN_2 injection, to to 200°, which is the upper temperature range of the *Isotemp 6200 R35*.
- Instrumentation Feedthrough ports: Up to 17 feed-through ports and the *SDM3065X Digital Multimeter* are able to measure temperatures, currents and voltage from the DUT.

2. As specified in Section 2.2 and shown in B, the total cost of the project is 1292 € below the approved budget of 10000 €.

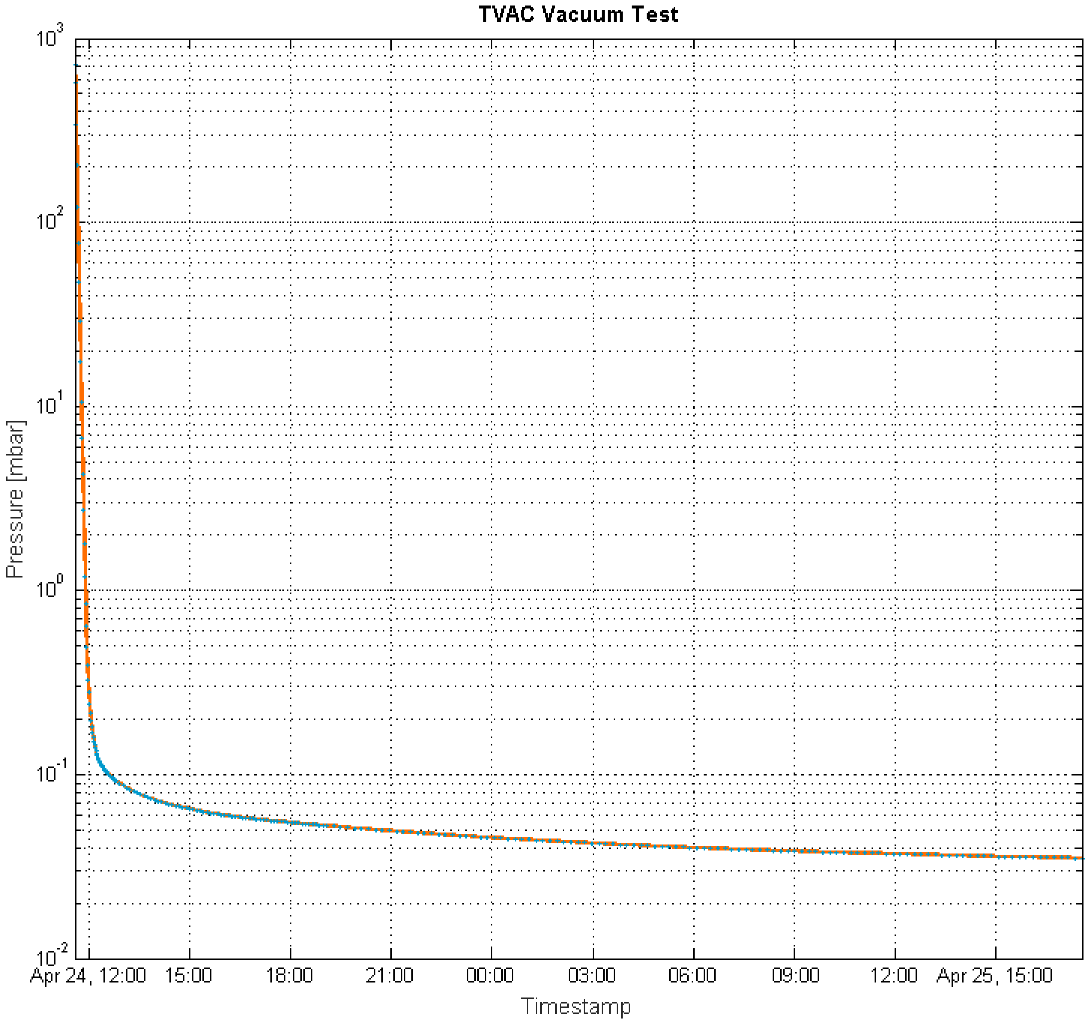


Figure 5.5 – *TVAC Chamber Performing Medium Vacuum Test without including Turbo-molecular pump.*

CHAPTER

6

CONCLUSIONS AND FUTURE LINES

Throughout the development of this Master's thesis, the analysis, design and implementation of a characterization equipment for aerospace photovoltaic panels have been carried out. This system is expected to be an important tool for the GranaSAT Group's development and growth. One purpose of the project was to make an approach to the problem from a business point of view, getting closer to standard procedures for the analysis and design phases. Furthermore, for the manufacturing and design of the TVAC Chamber, the daily communication and numerous discussions with the technicians of *Mecanizados Granada S.L.* have proved to be of great help and a compelling learning source. In addition, this project has allowed the student to acquire a deep knowledge in software tools not previously handled such as Solidworks, for 3D design, and LaTeX, a system for document preparation.

This project has been a great challenge for the student, which has tried to set out all the problems that have arisen along it, whose resolution, in some cases, have involved the application of concepts learned during the academic period at the University of Granada. Among the acquired capabilities, it is worth highlighting the application of knowledge in the electronics and programming areas.

Furthermore, this project has been conceived following a business approach that focuses on identifying the sated and hidden needs of a client. Consequently, the starting point has been the Client's requirement establishment. From that point, an engineering system approach has been used, that is, the extraction of the technical requirements from the analysis of the customer's requirement, the analysis of the system with a top-down approach, the material acquisition, the system design and, last but not least, the budget estimation.

The characterization equipment for aerospace photovoltaic panels presented is a complete and functional system, ready to exhaustively test solar panels in a laboratory environment. Nevertheless, some improvements and future lines of action have emerged during the development of this project and its documentation:

- It would have been desirable, the realization of a global validation test, involving the PV and TVAC System, in order to check the global behaviour of the whole system.
- Regarding to the TVAC System, several libraries for the control of the equipment have been implemented, and consequently, its use for the implementation of thermal and vacuum algorithms in the future is desirable.
- The future integration of the whole control algorithms for the TVAC in the MATLAB GUI is a relevant point to be taken into account. It would give an added value to the system in addition to avoiding the costs involved in the development of a new user interface.
- The implementation of several temperature and pressure test configurations in the improved GUI would provide the system with greater possibilities and capabilities for performing aerospace qualification tests in order to obtain space-worthy components.
- From a financial perspective, the renting of the PV System for the characterization of solar panels could be an additional source of income for the project. The TVAC System could also be rented but for a wider range of targets. Both systems, if integrated, could attract much interest from companies in the aerospace sector for testing its products before being officially tested by the ESA. Besides, the use of the TVAC Chamber in other reserach areas such as biomedicine, biology or biochemistry might be promising.
- Moreover, if both systems are integrated, the resulting one could be an exceptional tool for introducing Telecommunication students in vacuum technology and improved their skills in aerospace measuring procedures.
- Regarding to the data management, it could be captivating the creation of a cloud environment in which the data and parameters obtained from tests would be automatically updated to own servers. It could improve the document control and the competitiveness of the GranaSAT Group.

In summary, the author of this project is satisfied with the work accomplished. It should be noted the remarkable amount of knowledge and abilities acquired throughout the completion of the project.

REFERENCES

[1] Pyranometers. https://www.apogeeinstruments.com/pyranometer/.

[2] Pyranometers. need to know. https://www.campbellsci.es/blog/pyranometers-need-to-know.

[3] Solar irradiance spectrum above atmosphere and at surface. CC BY-SA 3.0. http://commons.wikimedia.org/wiki/File:Solar_spectrum_ita.svg.

[4] Reliability of cubesats – statistical data, developers' beliefs and the way forward. *Institute of Astronautics* (2016). Available at: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=3397&context=smallsat.

[5] Design of the electrical power system for the estcube-1 satellite. *Latvian Journal of Physics and Technical Sciences* (July, 2012). Available at: https://www.researchgate.net/publication/258680104_Design_of_the_Electrical_Power_System_for_the_ESTCube-1_Satellite.

[6] 2000, S. *Isotemp Circulator/Baths User's Manual*.

[7] Arthurcala. Solar spectrum and spectral response of pyranometers. CC BY-SA 4.0. https://commons.wikimedia.org/w/index.php?curid=51251652.

[8] Bailey, S., and Raffaelle, R. Space solar cells and arrays, 03 2011.

[9] Boddington, G. Electronic load. *Elektor Elektronics Magazine* (June 1990).

[10] Chawla, M. K. A step by step guide to selecting the "right" solar simulator for your solar cell testing application.

[11] Commission, I. E. Iec 60904-2:2007. Photovoltaic devices - Part 2: Requirements for reference solar devices. https://webstore.iec.ch/publication/18329&preview=1.

[12] et al, R. Development of a thermal-vacuum chamber for testing in small satellites. International Conference on Environmental Systems. https://ttu-ir.tdl.org/ttu-ir/bitstream/handle/2346/73024/ICES_2017_228.pdf?sequence=1&isAllowed=y.

[13] et al, T. S. Concentrating solar power: Best practices handbook for the collection and use of solar resource data. NREL Technical Report, 2010. https://www.nrel.gov/docs/fy10osti/47465.pdf.

[14] for Standardization, I. O. Solar energy.specification and classification of instruments for measuring hemispherical solar and direct solar radiation. Standard, 1990-10. https://www.iso.org/standard/16629.html.

[15] Galiaev, V. L. Large space chamber environments simulation for solar simulator components study. *Environmental Testing for Space Programms, Proceedings of the Third International Symposium held 24-27 June, 1997 at ESTEC, Noordwijk, the Netherlands. Edited by T.-D. Guyenne. ESA SP-408. Paris: European Space Agency, 1997, p.253*.

[16] Gueymard, C., Myers, D., and Emery, K. Proposed reference irradiance spectra for solar energy systems testing. *Solar Energy 73*, 6 (2002), 443 – 467.

[17] Harrison, K. Engineering a better vacuum chamber. GNB Corporation.

[18] Instruments, M. *925 MicroPirani Vacuum pressure Transducer Operation and Installation Manual*, 2009. https://www.mksinst.com/Docs/R/925man-revh.pdf.

[19] Instruments, O. Image. http://www.omniinstruments.co.uk/smp6-smart-pyranometer.html.

[20] International, A. Astm e927-10(2015), standard specification for solar simulation for photovoltaic testing. West Conshohocken, PA, 2015. https://www.astm.org.

[21] International, V. A. High and ultra-high vacuum.

[22] Jayaram, S., and Gonzalez, E. Design and construction of a low-cost economical thermal vacuum chamber for spacecraft environmental testing. *Journal of Engineering, Design and Technology 9*, 1 (mar 2011), 47–62.

[23] Jeon, J., Lee, S., Yoon, S., Seon, J., Jin, H., Lee, D., and Lin, R. P. Construction of a thermal vacuum chamber for environment test of triple CubeSat mission TRIO-CINEMA. *Journal of Astronomy and Space Science 30*, 4 (2013), 335–344.

[24] Juan Manuel López Torralba, A. R. A. Low cost tvac chamber for aerospace test. TAEE 2018, XII Congreso de Tecnología, Aprendizaje y Enseñanza de la Electrónica, Tenerife.

[25] Keithley. *KUSB-3116 User's Manual*, September 2010.

[26] Lin, M. Caracterització i implementació d'un simulador solar. Msc thesis, Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels, Castelldefels, June 2012.

[27] Marquart, M. Electronic load for testing power supplies. *Elektor Elektronics Magazine* (November 1993).

[28] O'Brien, T. P. Seaes-geo: A spacecraft environmental anomalies expert system for geosynchronous orbit. *Space Weather 7*, 9 (Sept 2009), 1–14.

[29] Patel, M. R. *Spacecraft Power Systems*. CRC Press, 2005.

[30] SIGLENT TECHNOLOGIES CO., L. *SDM3065X Digital Multimeter Remote Manual*, 2017.

[31] Solar, C. What is a pyranometer?

[32] Standard, J. I. Jis c 8912:1998: Solar simulators for crystalline solar cells and modules.

[33] Systems, T.-S. Solar simulator classifications.

[34] Technologies., A. Ieee 488.2 gpib general purpose interface bus.

[35] Technologies, K. I-v curve characterization in high-power solar cells and modules. Application Note. http://literature.cdn.keysight.com/litweb/pdf/5990-4854EN.pdf.

[36] Technologies, K. Key features specifications.

[37] University, C. S. P. *CubeSat Design Specification (CDS) REV 13*, 2014-07-07. Available at http://cubesat.calpoly.edu/images/developers/cds_rev13_final.pdf.

[38] Wang, W. Simulate a 'sun' for solar research: A literature review of solar simulator terchnology. KTH.

[39] Würfel, P. *Physics of Solar Cells: From Basic Principles to Advanced Concepts, 2nd Edition*. Wiley, Mar 2009.

ADDENDUM

A

PROJECT BUDGET

A.1 Hardware Cost

In this section, the project cost regarding to the Photovoltaic and TVAC Hardware Subsystem are detailed.

In the tables A.1 and A.2, both the cost associated with the PV and TVAC System are detailed.

Element	**Cost (€)**
Pyranometer	275.00 €
Electronic Load	1100.00 €
Power Supply	0.00 €
Xenon Solar Simulator	0.00 €
Solar Panel	0.00 €
TOTAL	**1375.00 €**

Table A.1 – *Photovoltaic System Cost.*

Element	Cost (€)
MKS 925 MicroPirani	680.00 €
Telstar Rotary Vane Pump	0.00 €
Isotemp 6200 R35	3295.00 €
Kodial View-port	253.00 €
Instrumentation Equipment	105.00 €
Chamber Manufacturing	3000.00 €
TOTAL	**7333.00 €**

Table A.2 – *TVAC System Cost.*

A.2 Software

In this chapter the cost associated with the sofftware licenses and packages are detailed in Table A.3.

Software	License Owner	Cost (€)
Solidworks 2017	GranaSAT Team	Free License
Matlab R2013a	MATLAB	Free License
TeXnicCenter	Juan Manuel López Torralba	Free License
Miktex	Juan Manuel López Torralba	Free License
SumatraPDF	Juan Manuel López Torralba	Free License
PencilProject	Juan Manuel López Torralba	Free License
	TOTAL	0 €

Table A.3 – *Software Costs.*

A.3 Human Resources

The realization of this project has been possible due to the commitment of two people. These people have generated some expenses related to its production. The first one, the Project Supervisor, is a Senior Engineer (50 €/h) computing about 8 hours per week. The second one is a Junior (20 €/h) Telecommunication Engineer accomplishing its Master Thesis. Then, the human resources expenses rise up to 32400 €, as shown in Table A.4.

Cargo	Horas	Coste(€)
Junior Engineer	1440	28800.00 €
Senior Engineer	360	18000.00 €
TOTAL		**46800.00** €

Table A.4 – *Human Resources Costs.*

ADDENDUM

B

TVAC CHAMBER PICTURE DOCUMENTATION

In this appendix, some photos related to the manufacturing process of the TVAC Chamber, carried out at the *Mecanizados Granada S.L.* facilities, are shown. In Figure B.1, the TVAC Chamber is shown with the SS door configuration. In Figure B.2, the chamber is open and the internal chamber is can be observed.

In the figures B.4 and B.3 both sides of the chamber are shown. In the right side of the chamber, a total of five KF-25 (above) and four KF-16 (below) feed-through ports are presented, in which three of them are thermocouples. In the left side, at the bottom of the image B.3,a KF-50 Dual LN_2 can be observed.

Finally, the TVAC Chamber rear view with the Telstar Rotary Pump is exhibited in Figure B.5.

Figure B.1 – *Closed TVAC Chamber with Stainless Steel Door. Front View.*

Figure B.2 – *Opened TVAC Chamber with Stainless Steel Door. Front View.*

Figure B.3 – *TVAC Chamber with Stainless Steel Door. Right Side.*

Figure B.4 – *TVAC Chamber with Stainless Steel Door. Left Side.*

Figure B.5 – *TVAC Chamber Rear View.*