GLASGOW CALEDONIAN UNIVERSITY

MEng Group Research Project

MMH723842-24-AB-GLAS

Design and Implementation of a Photodiode Array-Based Analogue 2D Sun Sensor

word count: xxx

by Zac McCaffery, Alexandru Belea,
Sebastian Alexander, William Kong, Nassor Salim,

Date: April 16, 2025

Contents

List of Figures

List of Tables

List of Equations

ADC Analog to Digital Converter

ADCS Attitude determination and control system

ATP Acquisiton, Tracking and Pointing

CCD Charge-Coupled Device

CSV Comma Separated Values

CTE Coefficient of thermal expansion

DAQ Data Acquisition System

FOV Filed of View

GSFC Goddard Space Flight Centre

LEO Low Earth Orbit

OPS Optical Position Sensor

PSD Position Sensitive Detector

PV Photovoltaic

RED Renewable Energy Demonstrator

SGP4 Simplified General Perturbations 4

SMPS Switch-Mode Power Supply

TIA Transimpedance Amplifier

TLE Two-line element set

VLC Visible Light Communication

WSN Wireless Sensor Networks

SMD Surface Mount Device

OpAmp Operational Amplifier

PCB Printed Circuit Board

PLA Polyactic Acid

LED Light Emiting Diode

EDA Electronic Design Automation

LUT Look-Up Table

Abstract

add abstract here

1. Acknowledgements

We would like to express our sincere gratitude to our supervisors, Dr. Roberto Ramirez-Iniguez and Geraint Bevan, for their invaluable guidance and unwavering support throughout this project.

Our appreciation extends to the 3rd Floor EEE Lab Technicians and Dr. Carlos Gamio-Roffe, whose technical expertise and assistance were instrumental in the successful construction of the prototype.

We are particularly grateful to the European Project Semester RED Team members—Nikolay Ivanov Shopov, Stef Hannisse, and Samridhi Gupta—for generously allowing the use of their Renewable Energy Demonstrator as a testbench. Their contribution provided an ideal platform for positioning light sources during the testing phase of this project.

2. Introduction

2.1. Problem Statement

With the ever-increasing commercialization of the space and satellite industry there is a growing need for a cost-effective method of attitude tracking for smaller satellite missions of such as CubeSat as these missions are purpose built for very specific objectives. Whilst the larger commercial satellite missions make use of expensive digital camera systems for tracking purposes, this is not feasible for much smaller CubeSat setups. CubeSats are defined from 1 unit to 12 – where 1U is a 10x10x10 cm satellite. Consequently, there is a demand for a cost-effective and easily implementable attitude tracking system that can provide accurate measurements for CubeSat missions, such as a Position Sensitive Detector (PSD) using photodiodes.

2.2. Aim of the Project

"To investigate and develop a cost-effective and reliable sun sensing solution suitable for Low Earth Orbit (LEO) nanosatellite attitude determination."

2.3. Objectives of the Project

To investigate the design of a sun sensing system for nanosatellites, used in orientation determination, through detection of its relative position to the sun using analogue sensors located on the satellite's body. Our goal is to create a system which balances cost-effectiveness and simplicity. To achieve this, we will create a software model of the analogue sensor(s) to simulate the system's ability to track the sun from various angles in orbit. After which, we aim to build a physical prototype and use a movable light source to simulate the sun's movement, allowing comparison between the real sensor's performance against our simulations. Although the physical prototype will be built using non-space-grade materials, one of the objectives is to look at and analyse materials required for building a space-grade PCB and sensor. For this step, the Mechanical side of the team will perform Printed circuit board (PCB) and aperture device finite analysis using ANSYS to determine resilience to environmental factors such as stress and thermal simulation. The application of signal processing will be explored to provide usable data, filter out

noise, and improve the system's accuracy. This approach aims to develop a cost-effective and reliable, in-house sun sensing solution specifically for nanosatellites operating in Low Earth Orbit. Major Objective points:

- **Conduct literature review:**
  - Analyse existing research on sun sensing technologies, with a focus on PSD-based analogue sensors and their applications in nanosatellites.
  - Identify current challenges, best practices, and advancements in attitude determination in Low Earth Orbit. Use these insights to guide the design and optimisation of the proposed sun sensing system.
- **Develop software model:**
  - Simulate the performance of the PSD-based analogue sun sensor in tracking the sun's position from various angles in Low Earth Orbit.
- **Design and fabrication of physical prototype:**
  - Integrate analogue sun sensor components, test and validate its performance under controlled conditions.
- **Compare simulated and experimental results:**
  - Establish evaluation methodology between simulated and experimental test results to ensure that topology evaluation is applicable.
- **Optimise sensor topology:**
  - Research and evaluate various configurations of analogue sun sensing systems to maximise sun detection accuracy and minimise blind spots.
- **Investigate environmental factors:**
  - Evaluate the material requirements of the PCB and aperture device.
- **Implement signal processing algorithms:**
  - Investigate the filtering of noise to enhance the signal-to-noise ratio and otherwise ensure the acquisition of usable data for accurate sun position determination.
  - Implement data handling which optimises scanning rates and efficiently processes the analogue signal data for real-time attitude determination.
- **Document results and overall cost-effectiveness:**
  - Develop criteria for final evaluation of sun sensing systems, on which to base the final presentation of project findings.

3. Literature Review

3.1. CubeSat Design

Puig-Suari, Turner and Ahlgren published an IEEE paper in 2001 with the help of their students at California Polytechnic State University exploring a need for micro satellites for use by universities in an ever-expanding space programme. They provide as a solution a standard satellite form-factor that will bring down the cost of both manufacture and deployment of satellites by smaller entities: the CubeSat. The paper identifies a key component for the success of this form factor a need for a standard CubeSat deployer mechanism which can deploy several satellites safely and develop such a platform, called Poly Picosatellite Orbital Deployer or P-POD. They point out the need and provide microsatellite size and shape of the CubeSat form factor [7].

Sai balaji et al. performed a study using MATLAB simulation of several attitude control algorithms to look at the ability to control a CubeSat of size 1U. They also simulated sensors such as sun sensors, magnetometer, and gyroscope. They concluded that it is possible to operate the satellite using a magnetorquer type actuator and an array of mathematical models and algorithms: it would take 2000 seconds for a 1U satellite to stabilize at 505km, 98° degree attitude in orbit with the methods utilized by them [8].

Incentivised by the rapidly increasing use of LEO, Lopez-Calle and Franco perform a quantitative comparative study on the catastrophic failure of CubeSats and Nanosats from radiation exposure due to the harsh environment of space versus failure due to collisions in the increasingly busy Low Earth Orbit (LEO). The authors concluded that while sustained damage and damage protection from radiation exposure used to be and currently still is the most crucial factor in protecting LEO microsatellites, increasingly the risk of debris collisions is becoming more important and will become the most important in the following 50 to 70 years. The authors conclude that microsatellite designers need to move their focus more towards defence from debris impacts as these, even if not resulting in catastrophic failure of the satellite, they will impact the attitude of the satellite [9].

3.2. PSD Enabled Sun Sensor

The need for position estimation based on light sources preceded current requirements in microsatellites. Qian, Wang, Busch-Vishniac and Buckman describe in 1993 a position

sensitive detector (PSD) method using a single two-dimensional lateral-effect PSD capable of keeping track of multiple light sources at the same time. Their method of tracking multiple light sources involves modulating each light source, LEDs, of interest to a different frequency. They succeeded in correctly tracking two light sources modulated at 10kHz and the other at 5kHz. They point out that the light sources can correctly be tracked even in the presence of background light and that several light sources can be tracked, with the only restrictions being the bandwidth of the PSD and the sampling time of the sample and hold device being used. Although their method of tracking several lights might be out of scope of a sun sensor, the methods of design may be of some use to the design of a PSD.

Similarly, Guanghui and his colleagues developed the AirLink-E100 system which makes use of a PSD and describe it in a 2007 paper. This system is not directly related to sun sensing; however, their finding may be of some use. In a position sensor on earth, they point out, due to background light the PSD does not work with the precision necessary. They introduce methods of achieving better precision using analogue and digital signal processing. They modulate the light to be sensed, in this instance a laser, with a square wave, in essence turning the laser on and off. This allows for sensing of the background noise (when the laser is off) and subtracting the noise from the next time period when the laser is on. The authors conclude that this is a sound method of filtering out background noise in PSD devices where the light source can be modulated[10].

Building on this and other work, Ortega, Lopez-Rodriguez, Ricart et al. describe in a 2010 paper a miniaturized two axis sensor with a $\pm 60^\circ$ FOV, totalling 120°, and angle accuracy better than 0.15°. Their method is directly aimed at sun sensing. They not only design, fabricate and characterize the sensor, but also successfully integrate it in a Spanish nano-satellite NANOSAT-1B. The team used monolithically integrated silicon photodiodes in a crystalline silicon substrate protected by a glass cover. The entire size of the sensor is 3cm $\times$ 3cm and weighs in at just 24 grams. The NANOSAT-1B which launched in 2009 contained three of these sensors[11].

Ortega et al.'s research directly aligns with the paper's objectives, rendering their work highly relevant, and was replicated by Dwik and Somasundaram's research modelling and simulating a PSD using MATLAB[12]. The authors point out the superiority of PSD over Charge-Coupled Devices (CCDs) because of the higher resolution and rapid response time. The researchers modelled two-dimensional photodiode arrays providing four output currents using included photodiode element in MATLAB. They also modelled simplified versions with single diode and one-dimensional array. They conclude that a PSD using photodiodes is a viable method of detecting the location of a light source from the current change readings of the diodes. They point out that for use in a fully working Acquisition, Tracking and Pointing (APT) system, further work is necessary that is not covered in their paper, such as signal conditioning.

Furthering the previous work, Delgado et al. showed in a 2013 paper the design, fabrication and characterization of a solar sensor that takes advantage of subdivision of the field of view (FOV) into 4 quadrants. The research team was able to develop high-precision sensors with an FOV of $\pm 60^{\circ}$ and fine resolution of 0.05° while providing a coarse resolution of 0.5°. This high precision is achieved by splitting the sensor into four sub sensors, each concentrating on a different range of angles. They named the technology Sensosol and the paper states it will be used onboard the SeoSat satellite from Inta corporation[13].

3.3. Mechanical Design and Analysis

Although the CubeSat specifications are strict in reference to size and shape of nanosatellites that aim to respect these specifications, it allows the freedom for designers to choose many characteristics. Therefore, mechanical analysis is required, with a focus on thermal and structural characteristics. To this end, Ullah, Rehman, Bari and Reyneri published a paper in 2017 raising the struggle of heat dissipation in CubeSats due to their small size not allowing the installation of heat-dissipating radiators. The team considered all thermal resistances of the CubeSat panels either as separate layers or similar materials combined in their simulation. They conducted both simulation and real measurements of the AraMiS-C1 satellite developed by the Torino Polytechnic and found that the simulated model correctly aligned with real world measurements. They conclude that their model can therefore be used to model any microsatellite following the CubeSat standard. They also concluded that the thermal resistance measured was exceptionally low and therefore could be safely used on satellites to be deployed[14].

Similarly, Raslan, Michna and Ciarcia performed a thermal simulation of a CubeSat in a 2019 paper. Their goal was to discover the required framework to maintain the thermal stability of a CubeSat in orbit, with the overarching goal of creating a CubeSat that will contain a mammalian tissue sample for gathering experimental data of the effects of microgravity and space radiation on the sample. The CubeSat, therefore, must be able to maintain a very narrow range of temperatures without large fluctuations so that the experiment remains valid. They aim to find external coating materials and attitude control that limit these fluctuations. The mission will involve a 6U sized CubeSat containing a biohousing. With a black-chrome plated metal coating in combination with solar panels on the sun exposed side the team was able to maintain 37 degrees with only 2 degrees deviation in the biohousing. The other sides of the satellite all were covered in solar panels in the simulation. They conclude that they have identified a suitable single-coat material that in combination with PID attitude control algorithms is able to maintain the temperature within an admissible range. They also conclude that this can be done for a wide range of orbits and exposure time. They point out future research will focus on finding

other coat types that maintain temperature without attitude control as changing attitude to maintain temperature consumes satellite power, which is a valuable resource[15].

Concentrating on the structural resistance of the CubeSats, Dhariwal, Singh and Kushwaha performed a structural analysis using ANSYS software and released their findings in a 2023 paper analysing the structural behaviour of 1U CubeSats under various loads, static, modal random vibration, and shock loads. The team claims their paper establishes a methodological framework for CubeSat structural analysis and can be used for future work. They conclude that because the stress applied to the aluminium alloy 6061 used did not go above the yield strength, it is safe to assume the material operated safely and can be used on the CubeSats structure. They also conclude that their analysis was accurate and point out a need for a physical test on a real 1U CubeSat structure[16].

This review of carefully selected research papers serves as a robust foundation for the work that is to be conducted in this project. The critical insights and methodology described for thermal and structural analysis of CubeSats by the teams' previous work, as well as CubeSat designs and PSD sensor work conducted as mentioned above, is important and helpful for the project's successful contribution to this research.

3.4. Photodiode Simulation and Signal Analysis

A paper by Fuada et al. released in 2017 investigates the received power characteristics of commercially available photodiodes used as receivers in Visible Light Communication (VLC) systems with a line-of-sight (LOS) channel. The authors used MATLAB simulation to analyse how various parameters affect the received power, including:

- Transmitter semi-angle (half power)
- Distance between transmitter and receiver
- Room size
- Receiver Filed of View (FOV)
- Optical filter gain
- Lens refractive index

They also point out 6 key considerations when choosing a photodiode for VLC applications:

1. Surface area: A larger surface area (e.g. 10mm x 10mm) can support mobility in the VLC system, but this needs to be balanced against the impact on cut-off frequency and susceptibility to ambient light noise.

2. Generated short current: the photodiode should generate sufficient current (>100 μA) when exposed to light, as this affects the required gain and bandwidth of the amplifier circuit.

3. Wavelength detection capabilities: The photodiode needs to be sensitive to the visible light spectrum (380nm to 780nm) for VLC applications.

4. Cut-off frequency: A high cut-off frequency (in the GHz range) supports high-speed data transfer, but this often requires sacrificing a larger surface area.

5. Rise time: Fast rise time (in the nanosecond range) is also desirable for high-speed VLC, but again this trades off with surface area.

6. Dark current and junction capacitance: The photodiode should have low dark current and low junction capacitance to minimize noise and maximize response time.

The paper notes that it is difficult to find a single commercially available photodiode that optimizes all 6 of these factors simultaneously. This often requires making trade-offs or using custom photodiode designs for the specific VLC application. The results show that factors like distance, room size, FOV, and LED power have a linear relationship with the received power at the photodiode. Additionally, the optical filter gain and lens index play an important role in determining the received power characteristics. The authors note that this study was limited to the LoS channel and does not take into consideration indirect illumination[17].

Nathanael A. Fortune writes a paper in 2021 meant to help scientists with common signal processing tasks when handling experimental data. The paper provides examples of using the Numerical Python (Numpy) and Scientific Python (SciPy) packages, as well as interactive Jupyter Notebooks, to accomplish tasks such as interpolation, smoothing, propagation of uncertainty, curve fitting, plotting functions and data, and determining the goodness of fit. The goal is to enable an interactive, exploratory approach to data analysis while ensuring the original data is freely available and the resulting analysis is readily reproducible. The paper includes sample Jupyter notebooks containing the Python code used to carry out these tasks, which can be used as templates for analysing new data[18]. This paper should prove useful in the numerical simulation of the PSD sensor.

3.5. IoT Communication Enhancement with LEO Satellites

The use of LEO satellites, including microsatellites, can be extended to the current expansion of Internet of Things (IoT) devices. To this end, Koukis and Tsaoussidis investigated

the use of satellites for IoT sensors and devices. They created simulations using OMNeT++ software and real sensor data from Smart Santander testbed. They concluded that their initial hypothesis was correct, and an increase of LEO satellites is inversely proportional to ping loss and round-trip time of packets sent between a LEO constellation and IoT equipment on the ground. They also concluded that the placement of ground stations led to improved communication even with a lower number of satellites. They close by saying they did notice instances where signal was deteriorated, and that further work is necessary[19].

4. Methodology

4.1. Prototype Development

4.1.1. Lifecycle

This section provides an overview of the Prototype Development Lifecycle.

Conceptualization and Requirements Definition

- The prototype must have four photodiodes in an xy pattern with respective circuitry required to output 0-5 Volts that will be read by an Arduino based Data Acquisition System (DAQ). The circuit must be able to react to light intensity changes, however the change will be at low frequency (below 1Hz) as a satellite attitude is considered to change only gradually.
- While the prototype may not have a high accuracy, it is hoped that it will be enough to measure light position changes roughly, even if at a low accuracy of 10° or 20° but this will remain to be seen.
- The prototype within the scope of this paper will show the ability to detect the position of light at normal room conditions, therefore it does not need to withstand temperature changes or radiation that a final product would require if deployed in space.
- Interface requirements: the prototype electrical output needs to be compatible with the Arduino Analog to Digital Converter (ADC) input. Therefore, the signal shall not go below 0 Volts or exeed 5 volts.
- Size and weight are not of high importance, but the device must fit in the testing equipment, which is the Renewable Energy Demonstrator arch. Preferably a height not higher than 5cm.

Theoretical Design

The prototype will be composed of three parts: a stripboard containing the components for the amplification circuit, a 3D printed Photodiode enclosure which will allow placing

the photodiodes in the correct positions, as the photodiodes have the legs at a smaller distance than the stripboard, and need to be placed quite close to eachother, with the third part being the Arduino based DAQ. The third part to the

Sun Sensor Geometry and Aperture Design was decided to be in an orthogonal, T shape, providing an x-y layout with two photodiodes in the x direction and two photodiodes in the y direction. Combined with an aperture design that covers one half of each diode, as represented in Figure 4.1. This configuration is similar to Ortega et al. in their paper attempting to miniaturize a two axis Sun Sensor[11].

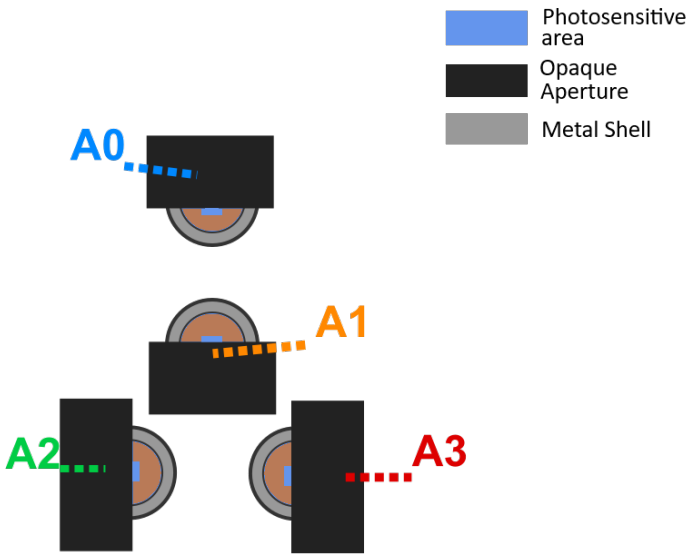


Figure 4.1.: Diagram of Photodiode Array with Apertures

Component Selection

The components chosen for the design of the prototype are detailed below. They were first placed on a BreadBoard and tests were performed to ensure the functionality of the circuit before moving them to a stripboard which would ensure the components are phisically more stable and reduce the chance of faulty connectoins. Therefore, the same components were used for both BreadBoard and StripBoard prototyping. The circuit design and testing is gone into more detail in Section 4.1.2.

Photodiodes were researched and several options were found, most of which were quite expensive, such as 2D PSD type sensors like the Hamamatsu S5990 but were both prohibitivly expensive and Surface Mount Device style (SMD), which would have been harder

Table 4.1.: Electrical Components Used in Amplification Circuit

Component Type	**Comp. Part Name**	**Component Value**	**Amount**
Op-Amp	TI LM324-N	–	2
Photodiode	Hamamatsu S5971	–	4
Resistor	–	1 MΩ	4
Resistor	–	150 kΩ	4
Resistor	–	10 kΩ	4
Capacitor	–	1 μF	4
Stripboard	Veroboard	–	1
Screw terminal block	–	2-input	3

Table 4.2.: Components Used for DAQ System

Component Type	**Component Part Name**	**Component Value**	**Amount**
Microcontroller	Arduino Uno	–	1
Cable	USB cable	–	1
Computer	Laptop/Python	–	1

to prototype but would allow for much higher resolutions, while also complicating the project due to the more complex nature of PSDs. A decision was made to base the project on 1D photodiodes, and four Hamamatsu S5971 were purchased, which offered a good compromise in price and specifications:

Table 4.3.: S5971 Photodiode Specifications

Parameter	**Value**
Spectral response range (λ)	320 to 1060 nm
Peak sensitivity wavelength (λ_p)	920 nm
Photosensitivity S (A/W) at λ_p	0.64
Photosensitivity S (A/W) at 780 nm	0.55
Photosensitivity S (A/W) at 830 nm	0.6
Short circuit current Isc	1.0 μA
Dark current Id (Typical)	0.07 nA[*3]
Dark current Id (Maximum)	1 nA[*3]
Cutoff frequency fc	0.1 GHz[*3]
Terminal capacitance Ct (f=1 MHz)	3 pF[*3]
Noise equivalent power NEP (V_R=10 V, λ=λ_p)	7.4×10^{-15} W/Hz$^{1/2}$

V_R = 10 V

Although the higher versions such as S5972-3 have better specifications, such as frequency cutoff of 1GHz and lower Dark current, these were not needed for our project, higher frequency cut off not needed due the static nature of the light source and dark current, while it could affect a case where one of the diodes is fully in the dark, a voltage offeset would be noticeable, but with a voltage resolution restricted by the Arduino DAQ

to 4.88mV/step, it was deemed acceptable:

$$\begin{aligned} V_{\text{offset-TIA}} &= I_d \times R_f \\ &= 0.07 \text{ nA} \times 1 \text{ M}\Omega \\ &= 70 \ \mu\text{V} \end{aligned} \tag{1}$$

$$\begin{aligned} V_{\text{offset-total}} &= V_{\text{offset-TIA}} \times \text{Gain}_{\text{post-amp}} \\ &= 70 \ \mu\text{V} \times 16 \\ &= 1.12 \text{ mV} \end{aligned} \tag{2}$$

Equation 2 above shows the final dark current would be a maximum of 1.12mV which is below what the ADC can detect.

Operational Amplifier (OpAmp) choice was once again not a complicated choice due to the same arguments mentioned above re. photodiode selection: low frequency signal and reduced DAQ resolution. The LM324-N is a low-cost OpAmp which provides acceptable performance. The advantages in choosing this OpAmp is its ability to function

Table 4.4.: Operational Amplifier Specifications

Parameter	**Value**
DC Voltage Gain	100 dB
Unity Gain Bandwidth	1 MHz
Supply Voltage Range (Single)	3 V to 32 V
Supply Voltage Range (Dual)	±1.5 V to ±16 V
Supply Current	700 μA
Input Bias Current	45 nA
Input Offset Voltage	2 mV
Input Offset Current	5 nA
Input Common-Mode Voltage Range	Includes Ground
Differential Input Voltage Range	Equal to Supply Voltage
Output Voltage Swing	0 V to V^+ - 1.5 V

Internally frequency compensated for unity gain
Temperature compensated

on a single pole Power Supply, it is rather cheap while still offering 1MHz Unity Gain Bandwidth and is recommended for DC Gain which the type of signal our project aims to amplify. For example, the slew-rate characteristic is not mentioned on the datasheet because it is aimed at low frequency operation.

BreadBoard Testing

Once the circuit design was completed as seen in Figure 4.6, the components were placed on a BreadBoard to test the real circuit, as seen in Figure 4.2. There were several

iterations, at first with only the Transimpedance Amplifier (TIA) and one photodiode- a design that when tested, resulted in a low Voltage output when testing with only the Light Emiting Diode (LED) light from the RED testbench. This is due to the low light power of LEDs. A decision was made to add a secondary amplification circuit which raised the Voltage to a maximum 5V as designed. Further details on design in Section 4.1.2. The final BreadBoard design can be found in Figure 4.2.

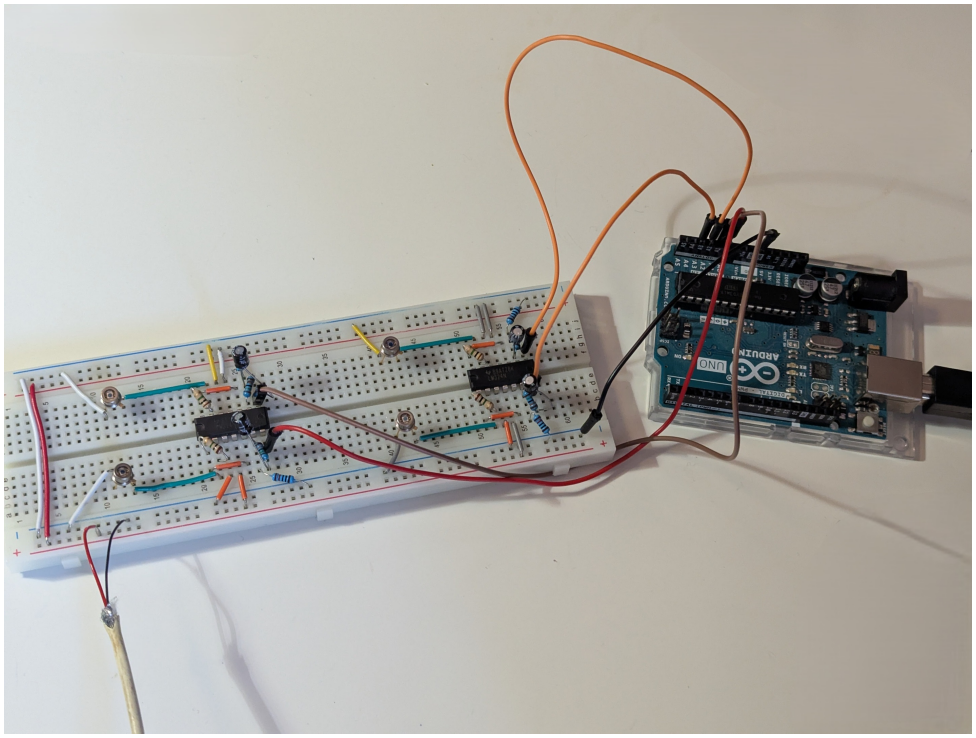


Figure 4.2.: Photo Of BreadBoard Prototype

Renewable Energy Demonstrator (RED)

The RED device borrowed from the EPS team [20] had to be modified to be used as our testbench providing illumination from desired angles in a repeatable fashion. Details on the modification can be found in 4.3. After modifying it to move the light in our desired location on the hemispheric plane both azimuth and elevation angles could be controlled, it was discovered that the servomotors in the RED were not always delivering the correct angles as requested in the code. It was discovered that "helping" the arch move towards the desired location, such as 90°from 0°by pushing with our hand, allowed the arch to move to the correct position. The reasons for this could be several, such as servomotors with insufficient torque, or damaged gear train. Due to time limitations it was decided not to further debug the device, instead a decision was made to use a protractor and manually "help" the arch when needed to reach the requred angle. A further issue with the RED device was that it would "randomly" change the location, which was only meant to be triggered by a button press. The reason for this was likely the same reason our recorded

signals had a lot of noise: interference from the RED power source. However this was rare enough that it did not impede our testing. A debouncing technique in software could likely have fixed this issue, if it was going to be fixed. Further, while overall the RED was very useful for our project, the servomotor issues causing the arch to not reach the correct angle easily and further affected by some control deadband - an inability of the servomotors to reliably make small changes in angle, meant that certain tests could not be performed, such as taking measurements at every 5°in order to record readings across the whole hemispere, which would allow creating a Look-Up Table (LUT) that would be used to correctly read the location of the light.

Design Refinement

Secondary Amplification was added to the circuit which was a major redesign for signal conditioning. This allows reading of the analog signal with the Arduino ADC,at the full range of 0V to 5V, as the Arduino ADC is 10 bit. This gives a resolution of $2^{10} = 1024$ steps, otherwise the readings would be from 0 V to about 300 mV, with a much lower resolution between steps - the DAQ would be using a resolution of only the region covered between 0V and 0.3V, ie. $1024/16 = 64$ steps. Details regarding this can be found in Section 4.1.2. Further, the Secondary Amplifier circuit was modified with the addition of a feedback resistor to reduce noise, as described in Section 4.1.2.

The Photodiodes were first placed in a square pattern around the circuit,which helped BreadBoard prototyping and would have also simplified the Stripboard design. However, it was quickly determined that the photodides must be as close to eachother as possible, due to the light in the RED testbench being closer to a "single point" spreading out, as opposed to the light coming from the sun, which is assumed to be parallel - in which case the distance relative between the photodiodes would not matter. More details about the lightsource on the RED testbench in Section 4.3 . This meant that soldering the photodides to the stripboard was not an option, due to the small distance between the photodiodes, the diodes having 3 pins (anode, cathode and case) at small distances in triangle patern and the relatively large distance between strips on the Veroboard stripboard (2mm). The decision was therefore made to move the photodiodes to their own 3D printed enclosure, which went through it's own redisign as described under Section 4.1.3. This meant soldering wires to the photodiode pins and ensuring they do not shortcircuit using heat shrink tubing.

The **Stripboard Prototype Development** was done in two stages to avoid making too many mistakes that would be too hard to fix: the design was first implemented using the Software Fritzing which is a free and Open Source Software suite for Electronic Design Automation (EDA) that allows planning on different BreadBoards and Stripboards. The

final CAD Stripboard design is reproduced in Figure 4.3. When soldering the components to the actual stripboard, besides some errors such as soldering a jack on the wrong stips, and having to de-solder and re-solder, the stripboard prototyping was quite straight forward due to having tested the design on the BreadBoard and creating the plan in Fritzing as mentioned. However, during testing, the readings seemed off and upon further debugging it was found that some flux remained on the stripboard, causing unwanted conductance between strips, creating a parallel resistance with one of the feedback resistors, which was causing the incorrect amplification. After cleaning the stripboard with support from Dr. Carlos, who had solvent in his office, the Stripboard prototype functioned as intended.

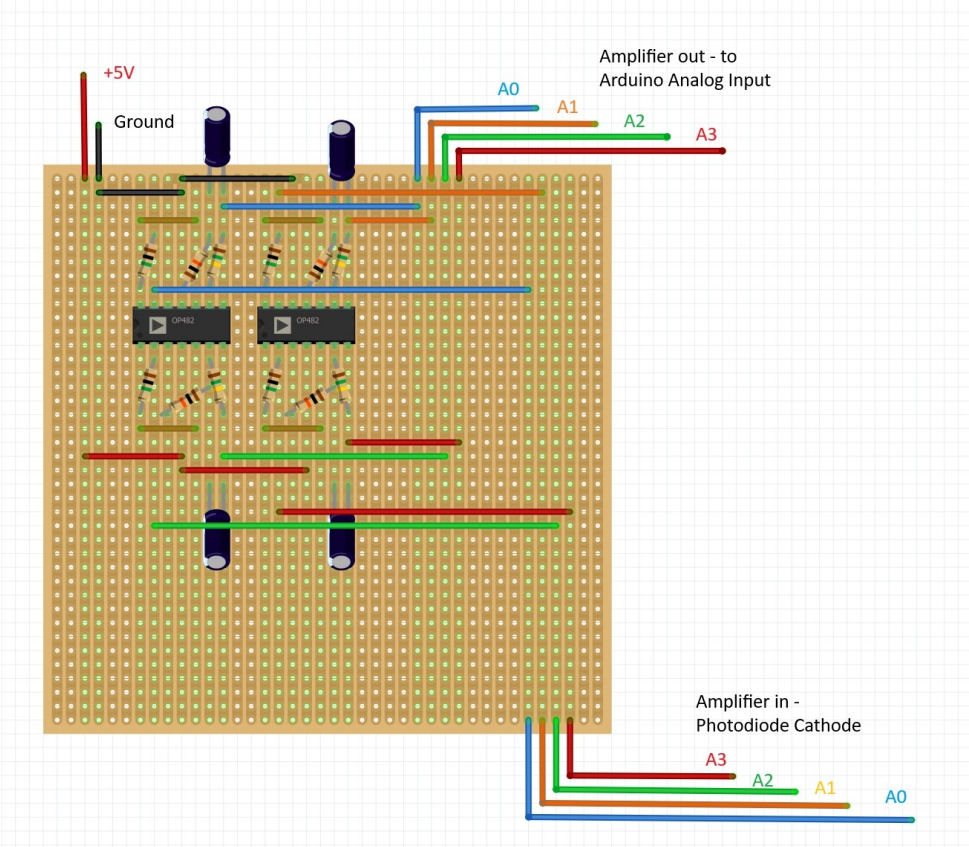


Figure 4.3.: Photo Of Stripboard Prototype on Fritzing

The **Aperture** on the 3D printed photodiode enclosure was harder to implement. Initially a 3D printed aperture was planned but this was not able to be produced. Another plan was to glue plexiglas to the enclosure and apply black electrical tape to it, an idea which was changed because the plexiglas available was considered too thick at 5mm. The worry was that light scattering in the plexiglas, refraction and especially lateral displacement due to the thickness, especially at a larger angle such as when the light is close to the horizon tangent point. The difference in displacement between zenith and when close to horizon was a worry. A solution was found by purchasing a screen protector glass for

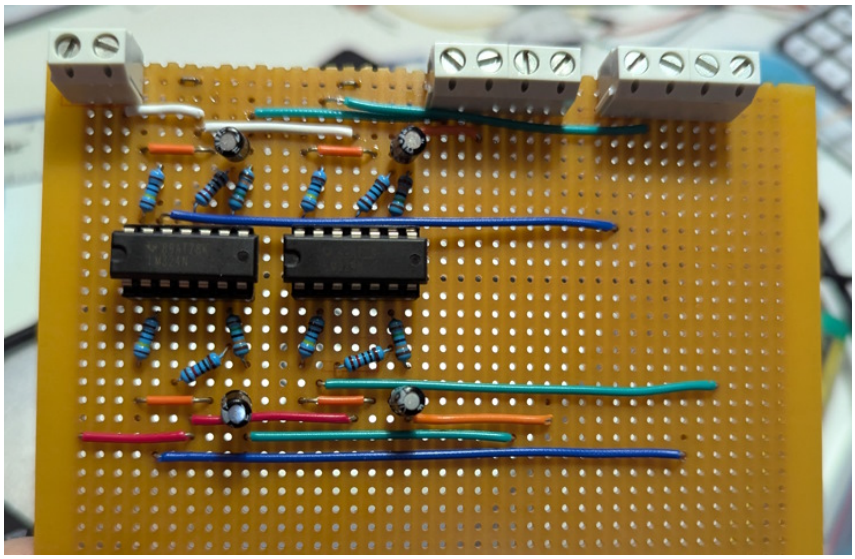


Figure 4.4.: Photo Of Stripboard Prototype

mobile phones. This glass was very thin, at under 1mm, and is fabricated to be very clear and transparent so as not to affect the quality of the mobile phone screen it is meant to protect. The black electrical tape was then placed on top of this glas as seen in Figure 4.5. As can be seen the apertures are not fully straight. This is because they were placed by hand.

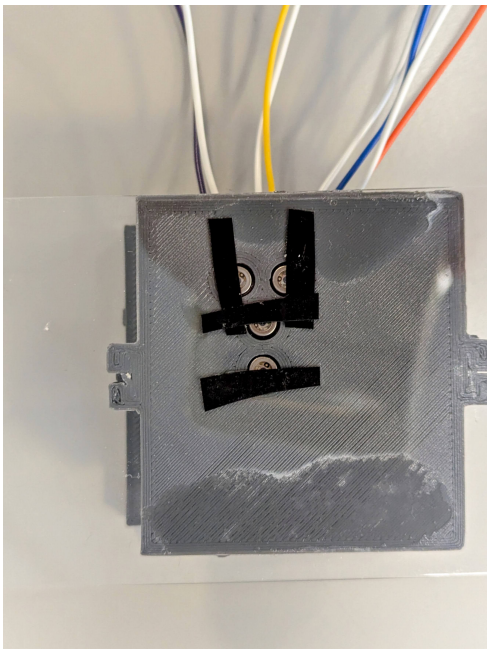


Figure 4.5.: Photo of aperture placement on screen protector glass

Comprehensive Testing

Afther all th

4.1.2. Signal Conditioning Circuitry

Photodiodes produce a certain amount of current when light hits the depletion region. Therefore, a larger depletion region is desirable, to capture more light and in turn produce more current. For this purpose the photodiode in our circuit is reverse-biased as can be seen in Figure 4.6 [21, p.155].

Transimpedance Amplifier (TIA)

A reverse-biased photodiode allows a current to flow from the cathode to anode which is connected to ground. This current is converted to a Voltage using a TIA with the following relationship:

$$V_{\text{out}} = -I_{\text{ph}} \cdot R_f \tag{3}$$

$$I_{\text{ph}} = P \cdot R_\lambda \tag{4}$$

Where P is Light Power (W) and R_λ is Responsivity (A/W).

$$V_{\text{out}} = -(P \cdot 0.5 \text{ A/W}) \cdot 1 \text{ M}\Omega \tag{5}$$

The TIA circuit makes use of an OpAmp as seen in Figure 4.6 that provides very high imput impedance (1GΩ) and allows the amplification of the signal without disturbing the photodiode current, therefore not affecting the readings. The inverting input is used in this configuration, which converts the negative current flowing from the cathode to the anode of the photodiode, into a positive voltage.

Secondary Amplification

Testing showed that even using a 1MΩ resistor, the output voltage was too low (around 310mV) at our RED testbench' LEDs maximum brightness, as explained in Section 4.1.1. To raise the maximum Voltage to the desired maxium of the ADC of 5V, a higher feedback resistor could be used, however this would introduced noise and would require more complicated TIA with feedback capacitors. Due to the LM324-N having 4 OpAmps, the decision was taken to implement a Secondary Amplification circuit. The non-inverting OpAmp configuration was chosen to maintain the voltage positive, which also means there is no need for a dual power supply and keeps the Voltage positive for the Arduino ADC.

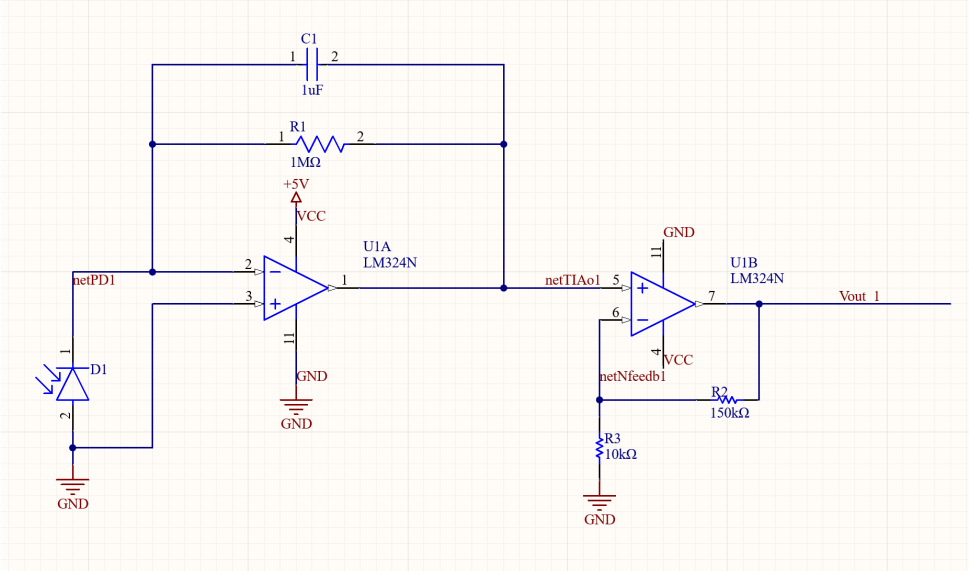


Figure 4.6.: TIA and Post Amplification Circuit in Altium Designer

A simple calculation was made to figure out the required Gain of the circuit:

$$A = \frac{\text{required Voltage}}{\text{measured}} = \frac{5\ \text{V}}{0.31\ \text{V}} = 16.1 \tag{6}$$

Knowing the gain required, the feedback resistor was calculated by choosing a 10kΩR_1 and rearranging the gain equation:

$$\begin{aligned} A &= 1 + \frac{R_f}{R_1} \\ 16 &= 1 + \frac{R_f}{10\ \text{k}\Omega} \\ 16 - 1 &= \frac{R_f}{10\ \text{k}\Omega} \\ 15 &= \frac{R_f}{10\ \text{k}\Omega} \\ R_f &= 15 \times 10\ \text{k}\Omega \\ R_f &= 150\ \text{k}\Omega \end{aligned} \tag{7}$$

This provides a gain $A = 16$ which is very close to the Gain required in Equation 6. Further it must be stated that the resistors used have a tolerance of 10% - therefore the actual final gain will fluctuate by that much. Once the design was tested on a BreadBoard, it was transfered to a stripboard as pictured in Figure 4.4. Later in the design during testing, a decision was made to add a 1μF capacitor in parallel with the feedback resistor

of the Secondary Amplifier. This creates a low-pass filter on the output as seen in Eq. 8.

$$
\begin{aligned}
f_c &= \frac{1}{2\pi RC} \\
f_c &= \frac{1}{2\pi \cdot 150\ \text{k}\Omega \cdot 1\ \mu\text{F}} \\
f_c &= \frac{1}{2\pi \cdot 150 \cdot 10^3 \cdot 1 \cdot 10^{-6}\ \text{s}} \\
f_c &= \frac{1}{2\pi \cdot 150 \cdot 10^{-3}\ \text{s}} \\
f_c &= \frac{1}{0.942\ \text{s}} \\
f_c &= 1.061\ \text{Hz}
\end{aligned}
\tag{8}
$$

This does mean that we are now restricting the design to not be able to show Voltage change rates at higher than 1Hz, and testing with a moving light source will have to be restricted to a frequency at least half of 1Hz. Otherwise, the rate of change of Voltage will appear gradual and not represent the real signal.

4.1.3. Enclosure Design 3D print

This section provides an overview on the design and fabrication of the enclosure for the prototype. The enclosure is a critical component that houses the electronic components and provides protection against environmental factors. The design process involves several steps, including conceptualization, modeling, and fabrication.

Conceptualization

Material Selection

Fabrication Process

Testing and Validation

Iterations and Improvements

Final Enclousure Specifications and Documentation

4.2. Data Acquisition System

4.2.1. Functional Requirements

The output signal from the photodiode array amplifier is required to be converted to digital form for post-processing. This requirement is filled by designing a Digital Acquisition System (DAQ) capable of recording the signal from the four photodiode circuits

simultaneously. The choice of design was conceived by analyzing the analog signal and determining some basic requirements of the Analog to Digital Converter (ADC) the DAQ must possess.

Analog Signal Characteristics

- The signal is four channel, one per photodiode, and between 0 and 5 Volts, as the TIA and post amplification was designed specifically for this output.
- Close to DC frequency, i.e., static in nature, due to light intensity remaining static under most tests. One test is performed at 0.2Hz, which is still still very low frequency, with the light completing a semicircular arc once in 130 seconds (26 positions of 5 seconds each).
- Later in testing it was found that the signal is impacted by interference of 400mVpp at a frequency fluctuating from 160kHz to 180kHz from the RED testbench power supply, as pictured in Figure 4.7.

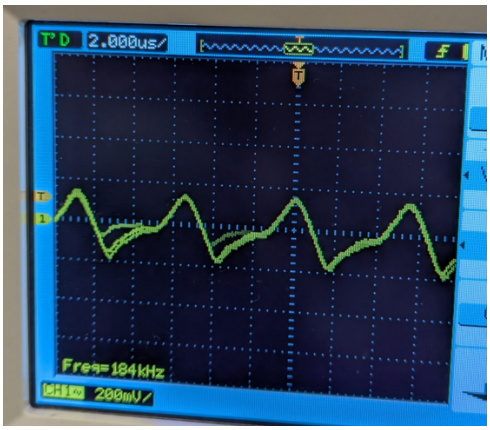


Figure 4.7.: Signal Noise Analysis, oscilloscope AC coupled

CSV Data Structure and Format Specification is as follows:

The output of the DAQ is to be saved in Comma Separated Values (CSV) file format, with columns as follows: Sample(nr.), Time(ms), A0(V), A1(V), A2(V), A3(V). This allows for easy post-processing and plotting.

4.2.2. Design Approach

The characteristics of the signal being low frequency, combined with the requirement to read all four signals simultanously and in sync, meant two things: the Sampling Rate could be quite low due to Nyquist theorem telling us that the sampling rate must be at least twice the frequency of the signal being sampled, in order to maintain the original

signal without aliasing [1, p. 146]. Therefore a low performing ADC is acceptable for a signal changing at under 1Hz. And secondly, the DAQ must support sampling from at least four analog inputs. These requirements meant that a cheap Arduino based DAQ could fit perfectly the needs of the project: it is powered by the Atmega328P which has an included ADC of 15 ksps [22, p.205]. And the Arduino Nano has four analog inputs.

Arduino Programming

The Arduino-based DAQ will require both a C++ program written for the Arduino itself, as well as a program or script on the PC receiving the digitized signal, this is because the Arduino lacks both the memory requirements and capability to store the recorded digitized signal to some internal memory.

The Arduino C++ Program must be able to listen to commands from the user on the PC receiving, start a recording, and immediatly transmit to the PC over serial communication.

4.2.3. Technical Specifications

Arduino Code

The Arduino Code which uses the Arduino ADC is formed of the setup() function triggered once at the start/reset of the device and a standard contnuous loop triggered after setup comletes. Inside the loop, two if statements check for instructions from the PC script. The recording time limit is hardcoded as a global function. Figure 4.8 shows the algorithm as a Flowchart that checks for Serial data in, waits for a command to start recording, and if recording time has reached the preset limit, it stops recoring, sends the last values to the Python script on the PC and a "recording_stopped" command. A FSM diagram is also available in Figure 4.9. The pseudocode used while designing the Arduino side of the DAQ system, is available in Listing 4.1. The final code is available in Appendix A.1.1.

The Atmega328P does not have a separate ADC clock input, therefore the CPU clock is used by first dividing by a default rate of 128, this divider is changed to 16 by changing bits 2-0 to 100, as per [22, p.219]. This increases the clock speed available to the ADC for a higher sampling rate. This results in a 1MHz clock signal to the ADC (16MHz/16) which seemed needed when dealing with multiplexing four analog inputs to a single ADC. The process is as follows:

Original ADC Clock Speed (with default prescaler of 128):

$$f_{\text{ADC-default}} = \frac{f_{\text{CPU}}}{Prescaler_{\text{default}}} = \frac{16 \text{ MHz}}{128} = 125 \text{ kHz} \tag{9}$$

Optimized ADC Clock Speed (with modified prescaler of 16):

$$f_{\text{ADC-optimized}} = \frac{f_{\text{CPU}}}{Prescaler_{\text{optimized}}} = \frac{16 \text{ MHz}}{16} = 1 \text{ MHz} \tag{10}$$

Conversion Time Calculations: ADC requires approximately 13 clock cycles for each conversion [22, p.208] Optimized ADC Clock Speed (with modified prescaler of 16):

$$T_{\text{conversion-default}} = 13 \times \frac{1}{f_{\text{ADC-default}}} = 13 \times \frac{1}{125 \text{ kHz}} \approx 104 \ \mu\text{s} \tag{11}$$

$$T_{\text{conversion-optimized}} = 13 \times \frac{1}{f_{\text{ADC-optimized}}} = 13 \times \frac{1}{1 \text{ MHz}} \approx 13 \ \mu\text{s} \tag{12}$$

Time required to sample all 4 analog inputs:

$$T_{\text{4channels-default}} = 4 \times T_{\text{conversion-default}} = 4 \times 104 \ \mu\text{s} \approx 416 \ \mu\text{s} \tag{13}$$

$$T_{\text{4channels-optimized}} = 4 \times T_{\text{conversion-optimized}} = 4 \times 13 \ \mu\text{s} \approx 52 \ \mu\text{s} \tag{14}$$

Maximum theoretical sampling frequency for all 4 channels:

$$f_{\text{sampling-max-default}} = \frac{1}{T_{\text{4channels-default}}} = \frac{1}{416 \ \mu\text{s}} \approx 2.4 \text{ kHz} \tag{15}$$

$$f_{\text{sampling-max-optimized}} = \frac{1}{T_{\text{4channels-optimized}}} = \frac{1}{52 \ \mu\text{s}} \approx 19.2 \text{ kHz} \tag{16}$$

Actual limited sampling frequency (based on minSampleInterval = 2ms):

$$f_{\text{sampling-actual}} = \frac{1}{2 \text{ ms}} = 500 \text{ Hz per channel} \tag{17}$$

Effective data rate across all channels:

$$\text{Data Rate} = 4 \text{ channels} \times 500 \text{ Hz} = 2000 \text{ samples/second} \tag{18}$$

In real testing the actual sampling rate was closer to 330Hz for 5 second recordings or 100Hz for a 2 minute recording - after some investigation the only explanation was the relativly small size of the transmission buffer implemented by the Serial C++ library. The buffer is of only 64 bytes, and when it fills, the function Serial.write() (used by println()) will block the write untill there is space in the buffer[ref:arduino.cc/serial.write]. As our line of text is quite long "498,5000,0.059,0.054,0.073" for example has 26 characters (last line of a 5 second recording). For larger recording length, where the first and second column, Sample and Time, can get quite large, the sampling rate decreased considerably, but was kept constant (around 10ms for a 2 minute recording). Presumably due to optimization in the Arduino Serial Hardware/Software or compiler, it remains constant at 10ms. However this was not investigated further as for our near-DC signal, even 10ms was a fast enough sampling rate for our DC-like signal.

```
recordingDuration = 5000 // for how long to record in milliseconds

minSampleInterval = 2    // control how fast to sample to avoid
                         // relying on Arduino performance
// Initialize serial communication
// Initialize analog inputs
// Setup ADC

// Infrom PC listening on Serial Connection: Arduino is ready to
 record
Serial.print("Arduino_DAQ_Ready")
// enter the loop
void loop(){
  //listen for command from PC script:
    String command = Serial.read()
  //set system state
  if (command == "START"){
    // Send header of csv
    Serial.println("Sample,Time(ms),A0(V),A1(V),A2(V),A3(V)")
    // keep track of system state
    state = recording
    // keep time
    startTime = currentTime()
    // send confirmation
    Serial.println("recording in progress")
    }
  // check if recording
```

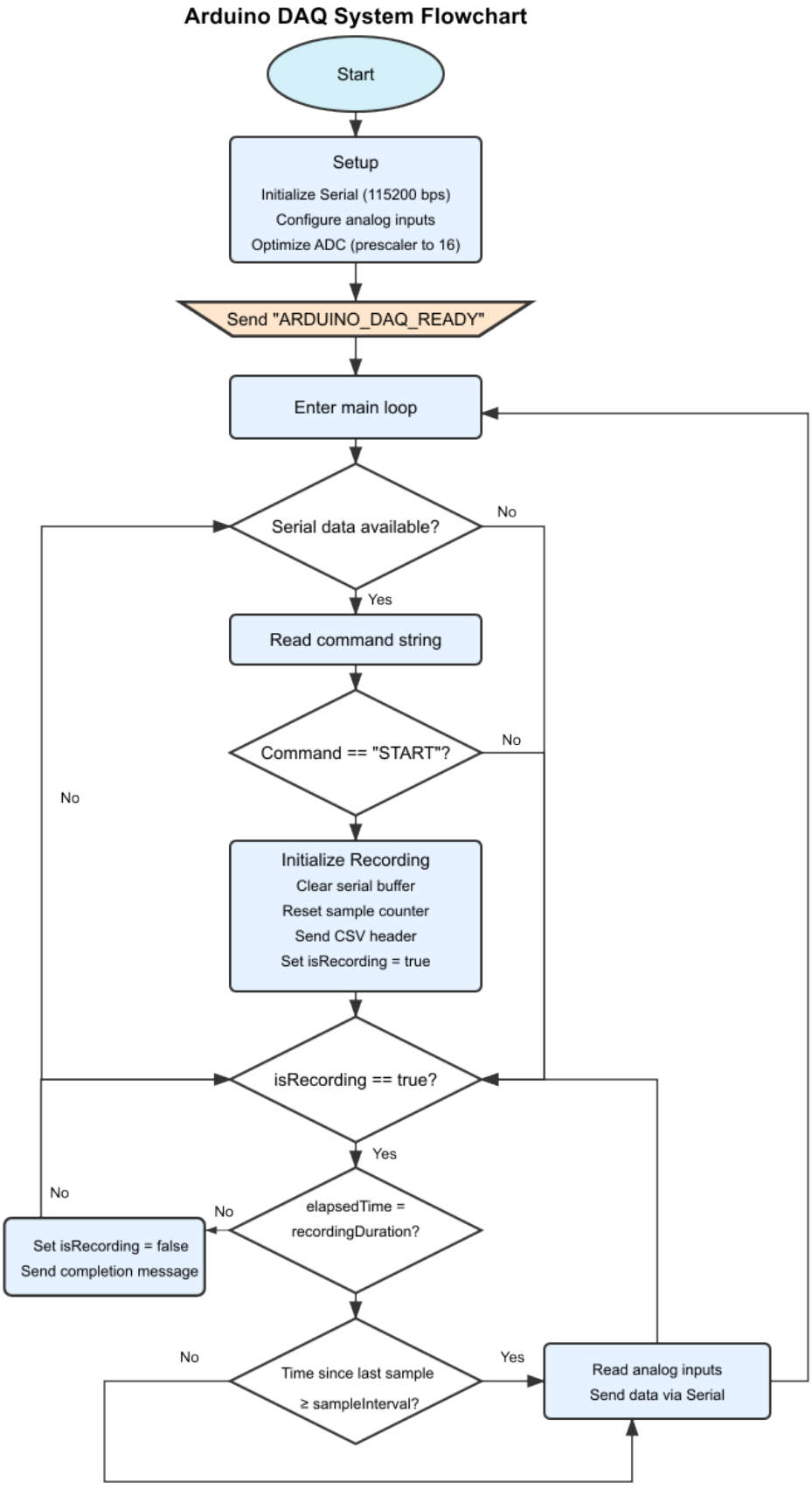


Figure 4.8.: Flowchart Arduino DAQ C++ program

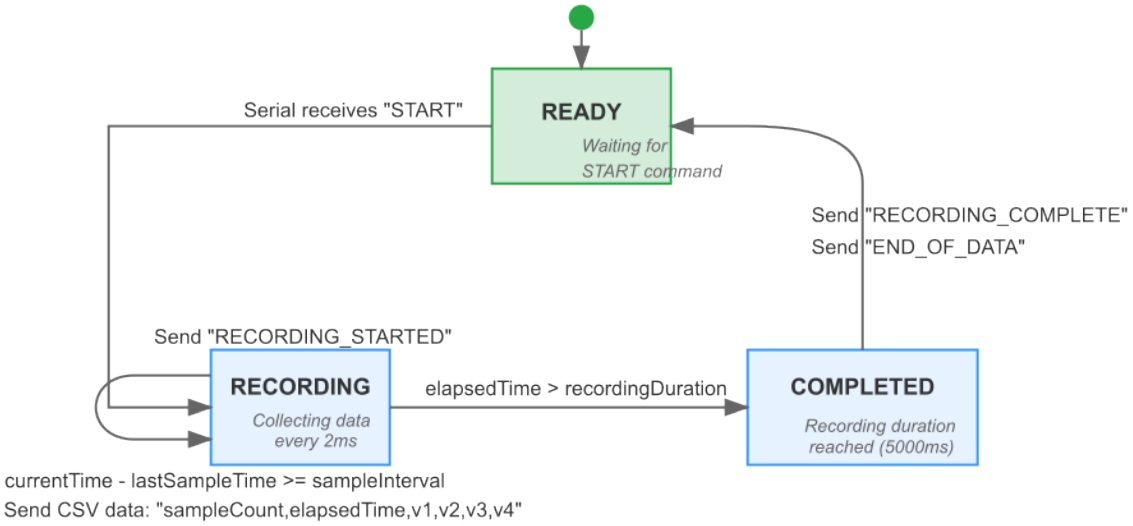


Figure 4.9.: FSM Arduino C++ LOOP

```
27    if (state == recording){
28      // check if within recording period
29      currentTime = currentTime()
30      elapsedTime = currentTime - startTime
31      if(currentTime <= recordingDuration){
32        // Also check not recording too fast
33        if(currentTime - lastSampleTime >= minSampleInterval){
34          sampleCount++;
35          // Start each row with sample count and time of sample
36          String currentCSVrow = String(sampleCount) , string(
    elapsedTime)
37          // Multiplex through all analog inputs
38          for (int = 0;i<4;i++){
39            //read raw values
40            rawValue = analogRead(analogInputs[i]);
41            // compute real value
42            voltage = rawValue * 5/1023
43            // add value to current row to send
44            currentCSVrow += String(voltage)
45            }
46          // Send completed row to PC
47          Serial.println(currentCSVrow)
48          }
49        }
50      }
```

```
51    // end recording
52    else{
53      state = notRecording
54      // tell PC recording finished
55      Serial.println("Recording_finished")
56      }
57  }
```

Listing 4.1: Arduino DAQ PseudoCode

Sampling Rate Details Several factors restrict the sampling rate:

Sample Interval Setting The most direct limitation is the `sampleInterval` constant set to 2ms in the code. It was meant to avoid having random sampling rates based on the number of computations required. This means samples are taken no more frequently than every 2 milliseconds (500 Hz theoretical maximum) of all four channels. The "jump" to sample the next channel is not limited in the code, but it will take 13 clock cycles, (ie. around 13µs at 1MHz ADC clock) to switch to the next channel.

ADC Prescaler Configuration The ADC prescaler is set to 16 (from the default of 128) with this line:

```
ADCSRA = (ADCSRA & 0xF8) | 0x04;
```

This increases the ADC clock to 16MHz/16 = 1MHz. With each conversion taking 13 ADC clock cycles, the theoretical maximum sampling rate is about 76.9kHz for a single channel.

Serial Transmission Overhead Each sample requires formatting and sending data over serial:

```
String dataString = String(sampleCount) + "," + String(elapsedTime);
// ... format and add voltage values ...
Serial.println(dataString);
```

This string creation and serial transmission takes some time to process as mentioned earlier.

Serial Baud Rate The code uses 115200 bps, which limits how quickly data can be transmitted. Each sample in this format might be around 30-40 bytes, which means ~3000-3800 samples/second theoretical maximum throughput.

String Operations The use of the Arduino `String` class is memory-intensive and can cause fragmentation over time, potentially causing the slowdowns noticed during testing with larger timeframes (and longer strings).

Python Script

The digitized signal must be interpreted and saved on the PC. This is done via a python script which listens to the Serial port from the arduino. The signal then also required cleaning from RF interference discovered during testing. In Figure 4.10 a flowchart is produced showing the way the script works: after initial setup that sets up the Serial Communication, the script waits for a "DAQ_READY" signal from the Arduino. Once this is received, a csv file is created and the script sends a "START" signal which the Arduino interprets and starts sampling and sending the data. The script receives each line and saves it in the new CSV, and continues to record data until the Arduino sends a "RECORDING_COMPLETE" signal - which will happen when the recordingDuration is reached. At this point the python script performs the following post-processing steps:

The filter_and_save_data() function' purpose is mainly to correctly interpret the CSV. It takes a csv with the raw voltage values, saves them into a Pandas DataFrame for easier manipulation and sends each channel to the apply_lowpass_filter() function which will be described below. filter_and_save_data() pseudocode is produced in Listing 4.2.

```
FUNCTION filter_and_save_data(filename)
// Load data from CSV file into a table structure
data_table = READ_CSV(filename)

// Streamline the data by converting text to numbers
FOR EACH column IN data_table
    CONVERT column values to numeric type
    IF conversion fails for any value
        REPLACE with NaN (Not a Number)
END FOR

// Remove any rows containing NaN values
REMOVE all rows with NaN values from data_table

// Calculate sampling frequency
time_differences = CALCULATE differences between consecutive time
 values
typical_time_difference = FIND median of time_differences
sampling_frequency = 1000.0 / typical_time_difference  // Convert ms
 to Hz

// Process each data channel
channel_list = ["A0(V)", "A1(V)", "A2(V)", "A3(V)"]
```

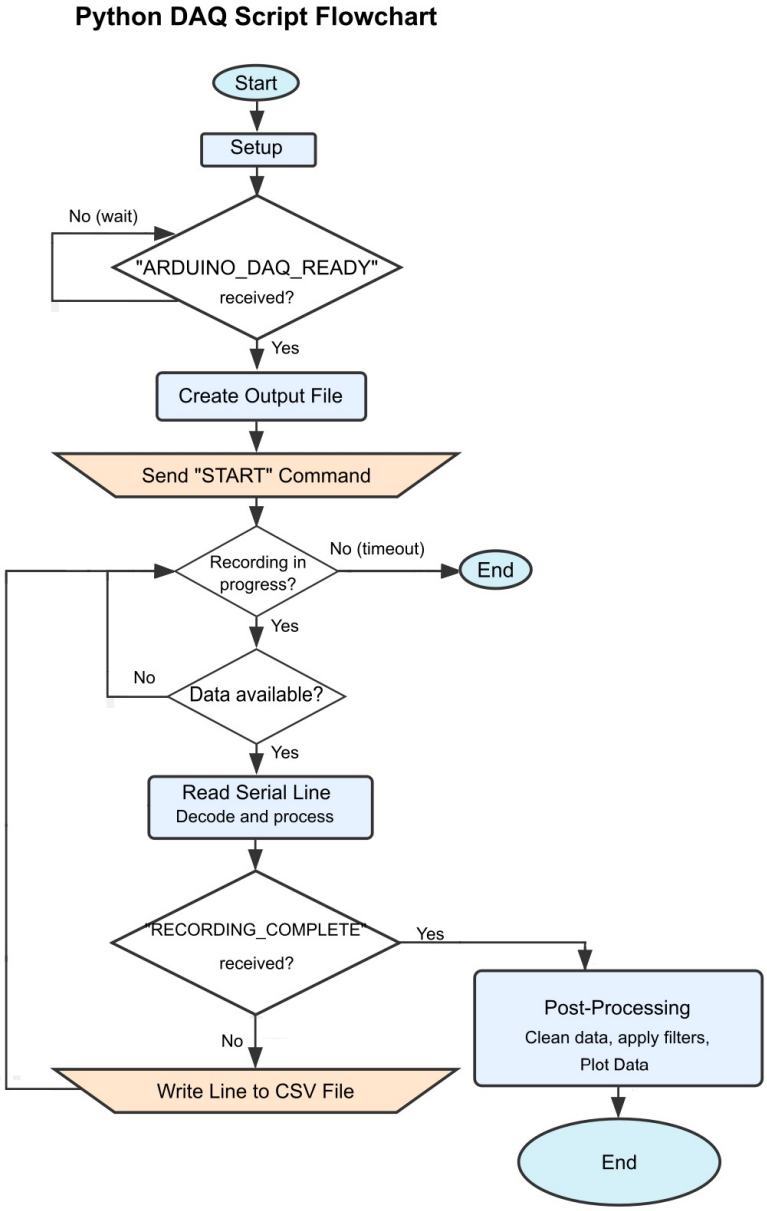


Figure 4.10.: Python Script Flowchart

```
22
23    FOR EACH channel_name IN channel_list
24        IF channel_name EXISTS in data_table
25            filtered_values = APPLY_LOWPASS_FILTER(original_values,
      sampling_frequency)
26            ADD new column named channel_name + "_filtered" with
      filtered_values
27        END IF
28    END FOR
29
30    // Save results to new file
31    new_filename = REMOVE_EXTENSION(filename) + "_filtered.csv"
32    WRITE data_table TO new_filename
33
34    RETURN new_filename
35 END FUNCTION
```

Listing 4.2: Python filter_and_save_data() PseudoCode

The apply_lowpass_filter() function was created and integrated into the script once it was observed that the RED testbench used was introducing noise as seen in Figure 4.7. The benefit of using an already built testbench that could place the light at exact positions repeatedly, meant it would make sense to accept the noise and just filter the data, as the signal of interest was very low frequency while the noise was around 170kHz. The filter could be relativly simple, due to the large frequency difference beteween noise and signal. A 4th order Butterworth IIR filter was deemed acceptable and a low cut-off frequency of 1Hz or 2Hz was used for the static readings. This was found to be acceptable for the test involving light location at a frequency of 0.2Hz the transition of the light from one position to another was still visibly sharp. The post-processing is not the only reason light transition would not appear instantaneous on the Voltage graph, another reason is the amplification circuit which has a 1Hz cutoff frequency via the feedback capacitor on the secondary-amplification OpAmp circuit. The pseudocode of the function that creates the low-pass digital filter and filters the data is reproduced in Listing 4.3. The butter() function from the library signal is used, from the scipy package [23] which is a free and open source library offered to the scientific comunity. Before feeding the cutoff frequency to the filter, it is normalized to the Nyquist rate, which means it is between 0 (DC) and 1 (Nyquist frequency), and therefore the filter can be applied no matter the sampling rate. As Schafer and Oppenheim explain in "Discrete-Time Signal Processing" Third Edition: "The frequency scaling or normalization in the transformation from $X_s(j\Omega)$ to $X(e^{j\omega})$ is directly a result of the time normalization in the transformation from $x_s(t)$ to $x[n]$" [1, p.171]. When creating a digital filter, this normalization becomes crucial because in the continuous-time domain, frequencies are measured in Herz and in the discrete-time

domain, frequencies become relative to the sampling rate. The relationship between these two frequency domains is given by $\omega = \Omega T$, where:

- ω is the normalized digital frequency (radians/sample)
- Ω is the analog frequency (radians/second)
- T is the sampling period (seconds)

However, when implementing IIR filters like the Butterworth filter used for our purpose, the bilinear transformation is employed to convert from the continuous-time domain to the discrete-time domain. This transformation introduces frequency warping, where the relationship between the analog and digital frequencies becomes:

$$\omega = 2 \arctan(\Omega T_d/2)$$

where T_d is the sampling period. This nonlinear relationship compresses the infinite analog frequency range $(-\infty, \infty)$ into the finite digital frequency range $(-\pi, \pi)$. The warping effect is more pronounced at higher frequencies, meaning that in our case it would not be noticeable [1, p.529-530].

A critical property of the bilinear transformation is stability preservation. In the analog domain, a stable system has all poles in the left half of the s-plane reproduced in Figure 4.11. The bilinear transformation maps the entire left half of the s-plane to the interior of the unit circle in the z-plane. This ensures that the 4-pole Butterworth filter, which is stable in the continuous-time domain, remains stable when converted to its discrete-time equivalent.

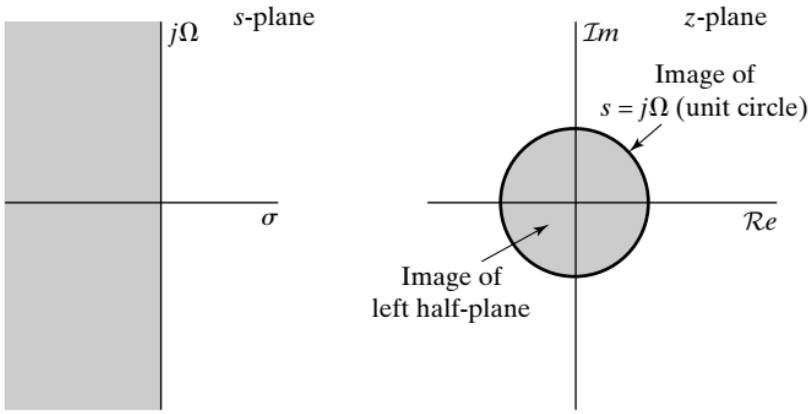


Figure 4.11.: Mapping of the s-plane onto the z -plane using the bilinear transformation [1, p.130]

The filter Frequency Response was reproduced in Figure 4.12 using the scipy.signal-.freqz() method. The actual implementation uses filtfilt() which applies the filter twice,

effectively doubling the filter order [24]. This affects the transition steepness and phase response. The Discrete-Time Transfer Function is reproduced in Equation 19.

$$H(e^{j\omega}) = \frac{b[0] + b[1]e^{-j\omega} + b[2]e^{-j2\omega} + b[3]e^{-j3\omega} + b[4]e^{-j4\omega}}{a[0] + a[1]e^{-j\omega} + a[2]e^{-j2\omega} + a[3]e^{-j3\omega} + a[4]e^{-j4\omega}} \tag{19}$$

```
from scipy import signal
FUNCTION apply_lowpass_filter(data, sampling_rate)
  // set hardcoded filter values
   cutoff_freq = 2;
   filter_order = 4;
  // calculate Nyquist Frequency and Normalized cut_off
  nyquist = 0.5 * sampling_rate;
  norm_cutoff = cutoff_freq / nyquist;
  // generate numerator b and denominator a polinomials
  b, a = signal.butter(filter_order, norm_cutoff, filterType = LOW);
  // filter the data
  filtered_data = filter(b,a,data);

  return filtered_data;
```

Listing 4.3: Python apply_lowpass_filter() PseudoCode

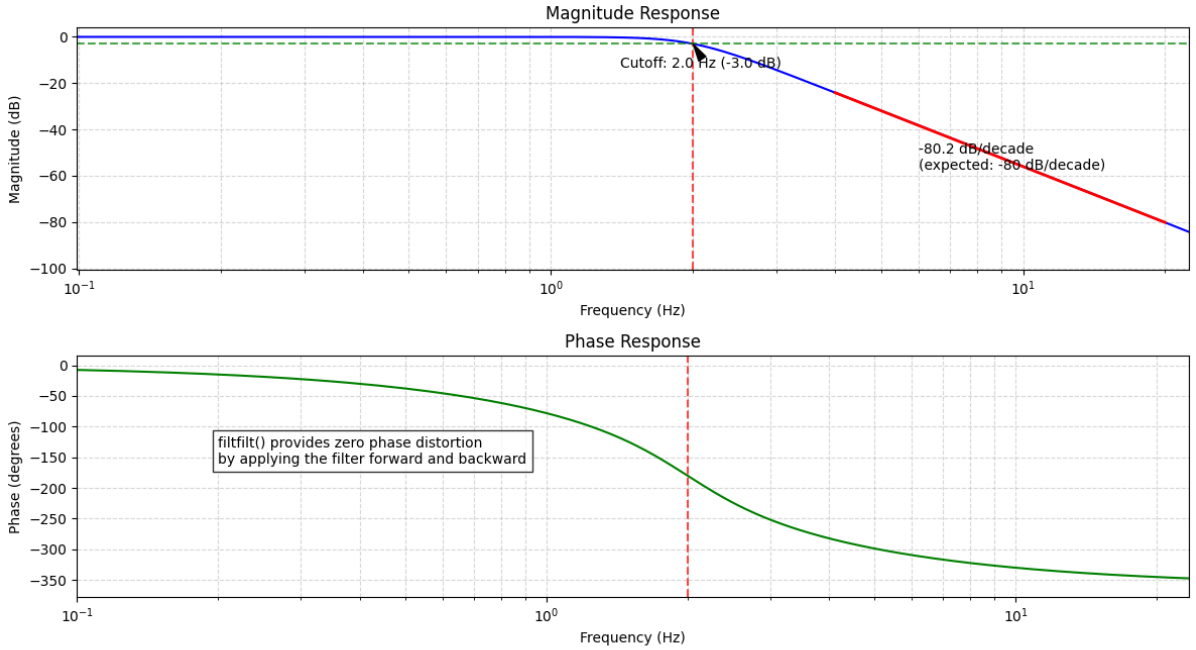


Figure 4.12.: Frequency Response of 4-pole Butterworth Low-Pass Filter (Cutoff: 2.0 Hz, Order: 4, Fs: 500 Hz)

4.2.4. Testing Strategy

The testing was done incrementally with each new version of the program, many tests were carried out as the development was dependent on both C++ code on the Arduino and the Python script on the PC to somewhat work together. At first the C++ program was tested by listening to the output in the Serial Monitor of the Arduino IDE and sending commands the same way. Debug Serial.println() lines were used to ensure the loop on the Arduino was in the correct state. Once the C++ program seemed to somewhat work as intended, the Python script was implemented to send / receive.

4.2.5. Evaluation of design

The design was tested iterartivly while making changes to the code with a signal generator at first to simulate a stead known input signal and compare readings on the Arduino IDE Serial Monitor. Once the Python script was fully working, tests were performed again for the python script's ability to read the values sent correctly. An issue was identified where sometimes the Arduino would send the column headers not at the beggining of the CSV and these lines had to be cleaned manually. This could be due to Serial buffer issues but was not investigated. Later an attempt was made to perform the cleaning programatically by creating a function in the script to do so.

4.3. Renewable Energy Demonstrator Testbench

For testing the capability of the Sun Sensor to correctly detect the location of the light source, a test bench was required that could reliably place the location of the light at a precise location repeatedly. For this purpose we used a project built by our colleagues in the European Project Semester year 2021/22 who createad just such a device intended for demonstrating renewable energy creation live [20]. Their device was able to demonstrate the energy levels created by a Photovoltaic (PV) cell by light emited at different angles. The light emission would change location based on time of day and the PV cell readings would show the difference in energy. Futher the PV cell was controllable by a joystick to point the PV Cell at the optimum angle for the highest energy capture. For our project, the arch and LED strip were used for outputing light from different angles.

Analysis of High Frequency Noise in AC-DC Power Supply

Interference structure of around 170kHz with 400mV peak-to-peak was detected on the signal being received while the RED testbench was on as shown in Figure 4.7. This noise could be generated by several factors in the AC power supply used by the RED testbench:

1. **Switching frequency harmonics** — If it's a switch-mode power supply (SMPS),

the fundamental switching frequency or its harmonics might be causing the noise. Many SMPS operate in the 50–200,kHz range.

2. **Poor filtering** — Inadequate output filtering (insufficient capacitance or poor quality capacitors) can allow switching noise to appear on the output.

3. **Improper design of magnetics** — Issues with the transformer or inductor design could cause ringing or oscillations.

4. **Resonance in the circuit** — Parasitic capacitance and inductance forming a resonant circuit at around 170,kHz.

5. **Control loop instability** — PWM controller instability can cause oscillations.

6. **Ground loops or poor PCB layout** — Improper grounding or PCB layout can create noise paths. [25]

To avoid spending time diagnosing and trying to repair the testbench, an easier solution was reached: performing digital filtering of the acquired signal in post processing. Due to the signal of interest being close to DC - frequencies much lower than 1Hz, and the noise being high frequency, around 175kHz, a simple digital Butterworth filter with a cutoff frequency at around 1-2 Hertz was found to be a good solution.

The only remaining issue was that this noise would sometimes trigger the internal components of the testbench, unintentionally triggering the button press from the control interface that was changing the light position, but it happened so rare that it was not a major concern.

4.4. Software Model

4.4.1. Functional Requirements

A Python model was constructed to provide a simulation of the movement and intersection of rays from a movable source to evaluate sensor performance and compare these results with practical experiments. The model allows for a number of configurable parameters:

- Trajectory of the light source 3D space, which moves in configurable discrete increments.

- Placement of any number of sensors and apertures, including their dimensions.

- Output visualisation, as a static, or animated graphic.

Affording flexibility for the model to simulate any sensor topology under a variety of conditions.

4.4.2. Theory and Concept

In order to encapsulate the real life behaviours required to simulate this system, the software model is designed to around a structure based on the classes "Planes", "Areas", and "Lines". These classes contain methods related to each of their properties and stores data on their states.

The basis for the 3D geometry involves defining:

- Each line by vectors representing position and normal direction, $\vec{A} = (a, b, c)$ and $\vec{u} = (\alpha, \beta, \gamma)$
- Each plane by the vectors $\vec{P} = (l, m, n)$ and $\vec{n} = (\lambda, \mu, \nu)$, respectively.

Coordinate System

Each ray is initially defined in the local coordinate system of the source plane. This local frame is established by three orthonormal basis vectors:

- $\vec{r}$ — right vector (local x-axis)
- $\vec{u}$ — up vector (local y-axis)
- $\vec{n}$ — normal vector (local z-axis)

These vectors form the columns of a rotation matrix R which transforms local coordinates to global:

$$R = \begin{bmatrix} \vec{r} & \vec{u} & \vec{n} \end{bmatrix} = \begin{bmatrix} r_x & u_x & n_x \\ r_y & u_y & n_y \\ r_z & u_z & n_z \end{bmatrix} \tag{20}$$

A point $\vec{p}_{\text{local}} = \begin{bmatrix} x_l & y_l & z_l \end{bmatrix}^\top$ defined in local coordinates is converted to global coordinates via:

$$\vec{p}_{\text{global}} = \vec{p}_{\text{plane}} + R \cdot \vec{p}_{\text{local}} \tag{21}$$

Where:

- $\vec{p}_{\text{plane}}$ is the global position of the origin of the plane
- R rotates the local point into the global frame

This ensures that all rays originating from the source plane move coherently when the plane is rotated or translated.

Ray generation

Rays are randomly generated within the bounds of the source plane, whose position, size, and direction are defined in the configuration. The position of each ray is calculated using:

$$X_i \sim \mathcal{U}\left(-\frac{w}{2}, \frac{w}{2}\right), \quad Y_i \sim \mathcal{U}\left(-\frac{l}{2}, \frac{l}{2}\right), \quad \text{for } i = 1, \ldots, N \tag{22}$$

Ray projection, from a source plane to a sensor plane, is modelled using the parametric equation of a 3D line (23). This allows each ray to be described in terms of a parameter t, which enables the calculation of the intersection points between the light rays and the sensor plane.

Line-Plane Intersection

$$\frac{x-a}{\alpha} = \frac{y-b}{\beta} = \frac{z-c}{\gamma} (= t) \tag{23}$$

Where the intersection coordinates (x, y, z) occur within a target area, a hit occurs, representing illumination.

For any given combination of source plane, and sensor plane, the t parameter is calculated using the Line-Plane Intersection equation.

$$t = \frac{\vec{n} \cdot \vec{P} - \vec{n} \cdot \vec{A}}{\vec{n} \cdot \vec{u}} \tag{24}$$

Arc movement

The simulation converts between spherical (polar) and Cartesian coordinate systems to define the movement of the source plane along an arc trajectory. This transformation enables position and orientation of the plane in 3D.

The spherical coordinates are expressed as (r, θ, φ) and are converted to Cartesian as:

$$x = r \cdot \sin(\varphi) \cdot \cos(\theta) \tag{25}$$

$$y = r \cdot \sin(\varphi) \cdot \sin(\theta) \tag{26}$$

$$z = r \cdot \cos(\varphi) \tag{27}$$

This enables generation of plane positions around a defined arc, supporting vertical, horizontal, and rigid trajectory configurations.

Movement types

- Azimuthal Scanning / Vertical Circles: The azimuthal angle θ remains fixed, while the elevation angle φ is varied.

- Polar Scanning / Horizontal Circles: The elevation angle φ remains fixed, while the azimuthal angle θ is varied.
- Rigid Arc: A semi-circlular path in the $xz - plane$ is generated, with the plane rigidly rotated to always face the origin. This mode uses the Rodrigues rotation formula to reorient the source plane toward the target, as it required rotation about an arbitary axis.

$$R = I + \sin\theta \cdot K + (1 - \cos\theta) \cdot K^2 \tag{28}$$

$$K = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \tag{29}$$

For each position along the trajectory, the plane is rotated such that its normal direction vector $\vec{n}$ points towards to the origin, about which the sensor plane is defined.

Table 4.5.: Definition of vectors and symbols used in ray and plane calculations

Symbol	**Description**
$\vec{A} = (a, b, c)$	Position vector of the ray origin
$\vec{u} = (\alpha, \beta, \gamma)$	Direction vector of the ray
$\vec{P} = (l, m, n)$	A known point on the target plane
$\vec{n} = (\lambda, \mu, \nu)$	Normal vector of the target plane
t	Ray parameter defining point along the ray path
r	Radius in spherical coordinates
θ	Azimuthal angle in spherical coordinates
φ	Elevation angle in spherical coordinates

4.4.3. Implementation

The software model implements the theoretical foundation to provide a flexible framework for simulating ray projection and intersection calculations. The implementation follows a modular object-oriented approach with components that can be configured to represent different experimental setups.

Architecture Overview

The architecture consists of several key components:

- Core geometric objects (Planes, Areas, Lines) that encapsulate the mathematical properties

- Simulation engine for trajectory generation and intersection testing
- Configuration system for experiment setup
- Visualisation pipeline for result analysis
- User interface for interactive control

Model Configuration

A .JSON file is used to allow for easy setup of different experimental scenarios without modifying code:

- Detailed definition of planes, including position, orientation, and dimensions
- Sensor and aperture areas with specific positions and sizes
- Trajectory specifications for various movement types
- Simulation parameters (number of rays, iterations)
- Visualisation options - Static, or animated plot
- Debugging and performance settings

The `Config` class (Listing 4.4) loads and parses the configuration file, making parameters accessible throughout the application:

```

    class Config:
        def init(self, file_path=None, data=None):
        if data is not None:
        self.load_from_dict(data)
        elif file_path:
        with open(file_path, "r") as f:
        data = json.load(f)
        self.load_from_dict(data)
        else:
        raise ValueError("Must provide either file_path or data")

    def load_from_dict(self, data):
        self.planes = data["planes"]
        self.sensor_areas = data["sensor_areas"]
        self.aperture_areas = data["aperture_areas"]
        self.arc_movement = data["arc_movement"]
        self.simulation = data["simulation"]
        self.intersection = data["intersection"]
        self.visualization = data["visualization"]
        self.debugging = data["debugging"]
```

```
22          self.performance = data["performance"]
23          self.output = data["output"]
24
```

Listing 4.4: Model configuration - Config Class

4.4.4. Results and Evaluation

4.5. Material Analysis and Selection

4.5.1. Material Selection and Requirements

The PCB will be operated in harsh environmental obstacles from strict requirements in both space and weight. PCB must withstand the extremities of immense pressures and vacuums, polarising temperatures, vibrations, impacts, space radiation and more. Generally, the materials for PCB made for space conditions are either polyimide or ceramic, as they can withstand extremely harsh conditions [26]. Commonly materials used for space PCB include polyimide, PTFE and alumina [27].

4.5.2. External Factors

The PCB needs to sustain itself when it is launched into and deployed to outer space from the extreme heat it will experience from -200°C in the shadow of a celestial body to over 200°C when sunlight exposes on it

Extreme Temperature Variations

Space exposes PCBs to extreme temperature fluctuations, which can affect their structural integrity and performance. Materials must maintain their thermal stability and dimensional consistency across a wide temperature range.

The temperature in space varies dramatically, ranging from extreme cold in the shadow of celestial bodies to scorching heat under direct solar exposure [28]. Materials expand and contract, with the magnitude of the dimensional change determined by the material's Coefficient of Thermal Expansion (CTE) [29]. When different materials in a PCB assembly have differing CTEs, the repeated cycles of expansion and contraction can result in considerable mechanical stresses at their interconnections [28].

These stresses may result in severe failures such as solder joint cracking, PCB layer delamination, and even circuit malfunction [28]. As a result, materials with naturally low CTE values, such as ceramic PCBs, are widely used in space applications to reduce these concerns [28]. A material's capacity to resist multiple temperature cycles without degradation is also critical for long-term reliability in space missions [27].

Ionizing Radiation

High-energy radiation, including cosmic rays, solar particles, X-rays, and UV radiation, exists throughout the space environment [28].

This ionising radiation poses a significant risk to the performance and durability of electronic components, particularly those found on PCBs. Radiation exposure can impair semiconductor performance, potentially causing data corruption via bit flips and resulting in a steady loss of material properties with time [30].

Celestial sources and the sun emit ionizing radiation that can disrupt the PCB's functionality and degrade the semiconductors' performance, hence, to protect the sensitive electronic components from being damaged by radiation, the PCBs must be made from radiation-hardened materials.

To mitigate these negative effects, radiation-hardened materials are frequently used in space-grade PCBs. Ceramic substrates, for example, are more resistant to radiation than many biological materials. Specialised coatings are also used to give another layer of protection to delicate electrical components [30].

The use of materials with intrinsic radiation resistance is an important design concern for ensuring the ongoing operation of electronic devices in the harsh radiative environment of space [31].

4.5.3. Internal Factors

Mechanical Stresses

The PCB, housing and aperture will experience mechanical stresses during the launch and deployment in the spacecraft, from the vibrations generated from the launch and deployment phase it can cause structural damage. To mitigate this issue there are in place shock-absorbing mechanisms that include the PCB material being flexible and conformal coating to safeguard the electronic components' integrity. Flexible PCBs are more effective at absorbing disturbances and vibrations compared to rigid PCBs.

PCBs are subjected to immense mechanical strains throughout a space mission's launch and deployment phases. The tremendous vibrations and shocks produced during launch, along with the high gravitational forces experienced, can put great stress on every component of the PCB [28].

Furthermore, the deployment of solar arrays and other spacecraft features may cause additional mechanical stress [28]. These forces can cause structural damage to PCBs, such as board cracking, layer delamination, and the breakdown of solder junctions that connect components [28].

To address these issues, designers frequently use shock-absorbing mechanisms, such as flexible PCB materials like polyimide, which are more effective at absorbing vibrations than rigid counterparts. Conformal coatings are also used to give an extra layer of pro-

tection against physical damage during launch and deployment [28]. Furthermore, precise PCB layout design is required to guarantee a more even distribution of mechanical loads across the board [28].

Outgassing and Vacuum

Outgassing is another internal factor which is when manufacturing the PCB, outgassing is a soldering wave defect that traps air within a PCB. This is a major issue as it can lead to the PCB impairment from the cavities or blowholes created by the air inside. This is often a result of defective manufacturing and poor material selection choices.

There is a strict size and weight limit for the PCBs because of the tight space in the spacecraft, thus the PCB must be compact and lightweight whilst maintaining equilibrium that it does neither compromise on the functionality and the structure of the PCB.

The near-perfect vacuum of space poses an additional notable challenge for PCB materials. Materials in this environment can undergo outgassing, which is the release of trapped volatile chemicals [27]. These outgassed compounds can have harmful impacts on delicate spacecraft equipment [32]. Optical instruments, such as cameras and telescopes, are especially susceptible to contamination by outgassed materials, which can deposit on their surfaces and degrade performance [28]. Similarly, thermal control surfaces can be influenced, affecting their ability to regulate the spacecraft's temperature [29].

Furthermore, the release of gases near high-voltage components can raise the risk of corona discharge, which could hinder electronic operations and cause damage [29]. To address these difficulties, the materials used in space-grade PCBs must have low outgassing qualities. Polymers such as polyimide and PTFE (Teflon) are widely used because of their extremely low outgassing properties, which help to protect the integrity of sensitive spacecraft components and systems [28].

4.5.4. PCB Material Selection

High-Tg FR-4

While not as extensively used as polyimide or ceramics in the most extreme space environments, high-Tg FR-4 laminates find application in scenarios where thermal conditions are less severe, such as within crewed spacecraft like the International Space Station [27]. These epoxy-based laminates possess a higher glass transition temperature (Tg) compared to conventional FR-4, providing greater stability at elevated temperatures. Epoxy laminates typically have a glass transition temperature between 150-170°C [27].

Using GRANTA EduPack 2022 R2, we found that FR-4 variations with dissipation factors (DF) of 0.015, 0.02, and more than 0.02 would be excluded from our nanosatellite PCB application. These materials were eliminated because they have larger signal losses at communication frequencies, reducing power efficiency in our power-constrained

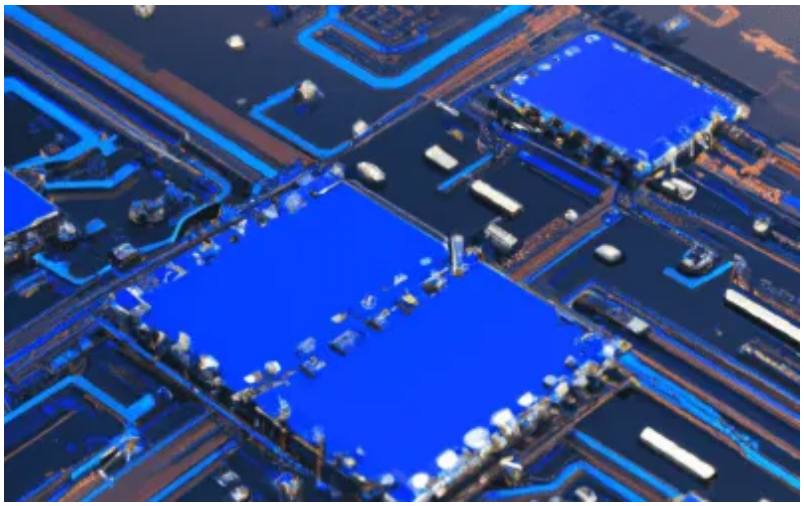


Figure 4.13.: FR-4 PCB [2]

nanosatellite design. The increased signal attenuation will compromise the reliability of data transmission and could also raise thermal loads on the board. Furthermore, ordinary FR-4 materials often have lower glass transition temperatures, which poses dimensional stability problems during the severe heat cycling encountered in the space environment (-60°C to +100°C). These restrictions would negatively impact both the electrical performance and long-term dependability of the PCB, making these materials unsuitable for the demanding circumstances of the space environment.

Even FR-4.0 with DF < 0.01, while superior to other FR-4 varieties, has limits for space applications. Its glass transition temperature range (130-180°C) has a lower limit that may not be adequate for intense orbital thermal cycling. Its thermal expansion coefficient and mild outgassing properties in vacuum may jeopardise long-term reliability and contaminate sensitive optical components. Although more expensive, more specialised space-grade materials would be preferable for maximum performance in nanosatellite applications.

Polyimide

Research is conducted about materials suited for space PCBs, a prominent material used is Polyimide (PI), the common "space age" material with the highest performing class out of plastics.

Polyimide is a flexible polymer that is commonly used as a substrate material in space-grade printed circuit boards due to its inherent flexibility and lightweight nature [27]. This flexibility enables polyimide-based PCBs to effectively absorb mechanical stresses encountered during the dynamic stages of launch and spacecraft deployment. Furthermore, polyimide has strong thermal stability, which allows it to survive temperature fluctuations in space to some extent [27]. It also has low outgassing qualities, which are critical for avoiding contamination of sensitive spacecraft components [28]. However,

polyimide is generally regarded as less suited for applications involving high levels of radiation exposure than other materials such as ceramics [27]. For over three decades, DuPont's Pyralux laminates and Kapton polyimide films have been widely used in the aerospace and defence markets, with applications ranging from multi-layer insulation to wire wrapping and flexible circuit interconnects on solar panel backplanes [33]. The Mars Rover Pathfinder also pioneered the use of adhesiveless laminate in space, using Pyralux AP material [33].

A material analysis was conducted using GRANTA Edupack and provides the general information about the material such as strengths and weakness shown in Appendix D. The main benefit of PI is that it has excellent heat resistance [260°C (500°F)] in continuous use, in tandem with having inherent fire-retardance and low smoke emission and excellent heat distortion temperature makes it a prime candidate for PCB. The limitations that Polyimide has is the high cost to produce, with the melt process being very difficult, making the manufacturing process of it be a slower process. It also has a relatively low impact strength, and worse elongation at break which can be a problem during the deployment process.

Alumina

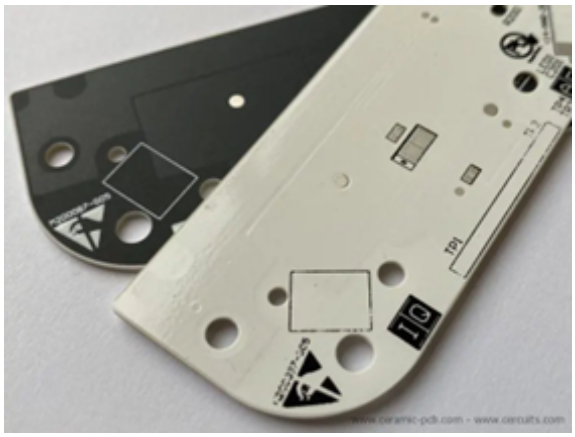


Figure 4.14.: Alumina Oxide PCB [3]

Alumina (Aluminium Oxide or Al_2O_3) is a ceramic that is a chemical compound of aluminium and oxygen, the 96% refers to the purity of the aluminium. A material analysis was also conducted using GRANTA Edupack and is shown in Appendix D. In abrasive environments they can maintain a long service life due to their high hardness and wear resistance, which would be ideal for the space usage because it has one of the extreme environments imaginable. They also have a high-temperature resistance with a high melting point at 2000-2100°C and outstanding thermal stability. It also has excellent corrosion resistance which is ideal for space usage where it can be a corrosive environment. It also is lightweight with its density ranging from 3.69 to 3.73 g/cm^3. The disadvantages

of it are the brittleness of the ceramic and can fracture on impact which limits the lifespan in the space usage to mechanical shocks. It is also difficult to machine from the high hardness so initial costs for manufacturing and lead time are higher than other materials [34], [35].

PTFE(Teflon)

Figure 4.15.: PTFE/Teflon sheets [4]

Polytetrafluoroethylene (PTFE) is a synthetic Fluoropolymer they have been known to be chemically inert, low friction properties and has a high heat resistant.

PTFE, also known as Teflon, is renowned for its great electrical characteristics, particularly its low loss tangent and dielectric constant [36]. These characteristics make it an excellent candidate for high-frequency applications, which are common in space communication systems [36]. Furthermore, PTFE is highly resistant to chemicals and outgassing, making it ideal for the hostile space environment [36]. Teflon is also used as a solid lubricant for numerous moving parts in spacecraft because of its resistance to heat and non-stick qualities [37].

From the GRANTA Edupack it has a melting point that ranges from 315-339°C and can operates at its maximum service temperate from 250-270°C which allows the PCB to work in high temperature environments. Because it is chemically inert it is suitable for it to space PCB [38].

Copper Foils

High-performance copper foils are essential in space-grade PCBs to ensure excellent signal integrity and efficient heat dissipation [27]. Copper's excellent electrical and thermal conductivity makes it the ideal material for conductive layers on a PCB. In applications requiring larger current carrying capability, thicker copper traces are frequently used [39]. The quality and thickness of the copper foil are crucial parameters that influence both the

Figure 4.16.: Copper foils [5]

electrical performance and thermal management capabilities of the PCB in the demanding space environment.

Kapton

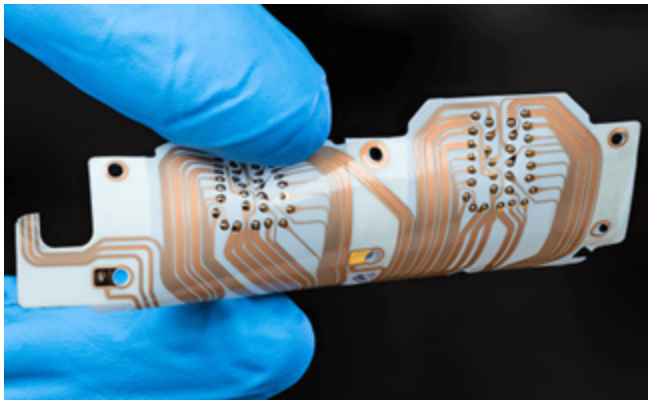


Figure 4.17.: Kapton PCB [6]

Kapton is highly radiation resistant and compatible with harsh conditions. Engineers typically use it to insulate cables and components in high vacuum chambers because it withstands radiation while having minimal influence on instrument base pressure [33]. Kapton, which can withstand temperatures of up to 400°C, can be combined with other materials like gold to create heat-resistant blankets for use in spacecraft. Kapton's flexibility and outstanding performance under harsh temperatures make it an ideal material for a variety of insulation and protection applications in space hardware [33].

Material Selection for PCB

The Comparing the elongation between the polyimide, alumina and PTFE, polyimide has the most ideal value being 75%–80% compared to alumina's 0.07%-0.09% or PTFE's 200%-400%, because having too little elongation can have a major risk of under deformation it will break, and having too much elongation causes the PCB to irreparably warp,

having an in-between ensures it can deform when under forces without breaking. Polyimide has the lowest density which is ideal for having weight limit on the shuttle craft. Polyimide has the highest cost per kg which can be a major restraint on the budget whilst alumina and PTFE are far cheaper in comparison. All 3 have the necessary operating temperature criteria of space temperatures from -200°C to +200°C. From this it is decided that the best material for PCB is polyimide, because it has the required strength, operating temperatures and it has the best middle road on its elongation (%) and flexural modulus making it ideal for space usage.

4.5.5. CubeSat Chassis Material

The standard material that is used in the chassis of the CubeSat are aluminium alloys such as 6061 or 7075, which is to remain lightweight but without sacrificing strength. There's also need to thermal management to it to manage the fluctuations in space, commonly using multi-layer insulation to achieve this. The 4-digit designation represents what type of aluminium it is; the first digit can be 1 to 8 which represents which wrought it is. The second digit is often 0 meaning it is in the base form, any number other than this indicates it has been altered. The third and fourth digits identifies what individual alloys they are. A comparison table was created in GRANTA Edupack to compare all the Aluminium 6061 variants would be the best, and 6061-T651 was found to be the best out of all of them, as seen in the appendix.

Aluminium 6061- T651

Aluminium 6061-T651 has a good strength-to-weight ratio which offers a good balance between the strength and weight without compromising one to a detrimental degree. It's also strong with the tensile strength up to 320 MPa. It is used in many structural applications because it is strong by itself and is lighter than steel making it ideal in the automotive and aerospace industries. It is also highly recyclable and non-toxic so after usage it can be reused and does not cause harm to be in proximity with. It is also not suspectable to stress corrosion cracking making it ideal for space use when external and internal forces are applied towards it.

Aluminium 7075

Aluminium 7075 is one of the strongest aluminium available with a tensile strength listed at 580 MPa making it an ideal material for structural and load-bearing applications. Just like 6061 it is also lightweight with a good strength to weight ratio and highly recyclable. However, it is highly susceptible to stress corrosion cracking which is a major weakness especially for space applications from the external and internal forces applied to it.

6061-T651 vs 7075

In terms of strength Aluminium 7075 is considerably stronger than 6061, this makes 7075 a suitable material for applications requiring high strength such as Construction. However, 6061 is much more flexible with having better ductility making it easier to form weld into shape what the design requires. 6061 also has better resistance in corrosion due to things in space such as atomic oxygen oxidising the metal. And 6061 is cheaper and simpler production process because of its machinability compared to 7075. It is decided that the most suitable for the chassis should be 6061-T651 because of the corrosion resistance, and machinability keeping cost down, with the better ductility to avoid the chassis structure to snap under forces, whilst 7075 is stronger, they already are suitable strength wise in space that it is a moot point.

Emerging Materials and Future Trends

Emerging materials and future trends in space PCB technology are being shaped by advances in materials science and manufacturing techniques that try to solve the unique challenges provided by the space environment. Next-generation materials such as 2D materials, organic electronics, and metamaterials are being investigated for their potential to increase the performance and reliability of space electronics by offering severe temperature resistance, radiation shielding, and enhanced thermal conductivity [40].

Furthermore, hybrid and composite materials, including self-healing polymers and multifunctional carbon fibre composites, are being developed to withstand the extreme conditions of space, such as micrometeorite impacts and electromagnetic interference, thus improving the feasibility and safety of space exploration [41]. All these developments point to a future in which space PCB technology is more integrated, efficient, and capable of meeting the expanding demands of space missions.

4.5.6. Quality Assurance and Reliability Standards in Space PCB Manufacturing

NASA Standards

NASA has comprehensive standards to ensure PCB quality and reliability PCBs used in space missions. NASA-STD-8739.1 defines the workmanship standards for polymeric applications on electrical assemblies, such as conformal coatings used to protect PCBs in defence and aerospace applications [42]. The Goddard Space Flight Centre (GSFC) has released GSFC-STD-8001, which specifies quality assurance standards for the design, procurement, fabrication, and use of high-reliability PCBs in GSFC project mission hardware [43]. GSFC-STD-8002 provides the requirements for the validation of manufacturing processes for printed wiring assemblies using water-soluble fluxes [44]. The NASA PCB

Working Group (PCB WG) is a helpful resource, offering expertise in PCB technology assessment and recommendations for quality assurance measures [36]. NASA generally uses polyimide-based laminates with glass reinforcements in its PCB constructions [45]. The agency also emphasises the necessity of visual acuity testing in accordance with NASA-STD-8739.6 or IPC-QL-653, and PCB inspectors must hold IPC-A-600 Certified IPC Specialist (CIS) accreditation to ensure complete and accurate quality control [43].

ESA Standards

The European Space Agency (ESA) also enforces strict requirements for materials and procedures used in space applications, including PCBs. These standards are extensively described in the European Cooperation for Space Standardisation (ECSS) series. ECSS-Q-ST-70-02C describes the thermal vacuum outgassing test protocols for screening space materials [26]. ECSS-Q-ST-70-10C specifies the requirements for qualified printed circuit boards [46], whereas ECSS-Q-ST-70-60C Corrigendum 1 specifies the requirements for their procurement [47]. Other relevant ECSS standards address a wide range of materials, mechanical parts, and procedures, including cleaning, heat testing, soldering, and crimping [47]. These standards are intended to ensure the reliability and performance of electronic assemblies in the challenging space environment.

IPC Standards

IPC, a global trade group for the electronics industry, has established a number of standards that are commonly used in the production of space-grade PCBs. IPC-2221 is a generic standard that covers practically every area of PCB design [27]. Specific subsection standards are IPC-2222 for rigid boards, IPC-2223 for flex circuits, and IPC-2226 for HDI structures [27]. IPC-6012 specifies the certification and performance standards for rigid PCBs, whereas IPC-6013 accomplishes the same for flexible printed boards [27]. IPC-A-600 specifies the visual inspection standard for PCB acceptability [27]. For surface mount designs, IPC-7351 provides critical guidelines [27].

For military and aerospace applications, IPC Class 3/A, as defined in IPC-6012 Class 3/A, denotes high reliability requirements [48]. Furthermore, the IPC 6012 Space Addendum specifies standards for class 3 boards used in the space and military avionics industries, including requirements to endure vibration, ground testing, and temperature cycling [49].

US Militarty Standards(MIL-SPEC)

The United States Military has developed its own performance criteria for PCBs used in defence and aerospace applications. MIL-PRF-31032 specifies the requirements for high-reliability, stiff PCBs, whereas MIL-PRF-50884 addresses flexible PCBs [26]. MIL-

PRF-55110 covers stiff PCBs used in military applications [50]. These standards set strict requirements on material selection, design, manufacturing processes, and testing to assure PCB dependability in hostile operating environments [50]. Furthermore, standards such as MIL-STD-810 explain environmental engineering issues for military equipment, while MIL-STD-461 establishes requirements for electromagnetic compatibility [48]. Compliance with MIL-SPEC standards is commonly required for PCBs used in military and aerospace systems.

4.6. Thermal Analysis of the PCB

4.6.1. Hypothesis

The purpose of the thermal analysis is to validate that the Polyimide PCB and the components can operate under the space conditions. The operational temperature of Polyimide minimum is $-240\,^{\circ}\mathrm{C}$ and the maximum is $260\,^{\circ}\mathrm{C}$. If the components are within the operational temperature range, then it is validated to be operational under dynamic heat conditions, however if it exceeds it, then the PCB material, and the PCB design with the component locations must be altered and reworked. It is expected that the photodiodes area of the PCB to have the most thermal activity, with the lowest thermal activity occurring to the resisters.

4.6.2. Thermal Analysis Parameters

After a PCB CAD model is designed and material selection is completed, the thermal analysis for the PCB can now be conducted. The PCB CAD model was imported into ANSYS workbench using steady-state thermal analysis and a mesh was applied generated to the model to influences the accuracy and convergence of the simulation. Each off the components have their respective materials applied to the model, such as the photodiode with silicon and aluminium, connectors being made of copper, nickel, zinc, amplifiers and resisters made from aluminium, and the PCB being made of polyimide. Then a convection of $22\,^{\circ}\mathrm{C}$ was applied to the side of the PCB to simulate the transfer of heat of the PCB from the shuttlecraft it is housed in, assumingly room temperature of the spacecraft is the same as on Earth, it would be $22\,^{\circ}\mathrm{C}$. Each of the components have their respective temperatures when operating. Finally, two tests will be conducted using $200\,^{\circ}\mathrm{C}$ and $-200\,^{\circ}\mathrm{C}$ of radiation applied to the entire PCB model to simulate when the PCB is facing to the sun, and when the PCB is in the shadows of celestial bodies respectively.

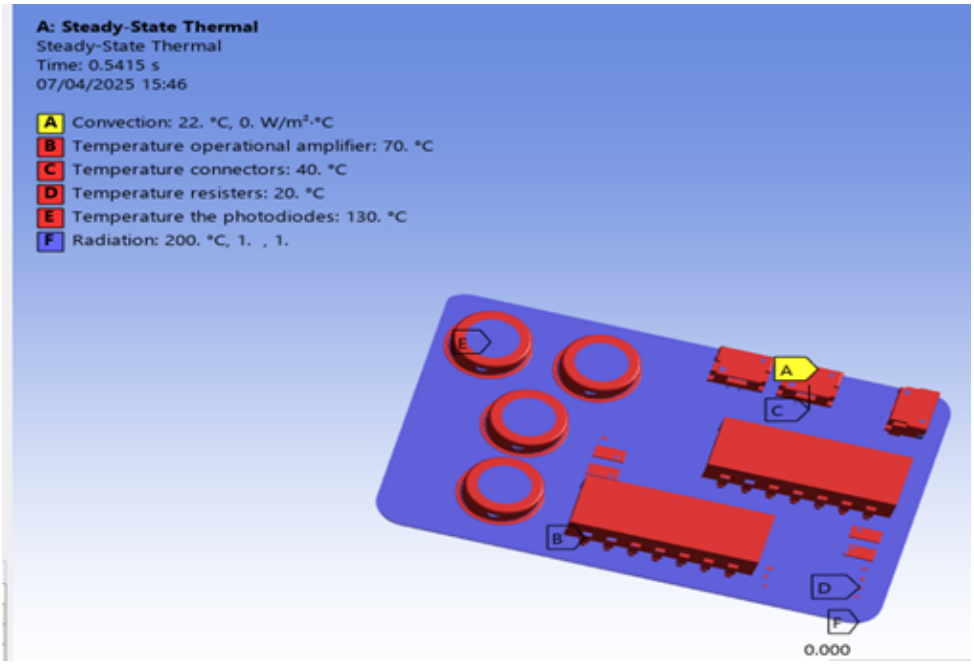


Figure 4.18.: PCB model with Parameters applied

4.6.3. Thermal results

4.6.4. Finite Analysis Evaluation 1 - Under 200 °C in space

The results for the PCB when facing towards the sun depicts the most heat being towards outside of the photodiodes area and the cooler parts are around the other components. As Figure 4.19 shows the maximum temperature experienced is 199.35 °C and the minimum temperature is −31.168 °C, this shows that the PCB can be operable when facing towards the Sun.

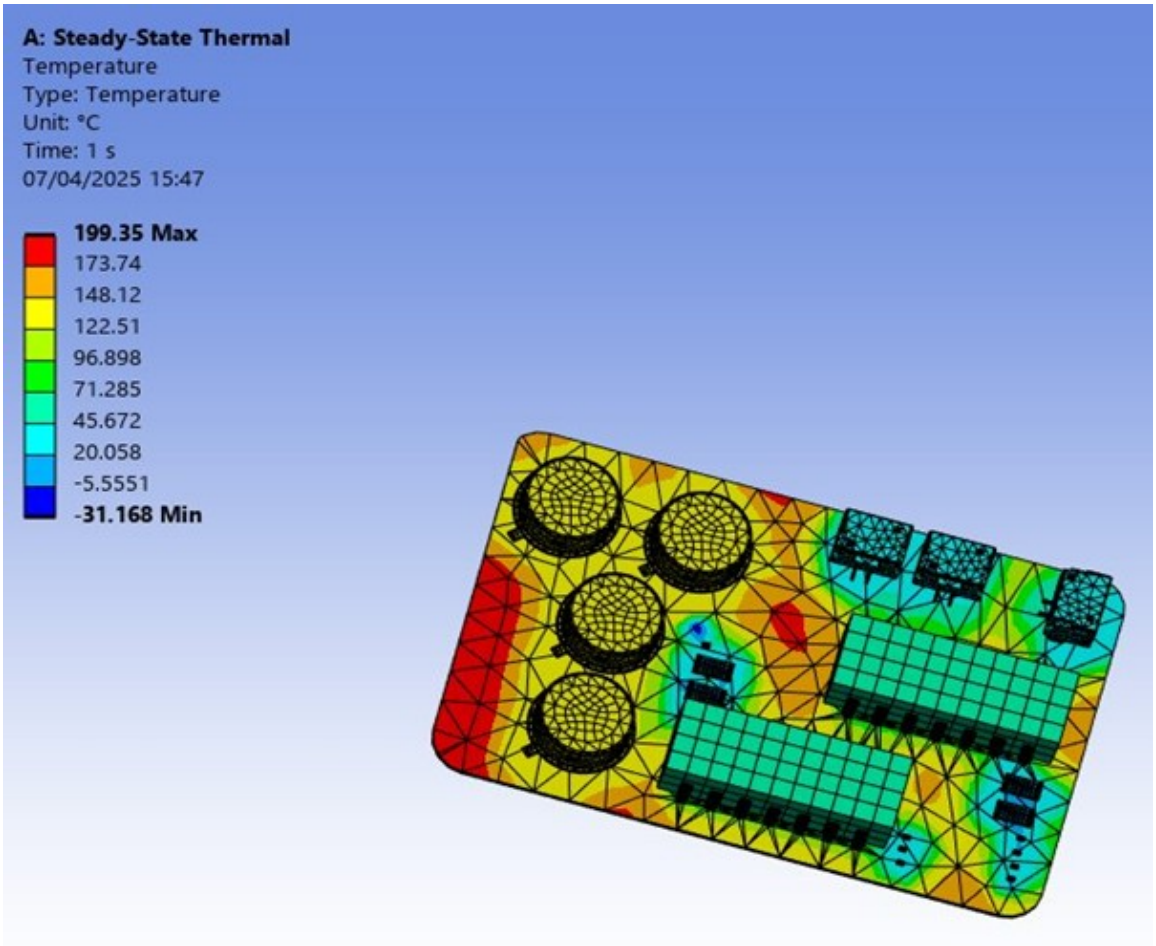


Figure 4.19.: Results under 200 °C radiation

Finite Analysis Evaluation 2 - Under $-200\,^{\circ}\mathrm{C}$ in space

The results for the PCB, when it is in the shadows of a celestial body depicts the most heat coming from the photodiodes themselves which is to be expected as it creates the most heat in the PCB components followed by the amplifiers then connectors. As Figure 4.20 shows the maximum temperature experienced is $136\,^{\circ}\mathrm{C}$ and the minimum temperature is $-173.83\,^{\circ}\mathrm{C}$, this shows that the PCB can be operable when in the shadow of a celestial body.

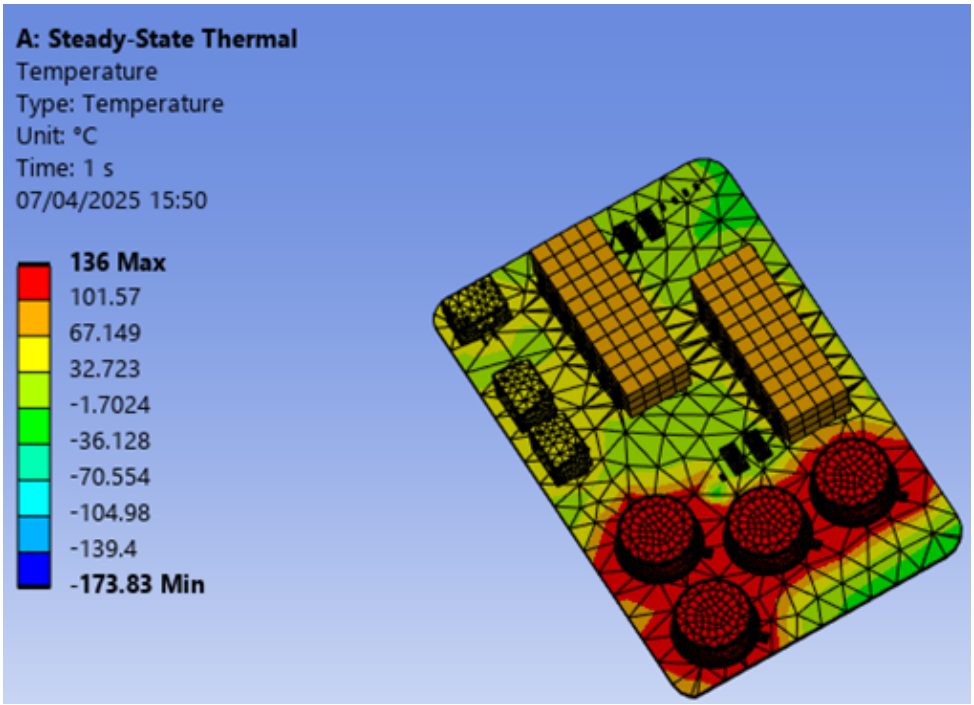


Figure 4.20.: Results under −200 °C radiation

4.6.5. Results Evaluation

The results shown for both under 200 °C and −200 °C radiation shows that the PCB is operable under the harsh thermal conditions in space. That validates the design of the PCB and their component placements, and the material selection. The polyimide having about a clearance when facing the sun at 60 °C and when in the shadows of a celestial object at −66 °C shows that it would not be on the brink of melting or freezing during operation. The next step would be analysis the stresses and forces at work in space, unfortunately, a mechanical analysis of the stress cannot be conducted because it requires further work with the complete work of the entirety of the CubeSat chassis and all the power source such as the solar panels.

4.7. CubeSat Chassis Design

For the purposes of visualization, there were 2 potential designs for the CubeSat chassis. The CubeSat chassis were design dependant on where the PCB would be mounted. The first CubeSat chassis design was taken from GrabCAD.com and an aperture is mounted to this design to show where they would be mounted. The second CubeSat chassis was based on a design found online and recreated as much as possible, however this did not have any measurements given so dimensions were assumed.

4.7.1. CubeSat Chassis Design 1

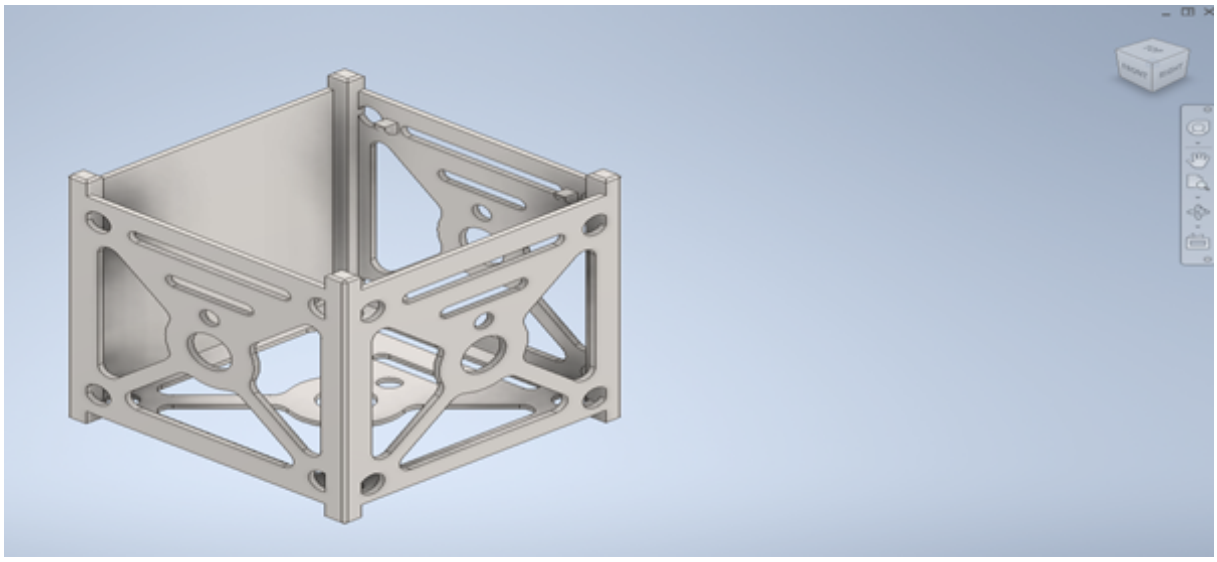


Figure 4.21.: Front view of the first CubeSat chassis design

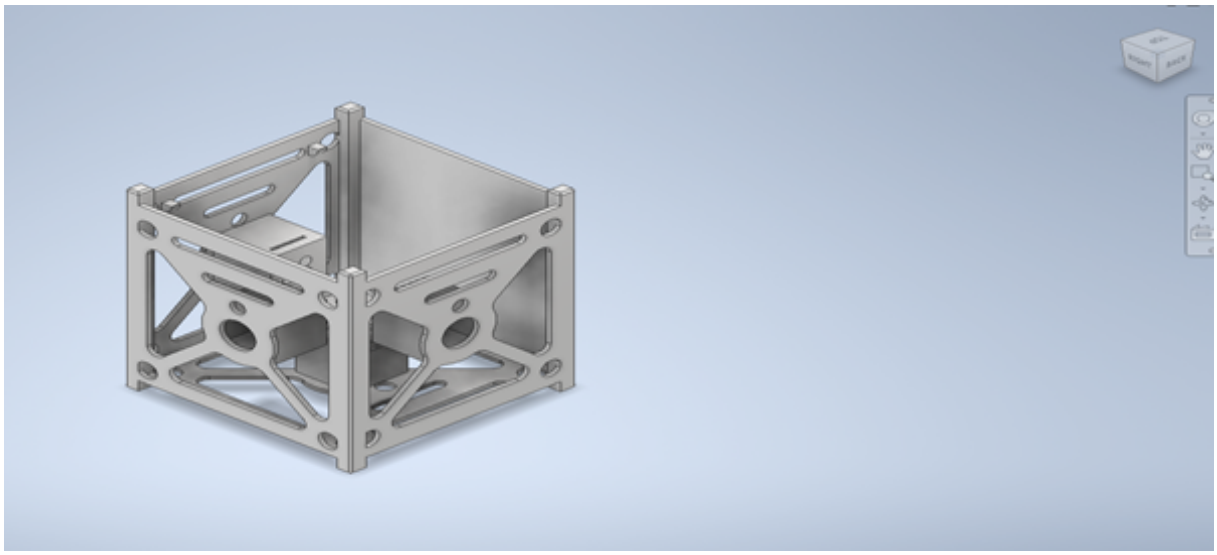


Figure 4.22.: Angular view of the first CubeSat chassis design

The first design taken from `https://grabcad.com/library/cubesat-1u-5` as seen in Figure 4.21 has exposed holes made to allow for easy removal, the circular holes at the centre are to let the light to expose to the aperture where the PCB photodiodes are located. The software used to show these are AutoCAD Inventor, with a different angle and the aperture shown in Figure 4.22.

4.7.2. CubeSat Chassis Design 2

The second hypothetical design is a much more basic skeleton frame made to house the aperture as seen below in Figure 4.23 and Figure 4.24.

Figure 4.23.: Front view of 1U Cubesat Skeleton Chassis

Figure 4.24.: Angular view of 1U Cubesat Skeleton Chassis

This was recreated in inventor with rough estimation on the thickness of the PCBs and skeletons itself and the apertures. The PCBs are mounted on top of each other compared to being the aperture housing the PCB mounted on each side of the CubeSat, the PCB cannot be more than 96×96 mm. As seen below in Figure 4.25 and Figure 4.26, are the CubeSat Chassis with and without the PCBs, additional angles and drawings are found in the appendix.

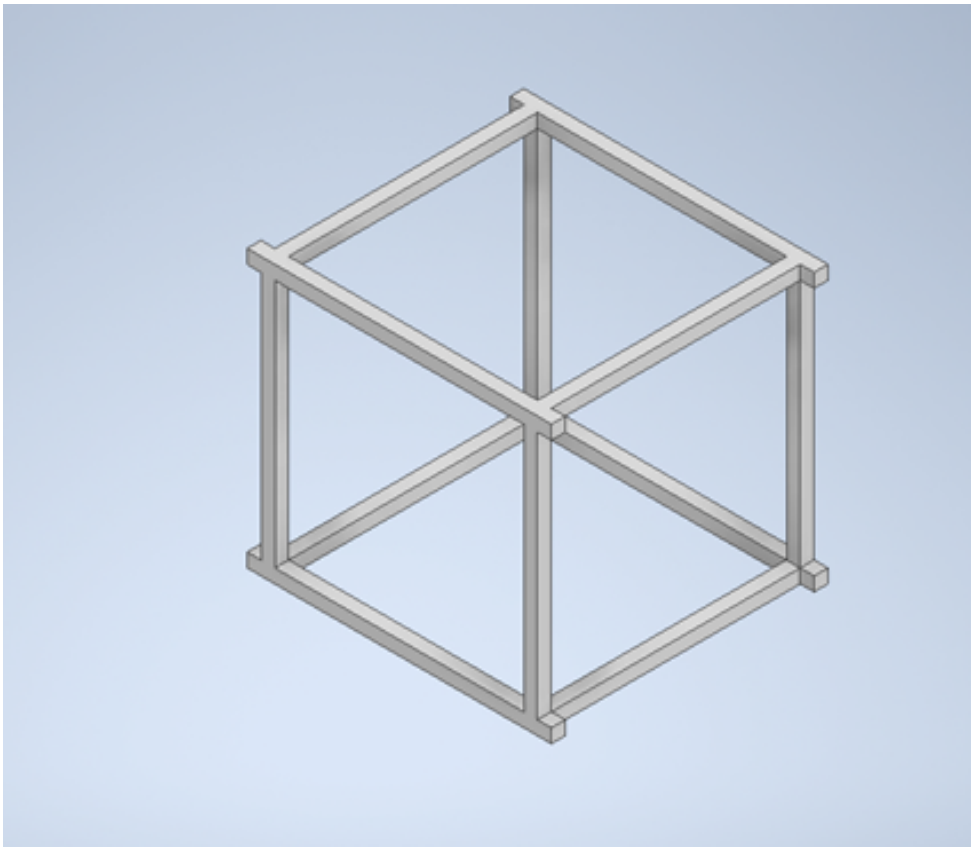


Figure 4.25.: Second chassis design without PCBs

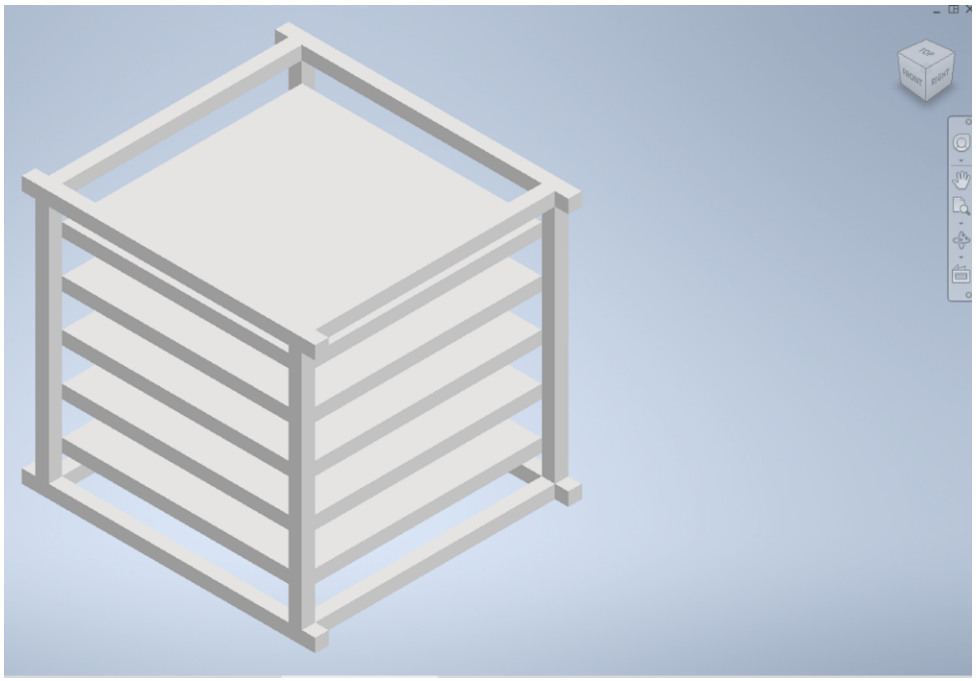


Figure 4.26.: Chassis with the aperture

5. Results

5.1. Sensor Characterization

TO FINISHTO FINISHTO FINISHTO FINISHTO FINISHTO FINISHTO FINISH focus on the fundamental properties and performance of your photodiodes themselves, distinct from the other subsections. Here are some key elements that would belong specifically under SensorCharacterization:

Basic Photodiode Electrical Characteristics:

Dark current measurements Junction capacitance I-V characteristics in different lighting conditions Spectral response profiles (sensitivity vs. wavelength)

Individual Sensor Benchmarking:

Performance comparison between the 4 photodiodes (matching/differences) Responsivity measurements (A/W) Quantum efficiency calculations Detection threshold levels

SNR!

Response Linearity:

Measurements showing linear range of the photodiodes Saturation point characterization Recovery time from saturation

Temperature Dependency:

Performance drift with temperature Baseline shift measurements Temperature compensation data

Aging/Stability Tests:

Long-term drift measurements Repeatability of measurements over time

This section should focus on the inherent properties of the photodiodes themselves - essentially providing the baseline characterization data that underpins all the other analysis. The other sections then build on this foundation by examining how these sensors perform when integrated into the complete system with amplification, angular positioning, enclosure effects, etc.

5.2. Amplification Performance

This section provides results of the amplifier performance.

5.3. Photodiode Angular Response

This section discusses the results of the response of the solar sensor to angular changes of the light source.

5.4. Enclosure Effectiveness

This section discusses the effectiveness of the Photodiode enlosure.

5.5. Data Acquisition System Evaluation

This section provides results related to the Arduino DAQ.

5.6. System Performance Analysis

5.6.1. Operational Constraints Identified

5.6.2. Environmental Factors Impact

5.6.3. System Stability and Repeatability

5.6.4. Recommendations for Improvement

5.7. Comparative Analysis

This section compares the simulation with the prototype results.

5.7.1. Breadboard vs. Stripboard Results

5.7.2. Iteration Improvements Analysis

5.7.3. Performance Against Design Requirements

The performance ...

5.7.4. Design Evolution Assessment

The what now?

5.8. System Limitations And Considerations

This section discusses the limitations and future work.

5.8.1. Renewable Energy Demonstrator (RED) testbench limitations and impact

The Renewable Energy Demonstrator (RED) device borrowed from the EPS team [20] proved to be valuable as a testbench for our project, but several limitations affected our ability to gather comprehensive data for accurate sun vector determination.

Servo Motor Limitations

There were at least two issues with the Servo Motors which reduced our ability to gether data:

- Inablity to reach the target angles requested in code without phisical intervention, presumably due to insufficient torque or gearing issues.
- Control deadband did not allow us to gather a large enough sample (small-angle adjustment) to create a LUT that would allow the prediction of light position by interpolation.

When attempting to move from 0° to 90°, for example, the servomotors would often stall, requiring manual assistance to “help” the arch reach the desired position. This introduced inconsistency in our testing methodology and required manual adjustment using protractors.

Signal Interference Issues

The RED’s Power Supply interference made the signal very noisy. However due to the nature of the high frequency noise and low frequency signal, it meant that it was relativly easy to filter using the scipy.signal library [23], detailed in Section 4.2.3. Further, arch position changing without pressing the button likely due to noise in the RED from the Power Supply made testing slighlty more annoying, but it was not happening often enough to disrupt testing too much, it was considered an acceptable quirk.

Impact on Look-Up Table development

The most significant impact to the project caused by these issues with the RED are that we were unable to generate a LUT or a "Sensor Response Surface" plot using the prototype, as the angle precision from the arch was not good enough. Originally, our plan included:

- Taking measurements at fine angular increments (every 5°) across the hemisphere
- Mapping between sensor readings and light source position

- Using said mapping to develop interpolation algorithms for positions between measured points
- Validating position determination accuracy across the sensor's field of view

The compbination of these limitations of the RED testbench made it impractical to gather the comprehensive dataset required to fulfill these goals, as withouht a reliable dataset across the entire FOV it is not possible to predict the location of light. This limitations ultimatly directly affected the ability to detect the light position, and we could only take some readings and compare them to the simulated environment.

5.8.2. Aperture placement accuracy

6. Conclusions

7. Future Work

Mention: methods to avoid detecting sun reflections off the moon and earth (such as light intensity or light source width if possible).

Unfortunately, there was no other easily available method of fabricating the apertures, such as having the apertures printed on glass with a fully opaque ink. Another method considered and attempted was to 3D print the apertures, but this would have resulted in the apertures having a thickness that would have changed the way the light enters, depending on the angle of the light.

Bibliography

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signals and Systems*. India: Pearson, 2013.

[2] 86PCB, "Fr-4 pcb: Material properties, thickness selection, and application guide - 86pcb," Jul 19 2024. [Online]. Available: https://86pcb.com/blog/fr-4-pcb-material-properties-thickness-selection-and-application-guide/

[3] CERCuits, "Aluminum oxide pcb (alumina)," 2025. [Online]. Available: https://ceramic-pcb.com/aluminum-oxide-pcb-al2o3/

[4] Imimg, "Teflon/ptfe sheets," 2025. [Online]. Available: https://3.imimg.com/data3/JP/YY/MY-1996077/teflon-ptfe-sheets.jpg

[5] V. Solutions, "Copper foil," 2022. [Online]. Available: https://viin-solution.com/images/newimage/Copper-Foil2.jpg

[6] Goodfellow, "Kapton," 2025. [Online]. Available: https://www.goodfellow.com/media/wysiwyg/kapton-image-2_1_.png

[7] J. Puig-Suari and C. Turner, "Development of the standard cubesat deployer and a cubesat class picosatellite," 2001.

[8] K. S. Balaji, B. S. Anand, P. M. Reddy, V. C. B, M. D. P. Lingam, and V. K, "Studies on attitude determination and control system for 1u nanosatellite," *ICCPCT*, pp. 616–625, 2023.

[9] I. Lopez-Calle and A. I. Franco, "Comparison of cubesat and microsat catastrophic failures in function of radiation and debris impact risk," *Scientific Reports*, vol. 13, no. 1, -01-07 2023.

[10] W. Guanghui, P. Shum, X. Guoliang, and Z. Xuping, "Position detection improvement of position sensitive detector (psd) by using analog and digital signal processing," in *2007 6th International Conference on Information, Communications and Signal Processing*, 2007, pp. 1–4, iD: 1.

[11] P. Ortega, G. López-Rodríguez, J. Ricart, M. Domínguez, L. M. Castañer, J. M. Quero, C. L. Tarrida, J. García, M. Reina, A. Gras, and M. Angulo, "A miniaturized two axis sun sensor for attitude control of nano-satellites," p. 1623, -10 2010.

[12] S. Dwik* and N. Somasundaram, "Modeling and simulation of two-dimensional position sensitive detector (psd) sensor," p. 744, -11-30 2019.

[13] F. J. Delgado, J. M. Quero, J. García, C. L. Tarrida, J. M. Moreno, A. G. Sáez, and P. Ortega, "Sensosol: Multifov 4-quadrant high precision sun sensor for satellite attitude control," in *2013 Spanish Conference on Electron Devices*, 2013, pp. 123–126, iD: 1.

[14] A. Ali, K. Ullah, H. U. Rehman, I. Bari, and L. M. Reyneri, "Thermal characterisation analysis and modelling techniques for cubesat-sized spacecrafts," *The Aeronautical Journal*, vol. 121, no. 1246, p. 1858, -10-17 2017.

[15] A. Raslan, G. Michna, and M. Ciarcià, "Thermal simulation of a cubesat," Research paper, 2019. [Online]. Available: https://www.researchgate.net/publication/335123882_Thermal_Simulation_of_a_CubeSat

[16] A. Dhariwal, N. Singh, and A. K. Kushwaha, "Structural analysis of 1u cubesat designed for low earth orbit missions," in *2023 International Conference on Innovations in Communication, Automation and Technology (ICICAT)*. IEEE, 06 2023, p. 1.

[17] F. et. al., "Analysis of received power characteristics of commercial photodiodes in indoor los channel visible light communication," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 7, pp. 164–172, Nov 7 2017. [Online]. Available: https://doi.org/10.14569/IJACSA.2017.080722

[18] N. A. Fortune, "A short guide to using python for data analysis in experimental physics," Authorea Preprints, July 2021. [Online]. Available: https://www.authorea.com/doi/full/10.22541/au.152961025.52816543?commit=338281975198a5f5599acd5c0cb0535a1155bea5

[19] G. Koukis and V. Tsaoussidis, "Satellite-assisted disrupted communications: Iot case study," *Electronics*, vol. 13, no. 1, -12-20 2023.

[20] N. I. Shopov, S. Hannisse, and S. Gupta, "Renewable energy demonstrator (red)," Glasgow Caledonian University, Tech. Rep., 2022.

[21] G. Keiser, *Fiber Optic Communications*, 1st ed. Singapore: Springer, 2021.

[22] Atmel, "Atmega328p 8-bit avr microcontroller with 32k bytes in-system programmable flash datasheet," 2015. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[23] T. S. community, "butter scipy v1.15.2 manual." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html#scipy.signal.butter

[24] ——, "filtfilt — scipy v1.15.2 manual," 2025. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.filtfilt.html

[25] G. L. G. Burbui, U. Reggiani, and L. Sandrolini, "Prediction of low-frequency electromagnetic interferences from smps," *ISEMC*, vol. 2, pp. 472–477, 2006.

[26] W. Leverett, "Satellite technology: Pcbs in space exploration and communications | abl circuits," Feb 5 2024. [Online]. Available: https://www.ablcircuits.co.uk/blog/pcb-satellite-exploration-communication/

[27] C. D. Systems, "Applying ipc pcb design standards | cadence," 2025. [Online]. Available: https://resources.pcb.cadence.com/blog/applying-ipc-pcb-design-standards-cadence

[28] Proto-Electronics, "Space-grade pcbs: challenges in designing electronics for extreme environments," Jan 11 2024. [Online]. Available: https://youtu.be/ZgLfodLIgcI

[29] 911EDA, "Space pcb design challenges: Engineering for the cosmos," 2024. [Online]. Available: https://www.911eda.com/articles/space-pcb-design-challenges/

[30] Gatema, "Challenges in pcb design for space applications | gatema," Jul 17 2024. [Online]. Available: https://www.gatemapcb.com/challenges-in-pcb-design-for-space-applications/

[31] P. Directory, "Understanding pcb design techniques for space-qualified applications. - pcb directory," Apr 9 2025. [Online]. Available: https://www.pcbdirectory.com/community/understanding-pcb-design-techniques-for-space-qualified-applications

[32] F. Circuits, "Pcbs for space: Ensuring reliability in space applications," Feb 21 2025. [Online]. Available: https://www.fscircuits.com/pcbs-for-space/

[33] M. E. Tech, "Making a material difference in aerospace and defense electronics," December 1 2024. [Online]. Available: https://www.mobilityengineeringtech.com/component/content/article/52137-making-a-material-difference-in-aerospace-defense-electronics

[34] Mascera-Tec, "Advantages and disadvantages of alumina ceramic," Website, n.d., accessed: 15 April 2025. [Online]. Available: https://www.mascera-tec.com/news/advantages-and-disadvantages-of-alumina-ceramic

[35] V. Ceramics, "Advantages and disadvantages of alumina ceramics," Website, n.d., accessed: 15 April 2025. [Online]. Available: https://vhandy.com/pros-and-cons-of-alumina-ceramics/

[36] C. D. Systems, "Designing reliable pcbs for space applications," 2025. [Online]. Available: https://resources.pcb.cadence.com/blog/designing-reliable-pcbs-for-space-applications

[37] F. Lou, "How to select satellite components that can withstand space," Feb 5 2023. [Online]. Available: https://revolutionized.com/satellite-components/

[38] R. PCB, "Introduction to ptfe teflon," LinkedIn post, September 14 2023. [Online]. Available: https://www.linkedin.com/pulse/what-ptfe-teflon-material-pcb-rayming-techonloy

[39] S. international, "Design guidelines for military and aerospace pcbs," May 1 2022. [Online]. Available: https://www.mobilityengineeringtech.com/component/content/article/45799-design-guidelines-for-military-and-aerospace-pcbs

[40] A. E. Ona-Olapo, T. D. Iluyomade, T. M. Olatunde, and O. P. Igbinenikaro, "Next-generation materials for space electronics: A conceptual review," *Open Access Research Journal of Engineering and Technology*, vol. 6, no. 2, pp. 51–62, Apr 30 2024. [Online]. Available: https://oarjpublication.com/journals/oarjet/sites/default/files/OARJET-2024-0020.pdf

[41] J. C. Ince, M. Peerzada, L. D. Mathews, A. R. Pai, A. Al-Qatatsheh, S. Abbasi, Y. Yin, N. Hameed, A. R. Duffy, A. K. Lau, and N. V. Salim, "Overview of emerging hybrid and composite materials for space applications," *Advanced Composites and Hybrid Materials*, vol. 6, no. 4, -06-26 2023.

[42] S. C. Systems, "Nasa inspection criteria for conformal coating," Mar 10 2021. [Online]. Available: https://scscoatings.com/newsroom/blog/nasa-inspection-criteria-for-conformal-coating/

[43] G. S. F. Centre, "Standard quality assurance requirements for printed circuit boards (gsfc-std-8001)," Nov 21 2019. [Online]. Available: https://standards.nasa.gov/standard/GSFC/GSFC-STD-8001

[44] ——, "Standard quality assurance for use of water soluble flux(gsfc-std-8002)," Mar 09 2015. [Online]. Available: https://standards.nasa.gov/sites/default/files/standards/GSFC/Baseline/0/GSFC-STD-8002.pdf

[45] C. D. Systems, "Designing reliable pcbs for space applications," 2025. [Online]. Available: https://resources.pcb.cadence.com/blog/designing-reliable-pcbs-for-space-applications

[46] ECSS, "Ecss-q-st-70-02c," Nov 15 2008. [Online]. Available: https://ecss.nl/standard/ecss-q-st-70-02c-thermal-vacuum-outgassing-test-for-the-screening-of-space-materials/

[47] ——, "List of published ecss standards (long) | european cooperation for space standardization," 2019. [Online]. Available: https://ecss.nl/list-of-published-ecss-standards-long/

[48] S. international, "Design guidelines for military and aerospace pcbs," May 1 2022. [Online]. Available: https://www.mobilityengineeringtech.com/component/content/article/45799-design-guidelines-for-military-and-aerospace-pcbs

[49] R. Shashikanth, "5 military grade pcb design rules | sierra circuits," Jan 5 2021. [Online]. Available: https://www.protoexpress.com/blog/military-grade-pcb-design-rules-considerations/

[50] C. D. Systems, "Pcb design for military and aerospace applications," Oct 9 2024. [Online]. Available: https://resources.pcb.cadence.com/blog/2024-pcb-design-for-military-aerospace-applications

A. Appendix - DAQ Code

A.1. Arduino DAQ full code

A.1.1. Arduino C++ Code

```
/*
* Reads 4 analog inputs (0-5V) for recording_dur seconds
* Streams data to PC while recording
*/
// change to true to print debug messages on Serial Monitor
// printing debug lines impacts transmission speed and messes csv
//  do not leave on during measurements!
bool debug = false;

const int analogInputs[] = {A0, A1, A2, A3};


// set up the global varialbes
unsigned long start_time;
const unsigned long recording_dur = 5000; // 25 seconds in
milliseconds (ENSURE python code is greater than this)
unsigned long last_sample_time = 0;
const unsigned long min_samp_interval = 2; // Sample every 2ms (
adjust for stability)
bool recording = false;
int sample_count = 0;

void setup() {
  // serial communication at 115200 bps
  Serial.begin(115200);

  // Set pins as input
  for (int i = 0; i < 4; i++) {
    pinMode(analogInputs[i], INPUT);
  }

  // Optimize ADC for faster sampling
  // Set ADC prescaler to 16 (default is 128)
  //
```

```
33      // Bit: 7:enable; 6: initiate a convertion 5:
34      //
35      ADCSRA = (ADCSRA & 0xF8) | 0x04;
36
37      // Wait for serial connection to establish
38      delay(1000);
39
40      // Send ready message
41      Serial.println("ARDUINO_DAQ_READY");
42    }
43
44    void loop() {
45      // Check if we received a command
46      if (Serial.available() > 0) {
47        String command = Serial.readStringUntil('\n');
48        command.trim();
49
50        if (command == "START") {
51          if(debug) Serial.println("received START command");
52
53          // Clear any remaining data in serial buffer
54          while (Serial.available()) {
55            Serial.read();
56          }
57
58          // Reset sample counter
59          sample_count = 0;
60
61          // Send header once
62          Serial.println("Sample,Time(ms),A0(V),A1(V),A2(V),A3(V)");
63
64          // Start recording
65          recording = true;
66          start_time = millis();
67          last_sample_time = start_time;
68
69          // Send confirmation
70          Serial.println("RECORDING_STARTED");
71        }
72      }
73
74      // If we're recording, collect and send data immediately
75      if (recording) {
76        if(debug) Serial.println("Recording!");
77        unsigned long currentTime = millis();
78        unsigned long elapsed_time = currentTime - start_time;
79
```

```
80        // Check if we're still within the recording period
81        if (elapsed_time <= recording_dur) {
82          if(debug) Serial.println("elapsed time << duration");
83          // Only sample at the specified interval
84          if (currentTime - last_sample_time >= min_samp_interval) {
85            last_sample_time = currentTime;
86
87            // Increment sample counter
88            sample_count++;
89
90            // Start building the output string
91            String data_string = String(sample_count) + "," + String(
    elapsed_time);
92
93            // Multiplex through the four inputs sequentially
94            for (int i = 0; i < 4; i++) {
95              if(debug) Serial.println("reading input: " + String(i));
96              int raw_value = analogRead(analogInputs[i]);
97              float voltage = raw_value * (5.0 / 1023.0);
98              data_string += "," + String(voltage, 3);
99            }
100
101           // Send the complete data string at once
102           Serial.println(data_string);
103         }
104       }
105       else {
106         // End of recording
107         recording = false;
108
109         // Send notification that recording is complete
110         Serial.println("RECORDING_COMPLETE");
111         Serial.print("SAMPLES_COLLECTED:");
112         Serial.println(sample_count);
113         Serial.println("END_OF_DATA");
114       }
115     }
116   }
```

Listing A.1: C++ Code on Arduino

A.1.2. PC-side Python Serial Receive Script

```
1   import serial
2   import time
3   import matplotlib.pyplot as plt
4   import pandas as pd
```

```
    import os
    import numpy as np
    from scipy import signal

    def apply_lowpass_filter(data, fs):
        """Apply a 4-pole low-pass Butterworth filter with 5Hz cutoff"""
        cutoff_freq = 2.0
        filter_order = 4

        nyquist = 0.5 * fs
        normal_cutoff = cutoff_freq / nyquist
        # analog=False implies  bilinear Transformation
        b, a = signal.butter(filter_order, normal_cutoff, btype='low',
    analog=False)
        filtered_data = signal.filtfilt(b, a, data)

        return filtered_data

    # Load data from CSV, apply a 4-pole low-pass filter, and save the
    filtered data
    def filter_and_save_data(filename):

        # Read the CSV data to pandas DataFrame
        #  It knows what are the column names
        df = pd.read_csv(filename)

        # Clean the dataframe - convert all columns to numeric
        for col in df.columns:                    # v - write NaN where it
     can't convert to number (in teh data, not column names)
            df[col] = pd.to_numeric(df[col], errors='coerce')

        # remove rows with NaN (not a number)
        df = df.dropna()

        # Samples not at exact distance from each other
        # Calculate the sampling frequency (median of differences)
        # numpy.diff to get the difference between samples
        time_diffs = np.diff(df['Time(ms)'])
        # numpy.median to get the median
        median_time_diff = np.median(time_diffs)  # in milliseconds
        fs = 1000.0 / median_time_diff  # Convert to Hz

        # Filter each analog channel:
        # ID column head for each channel
        analog_channels = ['A0(V)', 'A1(V)', 'A2(V)', 'A3(V)']
        # take each channel one at a time
        for channel in analog_channels:
```

```
            # if name maches a column name
            if channel in df.columns:
                # add a new column _filtered, and send the array containing all the raw values to
                # have them filtered, and save them in the new _filtered column
                df[f"{channel}_filtered"] = apply_lowpass_filter(df[channel].values, fs)

        # Save the pandas Dataframe with filtered columns to a new CSV file
        filtered_filename = f"{os.path.splitext(filename)[0]}_filtered.csv"
        df.to_csv(filtered_filename, index=False)

        return filtered_filename

    #
    # Plot the DAQ data with original and filtered signals overlapped
    #
    def plot_data(filename):

        # Read the CSV data with pandas
        df = pd.read_csv(filename)

        # Initialize an empty list to store our analog channel names
        analog_channels = []

        # Look through all column names in the DataFrame
        for col in df.columns:
            # the column name starts with 'A'
            if col.startswith('A'):
                #  the column name ends with '(V)'
                if col.endswith('(V)'):
                    # the column name does NOT contain '_filtered'
                    if '_filtered' not in col:
                        # add this column name to our list
                        analog_channels.append(col)

        # Create color cycle for different channels
        colors = ['blue', 'green', 'red', 'purple']

        # Create a single plot with all channels overlapping
        plt.figure(figsize=(14, 8))

        # Plot original data (semi-transparent)
        for i, channel in enumerate(analog_channels):
```

```
            color = colors[i % len(colors)]
            plt.plot(df['Time(ms)'], df[channel], label=f'{channel} Original',
                     linewidth=1.5, alpha=0.4, color=color, linestyle='-')

        # Plot filtered data (solid lines)
        for i, channel in enumerate(analog_channels):
            filtered_channel = f"{channel}_filtered"
            if filtered_channel in df.columns:
                color = colors[i % len(colors)]
                plt.plot(df['Time(ms)'], df[filtered_channel], label=f'{channel} Filtered',
                         linewidth=2.5, color=color, linestyle='-')

        # Set the y-axis range from 0 to 5V
        plt.ylim(0, 5)

        # Add labels and title
        plt.xlabel('Time (ms)')
        plt.ylabel('Voltage (V)')
        plt.title('Arduino DAQ - 4-Channel Readings with 4-Pole 5Hz Low-Pass Filter')
        plt.legend()
        plt.grid(True)

        # Add data summary
        duration = df['Time(ms)'].max() - df['Time(ms)'].min()
        sample_count = len(df)
        sample_rate = sample_count/(duration/1000) if duration > 0 else 0

        info_text = f"Data summary:\n" \
                    f"Duration: {duration:.1f} ms\n" \
                    f"Samples: {sample_count}\n" \
                    f"Sample rate: {sample_rate:.1f} Hz\n" \
                    f"Filter: 4-pole Butterworth, 5Hz cutoff"

        plt.figtext(0.02, 0.02, info_text, fontsize=10,
                    bbox=dict(facecolor='white', alpha=0.8))

        # Save the plot
        plot_filename = f"{os.path.splitext(filename)[0]}_plot.png"
        plt.savefig(plot_filename, dpi=300, bbox_inches='tight')

        # Show the plot
        plt.tight_layout()
```

```
133         plt.show()
134 
135     def main():
136 
137         # ASk if to plot or measure
138         what_to_do = int(input("Record new measurement (1) or plot
    existing (2): "))
139 
140         # User selected to plot old csv
141         if(what_to_do == 2):
142             plot_data(input("Insert the name of the csv: "))
143 
144 
145         # User selected to record new measurement
146         elif(what_to_do == 1):
147             # Use a default port (COM3 for Windows, modify as needed)
148             default_port = "COM3"  # Change to match your system
149 
150             print(f"Using  port: {default_port}")
151 
152             # Configure serial port
153             try:
154                 ser = serial.Serial(default_port, 115200, timeout=2)
155                 print("Connected to Arduino!")
156             except serial.SerialException:
157                 print(f"Error: Could not open port {default_port}")
158                 print("Please modify the default_port variable in the
    script.")
159                 return
160 
161             time.sleep(2)  # Wait for Arduino to reset
162 
163             # Flush buffers
164             ser.reset_input_buffer()
165             ser.reset_output_buffer()
166 
167             # Wait for Arduino ready
168             print("Waiting for Arduino to be ready...")
169             ready = False
170             timeout = time.time() + 10 # don't wait too long
171 
172             while not ready and time.time() < timeout:
173                 line = ser.readline().decode('utf-8', errors='ignore').
    strip()
174                 if line == "ARDUINO_DAQ_READY":
175                     ready = True
176                     print("Arduino is ready!")
```

```
177
178             if not ready:
179                 print("Timed out waiting for Arduino. Make sure it's
    properly connected.")
180                 ser.close()
181                 return
182
183             # Create a filename for this recording session
184             filename = f"arduino_daq_data_{time.strftime('%Y%m%d_%H%M%S
    ')}.csv"
185
186             print(f"Starting data recording to {filename}...")
187             print("Press Ctrl+C to stop if needed.")
188
189             with open(filename, 'w', newline='') as file:
190                 # Send start command
191                 ser.write(b"START\n")
192
193                 recording = True
194                 data_lines = 0
195
196                 # Start time for timeout
197                 start_time = time.time()
198                 timeout_duration = 15  # seconds
199
200                 while recording and (time.time() - start_time) <
    timeout_duration:
201                     if ser.in_waiting:
202                         line = ser.readline().decode('utf-8', errors='
    ignore').strip()
203
204                         if "RECORDING_COMPLETE" in line:
205                             recording = False
206                             print("Recording complete!")
207                         elif "END_OF_DATA" in line:
208                             pass
209                         elif line:
210                             # Write the line to the file
211                             file.write(line + '\n')
212                             data_lines += 1
213
214                             # Show progress occasionally
215                             if data_lines % 100 == 0:
216                                 print(f"Recorded {data_lines} data
    points...")
217
218             # Close the serial port
```

```
            if ser.is_open:
                ser.close()
                print("Serial port closed.")

            print(f"Recorded {data_lines} lines of data.")

            # Process the data
            print("Applying filters to data...")
            filtered_filename = filter_and_save_data(filename)
            print(f"Filtered data saved to {filtered_filename}")

            print("Generating plot...")
            plot_data(filtered_filename)
            print("Done!")

        else:
            print("Wrong Choice. Goodbye!")
            exit()
    if __name__ == "__main__":
        try:
            main()
        except KeyboardInterrupt:
            print("\nProgram terminated by user.")
```

Listing A.2: Python Serial Receive Script

B. Appendix - Software Model Code

B.1. Section 1 Title

B.1.1. subsectiontitle

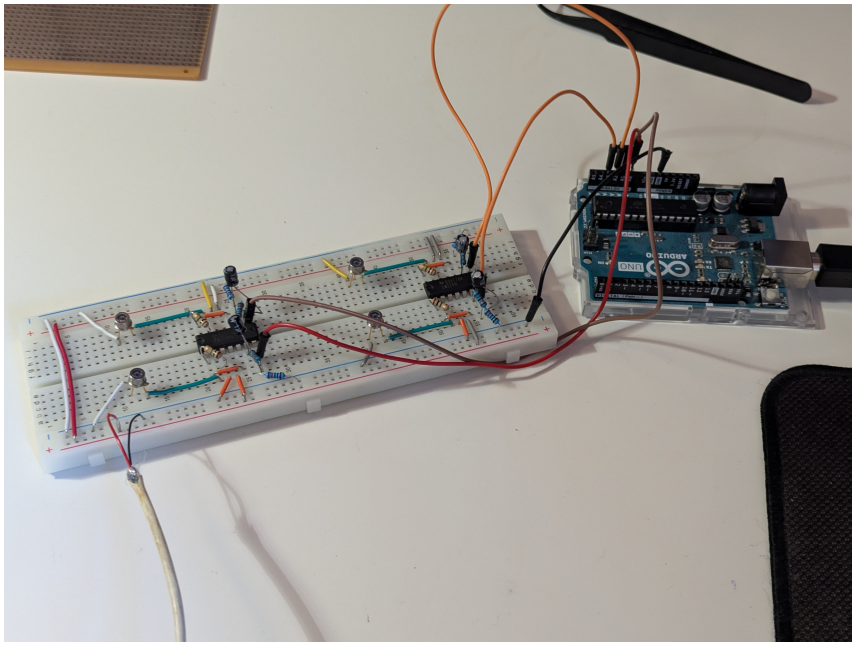
B.2. Section 2 Title

B.2.1. subsectiontitle

C. Appendix - Photos of Lab Work

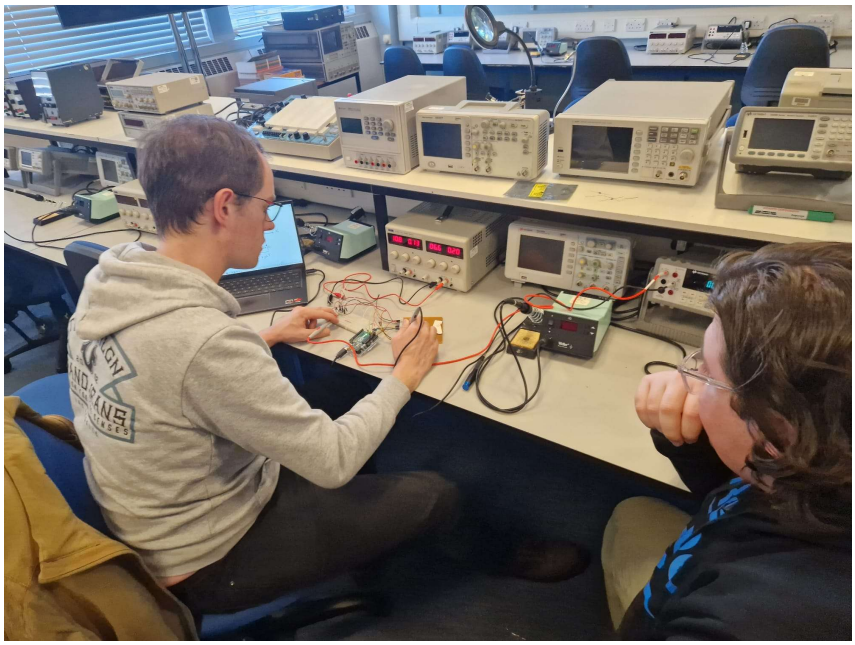
C.1. Prototype Images

C.1.1. BreadBoard Prototype

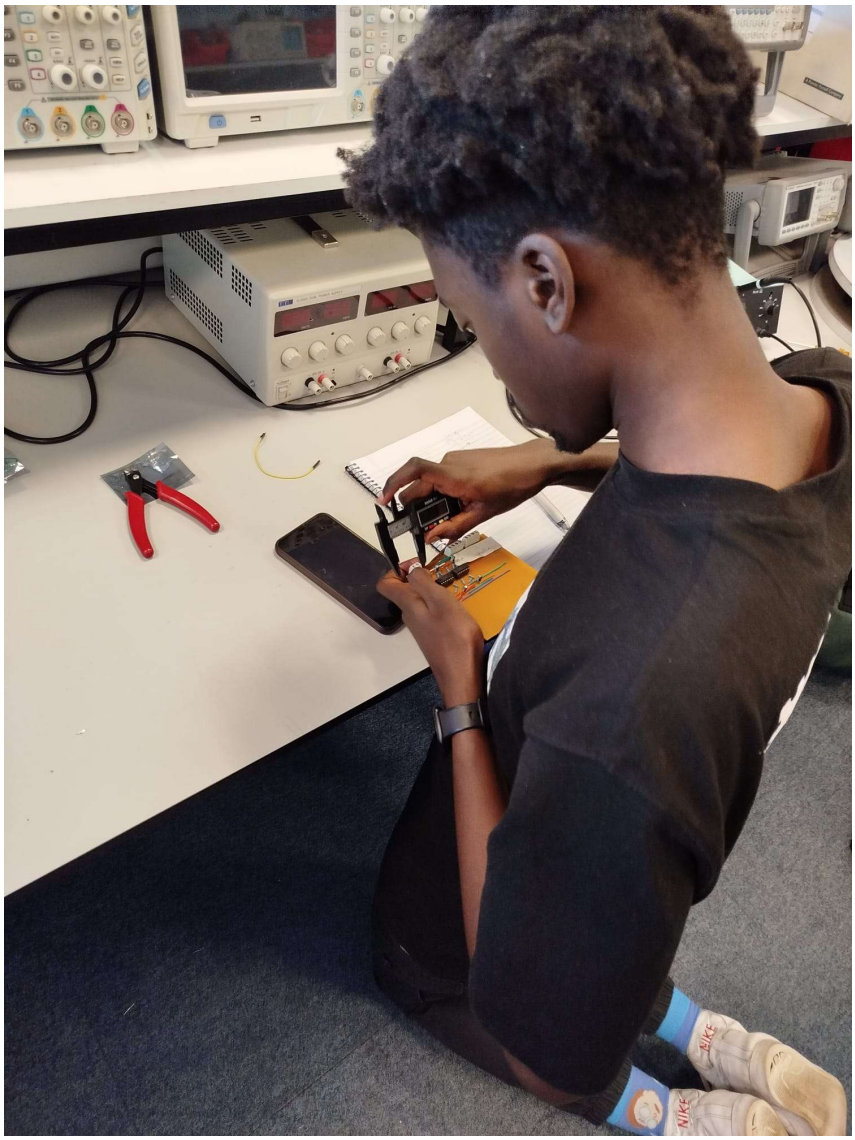


Intial BreadBoard Prototype of the Photodiode Circuit

C.1.2. Building the Prototype

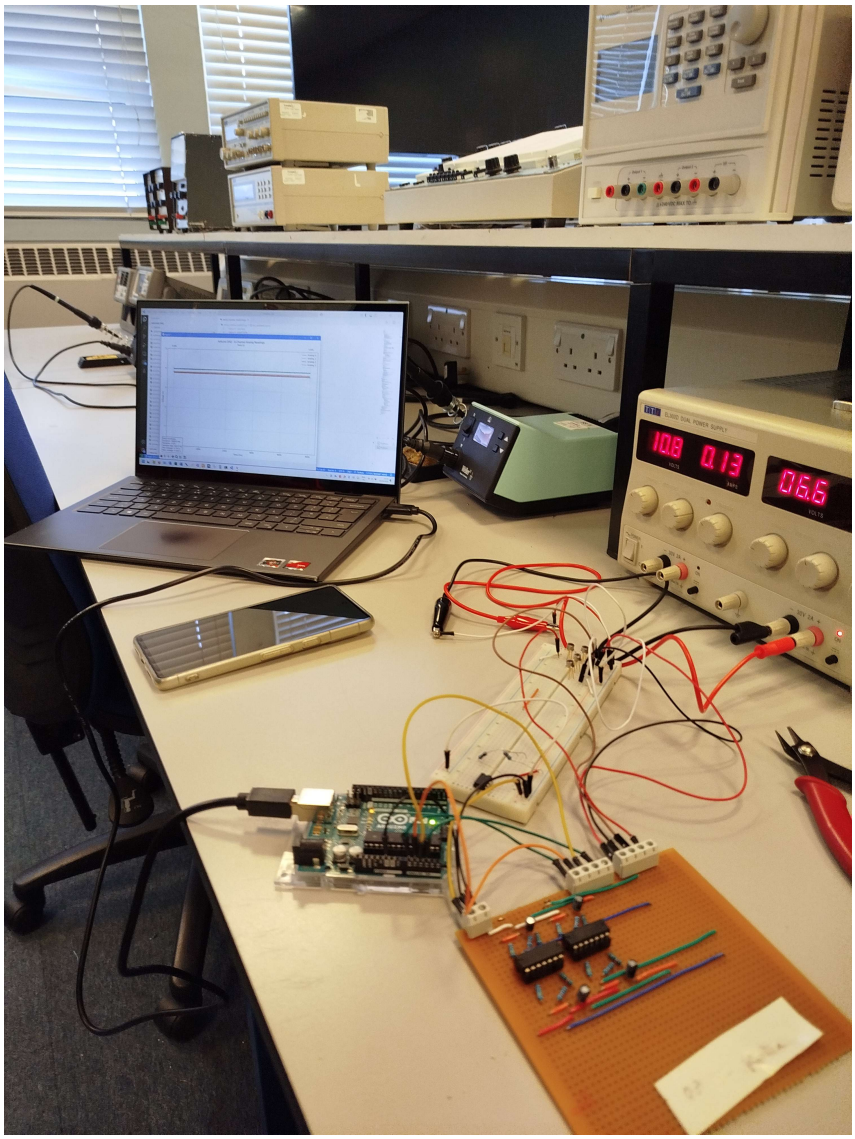


Lab Work for the Prototype (session 1)



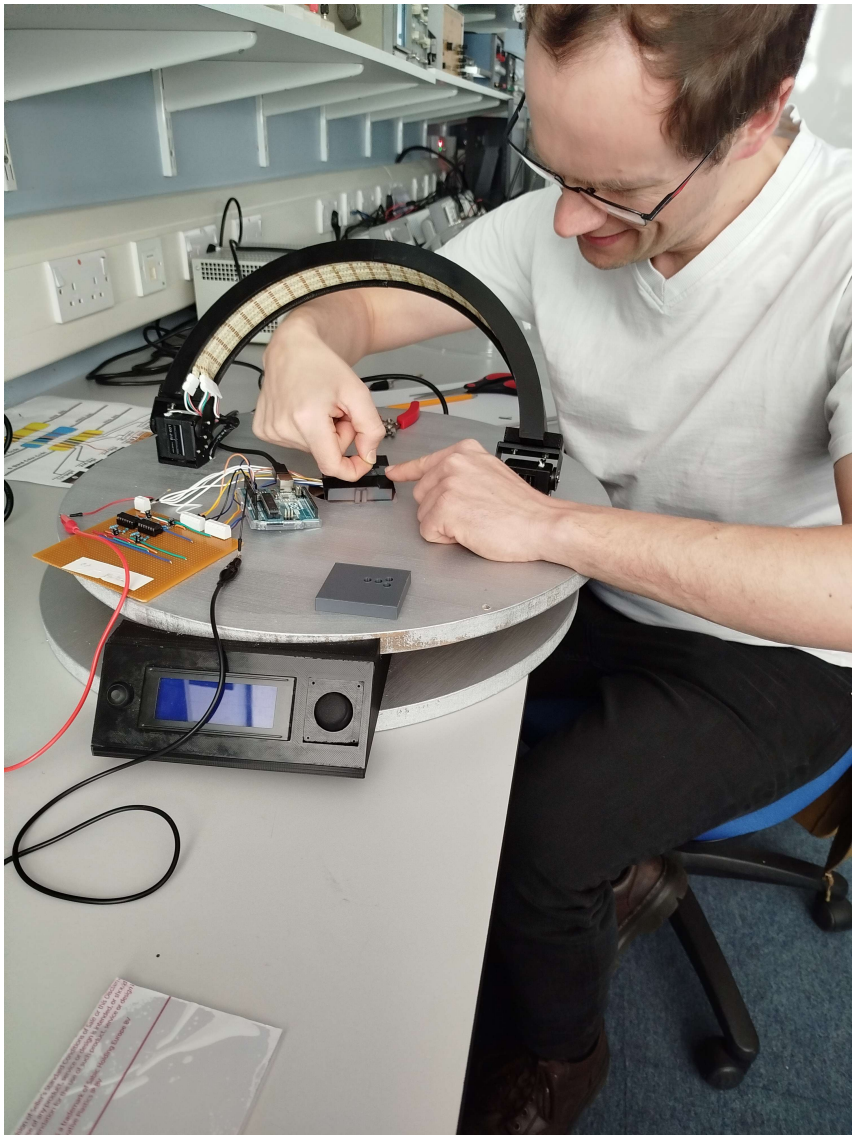
lab Work for the Prototype (session 2)

C.1.3. Prototype Testing



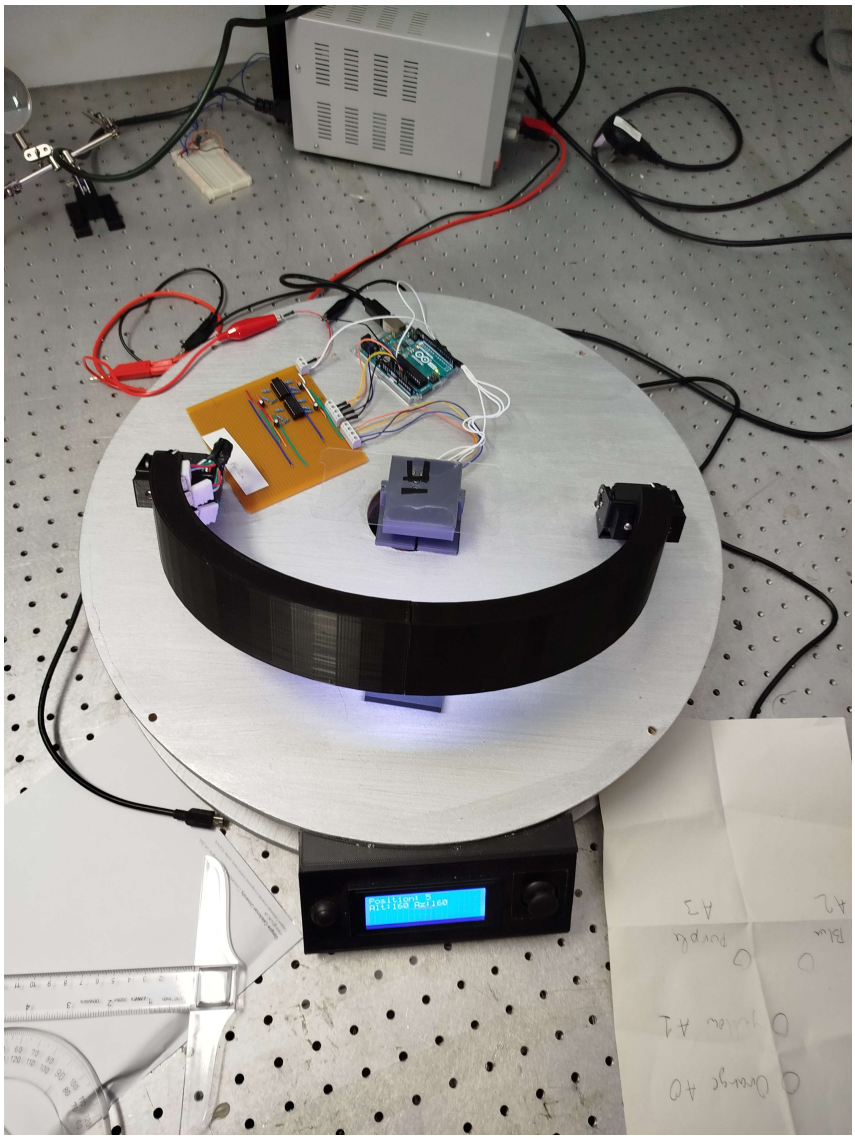
First Lab Test of the Prototype

C.1.4. RED testbench

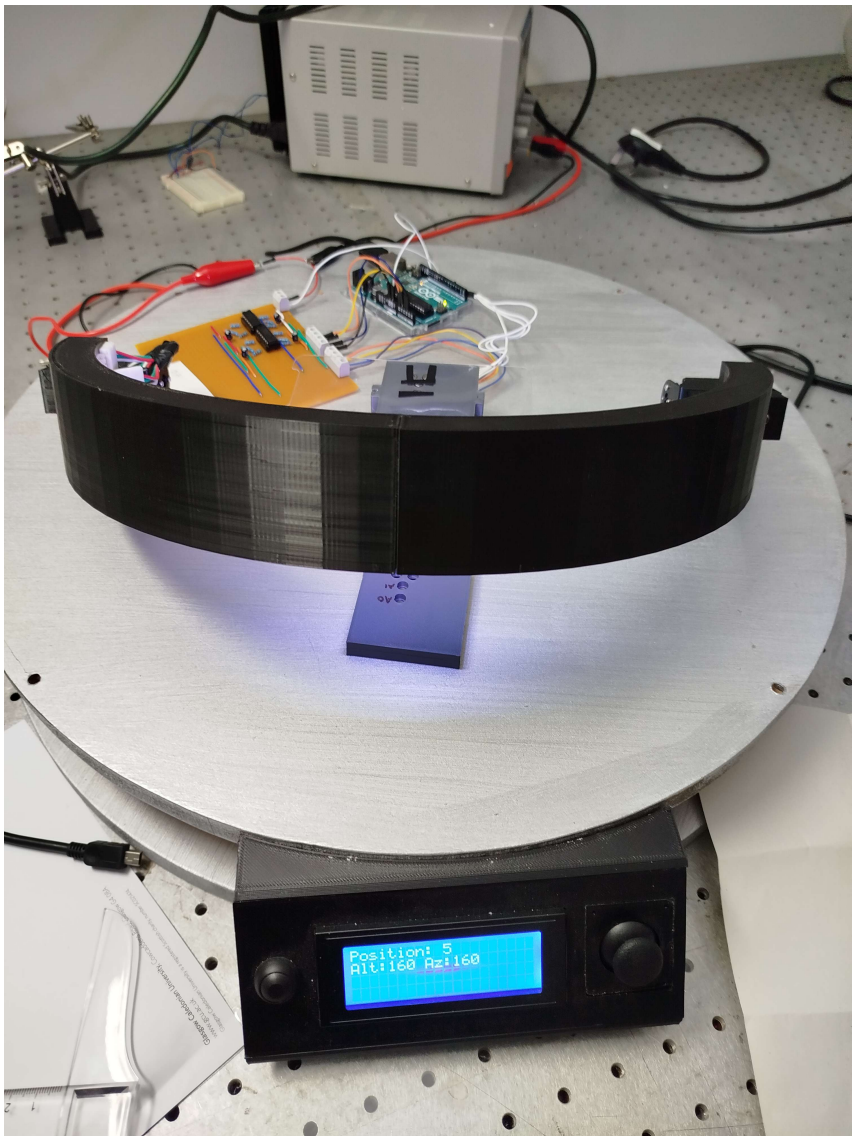


RED Testbench 1

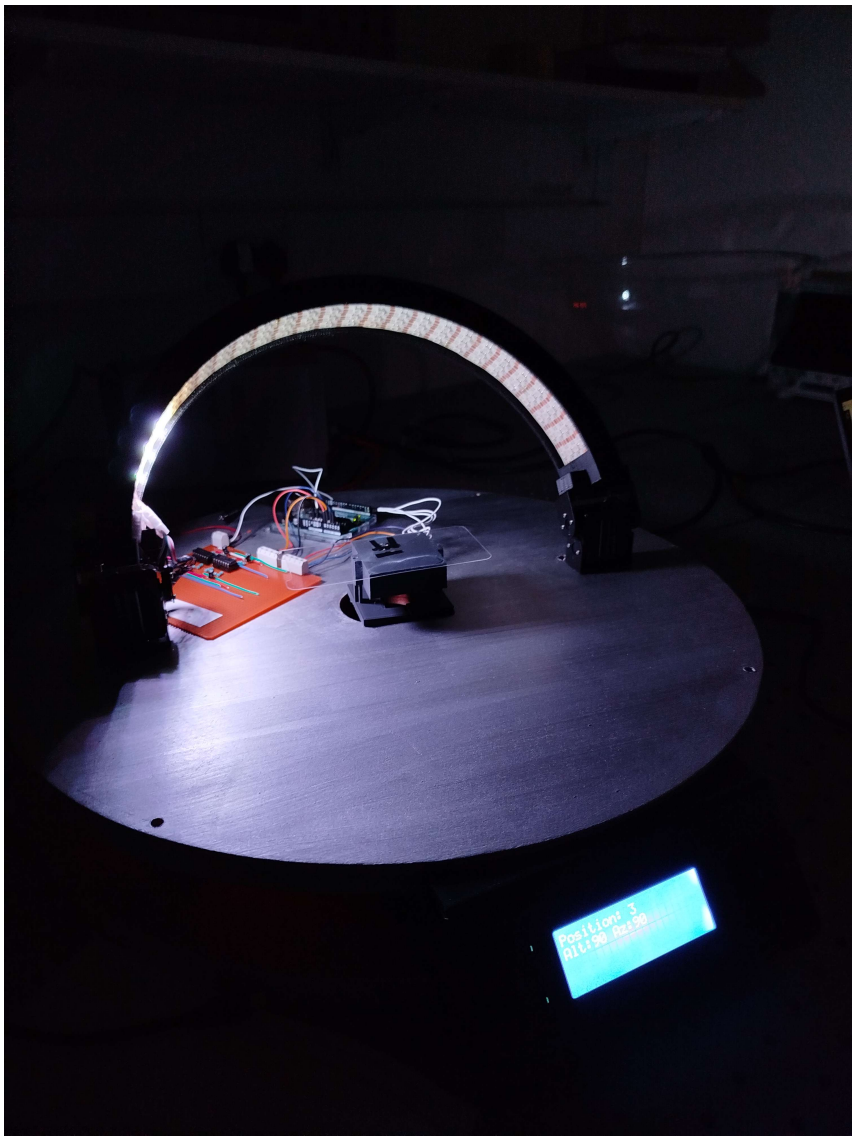
C.1.5. Solar Lab Prototype Testing



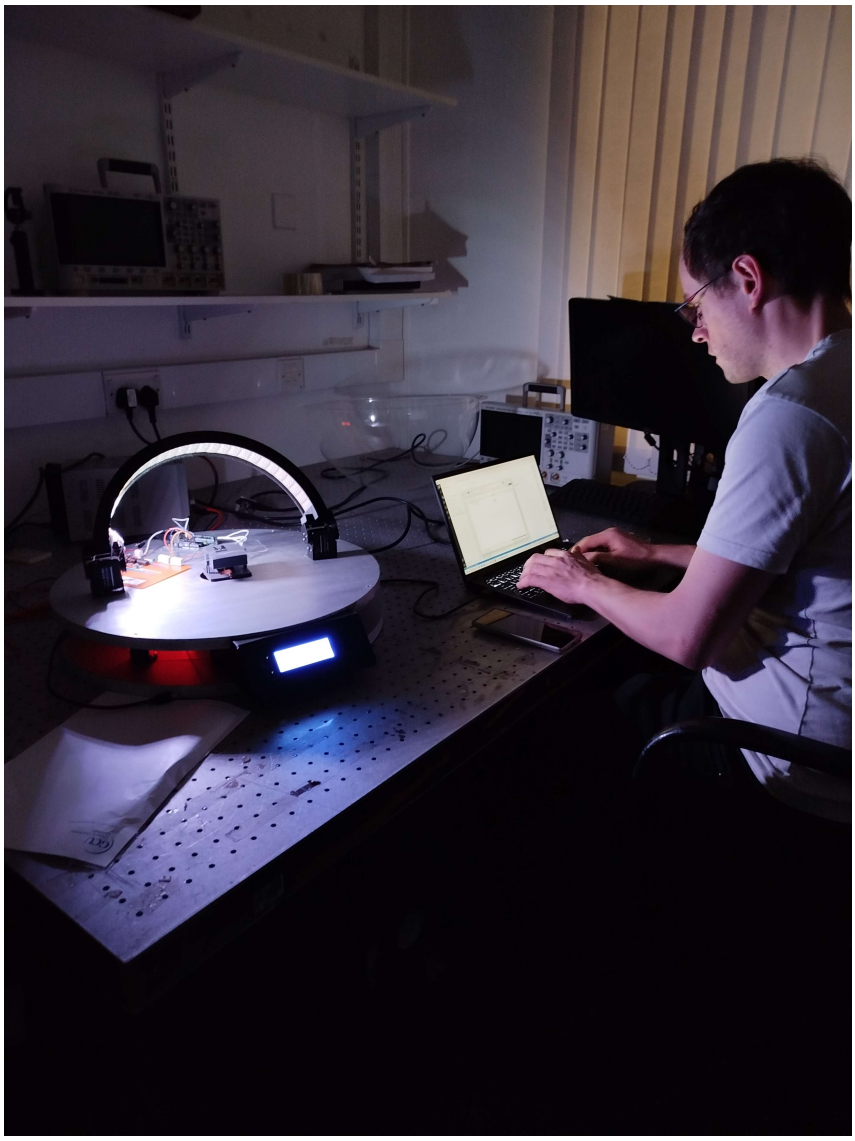
Solar Lab Test of the Prototype 1



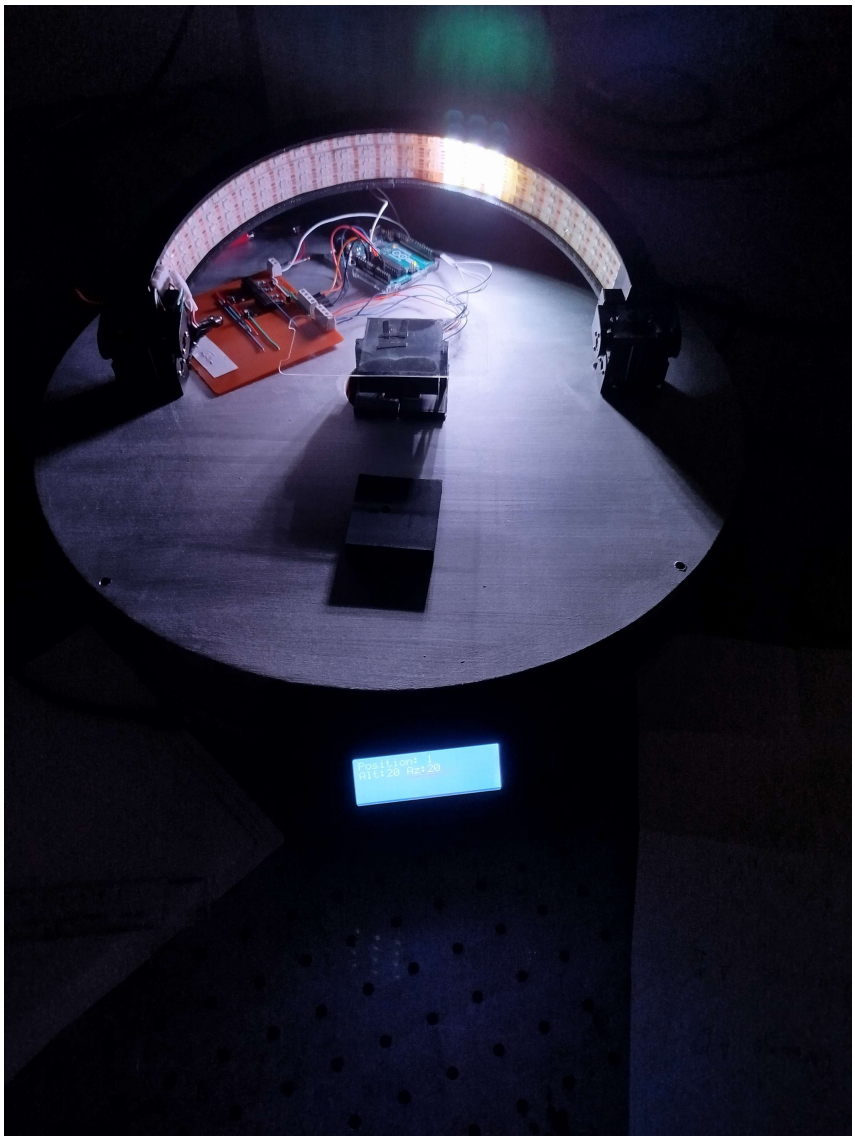
Solar Lab Test of the Prototype 2



Solar Lab Test of the Prototype 3



Solar Lab Test of the Prototype 4



Solar Lab Test of the Prototype 5

D. Appendix - Material Analysis

D.1. Material Analysis - ANSYS

D.1.1. PCB Comparison

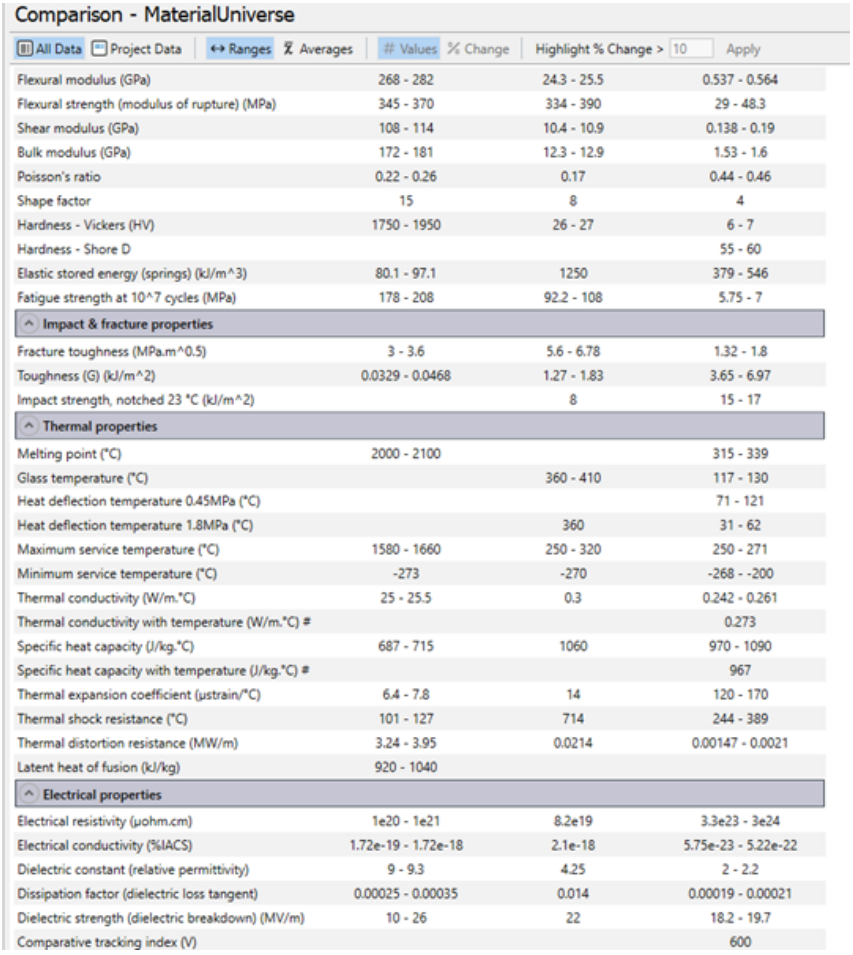
Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | Averages | # Values | % Change | Highlight % Change > 10 | Apply

	Alumina (96%)	Polyimide (PI), PCB laminate	PTFE (unfilled)
General information			
Included in Materials Data for Simulation		✓	✓
Materials Data for Simulation name		PCB laminate, Polyamide (PI)	Plastic, PTFE
Composition overview			
Form	Bulk material	Other	Bulk material
Material family	Ceramic (technical)	Composite (polymer matrix)	Plastic (thermoplastic, semi-crystalline)
Base material	Oxide	PI (Polyimide, aromatic)	PTFE (Polytetrafluoroethylene)
Polymer code		PI	PTFE
Composition detail (metals, ceramics and glasses)			
Al2O3 (alumina) (%)	96	0	0
Other (%)	4	0	0
Composition detail (polymers and natural materials)			
Polymer (%)	0	100	100
Price			
Price (GBP/kg)	11.2 - 16.8	60.8 - 72.1	9.16 - 10.3
Price per unit volume (GBP/m^3)	41400 - 62700	103000 - 123000	19600 - 22600
Physical properties			
Density (kg/m^3)	3690 - 3730	1700	2140 - 2200
Porosity (closed) (%)	0		
Porosity (open) (%)	0		
Mechanical properties			
Young's modulus (GPa)	268 - 282	24.9	0.4 - 0.552
Young's modulus with temperature (GPa) #			0.42
Specific stiffness (MN.m/kg)	72.3 - 76	14.6	0.184 - 0.255
Yield strength (elastic limit) (MPa)	210 - 231	249	19.7 - 21.7
Tensile strength (MPa)	210 - 231	249	20.7 - 34.5
Specific strength (kN.m/kg)	56.6 - 62.3	146	9.06 - 10
Elongation (% strain)	0.07 - 0.09	75 - 85	200 - 400
Compressive modulus (GPa)		24.3 - 25.5	0.402 - 0.423
Compressive strength (MPa)	2100 - 2310	231 - 269	11.2 - 12.3
Flexural modulus (GPa)	268 - 282	24.3 - 25.5	0.537 - 0.564

Edupack(PCB) part 1

D.1.2. Aluminium Comparison

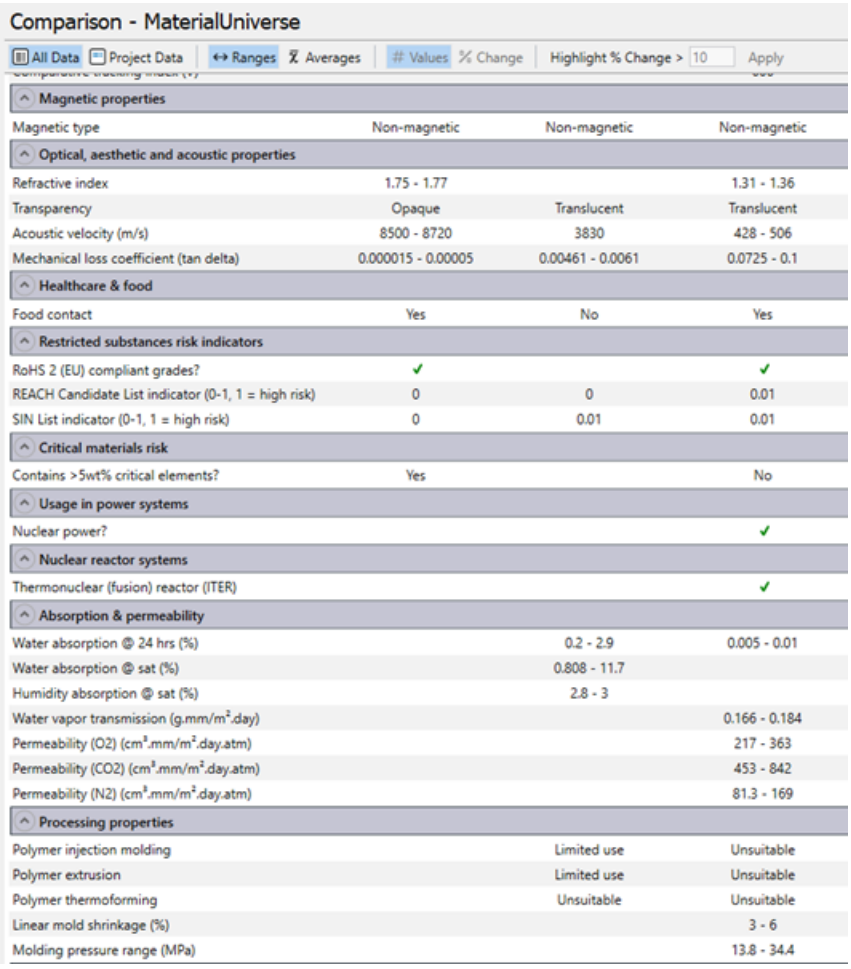


Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | % Averages | # Values | % Change | Highlight % Change > 10 | Apply

Flexural modulus (GPa)	268 - 282	24.3 - 25.5	0.537 - 0.564
Flexural strength (modulus of rupture) (MPa)	345 - 370	334 - 390	29 - 48.3
Shear modulus (GPa)	108 - 114	10.4 - 10.9	0.138 - 0.19
Bulk modulus (GPa)	172 - 181	12.3 - 12.9	1.53 - 1.6
Poisson's ratio	0.22 - 0.26	0.17	0.44 - 0.46
Shape factor	15	8	4
Hardness - Vickers (HV)	1750 - 1950	26 - 27	6 - 7
Hardness - Shore D			55 - 60
Elastic stored energy (springs) (kJ/m^3)	80.1 - 97.1	1250	379 - 546
Fatigue strength at 10^7 cycles (MPa)	178 - 208	92.2 - 108	5.75 - 7
Impact & fracture properties			
Fracture toughness (MPa.m^0.5)	3 - 3.6	5.6 - 6.78	1.32 - 1.8
Toughness (G) (kJ/m^2)	0.0329 - 0.0468	1.27 - 1.83	3.65 - 6.97
Impact strength, notched 23 °C (kJ/m^2)		8	15 - 17
Thermal properties			
Melting point (°C)	2000 - 2100		315 - 339
Glass temperature (°C)		360 - 410	117 - 130
Heat deflection temperature 0.45MPa (°C)			71 - 121
Heat deflection temperature 1.8MPa (°C)		360	31 - 62
Maximum service temperature (°C)	1580 - 1660	250 - 320	250 - 271
Minimum service temperature (°C)	-273	-270	-268 - -200
Thermal conductivity (W/m.°C)	25 - 25.5	0.3	0.242 - 0.261
Thermal conductivity with temperature (W/m.°C) #			0.273
Specific heat capacity (J/kg.°C)	687 - 715	1060	970 - 1090
Specific heat capacity with temperature (J/kg.°C) #			967
Thermal expansion coefficient (µstrain/°C)	6.4 - 7.8	14	120 - 170
Thermal shock resistance (°C)	101 - 127	714	244 - 389
Thermal distortion resistance (MW/m)	3.24 - 3.95	0.0214	0.00147 - 0.0021
Latent heat of fusion (kJ/kg)	920 - 1040		
Electrical properties			
Electrical resistivity (µohm.cm)	1e20 - 1e21	8.2e19	3.3e23 - 3e24
Electrical conductivity (%IACS)	1.72e-19 - 1.72e-18	2.1e-18	5.75e-23 - 5.22e-22
Dielectric constant (relative permittivity)	9 - 9.3	4.25	2 - 2.2
Dissipation factor (dielectric loss tangent)	0.00025 - 0.00035	0.014	0.00019 - 0.00021
Dielectric strength (dielectric breakdown) (MV/m)	10 - 26	22	18.2 - 19.7
Comparative tracking index (V)			600

Edupack(PCB) part 2

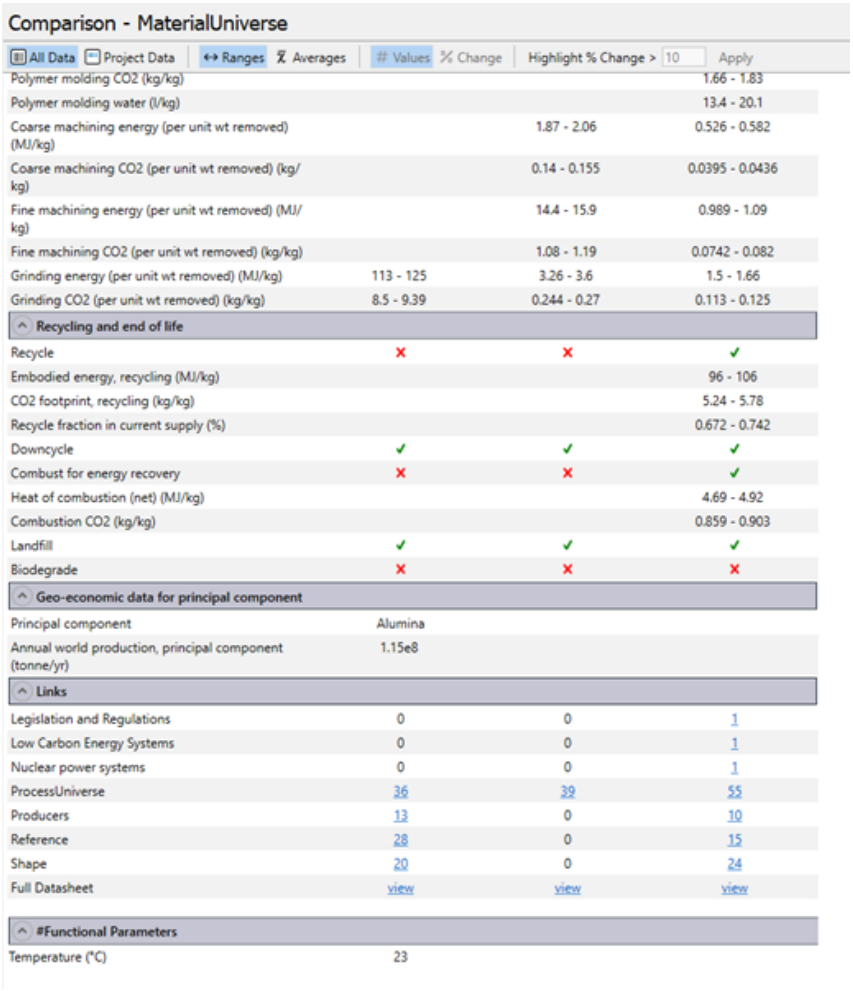


Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | Averages | # Values | % Change | Highlight % Change > 10 | Apply

Magnetic properties			
Magnetic type	Non-magnetic	Non-magnetic	Non-magnetic
Optical, aesthetic and acoustic properties			
Refractive index	1.75 - 1.77		1.31 - 1.36
Transparency	Opaque	Translucent	Translucent
Acoustic velocity (m/s)	8500 - 8720	3830	428 - 506
Mechanical loss coefficient (tan delta)	0.000015 - 0.00005	0.00461 - 0.0061	0.0725 - 0.1
Healthcare & food			
Food contact	Yes	No	Yes
Restricted substances risk indicators			
RoHS 2 (EU) compliant grades?	✓		✓
REACH Candidate List indicator (0-1, 1 = high risk)	0	0	0.01
SIN List indicator (0-1, 1 = high risk)	0	0.01	0.01
Critical materials risk			
Contains >5wt% critical elements?	Yes		No
Usage in power systems			
Nuclear power?			✓
Nuclear reactor systems			
Thermonuclear (fusion) reactor (ITER)			✓
Absorption & permeability			
Water absorption @ 24 hrs (%)		0.2 - 2.9	0.005 - 0.01
Water absorption @ sat (%)		0.808 - 11.7	
Humidity absorption @ sat (%)		2.8 - 3	
Water vapor transmission (g.mm/m^2.day)			0.166 - 0.184
Permeability (O2) (cm^3.mm/m^2.day.atm)			217 - 363
Permeability (CO2) (cm^3.mm/m^2.day.atm)			453 - 842
Permeability (N2) (cm^3.mm/m^2.day.atm)			81.3 - 169
Processing properties			
Polymer injection molding		Limited use	Unsuitable
Polymer extrusion		Limited use	Unsuitable
Polymer thermoforming		Unsuitable	Unsuitable
Linear mold shrinkage (%)			3 - 6
Molding pressure range (MPa)			13.8 - 34.4

Edupack(PCB) part 3

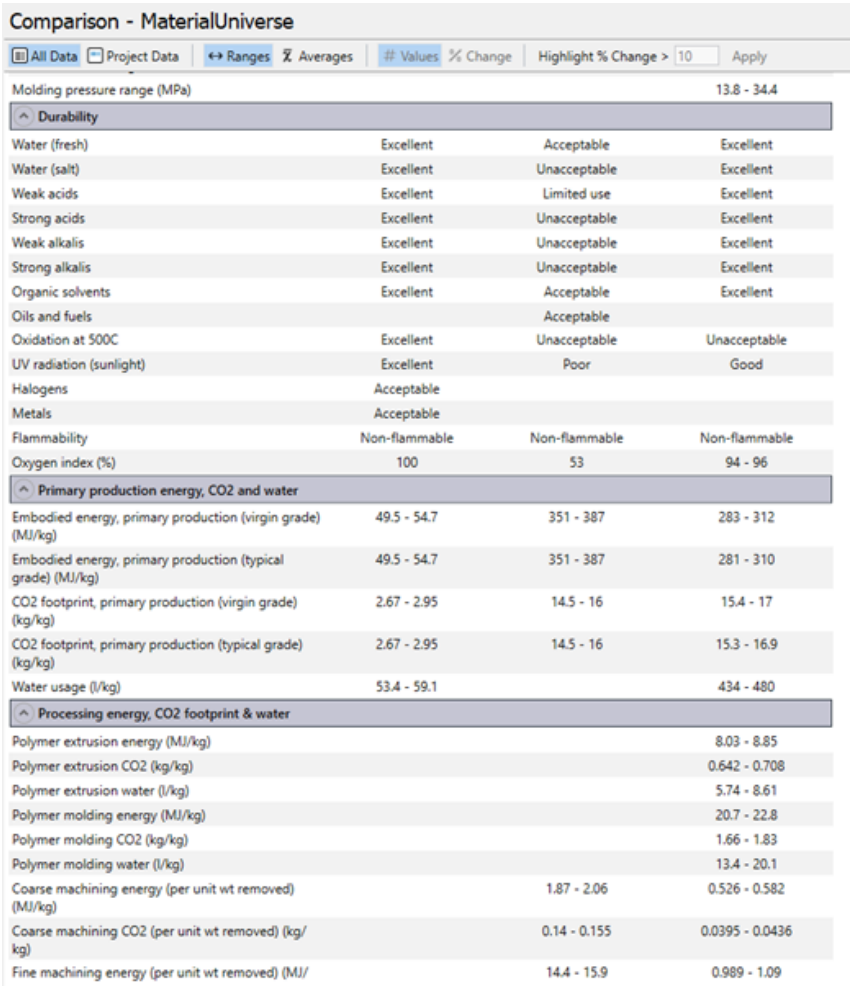


Comparison - MaterialUniverse

All Data | Project Data | Ranges | Averages | # Values | % Change | Highlight % Change > 10 | Apply

Polymer molding CO2 (kg/kg)			1.66 - 1.83
Polymer molding water (l/kg)			13.4 - 20.1
Coarse machining energy (per unit wt removed) (MJ/kg)		1.87 - 2.06	0.526 - 0.582
Coarse machining CO2 (per unit wt removed) (kg/kg)		0.14 - 0.155	0.0395 - 0.0436
Fine machining energy (per unit wt removed) (MJ/kg)		14.4 - 15.9	0.989 - 1.09
Fine machining CO2 (per unit wt removed) (kg/kg)		1.08 - 1.19	0.0742 - 0.082
Grinding energy (per unit wt removed) (MJ/kg)	113 - 125	3.26 - 3.6	1.5 - 1.66
Grinding CO2 (per unit wt removed) (kg/kg)	8.5 - 9.39	0.244 - 0.27	0.113 - 0.125
Recycling and end of life			
Recycle	✗	✗	✓
Embodied energy, recycling (MJ/kg)			96 - 106
CO2 footprint, recycling (kg/kg)			5.24 - 5.78
Recycle fraction in current supply (%)			0.672 - 0.742
Downcycle	✓	✓	✓
Combust for energy recovery	✗	✗	✓
Heat of combustion (net) (MJ/kg)			4.69 - 4.92
Combustion CO2 (kg/kg)			0.859 - 0.903
Landfill	✓	✓	✓
Biodegrade	✗	✗	✗
Geo-economic data for principal component			
Principal component	Alumina		
Annual world production, principal component (tonne/yr)	1.15e8		
Links			
Legislation and Regulations	0	0	1
Low Carbon Energy Systems	0	0	1
Nuclear power systems	0	0	1
ProcessUniverse	36	39	55
Producers	13	0	10
Reference	28	0	15
Shape	20	0	24
Full Datasheet	view	view	view
#Functional Parameters			
Temperature (°C)	23		

Edupack(PCB) part 4



Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | Averages | # Values | % Change | Highlight % Change > 10 | Apply

Molding pressure range (MPa)			13.8 - 34.4
Durability			
Water (fresh)	Excellent	Acceptable	Excellent
Water (salt)	Excellent	Unacceptable	Excellent
Weak acids	Excellent	Limited use	Excellent
Strong acids	Excellent	Unacceptable	Excellent
Weak alkalis	Excellent	Unacceptable	Excellent
Strong alkalis	Excellent	Unacceptable	Excellent
Organic solvents	Excellent	Acceptable	Excellent
Oils and fuels		Acceptable	
Oxidation at 500C	Excellent	Unacceptable	Unacceptable
UV radiation (sunlight)	Excellent	Poor	Good
Halogens	Acceptable		
Metals	Acceptable		
Flammability	Non-flammable	Non-flammable	Non-flammable
Oxygen index (%)	100	53	94 - 96
Primary production energy, CO2 and water			
Embodied energy, primary production (virgin grade) (MJ/kg)	49.5 - 54.7	351 - 387	283 - 312
Embodied energy, primary production (typical grade) (MJ/kg)	49.5 - 54.7	351 - 387	281 - 310
CO2 footprint, primary production (virgin grade) (kg/kg)	2.67 - 2.95	14.5 - 16	15.4 - 17
CO2 footprint, primary production (typical grade) (kg/kg)	2.67 - 2.95	14.5 - 16	15.3 - 16.9
Water usage (l/kg)	53.4 - 59.1		434 - 480
Processing energy, CO2 footprint & water			
Polymer extrusion energy (MJ/kg)			8.03 - 8.85
Polymer extrusion CO2 (kg/kg)			0.642 - 0.708
Polymer extrusion water (l/kg)			5.74 - 8.61
Polymer molding energy (MJ/kg)			20.7 - 22.8
Polymer molding CO2 (kg/kg)			1.66 - 1.83
Polymer molding water (l/kg)			13.4 - 20.1
Coarse machining energy (per unit wt removed) (MJ/kg)		1.87 - 2.06	0.526 - 0.582
Coarse machining CO2 (per unit wt removed) (kg/kg)		0.14 - 0.155	0.0395 - 0.0436
Fine machining energy (per unit wt removed) (MJ/		14.4 - 15.9	0.989 - 1.09

Edupack(PCB) part 5

Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | x̄ Averages | # Values | % Change | Highlight % Change > 10 | Apply

	Aluminum, 6061, T651	Aluminum, 7075, T6
General information		
Condition	T651 (Solution heat-treated and artificially aged)	T6 (Solution heat-treated and artificially aged)
UNS number	A96061	A97075
EN name	EN AW-6061 (EN AW-Al Mg1SiCu)	EN AW-7075 (EN AW-Al Zn5,5MgCu)
EN number	3.3211	3.4365
Included in Materials Data for Simulation	✓	✓
Materials Data for Simulation name	Aluminum alloy, wrought, 6061, T651	Aluminum alloy, wrought, 7075, T6
Composition overview		
Material family	Metal (non-ferrous)	Metal (non-ferrous)
Base material	Al (Aluminum)	Al (Aluminum)
Composition detail (metals, ceramics and glasses)		
Al (aluminum) (%)	95.8 - 98.6	87.2 - 91.4
Cr (chromium) (%)	0.04 - 0.35	0.18 - 0.28
Cu (copper) (%)	0.15 - 0.4	1.2 - 2
Fe (iron) (%)	0 - 0.7	0 - 0.5
Mg (magnesium) (%)	0.8 - 1.2	2.1 - 2.9
Mn (manganese) (%)	0 - 0.15	0 - 0.3
Si (silicon) (%)	0.4 - 0.8	0 - 0.4
Ti (titanium) (%)	0 - 0.15	0 - 0.2
Zn (zinc) (%)	0 - 0.25	5.1 - 6.1
Other (%)	0 - 0.15	0 - 0.15
Price		
Price (GBP/kg)	1.52 - 1.75	3.05 - 3.5
Price per unit volume (GBP/m^3)	4110 - 4780	8410 - 9910
Physical properties		
Density (kg/m^3)	2690 - 2730	2770 - 2830
Mechanical properties		
Young's modulus (GPa)	66.6 - 70	69 - 76
Young's modulus with temperature (GPa) #	69	69.6 - 76.6
Specific stiffness (MN.m/kg)	24.5 - 25.8	24.6 - 27.2
Yield strength (elastic limit) (MPa)	241 - 281	460 - 530
Yield strength with temperature (MPa) #	279	499

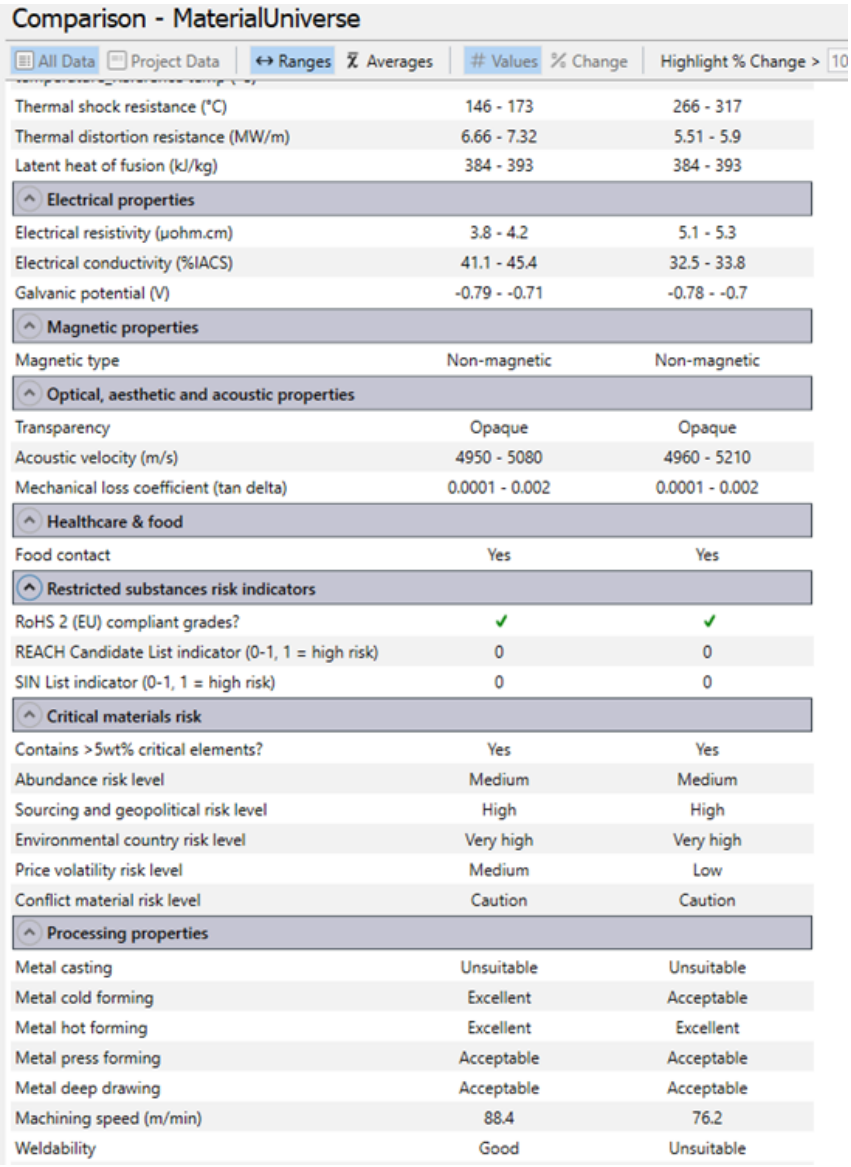
Edupack(Aluminium) part 1

Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | x̄ Averages | # Values | % Change | Highlight % Change > 10

Yield strength with temperature (MPa) #	279	499
Tensile strength (MPa)	274 - 320	530 - 580
Tensile strength with temperature (MPa) #		557
Specific strength (kN.m/kg)	88.8 - 104	164 - 189
Elongation (% strain)	10 - 14.4	2 - 10
Tangent modulus (MPa)	563	1190
Compressive modulus (GPa)	67.9 - 71.3	
Compressive strength (MPa)	241 - 281	460 - 530
Flexural modulus (GPa)	66.6 - 70	69 - 76
Flexural strength (modulus of rupture) (MPa)	241 - 281	460 - 530
Shear modulus (GPa)	25.6 - 26.9	26 - 28
Shear strength (MPa)	166 - 193	
Bulk modulus (GPa)	66.6 - 70	67 - 74
Poisson's ratio	0.325 - 0.335	0.325 - 0.335
Shape factor	25	15
Hardness - Vickers (HV)	100 - 107	152 - 168
Hardness - Brinell (HB)		145 - 165
Elastic stored energy (springs) (kJ/m^3)	427 - 576	1460 - 1940
Fatigue strength at 10^7 cycles (MPa)	106 - 124	152 - 168
Fatigue strength model (stress amplitude) (MPa) #	96.9 - 136	143 - 179
Impact & fracture properties		
Fracture toughness (MPa.m^0.5)	30 - 36	26.6 - 26.8
Toughness (G) (kJ/m^2)	13.3 - 18.9	9.38 - 10.3
Thermal properties		
Melting point (°C)	582 - 652	475 - 635
Maximum service temperature (°C)	130 - 150	80 - 100
Minimum service temperature (°C)	-273	-273
Thermal conductivity (W/m.°C)	161 - 174	131 - 137
Thermal conductivity with temperature (W/m.°C) #		146
Specific heat capacity (J/kg.°C)	934 - 972	913 - 979
Specific heat capacity with temperature (J/kg.°C) #	1020	
Thermal expansion coefficient (µstrain/°C)	23.4 - 24.6	22.9 - 24.1
Thermal expansion coefficient with temperature (µstrain/°C) #	22.8	
Thermal expansion coefficient with temperature_Reference temp (°C)	20	
Thermal shock resistance (°C)	146 - 173	266 - 317

Edupack(Aluminium) part 2

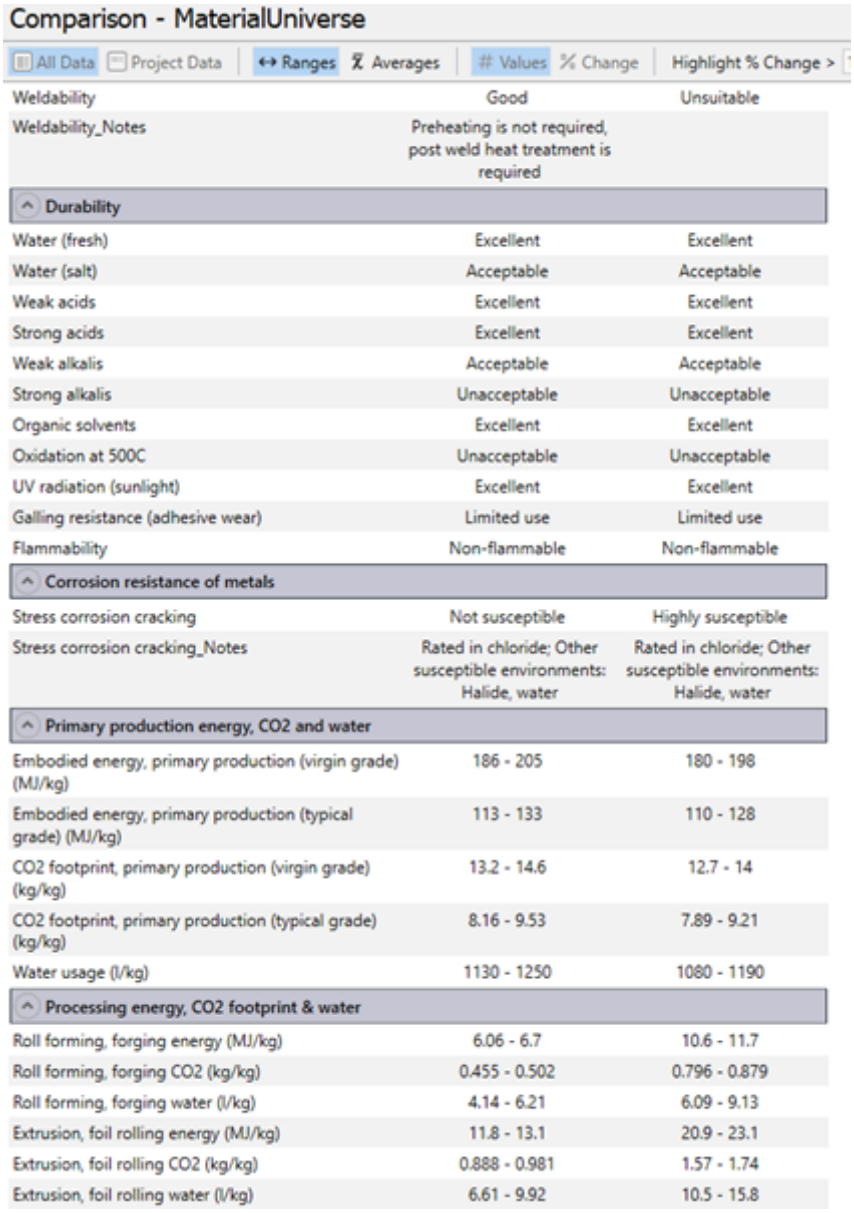


Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | x̄ Averages | # Values | % Change | Highlight % Change > 10

Thermal shock resistance (°C)	146 - 173	266 - 317
Thermal distortion resistance (MW/m)	6.66 - 7.32	5.51 - 5.9
Latent heat of fusion (kJ/kg)	384 - 393	384 - 393
Electrical properties		
Electrical resistivity (µohm.cm)	3.8 - 4.2	5.1 - 5.3
Electrical conductivity (%IACS)	41.1 - 45.4	32.5 - 33.8
Galvanic potential (V)	-0.79 - -0.71	-0.78 - -0.7
Magnetic properties		
Magnetic type	Non-magnetic	Non-magnetic
Optical, aesthetic and acoustic properties		
Transparency	Opaque	Opaque
Acoustic velocity (m/s)	4950 - 5080	4960 - 5210
Mechanical loss coefficient (tan delta)	0.0001 - 0.002	0.0001 - 0.002
Healthcare & food		
Food contact	Yes	Yes
Restricted substances risk indicators		
RoHS 2 (EU) compliant grades?	✓	✓
REACH Candidate List indicator (0-1, 1 = high risk)	0	0
SIN List indicator (0-1, 1 = high risk)	0	0
Critical materials risk		
Contains >5wt% critical elements?	Yes	Yes
Abundance risk level	Medium	Medium
Sourcing and geopolitical risk level	High	High
Environmental country risk level	Very high	Very high
Price volatility risk level	Medium	Low
Conflict material risk level	Caution	Caution
Processing properties		
Metal casting	Unsuitable	Unsuitable
Metal cold forming	Excellent	Acceptable
Metal hot forming	Excellent	Excellent
Metal press forming	Acceptable	Acceptable
Metal deep drawing	Acceptable	Acceptable
Machining speed (m/min)	88.4	76.2
Weldability	Good	Unsuitable

Edupack(Aluminium) part 3

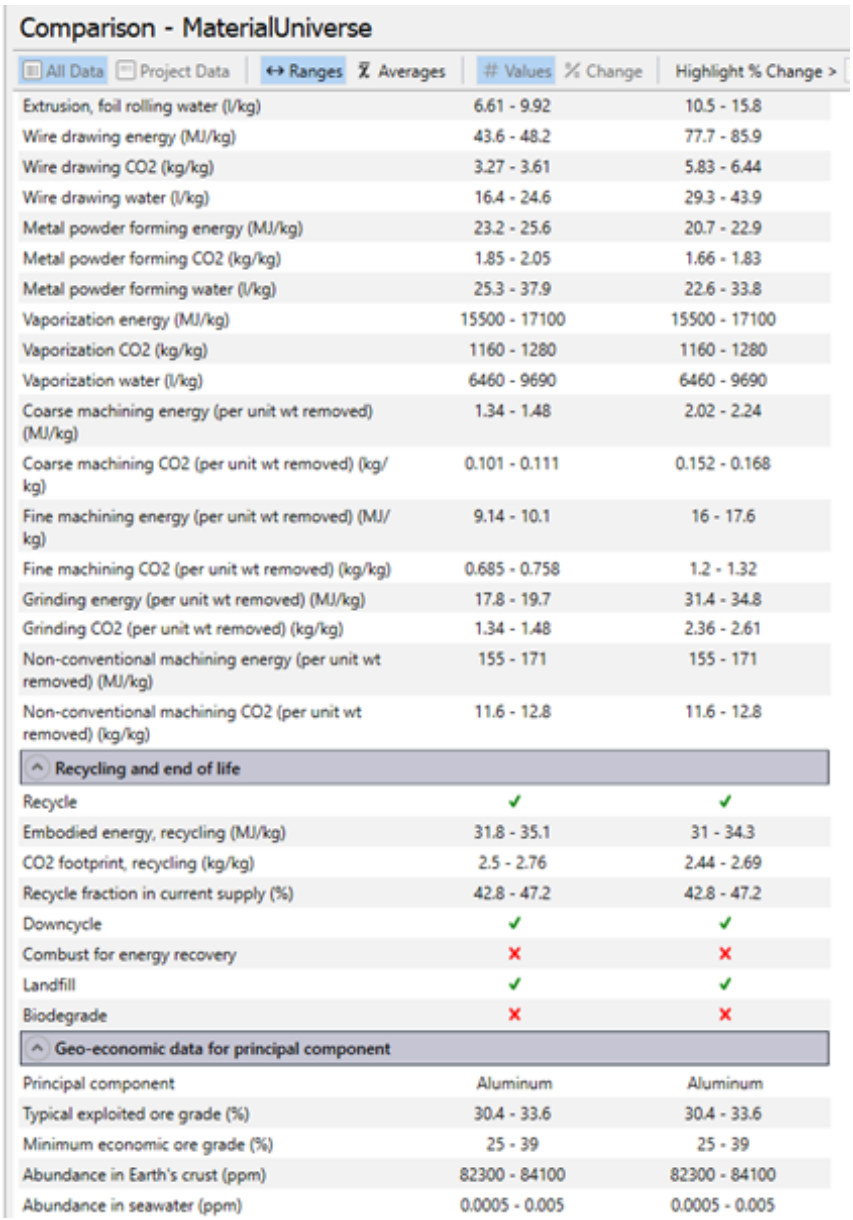


Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | Averages | # Values | % Change | Highlight % Change >

Weldability	Good	Unsuitable
Weldability_Notes	Preheating is not required, post weld heat treatment is required	
Durability		
Water (fresh)	Excellent	Excellent
Water (salt)	Acceptable	Acceptable
Weak acids	Excellent	Excellent
Strong acids	Excellent	Excellent
Weak alkalis	Acceptable	Acceptable
Strong alkalis	Unacceptable	Unacceptable
Organic solvents	Excellent	Excellent
Oxidation at 500C	Unacceptable	Unacceptable
UV radiation (sunlight)	Excellent	Excellent
Galling resistance (adhesive wear)	Limited use	Limited use
Flammability	Non-flammable	Non-flammable
Corrosion resistance of metals		
Stress corrosion cracking	Not susceptible	Highly susceptible
Stress corrosion cracking_Notes	Rated in chloride; Other susceptible environments: Halide, water	Rated in chloride; Other susceptible environments: Halide, water
Primary production energy, CO2 and water		
Embodied energy, primary production (virgin grade) (MJ/kg)	186 - 205	180 - 198
Embodied energy, primary production (typical grade) (MJ/kg)	113 - 133	110 - 128
CO2 footprint, primary production (virgin grade) (kg/kg)	13.2 - 14.6	12.7 - 14
CO2 footprint, primary production (typical grade) (kg/kg)	8.16 - 9.53	7.89 - 9.21
Water usage (l/kg)	1130 - 1250	1080 - 1190
Processing energy, CO2 footprint & water		
Roll forming, forging energy (MJ/kg)	6.06 - 6.7	10.6 - 11.7
Roll forming, forging CO2 (kg/kg)	0.455 - 0.502	0.796 - 0.879
Roll forming, forging water (l/kg)	4.14 - 6.21	6.09 - 9.13
Extrusion, foil rolling energy (MJ/kg)	11.8 - 13.1	20.9 - 23.1
Extrusion, foil rolling CO2 (kg/kg)	0.888 - 0.981	1.57 - 1.74
Extrusion, foil rolling water (l/kg)	6.61 - 9.92	10.5 - 15.8

Edupack(Aluminium) part 4

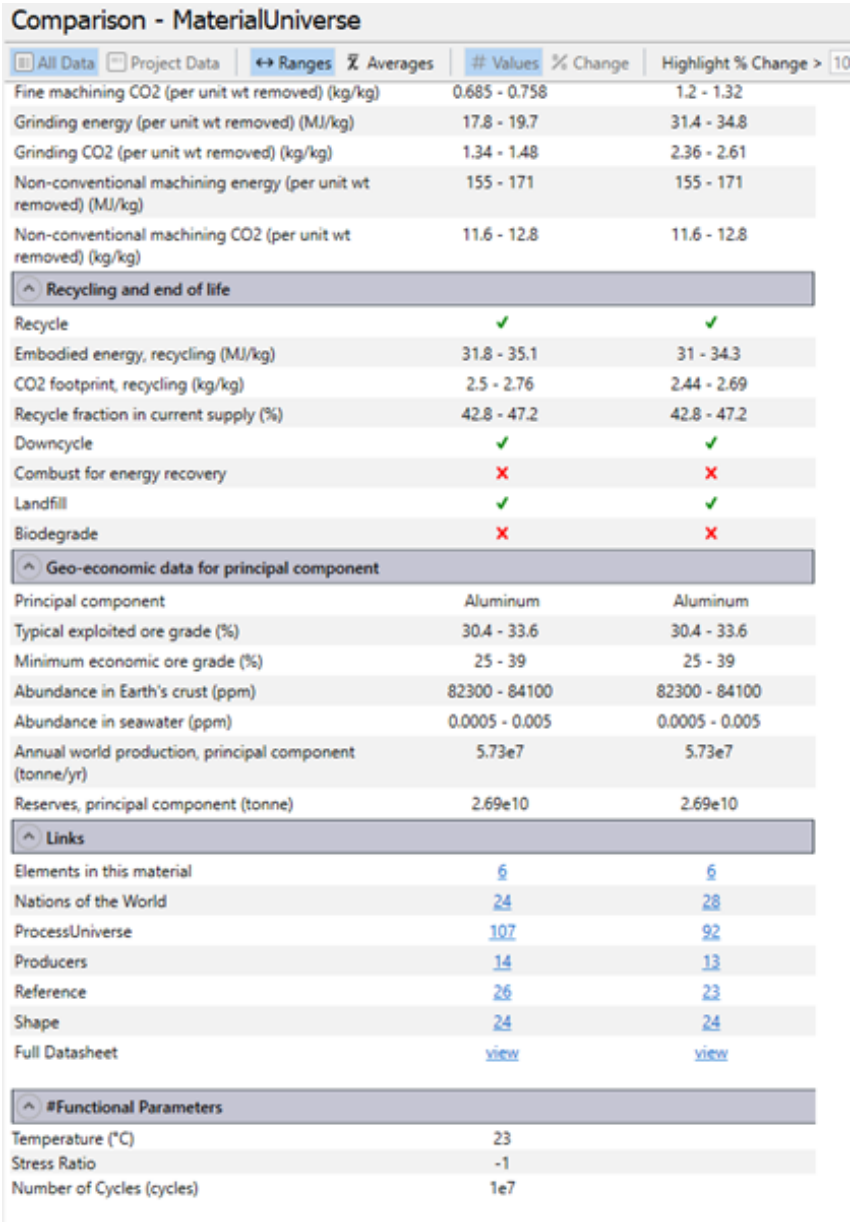


Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | Averages | # Values | % Change | Highlight % Change >

Extrusion, foil rolling water (l/kg)	6.61 - 9.92	10.5 - 15.8
Wire drawing energy (MJ/kg)	43.6 - 48.2	77.7 - 85.9
Wire drawing CO2 (kg/kg)	3.27 - 3.61	5.83 - 6.44
Wire drawing water (l/kg)	16.4 - 24.6	29.3 - 43.9
Metal powder forming energy (MJ/kg)	23.2 - 25.6	20.7 - 22.9
Metal powder forming CO2 (kg/kg)	1.85 - 2.05	1.66 - 1.83
Metal powder forming water (l/kg)	25.3 - 37.9	22.6 - 33.8
Vaporization energy (MJ/kg)	15500 - 17100	15500 - 17100
Vaporization CO2 (kg/kg)	1160 - 1280	1160 - 1280
Vaporization water (l/kg)	6460 - 9690	6460 - 9690
Coarse machining energy (per unit wt removed) (MJ/kg)	1.34 - 1.48	2.02 - 2.24
Coarse machining CO2 (per unit wt removed) (kg/kg)	0.101 - 0.111	0.152 - 0.168
Fine machining energy (per unit wt removed) (MJ/kg)	9.14 - 10.1	16 - 17.6
Fine machining CO2 (per unit wt removed) (kg/kg)	0.685 - 0.758	1.2 - 1.32
Grinding energy (per unit wt removed) (MJ/kg)	17.8 - 19.7	31.4 - 34.8
Grinding CO2 (per unit wt removed) (kg/kg)	1.34 - 1.48	2.36 - 2.61
Non-conventional machining energy (per unit wt removed) (MJ/kg)	155 - 171	155 - 171
Non-conventional machining CO2 (per unit wt removed) (kg/kg)	11.6 - 12.8	11.6 - 12.8
Recycling and end of life		
Recycle	✓	✓
Embodied energy, recycling (MJ/kg)	31.8 - 35.1	31 - 34.3
CO2 footprint, recycling (kg/kg)	2.5 - 2.76	2.44 - 2.69
Recycle fraction in current supply (%)	42.8 - 47.2	42.8 - 47.2
Downcycle	✓	✓
Combust for energy recovery	✗	✗
Landfill	✓	✓
Biodegrade	✗	✗
Geo-economic data for principal component		
Principal component	Aluminum	Aluminum
Typical exploited ore grade (%)	30.4 - 33.6	30.4 - 33.6
Minimum economic ore grade (%)	25 - 39	25 - 39
Abundance in Earth's crust (ppm)	82300 - 84100	82300 - 84100
Abundance in seawater (ppm)	0.0005 - 0.005	0.0005 - 0.005

Edupack(Aluminium) part 5



Comparison - MaterialUniverse

All Data | Project Data | ↔ Ranges | x̄ Averages | # Values | % Change | Highlight % Change > 10

Fine machining CO2 (per unit wt removed) (kg/kg)	0.685 - 0.758	1.2 - 1.32
Grinding energy (per unit wt removed) (MJ/kg)	17.8 - 19.7	31.4 - 34.8
Grinding CO2 (per unit wt removed) (kg/kg)	1.34 - 1.48	2.36 - 2.61
Non-conventional machining energy (per unit wt removed) (MJ/kg)	155 - 171	155 - 171
Non-conventional machining CO2 (per unit wt removed) (kg/kg)	11.6 - 12.8	11.6 - 12.8
Recycling and end of life		
Recycle	✓	✓
Embodied energy, recycling (MJ/kg)	31.8 - 35.1	31 - 34.3
CO2 footprint, recycling (kg/kg)	2.5 - 2.76	2.44 - 2.69
Recycle fraction in current supply (%)	42.8 - 47.2	42.8 - 47.2
Downcycle	✓	✓
Combust for energy recovery	✗	✗
Landfill	✓	✓
Biodegrade	✗	✗
Geo-economic data for principal component		
Principal component	Aluminum	Aluminum
Typical exploited ore grade (%)	30.4 - 33.6	30.4 - 33.6
Minimum economic ore grade (%)	25 - 39	25 - 39
Abundance in Earth's crust (ppm)	82300 - 84100	82300 - 84100
Abundance in seawater (ppm)	0.0005 - 0.005	0.0005 - 0.005
Annual world production, principal component (tonne/yr)	5.73e7	5.73e7
Reserves, principal component (tonne)	2.69e10	2.69e10
Links		
Elements in this material	6	6
Nations of the World	24	28
ProcessUniverse	107	92
Producers	14	13
Reference	26	23
Shape	24	24
Full Datasheet	view	view
#Functional Parameters		
Temperature (°C)	23	
Stress Ratio	-1	
Number of Cycles (cycles)	1e7	

Edupack(Aluminium) part 6

E. Appendix - CAD and 3D Printing

E.1. Section 1 Title

E.1.1. subsectiontitle

E.2. Section 2 Title

E.2.1. subsectiontitle

F. Appendix - Renewable Energy Demonstrator (RED) testbench code

F.1. Appendix - RED LED Strip Code

F.1.1. LED strip Code for automatic transition

This code is modified to change the LED position every 5 seconds automatically.

```
    #include <FastLED.h>

// LED strip configuration
#define NUM_LEDS 27
#define LED_PIN 5
CRGB leds[NUM_LEDS]; // define FastLED datatype

// automatic transition all leds:
const int timeInterval = 5000;  // miliseconds


// Communication pin from main Arduino
// const int led_status_pin = 12;  // Input pin to receive signals

// LED Positions
// MODIFY THESE VALUES TO CHANGE PRESET LED POSITIONS

//all positions version
const int NUM_PRESETS = 25;
// modified from 5 positions to all positions automatically every
   timeInterval seconds
int presetLedPositions[NUM_PRESETS] = {
  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25
};

// State variables
int currentPresetIndex = 2;  // start close to the edge
bool lastSignalState = LOW;
```

```
28
29 void setup() {
30   Serial.begin(9600);
31   Serial.println("LED Controller Starting...");
32
33   //  LED strip
34   FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, NUM_LEDS);
35   FastLED.setBrightness(255);  //  brightness (0-255)
36
37   //  communication pin
38   // pinMode(led_status_pin, INPUT); not used
39
40   // Clear LED
41   clearAllLeds();
42
43   //  starting position
44   currentPresetIndex = 0;  // begining position
45
46   // small delay
47   delay(50);
48
49   // Show initial LED position on screen
50   updateLedDisplay(presetLedPositions[currentPresetIndex]);
51
52   // Debug output
53   Serial.print("Starting with preset index: ");
54   Serial.println(currentPresetIndex);
55 }
56
57 void loop() {
58
59
60   // do not wait for input anymore, automatic
61   if (1) {
62
63     // Clear previous LEDs
64     clearAllLeds();
65
66     // Move to next preset
67     currentPresetIndex = (currentPresetIndex + 1) % NUM_PRESETS;
68
69     // Update LED display
70     updateLedDisplay(presetLedPositions[currentPresetIndex]);
71
72     // wait preset speed
73     delay(timeInterval);
74   }
```

```
75
76
77 }
78
79 void clearAllLeds() {
80   for (int i = 0; i < NUM_LEDS; i++) {
81     leds[i] = CRGB::Black;
82   }
83   FastLED.show();
84 }
85
86 // update the location of the LEDS,
87 // based on centerindex
88 void updateLedDisplay(int centerIndex) {
89   if (centerIndex > 0) {
90     leds[centerIndex - 1] = CRGB::White;
91   }
92
93   leds[centerIndex] = CRGB::White;
94
95   if (centerIndex < NUM_LEDS - 1) {
96     leds[centerIndex + 1] = CRGB::White;
97   }
98
99   FastLED.show();
100  Serial.print("LED updated at preset index: ");
101  Serial.print(currentPresetIndex + 1);  // Display 1-5 instead of 0-4
102  Serial.print(" (LED position: ");
103  Serial.print(centerIndex);
104  Serial.println(")");
105 }
```

Listing F.1: Cpp Code of the RGB strip with automatic changes

F.1.2. LED Strip Code for Manual 5 location transition

```
1     #include <FastLED.h>
2
3 // LED strip configuration
4 #define NUM_LEDS 27
5 #define LED_PIN 5
6 CRGB leds[NUM_LEDS];
7
8 // Communication pin from main Arduino
9 const int led_status_pin = 12;  // Input pin to receive signals
10
11 // Preset LED Positions Configuration
```

```
// MODIFY THESE VALUES TO CHANGE PRESET LED POSITIONS
const int NUM_PRESETS = 5;
int presetLedPositions[NUM_PRESETS] = {
  3,    // Position 1 - center LED index
  9,    // Position 2 - center LED index
  13,   // Position 3 - center LED index (middle)
  17,   // Position 4 - center LED index
  23    // Position 5 - center LED index
};

// State variables
int currentPresetIndex = 2;  // Start at center position (index 2, which
    is the 3rd preset)
bool lastSignalState = LOW;

void setup() {
  Serial.begin(9600);
  Serial.println("LED Controller Starting...");

  // Initialize LED strip
  FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, NUM_LEDS);

  // Initialize communication pin
  pinMode(led_status_pin, INPUT);

  // Clear all LEDs
  clearAllLeds();

  // Show initial LED position
  updateLEDstrip(presetLedPositions[currentPresetIndex]);
}

void loop() {
    // Read signal from main Arduino
    bool signalState = digitalRead(led_status_pin);

    // Apply debounce delay
    delay(5);

    // Read signal again after debounce
    bool debouncedSignalState = digitalRead(led_status_pin);

    // Only proceed if the signal is stable
    if (debouncedSignalState == signalState) {
        // If stable signal state changed from previous loop iteration
        if (debouncedSignalState != lastSignalState) {
            // Debug print for any signal change
```

```
58              Serial.print("Signal changed from ");
59              Serial.print(lastSignalState ? "HIGH" : "LOW");
60              Serial.print(" to ");
61              Serial.println(debouncedSignalState ? "HIGH" : "LOW");
62
63              // Detect rising edge (LOW to HIGH transition)
64              if (debouncedSignalState == HIGH && lastSignalState == LOW)
    {
65              Serial.println("Signal received from main Arduino");
66
67              // Clear previous LEDs
68              clearAllLeds();
69
70              // Move to next preset
71              currentPresetIndex = (currentPresetIndex + 1) % NUM_PRESETS;
72
73              // Update LED srip
74              updateLEDstrip(presetLedPositions[currentPresetIndex]);
75              }
76          }
77
78          // Update the last signal state
79          lastSignalState = debouncedSignalState;
80          }
81      // If signal is not stable after debounce, don't update
    lastSignalState
82      }
83
84  void clearAllLeds() {
85    for (int i = 0; i < NUM_LEDS; i++) {
86      leds[i] = CRGB::Black;
87    }
88    FastLED.show();
89  }
90
91  void updateLEDstrip(int centerIndex) {
92
93    if (centerIndex > 0) {
94      leds[centerIndex - 1] = CRGB::White;
95    }
96
97    leds[centerIndex] = CRGB::White;
98
99    if (centerIndex < NUM_LEDS - 1) {
100     leds[centerIndex + 1] = CRGB::White;
101   }
102
```

```
  FastLED.show();
  Serial.print("LED updated at preset index: ");
  Serial.print(currentPresetIndex + 1);
  Serial.print(" (LED position: ");
  Serial.print(centerIndex);
  Serial.println(")");
}
```

Listing F.2: Cpp Code of the RGB strip with Manual 5 locations

F.1.3. ARCH controller C++ code

```
#include <Servo.h>
#include <LiquidCrystal_I2C.h>

// Servo
Servo arch_left;
Servo arch_right;

// LCD screen
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Pin Definitions
const int pushButton = 13;
const int led_status_pin = 12;  // Signal to secondary Arduino for LED control

// Presets

const int NUM_PRESETS = 5;
int presetAngles[NUM_PRESETS] = {
  20,   // Position 1 - angle in degrees
  50,   // Position 2 - angle in degrees
  90,   // Position 3 - angle in degrees (center) - may not reach, due to servos?
  130,  // Position 4 - angle in degrees
  160   // Position 5 - angle in degrees
};

// State Variables
int current_index = 2;  // Start at center position (index 2, which is the 3rd preset)
bool lastButtonState = HIGH; // Using INPUT_PULLUP, so HIGH is not pressed

void setup() {
  Serial.begin(9600);
```

```
32
33    // Initialize servos
34    arch_left.attach(5);
35    arch_right.attach(6);
36
37    // Set initial position
38    change_state(current_index);
39
40    // Initialize LED communication pin
41    pinMode(led_status_pin, OUTPUT);
42    digitalWrite(led_status_pin, LOW);
43
44    // Initialize LCD
45    lcd.init();
46    lcd.backlight();
47    updateDisplay();
48
49    // Initialize button
50    pinMode(pushButton, INPUT_PULLUP);
51  }
52
53  void loop() {
54    // Read button state
55    bool buttonState = digitalRead(pushButton);
56
57    // Check for button press (transition from HIGH to LOW)
58    if (buttonState == LOW && lastButtonState == HIGH) {
59      // Move to next preset
60      current_index = (current_index + 1) % NUM_PRESETS;
61
62      // Update servo position
63      change_state(current_index);
64
65      // Update display
66      updateDisplay();
67
68      // Signal the LED Arduino to change LEDs
69      signalLedArduino();
70
71      // Debounce delay
72      delay(200);
73    }
74
75    // Save button state for next iteration
76    lastButtonState = buttonState;
77  }
78
```

```
79 void change_state(int presetIndex) {
80   // do a calibration each time to ensure correct position
81   arch_left.write(0);
82   arch_right.write(180); // Mirror  (180 - 0)
83 
84   // add delay to reach location
85   delay(5000);  // Adjust delay as needed for your servos
86 
87   int angle = presetAngles[presetIndex];
88 
89   // Update servos with the new location
90   arch_left.write(angle);
91   arch_right.write(180 - angle); // Mirror movement
92   // debug
93   Serial.print("Moving to preset ");
94   Serial.print(presetIndex + 1);
95   Serial.print(" (angle: ");
96   Serial.print(angle);
97   Serial.println(" degrees)");
98 }
99 
100 void signalLedArduino() {
101   // Trigger the RGB Arduino's LED change
102   digitalWrite(led_status_pin, HIGH);
103   delay(50);  // Brief pulse to trigger the other Arduino and pass
      debounce check
104   digitalWrite(led_status_pin, LOW);
105 
106   Serial.println("Sent signal to LED Arduino");
107 }
108 
109 void updateDisplay() {
110   lcd.clear();
111   lcd.setCursor(0, 0);
112   lcd.print("Position: ");
113   lcd.print(current_index + 1);
114 
115   lcd.setCursor(0, 1);
116   lcd.print("Alt:");
117   lcd.print(presetAngles[current_index]);
118   lcd.print(" Az:");
119   lcd.print(presetAngles[current_index]);
120 }
```

Listing F.3: C++ Code of the ARCH Arduino