BIOSPHERE SOLAR INTERNSHIP REPORT

4452437 - Keshav Shukla – Test and Product Engineer

AUGUST 1, 2023

COMPANY SUPERVISOR: SIEMEN BRINSKMA
TU Delft supervisor: Pieter Swinkels

Table of Contents

Acknowledgements

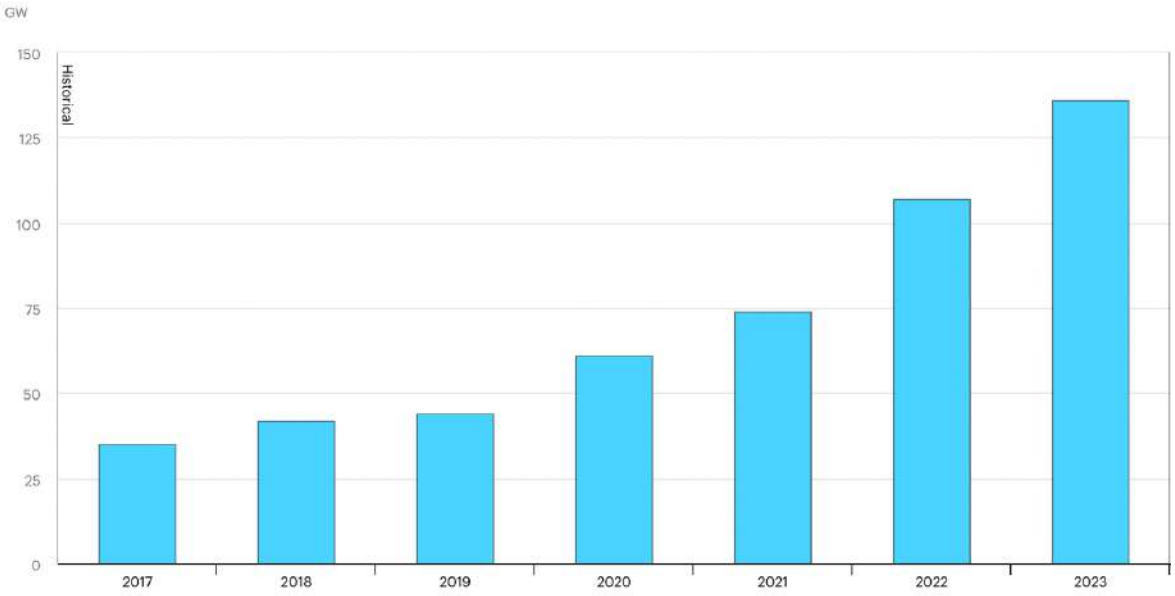
I want to first thank Biosphere Solar, especially the co-founders team Siemen Brinksma, Perine Fleury and Tim Kaasjager. It was with the combined effort of these three that Biosphere Solar is what it is today. Specifically, I would like to thank Siemen Brinksma for leading the company, it is his vision that started Biosphere Solar. Siemens presence at the workspace was always that of a calm guru, never shy in letting us know we're doing a great job (even when sometimes we were not). Perine was always a glowing figure at the workspace, never hesistating to share her food and smoothies, helping us stay energised during our work. Lastly. Tim Kaasjager, as Product Development lead I worked closely with Tim, and would like to thank him most for the learning I undertook during my internship. It was with his guidance and support that I was able to expand my knowledge of photo-voltaics, microelectronics, and perhaps most importantly he taught me not to be too nervous when experimenting with experimental set-ups, and to trust my knowledge.

I should also like to thank my colleagues, especially Mihir Limdi and Myrthe Coster, Mihir for sitting me with me for hours helping me decipher the IV Swinger's inner workings, and Myrthe as my fellow Test and Product designer taking lead on the Mechanical load test, and helping me understand the IEC regulations.

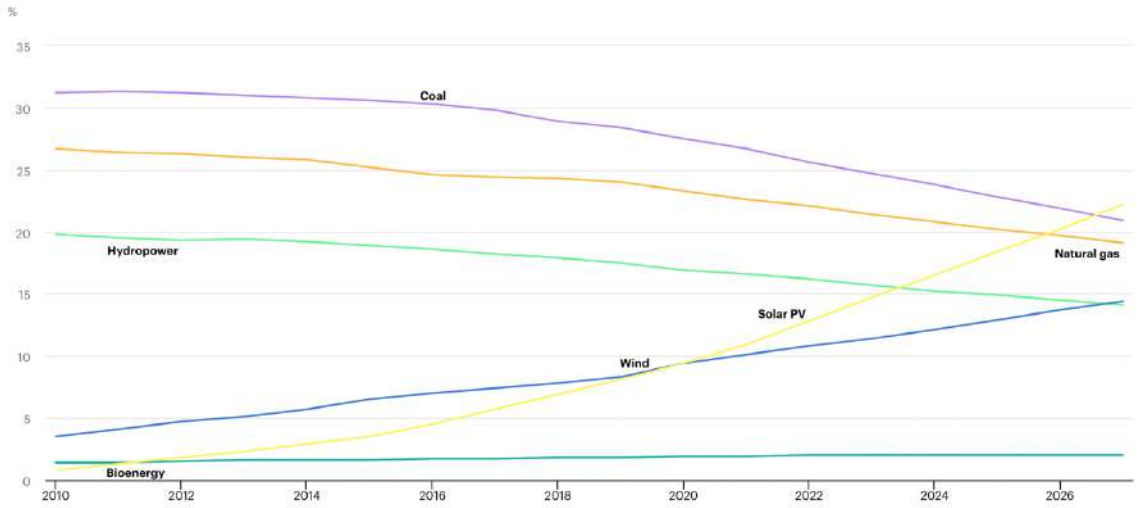
I would also like to share my deep appreciation for Chris Satterlee, the original designer of IV Swinger, by extension the entire open hardware community working so magnanimously, only out of their passion for technology and their desire to make technology more accessible to everyone.

Company summary

Biosphere Solar is a company that is working to develop an open-source design of a PV solar panel that is easier to maintain, repair and recycle. Being in midst of an energy transition, solar PV holds a dominant position, and arguably has the chance to make the largest impact in the switch to sustainable energy. Harvesting clean electricity, directly from sunlight, all the while being highly scalable, one can see why PV growth has skyrocketed in the past few years, even when compared to other sustainable options such as wind or hydropower.



(IEA, *Global distributed solar PV net additions per year, 2017-2024*, IEA, Paris https://www.iea.org/data-and-statistics/charts/global-distributed-solar-pv-net-additions-per-year-2017-2024, IEA. Licence: CC BY 4.0)



(IEA, *Share of cumulative power capacity by technology, 2010-2027*, IEA, Paris https://www.iea.org/data-and-statistics/charts/share-of-cumulative-power-capacity-by-technology-2010-2027, IEA. Licence: CC BY 4)

The increase popularity of Solar PV is also reflected in the money invested in PV technologies, with the IEA estimating over 320 billion USD being spent on solar in 2022, comprising of 45% of the total global electricity generation investment. (Bojek, sd)

The rush of investment and direction of research is currently focused on driving down the price and therefore increasing the access of Solar PV to as many households and industries as possible. This cost-driven approach has slashed prices immensely, since 2000, the cost of energy generated per watt has decreased by a factor of 25. (Bojek, sd) This approach, while undoubtedly effective, has one large caveat; once again the incentive to drive profits has left the industry to largely ignore the hidden costs associated with the manufacturing and production of solar panels. This linear mindset to manufacturing makes it increasingly difficult to prioritise maintenance, repair, and recyclability of the solar module. Why spend more money on developing a fixable and longer life product, when mass manufacturing is more convenient, at the expense of the environment. Ironically, while solar PV is a clean way of generating power, the manufacturing side of the solar PV is far from sustainable.

Referring to the Mission and Vision of the company, Biopshere Solar aims to create a world where the techno-sphere and the biosphere are brought in symbiotic relationship with one another; a world where solar energy and biodiversity regeneration work together, and the use of open-source hardware solar design enables rapid and resilient adaptations to be developed. This vision is guiding them to the development of an open-source, circular, and fair solar module: as Biosphere Solar, we aim to provide the knowledge and necessary push for the solar industry to transition away from its current business model.

Biosphere Solar started as the thesis project of Siemens Brinksma in 2021 in Industrial Ecology where he investigated a novel prototype of a recyclable solar panel, seeing its potential he decided to spin-off the idea as a company.

Biosphere Solar aims not only to integrate sustainability in the final product, but by considering circularity, sustainability, and transparency throughout the supply chain of a solar module. On top of that, they appreciate and understand the fundamental shift in priorities often required to be truly sustainable and aims to reflect that in their business model and goals of the company. Biosphere is focused on bringing about a disruption in the solar industry, it is also their belief that an open-source approach to problem-solving often leads to more collaboration and solutions that benefit more communities. Their additional goal is to act as a global initiative of researchers, entrepreneurs, and change-oriented individuals, bringing about a change in the solar industry one player at a time.

Following from this, the company has three major divisions, Strategy and Finance, Product Development and Execution and Network. I will be part of the Product Development team, working on in-house testing methods for the solar panel that is being developed. A deeper explanation of the department and its goals are given in the next section.

Activities during the internship

Being part of the product development team, I was tasked to contribute to the V1 version of the solar panel prototype. When working with a novel design, the development process of a product often requires multiple iterations, each with their own distinct goals, a neat way of breaking down the problem into smaller steps. Following the plan laid out in Table 1, I was tasked with helping with the V.1, the MVP or the minimum viable product. To achieve this, the main roadblock was getting the solar module certified for sales. Additionally, during the time of my internship Biosphere Solar was also developing what they call the 'Development Kit', a mini solar panel with a modular and repairable design, with contributions from our community, intended to get some experience in sales, production and community building prior to the V.2 and V.3. As such, part of my efforts and work time were also dedicated to the team that was working on the Development Kit.

Table 1. Design requirements based on prototypes

	Modular	Repairable	Upgradeable	High-quality recyclable	Certified for sales	Price Competitive	Locally sourced	Lightweig-ht
V.0 - prototype								
V.1 - Minimum Viable product								
V.2 - competitive Design								
V.3 - Locally disruptive Design								
V.4 - Globally Disruptive Design								

My work involved planning the work needed to obtain certification, this involved the route to certification for the CE marking, and planning and execution of tests required by the IEC (International Electrotechnical Commission). The CE marking is a trademark sign required to sell or trade any products in the EEA, the European Economic Area. It serves as a declaration by the manufacturer that the product complies with the minimum product safety, health and environmental requirements of certain EU directives. The IEC tests go deeper, testing the performance of the solar panel and ensuring the market-standard degradation rates. The IEC tests are often required to make deals with distributors and installers. In the

duration of the internship. Although the certification of the solar panel was my main priority often times my assistance was needed elsewhere. My work can be more accurately broken down into the following tasks:

- Establishing a roadmap of testing required for CE certification, involved digging through EU regulation documents and figuring out which tests are relevant for the CE mark.
- Developing in-house methods for relevant tests, in this case the Mechanical Load test and Electric Insulation test.
- Conducting and documenting the Mechanical Load test and Electric Insulation test to obtain the CE marking.
- Refining and documenting existing in-house testing methods for EL (electro-luminescence) and Hail Test.
- Construct and test the open-source PCB dubbed IV Swinger, a device used to generate Voltage-Current graphs of solar panels to test their power output.
- Assisting with troubleshooting of the solar panel installation located in the Green Village, TU Delft Campus.
- Assist the team with the iterations of the Development Kit, involves laser-cutting, assembly assistance, and presence at events.

Learning experience

At the start of the internship, and while applying I felt very nervous, this position did not entail much of what I learnt during Chemical Engineering. It was a test and product engineer role, requiring very hands-on skills and being part of an organisation that was full of PV and sustainability enthusiasts, and a team of co-founders that were incredibly dedicated to making their vision come true. If I had to keep up, I would have had to step outside my comfort zone and figure out for myself where I could make the best impact, the path forward wasn't completely clear from the beginning.

The internship began with a TU Delft MOOC on Solar Energy: Photovoltaic Materials, Devices and Modules. This course ranged from a deep dive into the mechanics of electron-hole pair generation that PV relies on, while also covering practical knowledge about how modules are linked together in solar farms to improve efficiency. I considered this my 'literature' phase, where I studied and understood the technology and the market behind PV. While doing this, I also learnt how to build a model solar module, which included soldering, glass cutting and quality control on assembled modules. It was also during this time that I realised my electric engineering knowledge was very much below par, while at the solar panel installation at Green Village, I was looking completely helpless while the electric team was troubleshooting and re-wiring the installation. This also motivated me to brush up on my basic circuitry and current flow applications, giving me a great excuse to dive back into my high-school physics book. Already at the first half month of my internship I was learning new things, refreshing my knowledge, and developing my hands-on skills as well.

Past my literature phase, the first task I undertook was also the most daunting, it was to get the IV Swinger working, which the current team had been struggling with. While the soldering of the components was quite simple, getting it reliably produce results posed a challenge. The learning curve for microelectronics was steep, and my initial assumption of circuit boards being purely logical was painfully dissolved. I was very thankful for having the access to Chris Saterlee, the developer of the IV Swinger, to answer my more pointed questions. The troubleshooting of the IV Swinger was particularly important since it was the only way to record and measure the power output of a solar module at varying loads. It involved gaining an insight of the details of microelectronic circuits and the role of individual components in those circuits. I wrestled with faulty components, busted ADC's, loose connections, opting in the end to rebuild the Swinger to get it working. With the help from a variety of sources; ElectroBOOM's videos on YouTube, Arduino stackoverflow forums and deep dive sessions with Chris, the IV Swinger was able to produce consistent results with conventional IV curves.

The development of the IV Swinger was part of the larger goal of Biosphere Solar, to get their panel certified. The second half of my internship was spent in organising and starting the road to certification. Aside from the sifting through needlessly wordy EU documents, my main take-away from this phase was my ability to trust theory and apply it in practice. As a Chemical Engineer, I did not often get a chance to apply what I learn in a hands-on setting, unless I get access to a reactor. Learning to adapt theory and apply it in an actual test was a little challenging, especially when it came to the Insulation test, which required handling

500V test set-ups. The first part of the challenge was applying my newly refreshed electric skills in formulating a test set-up designed to test the insulative properties of solar modules. Once the test set-up was finalised, the wire diagrams had to transform into a real-life set-up. I often found myself hesitating on simple tasks, such as soldering wires together, unsure of my technique and capabilities. It was here that Tim Kaasjager helped me get confidence in my own abilities, and always encouraged me to try with little fear of failure. Aside from this, constructing the set-ups for the Hail test, mechanical load test was also fun, albeit less challenging owing to their simplicity.

Lastly, working in a start-up as dynamic as Biosphere Solar was a challenge. I still am quite accustomed to strict work packages and less accustomed to fluid work responsibilities. On the side of the softer skills, my time at Biosphere Solar also forced me out of my comfort zone in terms of working style, where often it was up to me to decide my work schedule and priorities. I'm still unsure if I prefer this workstyle, the challenge was definitely appreciated!

Problem statement

Traditional solar panel design consists of a layer of EVA polymer designed to hold the solar cells in place, this glue type EVA layer is attached directly onto the solar cells, also protecting them from moisture ingress and debris, a critical function owing to the solar cells' sensitivity.

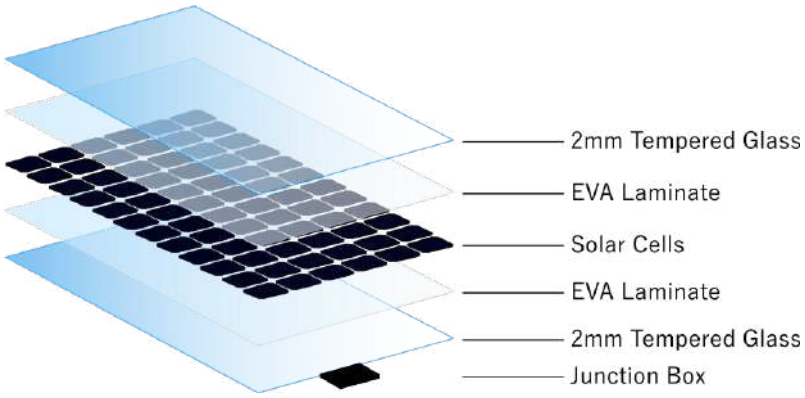


Figure 1 Typical Solar Panel Assembly

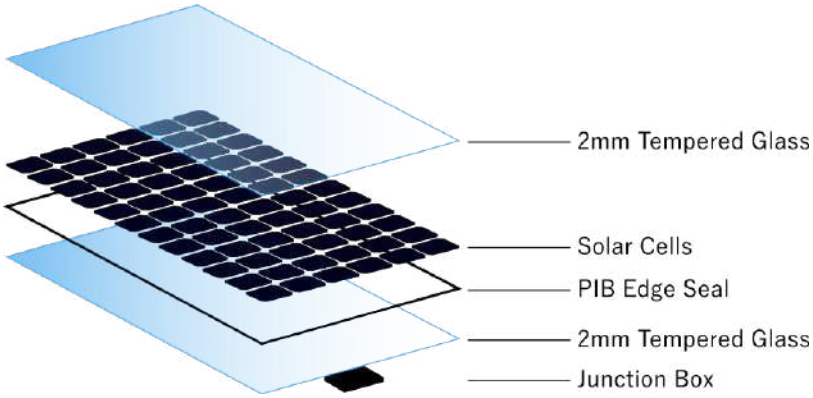


Figure 2 Solar Panel Assembly proposed by Biosphere Solar

Referring to Figure 1, the EVA laminate layer exists both on top and on the bottom of the solar cells, adhering them directly on the 2mm Tempered glass. Aside from protecting the solar cells the two EVA layers create a laminated assembly that is much more rigid than the individual parts, allowing for thinner glass to be used and reducing the overall weight of the solar panel. 2mm glass can be quite flexible, facing high wind forces on top of high rises or open fields could potentially cause the glass to shatter if not for the EVA layer. Perhaps more importantly, the solar cells enveloped between the glass layers are prone to develop cracks at the slightest shocks or bends, such as those by wind or hail.

Asides from the advantages, the EVA layer has several ecological disadvantages.

The EVA layer goes through UV degradation known as ‘yellowing’, rapidly decreasing the performance and energy generation capabilities of the solar cells. (Peplow, 2022)
The EVA layer makes it impossible to reach the solar cells in case of repair / refurbishing
Since the life cycle of a solar cell is typically longer than that of the EVA, all solar cells are severely downcycled when it comes to the end of life of the solar module (typically when the EVA starts to degrade), often being used for filler applications. (Han, 2018) Although a lot of funding has been diverted to proper recycling of the solar modules, Biosphere aims to create a design where such extensive techniques for recycling are not needed, and the design caters more to a philosophy repairability and modularity. (Khalifa, 2021)

Referring to Figure 2, Biosphere opts for a design that contains an edge seal between the tempered glass, known as PIB. Since the same material is used for double-insulated glass windows, its efficacy and ability to block moisture is well documented, and up to industry standards. Omitting the EVA however, brings about its host of problems, decreased rigidity, decreased protection are among the most critical ones. While there are other team-members working on the design iterations of the solar module, it was my job to come up with in-house testing methods to ensure the safety of the design.

An additional goal of Biosphere Solar is to lower the barriers to entry to the solar industry, and therefore make it more accessible. Currently, testing and certification hides behind a large paywall, with certification typically taking tens of thousands of Euros. If the module was to fail certification, the cost must be repeated. This is not typically a problem with traditional solar panels since the industry standard design has been proven countless times, a new design would not give the same assurances. Testing equipment is a further money sink, expensive machines were not readily made available to Biosphere Solar, a 2-year-old start-up. As such, it was my responsibility to develop in-house testing methods wherever possible, such that the Biosphere Solar can apply for certification with the assurances that the modules would be successful in the official tests.

Approach

Regulatory concerns

Biosphere Solar has a pilot project in the pipeline with the Gemeente in Den Haag, to place solar panels on the beach of Scheveningen, a critical step in testing long-term performance of their solar panels in harsh conditions. Although not formally necessary, providing a CE marking for this pilot project was desired, in part to set-up the framework of testing required for all future solar panel iterations.

This would require siffling through EU regulatory documents and figuring out which tests were most relevant. The following EU website, gives certain regulations which apply to a specific product classes. An alternative option is to try and fulfil the 'harmonised standards' but the requirements of complying to the harmonised standards were unclear, leading me to move forward with the documents on the aforementioned EU website. Following from this, a solar panel falls under the Low Voltage and Electromagnetic compatibility product groups. To comply with the directives, the following checklist must be adhered to:

Low voltage:

1. General conditions

(a) the essential characteristics, the recognition and observance of which will ensure that electrical equipment will be used safely and in applications for which it was made, shall be marked on the electrical equipment, or, if this is not possible, on an accompanying document;

(b) the electrical equipment, together with its component parts, shall be made in such a way as to ensure that it can be safely and properly assembled and connected;

(c) the electrical equipment shall be so designed and manufactured as to ensure that protection against the hazards set out in points 2 and 3 is assured, providing that the equipment is used in applications for which it was made and is adequately maintained.

2. Protection against hazards arising from the electrical equipment

Measures of a technical nature shall be laid down in accordance with point 1, in order to ensure that:

(a) persons and domestic animals are adequately protected against the danger of physical injury or other harm which might be caused by direct or indirect contact;

(b) temperatures, arcs or radiation which would cause a danger, are not produced;

(c) persons, domestic animals and property are adequately protected against non-electrical dangers caused by the electrical equipment which are revealed by experience;

(d) the insulation is suitable for foreseeable conditions.

3. Protection against hazards which may be caused by external influences on the electrical equipment

Technical measures shall be laid down in accordance with point 1, in order to ensure that the electrical equipment:

(a) meets the expected mechanical requirements in such a way that persons, domestic animals and property are not endangered;

(b) is resistant to non-mechanical influences in expected environmental conditions, in such a way that persons, domestic animals and property are not endangered;

(c) does not endanger persons, domestic animals and property in foreseeable conditions of overload.

Electromagnetic Compatibility

Opening the box firstly it states the following "Directive 2014/30/EU on Electromagnetic Compatibility (EMC) specifies in detail the essential requirements the product has to meet in order for the manufacturer to affix the CE marking." It also further states "The essential requirements regarding electromagnetic compatibility for equipment are set out in Annex I of the directive". Annex I of the directive is vague and fairly straightforward, stating the following has to be met:

- the electromagnetic disturbance generated does not exceed the level above which radio and telecommunications equipment or other equipment cannot operate as intended;

- it has a level of immunity to the electromagnetic disturbance to be expected in its intended use which allows it to operate without unacceptable degradation of its intended use.

Aside from these points, the EU directives gave no indication of tests to be conducted. The regulations also make multiple referrals to the IEC tests, leading me to the next step. Referring to the specific IEC document labelled IEC 61215-2-2106, a range of standard tests for solar panels are given. While ***ALL*** tests are relevant, and should be done, the MQT 03 Insulation test, MQT 16 Static mechanical load test are the most relevant, with MQT 02 Maximum power determination and MQT 17 Hail test also being relevant, albeit less so. The procedure and method for each test is given below.

MQT 03 Insulation test

According to the regulations the MQT 03 insulation test requires supplying a voltage of 500 V between the frame and the solar cells, while the details of the tests are confidential, to adhere to the required test conditions, the following test-set up was constructed. When attempting to recreate the test set-up, kindly look at the schematics given in tandem with the instructions.

Pre-requisites

- You have read the Moderate Voltage protocol (Appendix A)
- Gotten your preparations checked and signed by a third person
- You have read Section 4.3 in IEC 61215-2:2016
- You are conducting the test with more than one person, of which at least one has a BHV or first aid degree

Material

1. 500V power supply with current limitation <details>
2. DMM with an upper measuring range of at least 600V and 1A and either a resistance upto 40Mohm or current measurement lower than 10uA. <details>
3. All materials and clothing mentioned in the Moderate Voltage Protocol (Appendix A)
4. 2 MC4 Y splitters
5. Two cables (about 1.5m) <details>
6. Insulation connector:

    i. Conductive foil (when testing a frameless module) or conductive tape
    ii. 4 50cm cables brought together into one MC4 connector on one end and with 30x30cm pieces of conductive foil attached to all the loose ends <details>

7. Electrical tape
8. MC male and female heads, along with crimping pins
9. 7x4 = 24 cell solar module with Maxeon Gen II Solar Cells, encapsulated 2 panes of 6mm tempered glass with PIB seal <check Appendix B for composition of solar module and further instructions>

Procedure before test

10. If there is no frame, wrap a conductive foil around the edges. Cover all polymeric surfaces (front - / backsheet, junction box) of the module with conductive foil. Do this with a few drops of superglue. Alternatively, conductive tape could also be used, such as aluminium or copper tape.
11. Attach the high voltage cable to the primary end of the MC4 Y splitter, and split the other end into 4 separate cables, attaching each of the ends of the cables to the front-sheet, back-sheet, frame and junction box of the solar module respectively, either by soldering them on the foil, and gluing the foil to the locations, or using aluminium tape to attach the ends to the desired places.
12. Connect the shorted output terminals of the module to the primary end of the second MC4 Y splitter, with a voltmeter in between (in series).
13. For both the MC4 Y splitters, one split of the MC4 should run through the DC power supply, while the other split end should run through to a loose wire with a MC4 connector, used to discharge the set-up at the end of the test.
14. The shorted terminals should connect to the positive end of DC power supply, while the ends connected to the front and backsheet, frame and junction box should be connected to the negative end of the DC power supply.

15. Check each of the connections for continuity with a multi-meter to make sure the connections are OK

Set-up

16. Make sure you have created a space in compliance with the Moderate Voltage Protocol
17. Turn on DMM into either current or resistance reading mode, for Micro-Amps
18. Set the voltage to 0V. Turn on the power supply and incrementally increase the voltage to 500V, with an interval of 50V, keep at each interval for 30 seconds and measure the current reading on the voltmeter. Keep aware of any changes in the system before the next increment. Keep at 500V for at least 1 minute.
19. Set the voltage to 0V, turn the power supply OFF and connect the loose MC4 connectors to discharge the system (symbolised in drawing with a switch)
20. Disconnect all apparatus

Evaluation

If the insulation resistance is above 40MOhm or if the current is below 12.5 uA the test is successful.

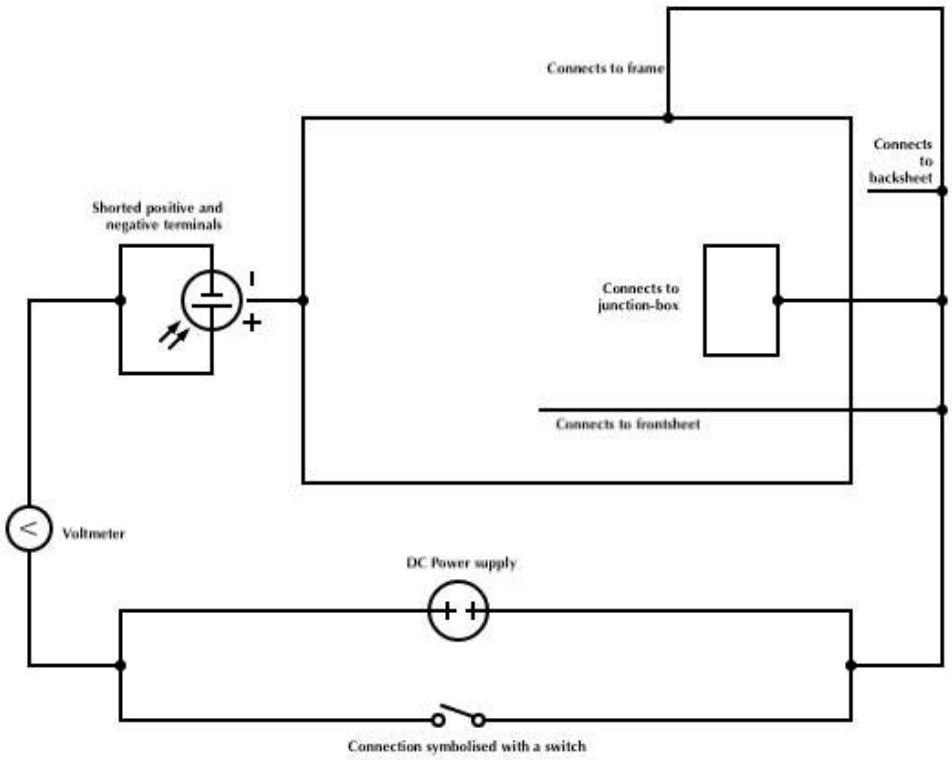


Figure 3 Schematic representation of test set-up

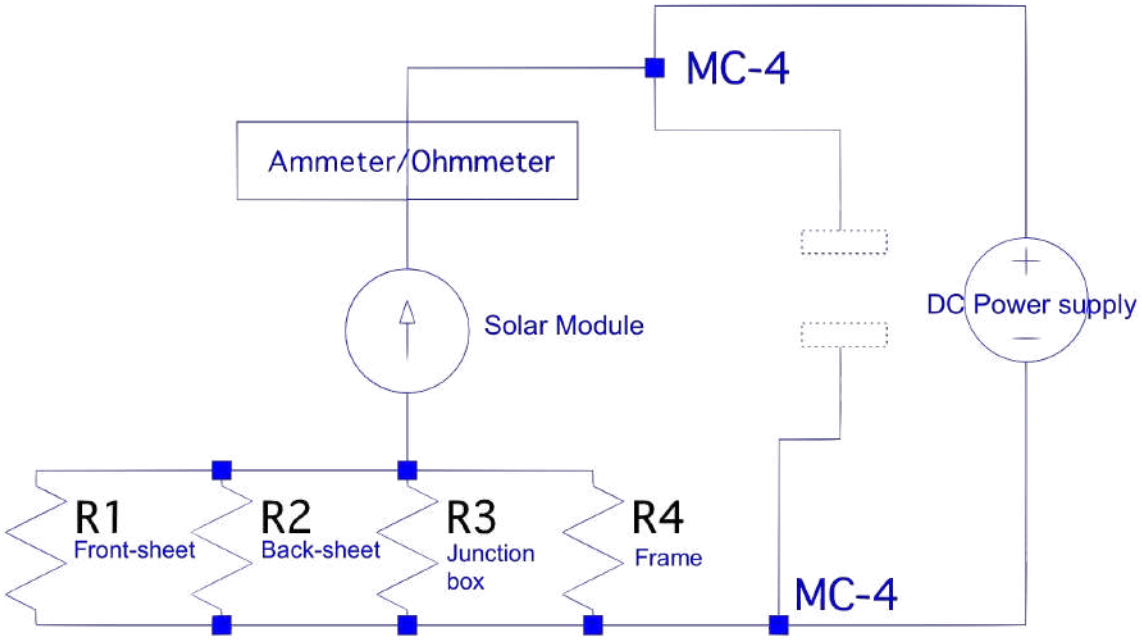


Figure 4 Electric diagram of test set-up

MQT 16 Static Mechanical Load test

The static mechanical load test is designed to ensure the solar module can withstand a minimum static load. According to the IEC regulations, the solar module should be able to withstand a sustained minimum test load of 2400 Pa for at least 1 hour. This translates to various loads depending on the module size.

Considering the risk of failure, sample tests can be done with two panes of PIB sealed glass with no solar cells in between, to prevent solar cell wastage. Keep in mind that the presence of solar cells, and any other structural elements may add to the load withstand ability. For the test, the following glass panes were tested:

Dimensions	Type of glass	Weight required
150 x 300 x 3mm	Non tempered	11 kg
926 x 555 x 3 mm	Non tempered	125 kg
926 x 555 x 4 mm	Tempered	125 kg
Proper solar module	Tempered	To be calculated

Materials

The materials include items that change depending on the size of the module.

1. Sand as the variable load
2. Distribution bag made to house the sand, dimensions according to the glass size
3. Solar module prototype
    1. Glass sheet 2x
    2. Spacers in grid
    3. 6mm IGU channels
    4. 4x IGU corners
4. Wooden frame such that the longer edge of the glass panel rests on the frame, whereas the shorter edge has no support. See Figures below.
5. Camera
6. Tripod
7. Textured ruler
8. Scale
9. 4x mounts intended to fasten the glass prototype to the wooden frame

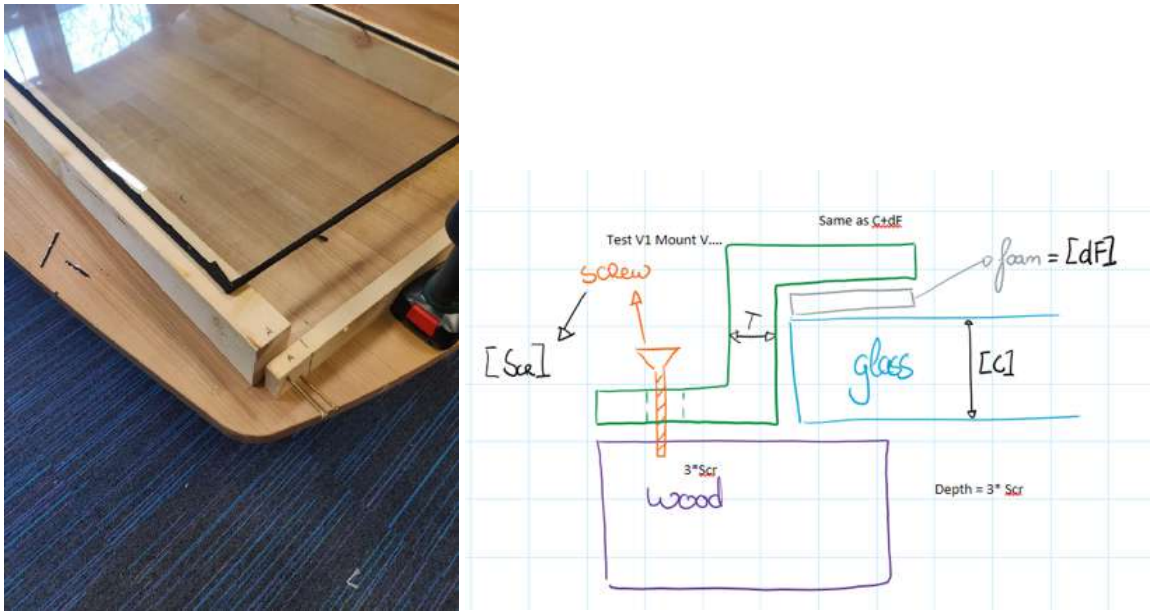


Figure 5 Pictures of test set-up

Procedure Test set-up

1. Cut IGU channels to the lengths of the glass
2. Loose fit IGU channels and corners along the perimeter of the bottom pane of glass
3. Place 2 layers of PIB through the perimeter of the glass
4. Place the IGU channels and corners on the PIB
5. Place 2 layers of PIB on top of the IGU channels
6. Place the top pane of glass firmly on the IGU channels, ensuring it is parallel and in-line with the bottom pane of glass

Procedure Test

7. Mount prototype to frame using the mounts
8. Set up the textured ruler such that the deflection of the glass throughout the test can be accurately measured (see Figure 6)
9. Place a camera such that the deflection can be measured (see Figure 6)
10. Place the distribution bag on the glass pane
11. Add sand into the distribution bag until the target weight is reached, ensuring that the bag continually rests on the glass, and does not fold over the glass pane
12. Start a timer when distribution bag has sand equal to the target weight
13. Leave the bag on for an hour, after which the set-up can be cleared

Figure 6 Deflection ruler set-up

MQT 02 Maximum power determination

Biosphere Solar required a device that could reliably measure the power output of a solar panel, as well as detect issues such as shading or failure of 1 or more of individual solar cells within the module.

Determining the maximum power is typically done through the generation of an IV curve for a solar module. Doing so also allows other attributes of the solar module to be recorded, including the open circuit voltage, the short circuit current and the response of the solar module over various loads. Typically done through laboratory grade machines, an IV curve is can be graphed by making the current generated by the PV module run through a circuit, with increasing resistance every cycle, and recording the associated voltage and current measurements. The variable load is typically accomplished through a series of resistors, or a variable resistor connected to a computer, which is also connected via a control loop to the voltmeter and ammeter. Biosphere Solar opted to construct an open-source IV curve generated dubbed the IV Swinger. With little or no resistance, the short circuit current of the PV module can be measured, when voltage is 0. Conversely, with infinitely high resistance, the open circuit voltage can be measured, when no current is allowed to flow. The generated current stays constant until the resistance increases to the point where not enough electron-hole pairs are being generated to keep up with the demand, leading to a drop in current.

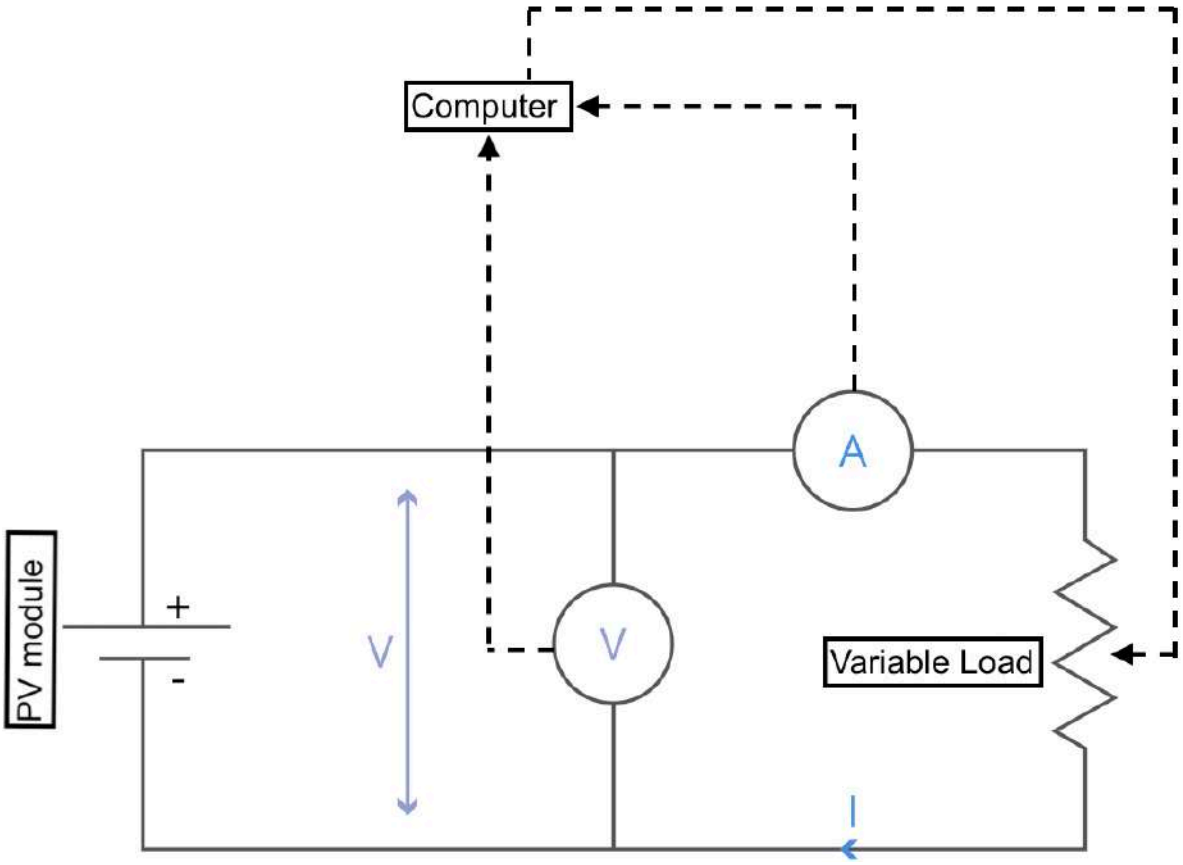


Figure 7 IV Curve generator schematic

The IV Swinger, is custom made circuit board, accompanied by a custom code (Appendix C), that can provide a reliable IV Swinger graph allowing us to measure the power output of our custom solar panels at a fraction of the cost. Simply put, the IV Swinger PCB takes its input from the solar panel, runs it through resistors and capacitors, and uses an op-amp and ADC to convert the analogue signal to a digital one, enabling the user to read a measurement on the computer screen. To delve in the workings of the Swinger, the electric diagram of the PCB must be investigated.

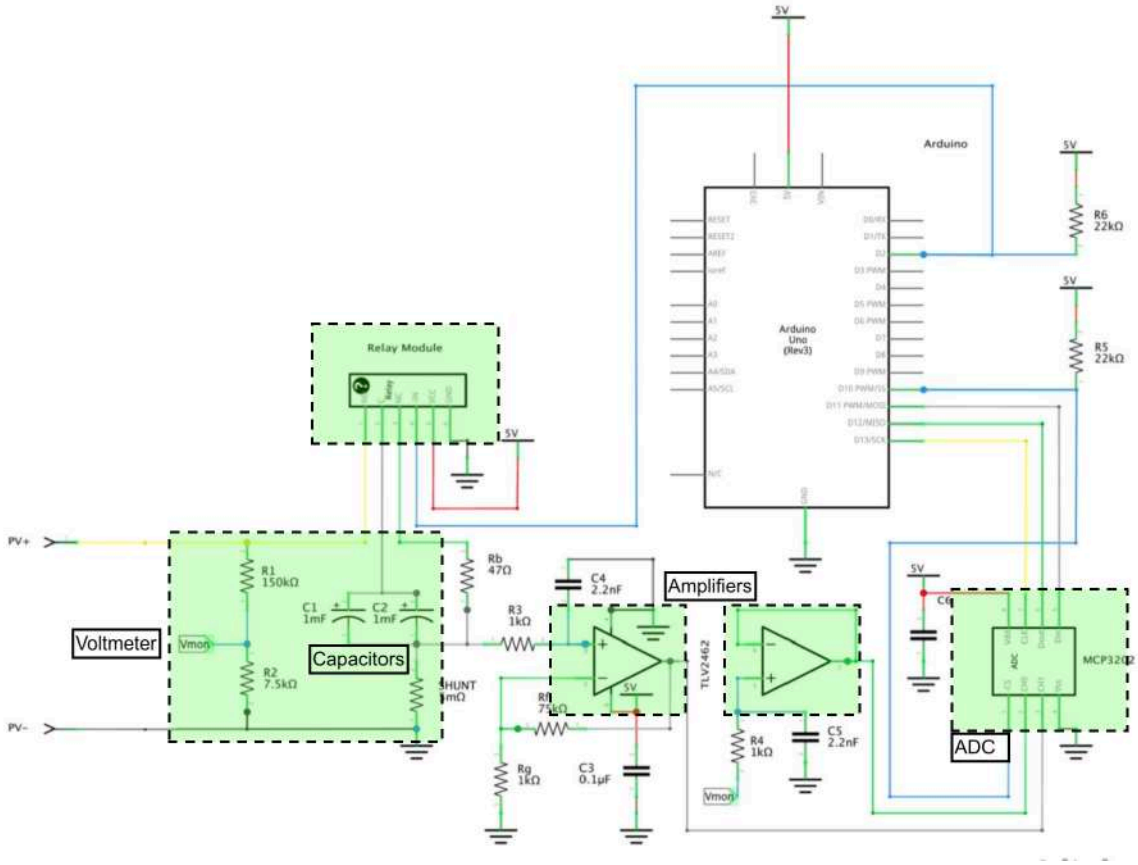


Figure 8 Electric diagram of IV Swinger

While this is a complicated diagram, it helps to focus on the elements that play key roles, highlighted in green, which will be investigated in detail further on, to help with troubleshooting in the future. Firstly, the input from the PV module is given on the left, labelled PV+ and PV-, following the current from PV+, we can see it first flows to the relay module through the NC channel, and then through the C channel in the relay. The current goes through capacitors labelled C1 and C2 and flows back to the PV-. In between the PV+ and PV- we have a voltmeter (Vmon) and resistors, designed to measure the potential across the circuit. Diverting from the capacitors C1 and C2 we see a connection going to the amplifiers, the first of which acts as an ammeter, and the second (connected through the Vmon) acts as a voltage follower (more details in the Op-amp section). The output of both the amplifiers enters the ADC (analogue to digital converter) at which point it is fed into an Arduino board (which processes the data) and transfers the data to the computer.

Let us first more closely examine the role of the relay module and capacitors.

Relay and Capacitors

The first thing that might speak out in Figure 8 is the lack of a variable load as described in Figure 7. The capacitors act as the required variable load; in this simpler model, the loading of the capacitor emulates the function of an increased load. In this instance, the interval between a charge and discharged capacitor accommodates for all the variability.

- A discharged capacitor acts as a short circuit (R = 0 ohms)
- A fully charged capacitor acts as like an open circuit (R = infinite ohms)
- A charging capacitor provides the variable loads (R = 0 ohms → infinite ohms)

The relay module controls the path of the current, when the test is started, the current from the solar panel goes through the capacitor, as the test is completed, the relay switches, connecting the capacitor to a resistor (Rb) and allows it to discharge. The Rb resistor therefore plays a key role in allowing the capacitor to be emptied.

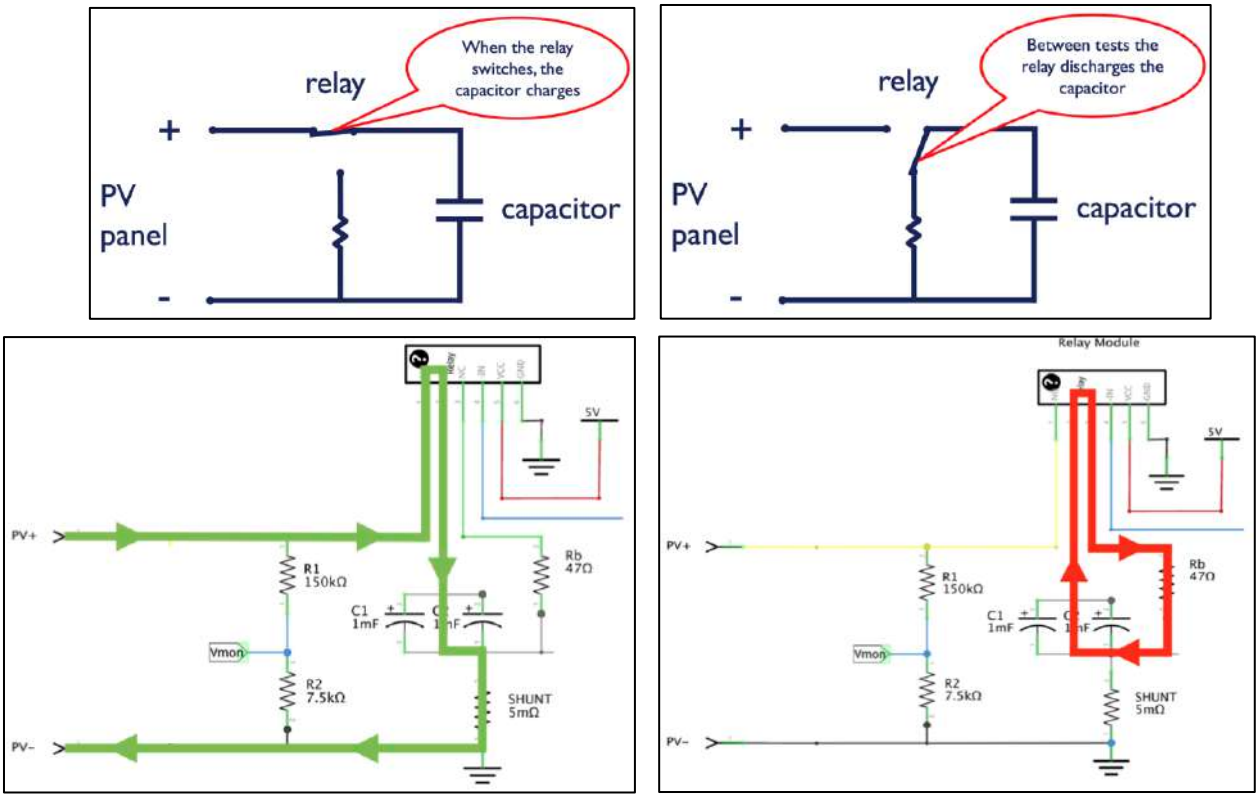


Figure 9 Electric diagram showcasing the role of the relay control

The malfunction of the relay and/or the capacitors is a bottle neck to the data gathering for the generation of an IV curve. If the capacitors are not discharging properly (can be checked with a DMM after every test), check the connection of the Rb resistor, ensuring that it is properly soldered.

Similarly, the relay dictates the path of the current. The relay used in this model is an EMR (electromagnetic relay). Looking at Figure 10, the NO, C and NC connections can be seen on the left, when there is no current flowing through the coil, the electromagnet is off, and the C contact is connected to the NC contact. When there is current flowing through the coil, the electromagnet is on, and it pulls the C contact down to the NO contact. There should be an audible click every time the relay switches. On the right of Figure 10 we also have 3 connections, labelled IN, GND and VCC. The GND refers to the ground connection, the VCC connects to the +5V from the Arduino and the IN also connects to the Arduino, and controls when the relay should be switched. Confusingly, this EMR is what is known as "active low", which means that a low voltage (near GND) activates the relay and a high voltage (near +5V) deactivates the relay.

Referring to Figure 8, the IN path goes to the Arduino and is also connected to the R6 resistor, whose function it is to hold the relay in the inactive state, while the Arduino takes over the control and dictates the relay functioning. In case the relay is behaving strangely the R6 resistor is a common culprit. Similarly, if an "active high" relay adjustments need to be made to the code to function properly.

Figure 10 EMR relay module

To confirm that the relay being used is indeed an "active low", the following procedure can be followed:

- With the Arduino board off:
  - Connect the relay GND to the Arduino GND
  - Connect the relay VCC to the Arduino 5V
  - Connect a cable to the relay IN (do not connect the other end to anything yet)
- Connect the Arduino board to a computer, the Arduino board green LED should be on. The relay module red LED should be on, indicating it is powered. There should be no green LED on the relay module. Assuming it is a new Arduino board, the yellow light should be blinking too. Taking a multimeter, there should be no resistance with the C terminal and the NC terminal, and infinite resistance with the C terminal and NO terminal.
- Next, connect the cable from the relay module IN pin to the GND socket near the yellow blinking LED from the Arduino board, the relay module should now click and the green LED on the relay board should be on. The C terminal should have 0 resistance with NO and infinite resistance with NC (check using a multimeter again).

Op-amp (Ammeter and Voltage follower circuits)

Before diving into the individual ammeter and voltage follower circuits, it is important to understand that both these circuits exist, in a single op-amp unit, which has a dual functionality. Referring to Figure 11, we can see that the two circuits (labelled 1 and 2) are detached and not interconnected. For our purpose, this op-amp unit helps in determining both the voltage and current.

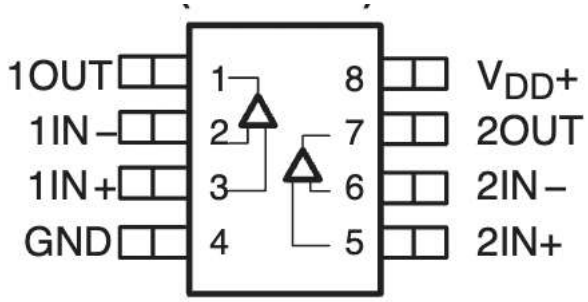


Figure 11 Op-amp internals

Voltmeter circuit

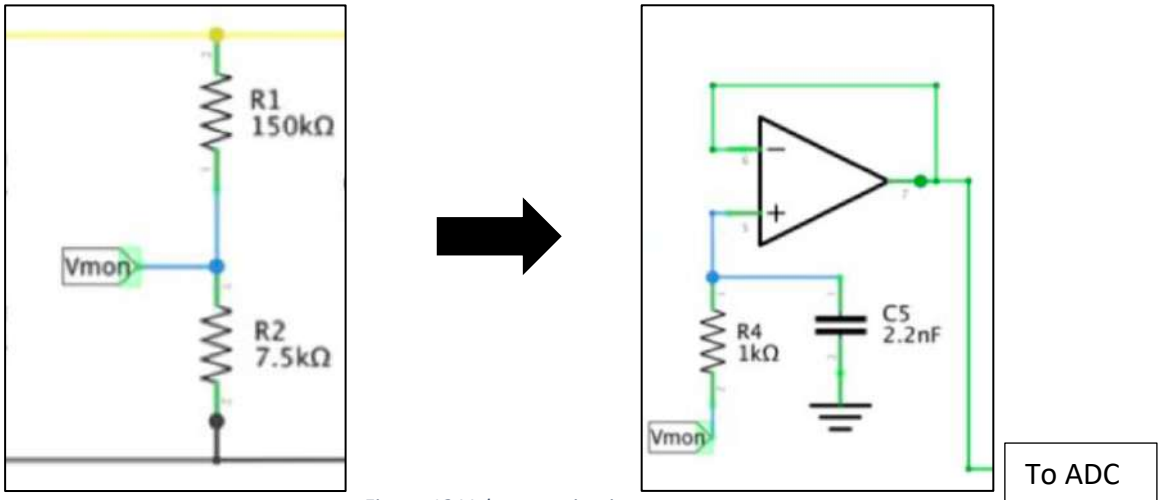


Figure 12 Voltmeter circuitry

The voltmeter circuit begins at the start of the overall circuit, where two resistors and a voltmeter are placed in series between the PV+ and PV- outputs. This PCB is designed to handle voltages up to around 80V, which needs to be scaled down to match the 5V ADC and Arduino voltages, with the resistors serving that function. The values of these resistors are also therefore optimised for a solar module for a Voc (open circuit voltage) of around 80V, if the module has values significantly higher or lower than 80V, the IV swingers' resolution will be lower.

Following this, the first circuit of the Op-amp (seen on the right of Figure 12), acts as a voltage buffer. The output of a voltage buffer is the same as the input, providing no amplification or attenuation to the signal, its function is to deliver the same voltage at a lower current, adjusted for the ADC input. The voltage buffer acts as a high-impedance load,

essentially providing the voltmeter reading to the ADC, at a reasonable and safe current, as required by the ADC. (What is Voltage Follower?, sd)

Ammeter circuit

Measuring current is typically done through hall-effect sensors, which detect the magnetic field generated by the current, but this only works with no other magnetic fields around. Since we are using an electromagnetic relay, the current would instead have to be measured by measuring a voltage drop across a resistor. Which means this circuit does not directly measure the current at all, instead it deduces the current through a corresponding voltage measurement.

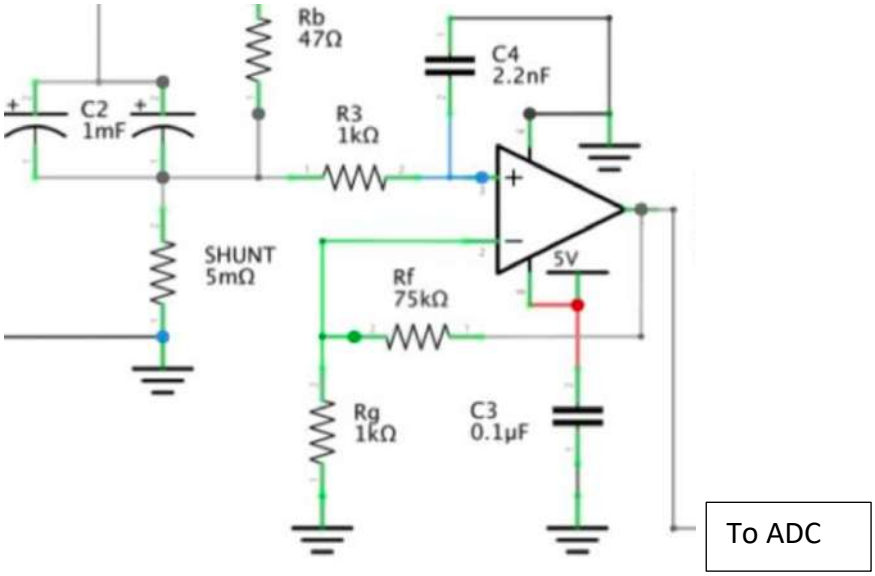


Figure 13 Ammeter circuitry

To not interfere with the circuit, a low resistance, high precision resistor would be required. This is the function of the Shunt resistor, with a resistance of 5mOhms. The reason for choosing a low resistance is to not disturb the circuit, in turn influencing the results. From the Shunt resistor, the current enters the + input of the amplifier. Choosing such a low resistor also means the voltage drop across the resistor is miniscule, to get a proper reading, this drop in potential needs to be amplified, which is where the amplification circuit comes into play. Although the mechanics of the amplification are a little complicated, in such a circuit, the gain (or amplification) is $1+ R_f/R_g$. For this it's important to understand that the resistors R_f and R_g play a role in in amplifying the signal, whereas the resistor R_3 and capacitors C3 and C4 exist to filter out any noise which also gets amplified.

ADC – Analogue to Digital converter

As explained earlier, the ADC turns the measured analogue voltage values (from the voltmeter and ammeter circuits) and turns them into digital signals used for processing the data for the IV curve. It therefore needs two input channels, labelled CH0 and CH1.

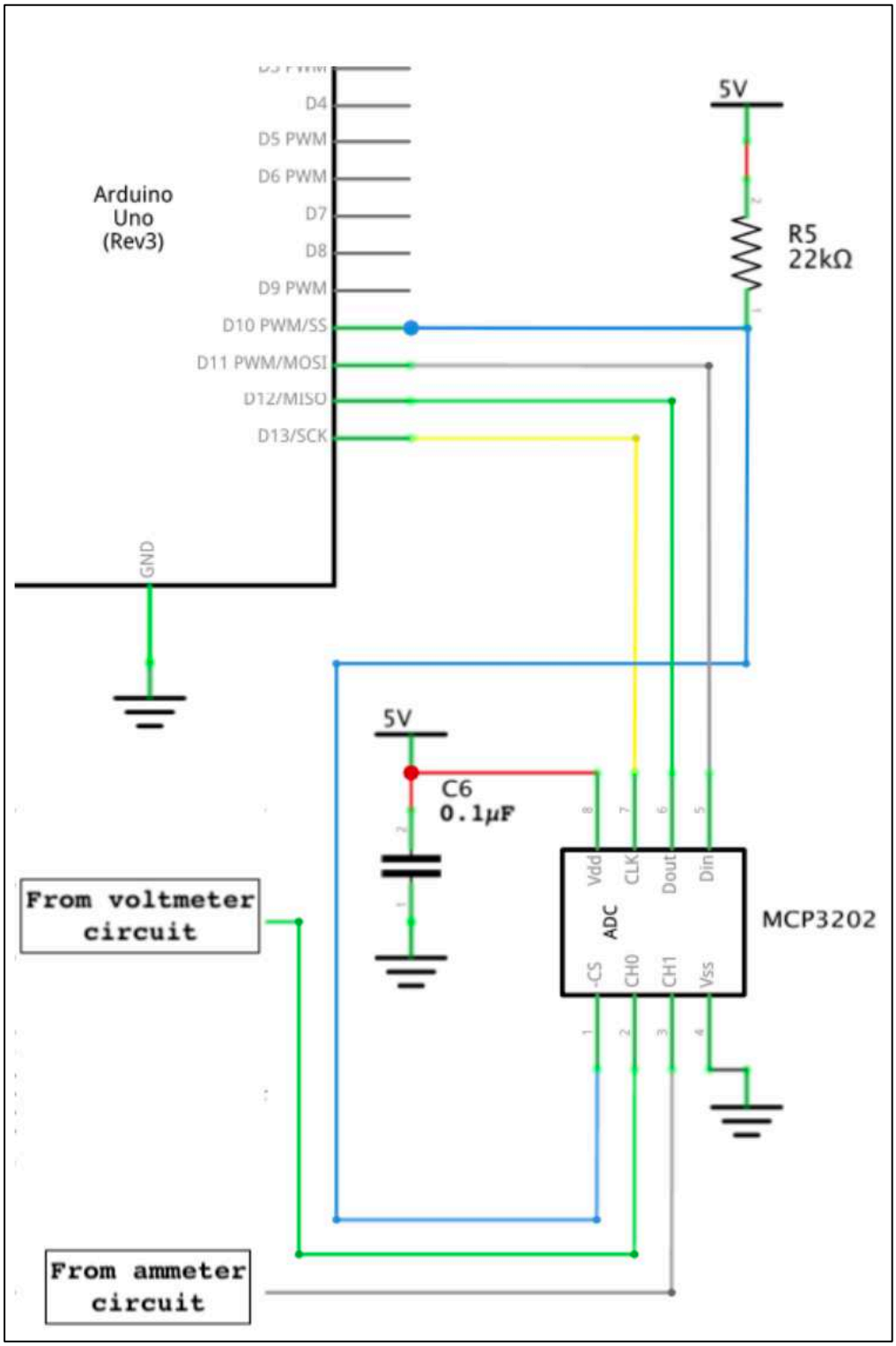


Figure 14 ADC

For the purpose of this report, the inner workings of the ADC are not discussed, nor do I have the technical background knowledge to discuss it in-depth. I did however do troubleshooting for the ADC, and the sanity checks are given below, along with each of the PIN functions.

- The VSS pin is connected to ground. Ground is connected to the Arduino board ground. The PV- input (black binding post) is also connected to ground. This is

important because the ADC voltage measurements are relative to the VSS pin and both the ammeter and voltmeter are measuring voltages that are relative to the PV- input.

- The VDD pin is connected to +5V from the Arduino. It is also connected to a 0.1 µF capacitor, C6, whose other lead is connected to ground. Its purpose is to filter noise from the power supply.
- The CH0 pin is connected to the voltmeter circuit output. This is the Channel 0 input.
- The CH1 pin is connected to the ammeter circuit output. This is the Channel 1 input.
- The CS pin is connected to Arduino pin D10. By convention, the Arduino pin D10 is always used for SS.
- The DIN pin is connected to Arduino pin D11. By convention, Arduino pin D11 is always used for the Din pin.
- The DOUT pin is connected to Arduino pin D12. By convention, Arduino pin D12 is always used for the Dout pin.
- The CLK pin is connected to Arduino pin D13. This is the serial clock, it essentially references the measurements with its own intervals, hence the word clock.

In order to make sure the ADC is functioning properly, a sanity check with the Arduino UNO can also be conducted. With the board connected to a computer, load the code given in Appendix C, and in the serial monitor window of Arduino IDE, paste the command “Config: READ_ADC 20” In case the outputted values are 0, the ADE is not functioning, if the values are non-zero and somewhat consistent, the ADE should be working fine! (Satterlee, sd)

MQT 17 Hail test

The Hail test was conducted to emulate natural hail falling on the solar panels. The requirements of the IEC are given in the table below, they include a range of ice balls, and the associated speed with which they should be launched at the solar panel.

Diameter	Mass	Test velocity	Diameter	Mass	Test velocity
mm	g	m/s	mm	g	m/s
25	7,53	23,0	55	80,2	33,9
35	20,7	27,2	65	132,0	36,7
45	43,9	30,7	75	203,0	39,5

The IEC regulations recommend using a pneumatic pump set up to shoot the ice balls at the solar module. In an attempt for simplicity, the following set-up below was proposed.

Material

1. Solar module
2. Backdrop of black cloth ~5 m2, for better contrast for visual analysis of ice ball speed
3. Silicon moulds for ice spheres of 40mm diameter
4. A kettle
5. Tracker – Video analysis and Modelling tool
6. Computer
7. High speed camera (100 FPS was used, but higher FPS camera would lead to more accurate speed calculations)
8. Freezer at -10ºC

Procedure before test

1. Allow twice boiled water to cool down and pour in the silicone mould for freezing.
2. While in the freezer, set up the test area. Clear a 5x5m area on the ground.
3. Set up the solar module, backdrop and camera as shown in the photo below. The length at which the speed was analysed was 2 meters
4. Conduct a visual analysis for any defects or cracks on the solar module.
5. Conduct an Electroluminescence test to detect for any microcracks on the solar cells.

Figure 15 Test set-up of hail test

Set-up

1. Once the ice balls have been frozen (minimum 1h in the freezer), inspect for cracks and pick out the ones with the least cracks (ideally no cracks)
2. Turn on the camera
3. Standing at 3 meters from the solar panel, throw the ice balls as hard as possible at the solar panel, aiming for the following points:
4. Any corner of the solar module
5. Any edge of the solar module
6. At the interconnections between cells
7. At the face of a solar cell
8. On the module, at any point farthest away from points 1-4
9. Any points, based on judgement that may prove most vulnerable (ie the junction box)
10. After each throw, inspect and note down any visible damage to the solar module, and mark the approximate area where the ice ball struck with a sticker
11. Conduct an Electroluminescence test following all throws, to detect for any crack formation in the solar cells
12. Stop recording
13. Open the video file on the Tracker software, and following the software determine the speed of each throw

Electro-luminescence testing for crack detection

EL testing for solar modules essentially uses the fact that solar cells are photo-diodes, and can act as light emitting sources when current is run through them, instead of solar cells acting as current generators when light is emitted on them. When current is passed through, and photographed through a camera, any cracks on the solar cells invisible to the eye can be captured easily. Cracks create an electric separation, resulting in an inactive cell part, that crack is therefore unable to emit light, appearing dark in the captured photograph.

Material

1. 2-cell solar module
2. Bench power supply upto the Voc of tested module, in this case only upto 2 V is needed
3. 2x 20cm wire with crocodile clips on both ends
4. Sony A58 camera with IR filter removed (follow this video for instructions: https://www.youtube.com/watch?v=0H1Om5AVX2w)
5. A completely dark room with a wall outlet

Procedure

1. Without plugging in the bench power supply, attach the positive end of the solar module to the positive end of the power supply, and the negative end of the power supply to the negative end of the solar module
2. Plug in the power supply but do not turn it on
3. Have the IR camera ready to take a photo of the solar module
4. Turn off the lights to ensure the environment is as dark as possible
5. The solar module should now start emitting IR light, photograph through the IR camera
6. Turn off power supply and turn the lights on

Results and discussion

MQT Insulation test

Voltage	Resistance / Current (including uncertainty)	Comments

An increase in voltage provided generated no current, up until the limit of 500V, this means that there is indeed no electrical continuity between the solar cells and the housing frame of the solar module.

The same test set up can be used to test larger modules up to 1000V as required by the IEC regulations stated in the document IEC 61215-2-2016.

MQT Mechanical load test

Dimensions of glass	Type of glass	Weight required	Weight tested	Pass/fail
150 x 300 x 3mm	Non tempered	11 kg	25 kg	Pass (1h)
926 x 555 x 3 mm	Non tempered	125 kg	125 kg	Fail (33min)
926 x 555 x 4 mm	Tempered	125 kg	125 kg	Pass (1h)
Proper solar module	Tempered	To be calculated	N/A	N/A

The first test, set with smaller dimensions glass was intended as a test for the proposed test set-up, to highlight any issues that might happen with the larger glass panes. The successful results of a smaller glass pane are not transferable to a larger sized panel, since the force of the load on an object increases as the object gets longer (see Figure 17).

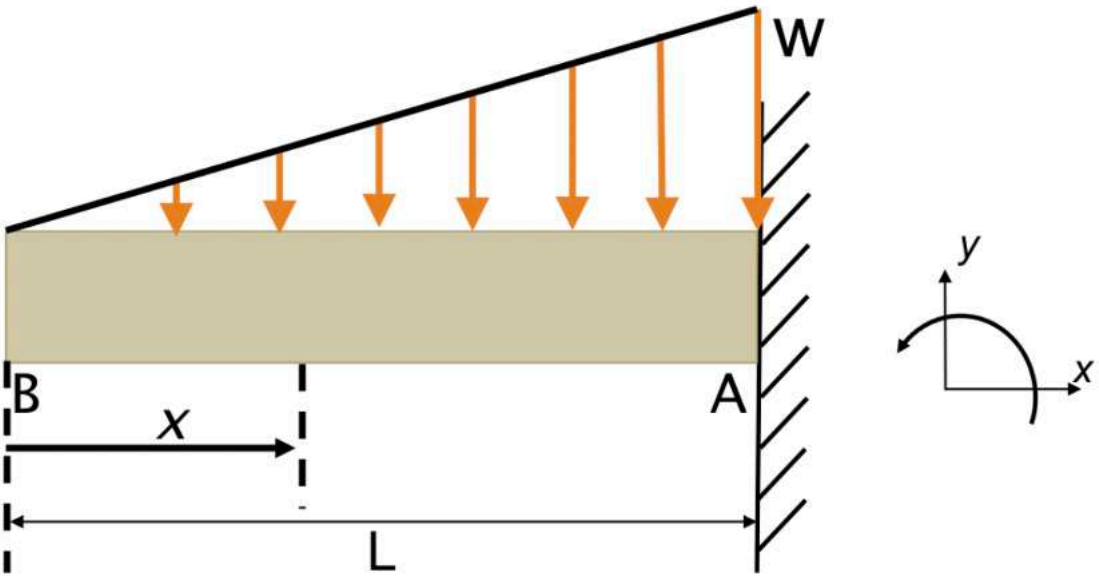


Figure 16 Load distribution

This is proven in the second test, where a similar 3mm non-tempered glass of higher size could not withstand the required 2400Pa of force (125 kg for the given area) for more than 33 min, leading to shattering of the glass (Figure 18).

Under the same set-up a 4mm tempered glass could withstand a similar load of 125kg, for 1 hour. It withstood this load with a midpoint deflection of 3mm.

Although successful, the test does not properly emulate the conditions in which a solar module is typically installed on roof-tops, by only providing support on the edge of the frames, a lot of the force is concentrated at the centre of the panel, when typically, a solar panel is mounted more securely. The IEC regulations clearly state the test should be conducted with the manufacturer's prescribed method of mounting. Having said that, Biosphere Solar can be assured that 4mm thick, tempered glass should be sufficient,

considering more rigid mounting. Once the final design of the solar module and mounting system is decided upon, the test should be repeated with 3mm non-tempered glass to ascertain whether a thinner (therefore lighter) and non-tempered (therefore cheaper) glass can work with a more rigid mounting frame under the same 2400 Pa pressure.

Figure 17 Failed static load test

MQT 02 Maximum power determination

There were 3 categories of solar modules that were tested with the IV swinger to get their Max power.

- Second-hand decommissioned solar power produced by Shell (put details here)
- Sample solar module (28 cell module (7x4))
- 2 cell solar module

The purpose of the IV Swinger was to give reliable results such that the maximum power of the in-house solar modules could be checked at any given time. The highest degree of accuracy was not required, since the purpose was to make sure we had a power meter that could detect shading, solar cell failure issues and just general power trouble-shooting capabilities.

First, the results for the decommissioned Shell solar module and the sample solar module are discussed. The IV swinger generated a typically shaped IV curve for the Shell panel, getting a short circuit current of around 2.5A and an open circuit voltage of around 30.5 V. The MPP, or maximum power point is determined to be 24.8V and 2.28A, leading to a power output of 56.5W. Aside from the dimension's composition of the solar module, the power generation is also highly dependent on the incident irradiation. Similarly for the sample solar module developed in-house the open circuit voltage is 17.4 V (makes sense since it has fewer cells), and a short circuit current of 3.17 Amps, with a maximum power rating of 39.5 W.

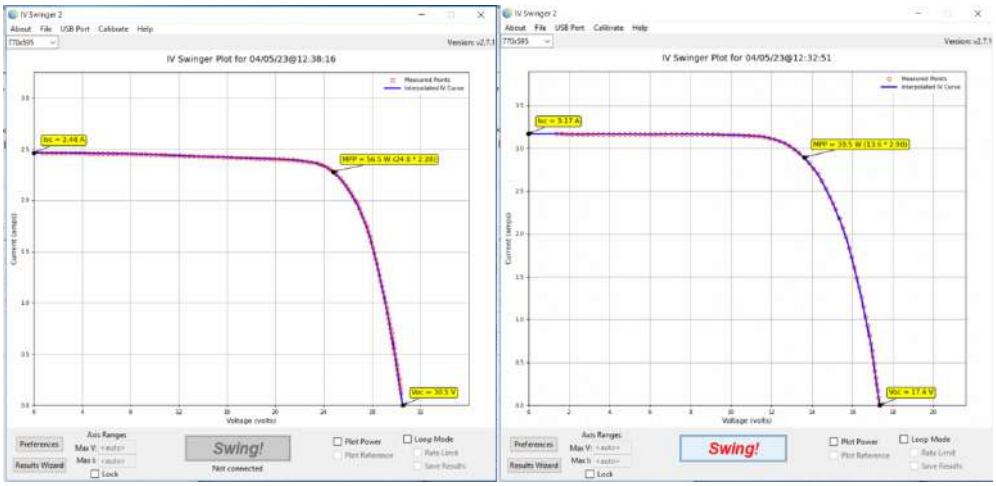


Figure 18 IV graphs generated by IV Swinger

The performance of a solar panel is highly dependent on the incident light, or the irradiance. A typical relationship between irradiance (G) and current (I)/voltage (V) is quite straightforward, however, the performance of a solar panel is also dependent on the temperature at which the solar cells are at, with reducing performance at temperatures above 25ºC. This adds another variable in the relationship, for which the IEC gives the following formula:

$$I_{SC} = \frac{G_{SC} * I_m}{G_m} * [1 - \alpha(T_{SC} - T_m)]$$

$(\ )_m$ stands for the value during measurement, whereas $(\ )_{SC}$ refers to standard conditions, which, in the case for irradiance is set at 1000 W/m^2 by convention. All standard condition voltage, current and power ratings by convention are given at presumed irradiance of 1000 W/m^2 and at 25ºC. α is the temperature coefficient, solar panels typically degrade in performance by α% for every degree above 25ºC. (Photovoltaic (PV) module performance testing and energy rating - Part 1: Irradiance and temperature performance measurements and power rating)

Most temperature coefficients are between 0.3 – 0.5 %, assuming a value of 0.4%, means for 1 degree above 25ºC, the solar module produces energy that is (1-0.4% = 99.96%) of its rated power. Even considering an ambient temperature of 35ºC, the solar module will perform at 96%, considering the measurement was taken on May 4th, the detriment to performance can be largely ignored. This means for the sake of estimation, the simpler formula given below can be used:

$$I_{SC} = \frac{G_{SC} * I_m}{G_m}$$

Biosphere had no access to a pyranometer, or a similar device that could measure irradiance, although they have access to the Green Village's weather data, the service was not available during the time of my internship. In any case, to validate the data collected, the irradiance data collected at the KNMI institute in Rotterdam was used, which provided an average irradiance per hour since 1991 for each day of the year. Although this is not an ideal solution, it should allow us to roughly validate the I_{sc} value of the tested solar panels.

PANEL	**I_{SC} (A)**	**IRRADIANCE (W/M^2)**	**I_{SC} (A) (CORRECTED)**	**Expected Isc (A)**
SHELL PANEL	2.28	434	5.7	N/A
Sample panel	3.17	434	7.3	6.3

The test, being conducted at 12:30 on the 5th of March (the app is American so the date and month are reversed), based on the KMNI data, the average irradiance between 12:30 and 13:30 is 434 W/m^2, using the formula given above the expected I_{sc} should be 5.7 Amps for the Shell panel, and 7.3 Amps for the solar module developed at Biosphere Solar. (Meteorological data portal, sd)

The Biosphere Solar module follows a typical template of 28 cells in series, each solar cell has an established I_{sc} value of 6.3, based on the manufacturers data sheet for the Maxeon Gen2 solar cell. This discrepancy is likely because the irradiance on the actual day was higher than the average recorded by KMNI, since the actual short circuit current cannot be higher than the capacity of the solar cells. The numbers do seem to be in a range that is expected, for example, if the irradiance was 500 W/m^2, the I_{sc} would have correctly been 6.3, instead of 7.3, which only represents a small change from the average data at KMNI.

The next set of data shows the IV curves for 4 separate 2-cell modules. The PVMD lab at TU Delft allowed us to use the solar emulator station, this station was optimised for a 2-cell set-up and to validate the constructed IV Swinger, 4 separate 2-cell modules were tested with the lab's IV curve generator, and the results were compared with the curves generated by the IV Swinger.

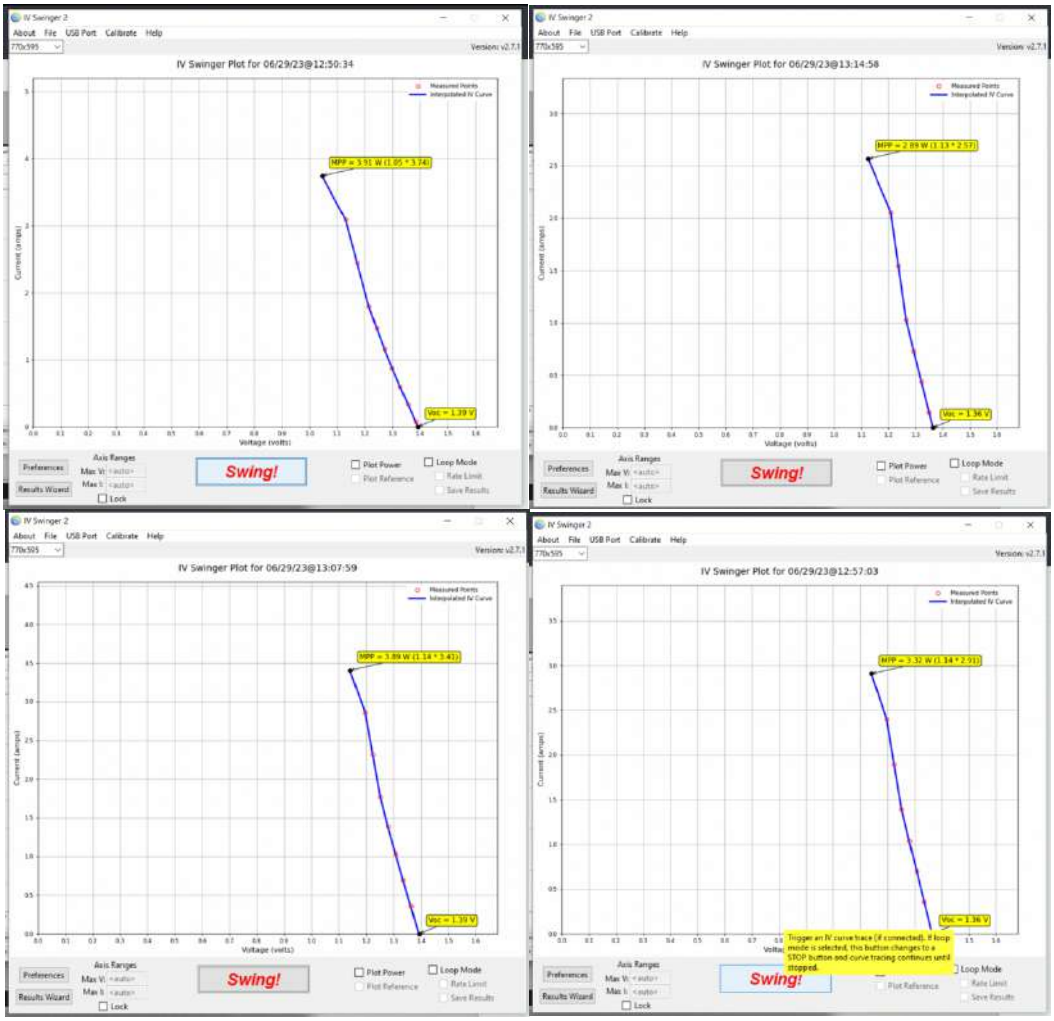


Figure 19 IV curves of 2-cell modules generated by the IV Swinger

Before comparing values with PVMD lab's solar emulator, the strange, almost linear relationship can be observed with the 2-cell module IV curves, which should not be the case. The entire initial part of the curve, a horizontal line is missing from these curves. After more troubleshooting with the IV swinger, I believe I know the cause of this incomplete graph.

The circuity of the IV Swinger is optimised for full scale solar panels, as such the internal resistances of the IV Swinger are too high for a 2-cell module (in relative terms, these resistances would be negligible for a solar module that has >10 cells). A typical IV curve should display a horizontal line at first, showcasing the solar modules capability to deliver a certain amount of current without a drop in voltage. In the case for the IV Swinger, even when simulating a 'short circuit' current, the internal resistance (for example the Rb resistor mentioned in the IV Swinger section) already causes a drop in voltage, distorting the IV

curve. The only relevant information we can get from these graphs is the recorded V_{oc} or the open circuit voltage.

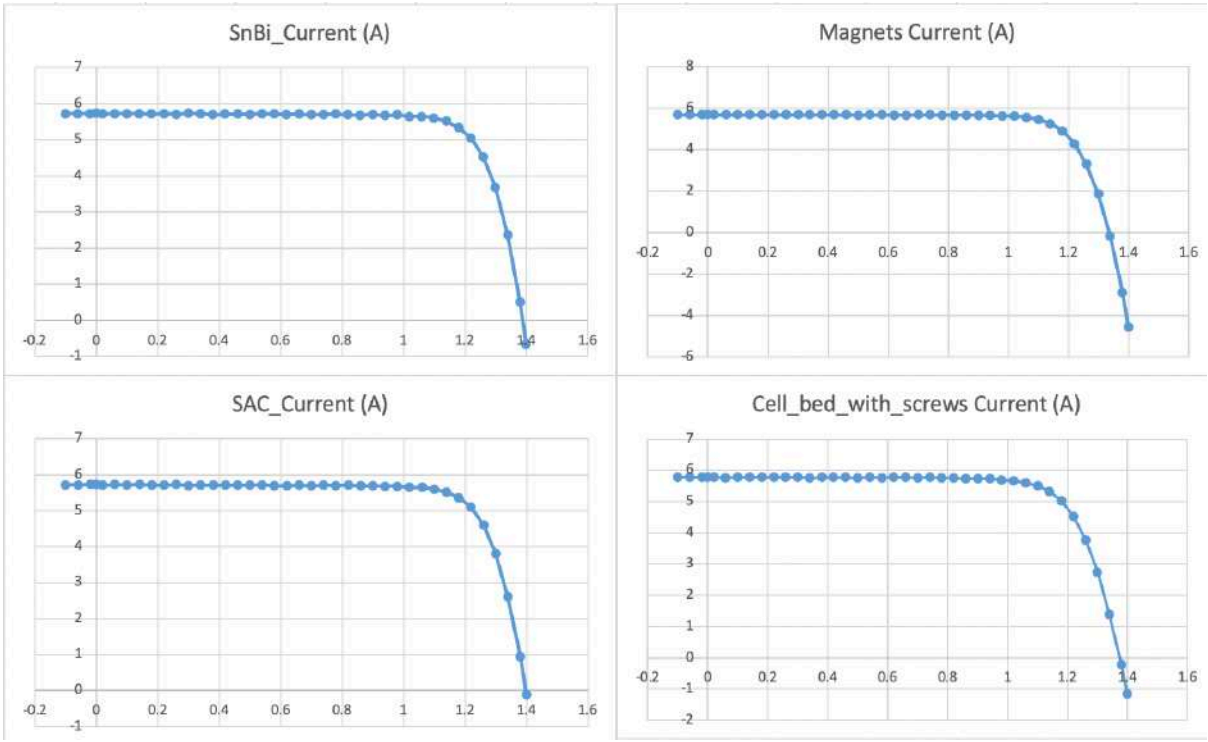


Figure 20 IV curves of 2 cell modules generated by solar emulator at PVMD

The V_{oc} can be approximated by predicting the x-axis intercept in each case. For the SnBi and SAC modules, the V_{oc} is clearly around 1.39 – 1.4 Volts, which is also confirmed by the IV Swinger, whereas for the Magnets and Screws sample, the IV Swinger V_{oc} is 1.36 for both. Digging into the data, the Magnets V_{oc} had a value of 1.34, whereas the screws value had a value of 1.37. The table below summarises these results.

Solar module	PVMD Lab V_{oc}	IV Swinger V_{oc}
SnBi	1.39	1.39
SAC	1.40	1.39
Magnets	1.34	1.36
Cell bed with screws	1.37	1.36

It is safe to say that the open circuit voltage values determined by the IV Swinger are, at most 1.5% different than the open circuit voltage values calculated by the set-up at PVMD. This lends confidence to the data gathered by the IV Swinger; however, no such check can be done for the current measurement. I would highly suggest Biosphere Solar invests in a high quality pyranometer, with which accurate irradiance data can be obtained, which is not only time sensitive, but by placing the pyranometer close to the solar module, can be highly local as well.

MQT Hail test

The picture below shows the positions on the panel where the ice ball struck the solar module, at 4 different locations. Each of the balls used had a diameter of 40mm. According to the density of ice, weighing around 30 grams for each of the balls, with slight variations.

Diameter (mm)	Weight (g)	Speed (m/s)
40	30.8	15.15
40	31.1	16.23
40	30.1	14.98
40	30.3	17.45

According to the IEC regulations, the mass of spheres to be tested for a hail test are given in the table below, along with their ideal test speeds and masses.

Diameter mm	Mass g	Test velocity m/s	Diameter mm	Mass g	Test velocity m/s
25	7,53	23,0	55	80,2	33,9
35	20,7	27,2	65	132,0	36,7
45	43,9	30,7	75	203,0	39,5

Since the mass lies in between 20.7 and 43.9 grams, this is in line with expectations. Nevertheless, the act of throwing an ice ball clearly does not generate enough speed, as required by the IEC regulations. Considering this, I would highly recommend formalising the set-up and using a pneumatic device as suggested in the IEC documents.

As for improvements to the current set-up, perhaps placed the solar module on the ground and throwing ice balls from an elevated platform, as to let gravity aid in increasing the speed might be worth a try, although the balls must go significantly faster, and aiming reliably might prove to be a bigger issue in this case.

Lastly, I would also suggest a camera with a higher frame rate be used, in the 2 meters given, only 3 data points were gathered, and motion blur drastically reduced the accuracy of the software used.

Figure 21 Motion blur during hail test data analysis

MQT EL (Electro-luminescence) testing

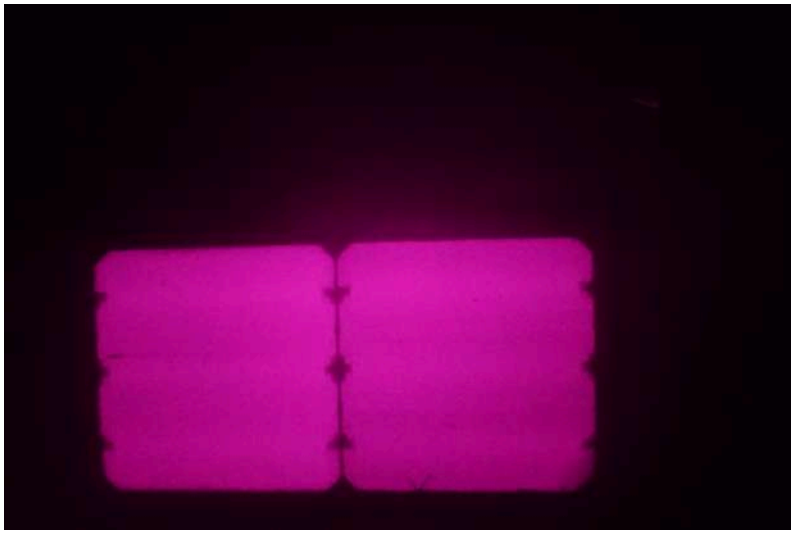


Figure 22 EL of module 1

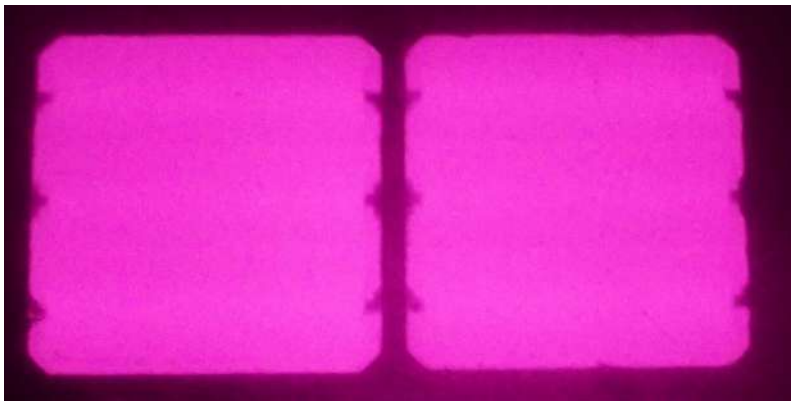


Figure 23 EL of module 2

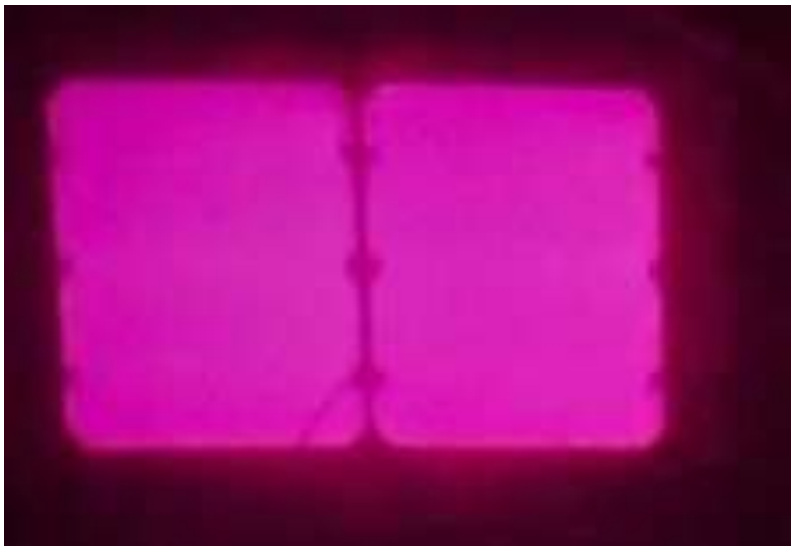


Figure 24 EL of module 3, showing a crack on the bottom right corner of the left solar cell

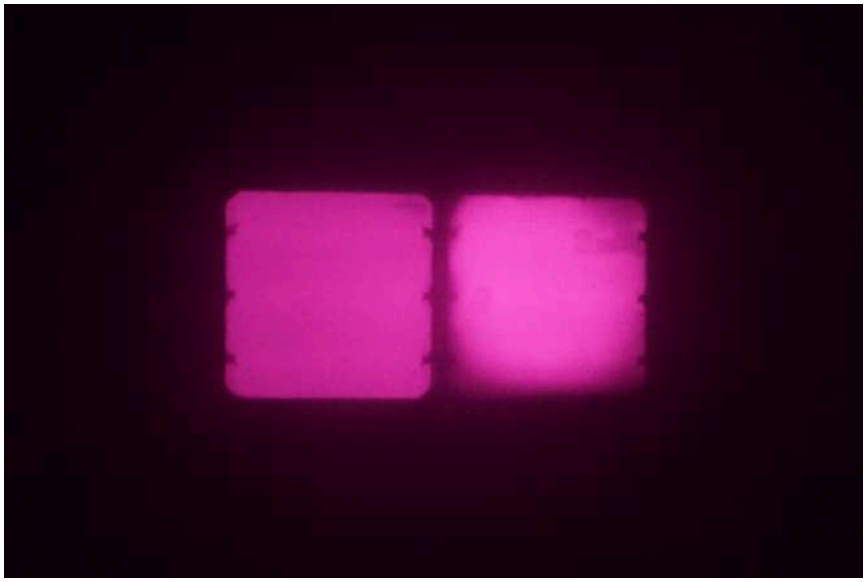


Figure 25 EL of solar module with dead zones (right solar cell)

EL testing shows the solar cells glowing red (radiating IR light) and could be used to tell problems not seen by the naked eye. Figure 24 for example shows a crack, such a prominent crack can sometimes be visible without EL testing, through discrepancies in reflected light. Figure 25 shows dead zones. This solar module was a test sample used to try a new soldering plate, and prolonged contact with a hot surface led to zones in the solar cell that do not facilitate current as well as they should. In this case, even a brighter red cell on the right can be seen, owing to the build-up of current.

EL testing should be done carefully, and prolonged reverse bias with the power supply should be avoided. For larger solar modules, a stronger power supply would be needed.

Appendix A: Moderate Voltage protocol

Moderate Voltage Protocol

Introduction

This protocol is based on the EEEL Safety Rules for Moderate and High Voltages and on practices from the TU Delft Solar Boat Team. It is **mandatory** to have read the safety rules before engaging in any procedure involving moderate voltages (120V-1000V RMS AC or DC). The protocol comprises of references with must be read and a checklist that must be **completely** ticked off and checked and signed by a third person.

Mandatory Practices

You must always:

- Have one hand in your pocket unless necessary for a certain action. Think twice about what you do with your second hand
- If you ever need to use a tool, make sure it is insulated with electrical tape
- Don't switch anything on before the setup is complete

Check-List

- You have read the EEEL Safety Rules
- You have passed the Moderate Voltage Exam
- You are wearing :
    - Insulating high voltage gloves that comply with EN60903
    - A fluorescent vest
    - Safety glasses
    - A piece of clothing with a pocket to place one hand in
    - Rubber soles
- You have read the testing protocol for the relevant test
- You have indicated a space where the test will be performed and informed any nearby individuals that high voltages are present
- If the test is performed outdoors, rain forecast must be <10% chance that day.
- There are no other objects in the indicated space than absolutely necessary
- There is a fire extinguisher at the perimeter of the space

Checked and signed by:

[Name] on [Date]

_____

Appendix B: Photos of solar modules tested and specifications

Appenix C – Arduino code

```
#define VERSION "1.4.6"         // Version of this Arduino sketch

// Uncomment one or more of the following to enable the associated
// feature. Note, however, that enabling these features uses more of the
// Arduino's SRAM, so we have to reduce the maximum number of IV points
// accordingly to prevent running out of memory.
//#define DS18B20_SUPPORTED
//#define ADS1115_PYRANOMETER_SUPPORTED
//#define CAPTURE_UNFILTERED_ISC_POLL     // Debug only
//#define CAPTURE_UNFILTERED_POST_ISC     // Debug only

#if defined(CAPTURE_UNFILTERED_ISC_POLL) || \
    defined(CAPTURE_UNFILTERED_POST_ISC)
#define CAPTURE_UNFILTERED
#endif

#include <SPI.h>
#include <EEPROM.h>

#ifdef DS18B20_SUPPORTED
#include <OneWire.h>
#include <DallasTemperature.h>
#define DS18B20_SRAM 44
#else
#define DS18B20_SRAM 0
#endif

#ifdef ADS1115_PYRANOMETER_SUPPORTED
#include <Wire.h>
#include <Adafruit_ADS1015.h>
#define ADS1115_SRAM 224
#define ADS1115_IRRADIANCE_POLLING_LOOPS 10
#define ADS1115_TEMP_POLLING_LOOPS 5
#define MAX_STABLE_TEMP_ERR_PPM 5000    // 5000 = 0.5%
#define MAX_STABLE_IRRAD_ERR_PPM 10000  // 10000 = 1%
#else
#define ADS1115_SRAM 0
#endif

#ifdef CAPTURE_UNFILTERED
#define MAX_UNFILTERED_POINTS 125
#define UNFILTERED_SRAM ((MAX_UNFILTERED_POINTS*4)+12)
#else
#define UNFILTERED_SRAM 0
#endif

#define MAX_UINT (1<<16)-1     // Max unsigned integer
#define MAX_INT (1<<15)-1      // Max integer
#define MAX_ULONG (1LL<<32)-1  // Max unsigned long integer
#define MAX_LONG (1<<31)-1     // Max long integer
#define MAX_MSG_LEN 40         // Maximum length of a host message
#define MSG_TIMER_TIMEOUT 1000 // Number of times to poll for host message
#define CLK_DIV SPI_CLOCK_DIV8 // SPI clock divider ratio
#define SERIAL_BAUD 57600      // Serial port baud rate
#define ADC_MAX 4096.0         // Max count of ADC (2^^num_bits)
#define ADC_SAT (ADC_MAX-1)    // ADC saturation count
#define ADC_CS_PIN 10          // Arduino pin used for ADC chip select
```

```
#define RELAY_PIN 2            // Arduino pin used to activate relay (or
SSR1)
#define ONE_WIRE_BUS 3         // Arduino pin used for one-wire bus
(DS18B20)
#define SECOND_RELAY_PIN 4     // Arduino pin used to activate 2nd
relay/SSR5
#define SSR2_PIN 6             // Arduino pin used to activate SSR2 (if
exists)
#define SSR2_ACTIVE HIGH       // SSR2 is active high
#define SSR2_INACTIVE LOW      // SSR2 is active high
#define SSR3_PIN 7             // Arduino pin used to activate SSR3 (if
exists)
#define SSR3_ACTIVE LOW        // SSR3 is active low
#define SSR3_INACTIVE HIGH     // SSR3 is active low
#define FET3_PIN 9             // Arduino pin used to activate FET3 (if
exists)
#define FET3_ACTIVE HIGH       // FET3 is active high
#define FET3_INACTIVE LOW      // FET3 is active high
#define SSR4_PIN 8             // Arduino pin used to activate SSR4 (if
exists)
#define SSR4_ACTIVE LOW        // SSR4 is active low
#define SSR4_INACTIVE HIGH     // SSR4 is active low
#define SSR6_PIN 5             // Arduino pin used to activate SSR6 (if
exists)
#define SSR6_ACTIVE LOW        // SSR6 is active low
#define SSR6_INACTIVE HIGH     // SSR6 is active low
#define CS_INACTIVE HIGH       // Chip select is active low
#define CS_ACTIVE LOW          // Chip select is active low
#define VOLTAGE_CH 0           // ADC channel used for voltage measurement
#define CURRENT_CH 1           // ADC channel used for current measurement
#define VOC_POLLING_LOOPS 400  // Number of loops measuring Voc
#define FULL_MAX_IV_POINTS 275 // Max number of I/V pairs to capture
#define IV_POINT_REDUCTION ((DS18B20_SRAM+ADS1115_SRAM+UNFILTERED_SRAM)/4)
#define MAX_IV_POINTS (FULL_MAX_IV_POINTS - IV_POINT_REDUCTION)
#define MAX_IV_MEAS 1000000    // Max number of I/V measurements (inc
discards)
#define I_CH_1ST_WEIGHT 5      // Amount to weigh 1st I ADC value in avg
calc
#define I_CH_2ND_WEIGHT 3      // Amount to weigh 2nd I ADC value in avg
calc
#define MIN_ISC_ADC 100        // Minimum ADC count for Isc
#define MAX_ISC_POLL 5000      // Max loops to wait for Isc to stabilize
#define ISC_STABLE_ADC 5       // Stable Isc changes less than this
#define MAX_DISCARDS 300       // Maximum consecutive discarded points
#define MIN_VOC_ADC 10         // Minimum value for Voc ADC value
#define ASPECT_HEIGHT 2        // Height of graph's aspect ratio (max 8)
#define ASPECT_WIDTH 3         // Width of graph's aspect ratio (max 8)
#define TOTAL_WEIGHT (I_CH_1ST_WEIGHT + I_CH_2ND_WEIGHT)
#define AVG_WEIGHT (int) ((TOTAL_WEIGHT + 1) / 2)
#define EEPROM_VALID_VALUE 123456.7890    // Must match IV_Swinger2.py
#define EEPROM_RELAY_ACTIVE_HIGH_ADDR 44  // Must match IV_Swinger2.py
#define SSR_CAL_USECS 3000000   // Microseconds to perform SSR current cal
#define SSR_CAL_RD_USECS 100000 // Microseconds to read/average current
#define CMD_BDGP_READ_ITER 1000 // Bandgap iterations (on READ_BANDGAP
command)
#define GO_BDGP_READ_ITER 1000  // Bandgap iterations (on every Go command)

// Compile-time assertion macros (from Stack Overflow)
#define COMPILER_ASSERT(predicate) _impl_CASSERT_LINE(predicate,__LINE__)
#define _impl_PASTE(a,b) a##b
#define _impl_CASSERT_LINE(predicate, line) \
```

```
    typedef char
_impl_PASTE(assertion_failed_on_line_,line)[2*!!(predicate)-1];

// Compile-time assertions
COMPILER_ASSERT(MAX_IV_POINTS >= 10);
COMPILER_ASSERT(MAX_IV_MEAS <= (unsigned long) MAX_ULONG);
COMPILER_ASSERT(TOTAL_WEIGHT <= 16);
COMPILER_ASSERT(ASPECT_HEIGHT <= 8);
COMPILER_ASSERT(ASPECT_WIDTH <= 8);

// Global variables
char relay_active;
char relay_inactive;
int clk_div = CLK_DIV;
int max_iv_points = MAX_IV_POINTS;
int min_isc_adc = MIN_ISC_ADC;
int max_isc_poll = MAX_ISC_POLL;
int isc_stable_adc = ISC_STABLE_ADC;
int max_discards = MAX_DISCARDS;
int aspect_height = ASPECT_HEIGHT;
int aspect_width = ASPECT_WIDTH;
const static char ready_str[] PROGMEM = "Ready";
const static char config_str[] PROGMEM = "Config";
const static char go_str[] PROGMEM = "Go";
const static char clk_div_str[] PROGMEM = "CLK_DIV";
const static char max_iv_points_str[] PROGMEM = "MAX_IV_POINTS";
const static char min_isc_adc_str[] PROGMEM = "MIN_ISC_ADC";
const static char max_isc_poll_str[] PROGMEM = "MAX_ISC_POLL";
const static char isc_stable_adc_str[] PROGMEM = "ISC_STABLE_ADC";
const static char max_discards_str[] PROGMEM = "MAX_DISCARDS";
const static char aspect_height_str[] PROGMEM = "ASPECT_HEIGHT";
const static char aspect_width_str[] PROGMEM = "ASPECT_WIDTH";
const static char write_eeprom_str[] PROGMEM = "WRITE_EEPROM";
const static char dump_eeprom_str[] PROGMEM = "DUMP_EEPROM";
const static char relay_state_str[] PROGMEM = "RELAY_STATE";
const static char second_relay_state_str[] PROGMEM = "SECOND_RELAY_STATE";
const static char do_ssr_curr_cal_str[] PROGMEM = "DO_SSR_CURR_CAL";
const static char read_bandgap_str[] PROGMEM = "READ_BANDGAP";
const static char read_adc_str[] PROGMEM = "READ_ADC";

#ifdef DS18B20_SUPPORTED
// Global setup for DS18B20 temperature sensor
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
int num_ds18b20s;
#endif
#ifdef ADS1115_PYRANOMETER_SUPPORTED
// Global setup for ADS1115-based pyranometer
Adafruit_ADS1115 ads1115;
#endif

void setup()
{
  bool host_ready = false;
  char incoming_msg[MAX_MSG_LEN];

  // Get relay type from EEPROM (active-low or active-high)
  relay_active = get_relay_active_val();
  relay_inactive = (relay_active == LOW) ? HIGH : LOW;

  // Initialization
```

```
  pinMode(ADC_CS_PIN, OUTPUT);
  digitalWrite(ADC_CS_PIN, CS_INACTIVE);
  pinMode(RELAY_PIN, OUTPUT); // Also SSR1
  digitalWrite(RELAY_PIN, relay_inactive);
  pinMode(SECOND_RELAY_PIN, OUTPUT); // Also SSR5
  digitalWrite(SECOND_RELAY_PIN, relay_inactive);
  pinMode(SSR2_PIN, OUTPUT);
  digitalWrite(SSR2_PIN, SSR2_ACTIVE);
  pinMode(SSR3_PIN, OUTPUT);
  digitalWrite(SSR3_PIN, SSR3_INACTIVE);
  pinMode(FET3_PIN, OUTPUT);
  digitalWrite(FET3_PIN, FET3_INACTIVE);
  pinMode(SSR4_PIN, OUTPUT);
  digitalWrite(SSR4_PIN, SSR4_ACTIVE);
  pinMode(SSR6_PIN, OUTPUT);
  digitalWrite(SSR6_PIN, SSR6_ACTIVE);
  Serial.begin(SERIAL_BAUD);
  SPI.begin();
  SPI.setClockDivider(clk_div);
  set_up_bandgap();
#ifdef DS18B20_SUPPORTED
  // DS18B20 temperature sensor init
  sensors.begin();
  num_ds18b20s = sensors.getDS18Count();
  if (num_ds18b20s) {
    sensors.setResolution(10);
  }
#endif
#ifdef ADS1115_PYRANOMETER_SUPPORTED
  ads1115.begin();
#endif

  // Print version number
  Serial.print(F("IV Swinger2 sketch version "));
  Serial.println(F(VERSION));

  // Tell host that we're ready, and wait for config messages and
  // acknowledgement
  host_ready = false;
  while (!host_ready) {
    Serial.println(F("Ready"));
    if (get_host_msg(incoming_msg)) {
      if (strstr_P(incoming_msg, ready_str)) {
        host_ready = true;
      }
      else if (strstr_P(incoming_msg, config_str)) {
        process_config_msg(incoming_msg);
      }
    }
  }
#ifdef DS18B20_SUPPORTED
  Serial.println(F("DS18B20 temperature sensor is SUPPORTED"));
#else
  Serial.println(F("DS18B20 temperature sensor is NOT supported"));
#endif
#ifdef ADS1115_PYRANOMETER_SUPPORTED
  Serial.println(F("ADS1115-based pyranometer is SUPPORTED"));
#else
  Serial.println(F("ADS1115-based pyranometer is NOT supported"));
#endif
#ifdef CAPTURE_UNFILTERED
```

```
  Serial.println(F("Debug capture of unfiltered IV points is SUPPORTED"));
#else
  Serial.println(F("Debug capture of unfiltered IV points is NOT
supported"));
#endif
  // Print value of MAX_IV_POINTS / max_iv_points
  Serial.print(F("MAX_IV_POINTS: "));
  Serial.print(MAX_IV_POINTS);
  Serial.print(F("   max_iv_points: "));
  Serial.println(max_iv_points);
#ifdef DS18B20_SUPPORTED
  // Print temp sensor info
  for (int ii = 0; ii < num_ds18b20s; ii++) {
    DeviceAddress rom_code;
    sensors.getAddress(rom_code, ii);
    Serial.print(F("ROM code of DS18B20 temp sensor #"));
    Serial.print(ii+1);
    Serial.print(F(" is 0x"));
    for (int jj = 7; jj >= 0; jj--) {
      if (rom_code[jj] < 16) Serial.print(F("0"));
      Serial.print(rom_code[jj], HEX);
    }
    Serial.println(F(""));
  }
#endif
}

void loop()
{
  // Arduino: ints are 16 bits
  bool go_msg_received;
  bool update_prev_i = false;
  bool poll_timeout = false;
  bool skip_isc_poll = false;
  bool count_updated = false;
  bool voc_adc_found = false;
  bool emr_isc_stable = false;
  bool ssr_isc_stable = false;
  char incoming_msg[MAX_MSG_LEN];
  int ii;
  int index = 0;
  int max_count = 0;
  int adc_v_delta, adc_i_delta, adc_i_prev_delta;
  int manhattan_distance, min_manhattan_distance;
  int pt_num = 1;   // counts points actually recorded
  int isc_poll_loops = 0;
  int num_discarded_pts = 0;
  int i_scale, v_scale;
  int adc_v_vals[MAX_IV_POINTS], adc_i_vals[MAX_IV_POINTS];
  int isc_adc, voc_adc;
  int adc_noise_floor, min_adc_noise_floor, max_adc_noise_floor;
  int done_i_adc;
  int adc_v_val_prev_prev, adc_v_val_prev, adc_v_val;
  int adc_i_val_prev_prev, adc_i_val_prev, adc_i_val;
  int isc_stable_adc_v_val = -1;
  int isc_stable_adc_v_val_prev = -1;
  int isc_stable_adc_v_val_prev_prev = -1;
  int isc_stable_adc_i_val = -1;
  int isc_stable_adc_i_val_prev = -1;
  int isc_stable_adc_i_val_prev_prev = -1;
  unsigned long num_meas = 1; // counts IV measurements taken
```

```
  long start_usecs, elapsed_usecs;
  float usecs_per_iv_pair;
#ifdef CAPTURE_UNFILTERED
  bool capture_unfiltered = false;
  int unfiltered_index = 0;
  int unfiltered_adc_v_vals[MAX_UNFILTERED_POINTS];
  int unfiltered_adc_i_vals[MAX_UNFILTERED_POINTS];
#endif

  // Wait for go (or config) message from host
  Serial.println(F("Waiting for go message or config message"));
  go_msg_received = false;
  while (!go_msg_received) {
    if (get_host_msg(incoming_msg)) {
      if (strstr_P(incoming_msg, go_str)) {
        go_msg_received = true;
      }
      else if (strstr_P(incoming_msg, config_str)) {
        process_config_msg(incoming_msg);
      }
    }
  }

  // Measure Vref (indirectly, by measuring bandgap)
  read_bandgap(GO_BDGP_READ_ITER);

  // Get Voc ADC value and current channel ADC noise floor
  voc_adc = 0;
  adc_noise_floor = ADC_MAX;
  min_adc_noise_floor = ADC_MAX;
  max_adc_noise_floor = 0;
  memset(adc_v_vals, 0, sizeof(adc_v_vals));
  memset(adc_i_vals, 0, sizeof(adc_i_vals));
  for (ii = 0; ii < VOC_POLLING_LOOPS; ii++) {
    adc_v_val = read_adc(VOLTAGE_CH);  // Read voltage channel
    adc_i_val = read_adc(CURRENT_CH);  // Read current channel
    // Update frequency count for this current channel value. We
    // temporarily use the adc_v_vals array for the values and the
    // adc_i_vals array for the counts
    for (index = 0, count_updated = false;
         (index < (int)sizeof(adc_v_vals)) && !count_updated;
         index++) {
      if (adc_i_vals[index] == 0) { // first empty slot
        adc_v_vals[index] = adc_v_val;
        adc_i_vals[index] = 1; // count
        count_updated = true;
      } else if (adc_v_vals[index] == adc_v_val) {
        adc_i_vals[index]++; // count
        count_updated = true;
      }
    }
    // The ADC noise floor is the value read from the ADC when it
    // "should" be zero.  At this point, we know that the actual current
    // is zero because the circuit is open, so whatever value is read on
    // the current channel is the noise floor value.
    if (adc_i_val < min_adc_noise_floor) {
      min_adc_noise_floor = adc_i_val;
    }
    if (adc_i_val > max_adc_noise_floor) {
      max_adc_noise_floor = adc_i_val;
    }
```

```
}

// The Voc ADC value is the most common value seen during polling
for (index = 0, voc_adc_found = false, max_count = 0;
     (index < (int)sizeof(adc_v_vals)) && !voc_adc_found;
     index++) {
  if (adc_i_vals[index] == 0) {
    // When we see a slot with a zero count, we're done
    voc_adc_found = true;
  } else if (adc_i_vals[index] > max_count) {
    voc_adc = adc_v_vals[index];
    max_count = adc_i_vals[index];
  }
}

adc_noise_floor = min_adc_noise_floor;
// Increase minimum Isc ADC value by noise floor
min_isc_adc += adc_noise_floor;

// Determine the current channel ADC value that indicates the curve
// has reached its tail.  This value is twice the noise floor value,
// or 20; whichever is greater.
done_i_adc = adc_noise_floor << 1;
if (done_i_adc < 20) {
  done_i_adc = 20;
}

// If Voc is valid, activate relay/SSRs)
//
if (voc_adc < MIN_VOC_ADC) {
  // If the Voc ADC value is lower than MIN_VOC_ADC we assume that it
  // is actually zero (not connected) and we force it to zero and skip
  // the relay activation and Isc polling
  skip_isc_poll = true;
  voc_adc = 0;
} else {
  skip_isc_poll = false;
  poll_timeout = true;

  // Turn on SSR3 (does nothing if this is not an SSR IVS2 or is a cell
  // version that has no SSR3)
  digitalWrite(SSR3_PIN, SSR3_ACTIVE);
  digitalWrite(FET3_PIN, FET3_ACTIVE);
  delay(20);  // Let it turn completely on before any current flows

  // Activate relay (or SSR1)
  digitalWrite(RELAY_PIN, relay_active);

  // Turn off SSR2 (does nothing if this is not an SSR IVS2 or is a cell
  // version that has no SSR2)
  digitalWrite(SSR2_PIN, SSR2_INACTIVE);
}

// Poll for stable Isc
//
adc_v_val_prev_prev = ADC_MAX;
adc_v_val_prev = ADC_MAX;
adc_i_val_prev_prev = 0;
adc_i_val_prev = 0;
for (ii = 0; (ii < max_isc_poll) && !skip_isc_poll; ii++) {
  adc_i_val = read_adc(CURRENT_CH);  // Read current channel
```

```
    adc_v_val = read_adc(VOLTAGE_CH);  // Read voltage channel
#ifdef CAPTURE_UNFILTERED_ISC_POLL
    if (((adc_i_val > min_isc_adc) || capture_unfiltered) &&
        (unfiltered_index < MAX_UNFILTERED_POINTS)) {
      unfiltered_adc_i_vals[unfiltered_index] = adc_i_val;
      unfiltered_adc_v_vals[unfiltered_index++] = adc_v_val;
      capture_unfiltered = true;
    }
#endif
    isc_poll_loops = ii + 1;
    if (adc_i_val > min_isc_adc) {
      // For the EMR version, Isc is considered stable when three
      // consecutive measurements:
      //   - have current greater than min_isc_adc
      //   - have increasing voltage
      //   - have decreasing or equal current
      //   - have a current difference less than or equal to isc_stable_adc
      //
      // For the SSR version, Isc is stable when both the voltage and
      // current have stopped changing, i.e. three of the same values
      // are seen in a row.
      //
      // Although we don't "know" whether the hardware is an EMR or SSR
      // version, it is very unlikely that the EMR conditions would
      // match on the SSR hardware or vice versa. But if they do, it
      // would most likely not matter.
      if (((adc_v_val > adc_v_val_prev) &&  // EMR conditions
           (adc_v_val_prev > adc_v_val_prev_prev) &&
           (adc_i_val <= adc_i_val_prev) &&
           (adc_i_val_prev <= adc_i_val_prev_prev) &&
           (abs(adc_i_val_prev - adc_i_val) <= isc_stable_adc) &&
           (abs(adc_i_val_prev_prev - adc_i_val_prev) <= isc_stable_adc)))
{
        emr_isc_stable = true;
      }
      if (((adc_v_val == adc_v_val_prev) &&  // SSR conditions
           (adc_v_val_prev == adc_v_val_prev_prev) &&
           (adc_i_val == adc_i_val_prev) &&
           (adc_i_val_prev == adc_i_val_prev_prev))) {
        ssr_isc_stable = true;
      }
      if (emr_isc_stable || ssr_isc_stable) {
        isc_stable_adc_v_val = adc_v_val;
        isc_stable_adc_v_val_prev = adc_v_val_prev;
        isc_stable_adc_v_val_prev_prev = adc_v_val_prev_prev;
        isc_stable_adc_i_val = adc_i_val;
        isc_stable_adc_i_val_prev = adc_i_val_prev;
        isc_stable_adc_i_val_prev_prev = adc_i_val_prev_prev;
        poll_timeout = false;
        break;
      }
      if (adc_v_val >= adc_v_val_prev) {
        // If voltage increases or is equal, shift previous to
        // previous-previous. But previous-previous keeps its value if
        // voltage decreases. This has the effect of discarding the
        // previous value, which handles the EMR "bounce" case.
        adc_v_val_prev_prev = adc_v_val_prev;
        adc_i_val_prev_prev = adc_i_val_prev;
      }
      // Shift current to previous
      adc_v_val_prev = adc_v_val;
```

```
    adc_i_val_prev = adc_i_val;
  }
}
if ((max_isc_poll < 0) && !skip_isc_poll) {
  // Special debug case (negative max_isc_poll). Just poll until a
  // non-zero current is found
  poll_timeout = true;
  for (ii = 0; ii < MAX_ISC_POLL; ii++) {
    adc_i_val = read_adc(CURRENT_CH);  // Read current channel
    if (adc_i_val) {
      poll_timeout = false;
      adc_v_val = read_adc(VOLTAGE_CH);  // Read voltage channel
      adc_i_val_prev_prev = adc_i_val;
      break;
    }
  }
}
// Turn off SSR3 (SSR4 in cell version) when polling is complete
digitalWrite(SSR3_PIN, SSR3_INACTIVE);
digitalWrite(FET3_PIN, FET3_INACTIVE);
digitalWrite(SSR4_PIN, SSR4_INACTIVE);

if (poll_timeout)
  Serial.println(F("Polling for stable Isc timed out"));

// Isc is approximately the value of the first of the three points
// at the end of Isc polling
isc_adc = adc_i_val_prev_prev;

// First IV pair (point number 0) is last point from polling
adc_v_vals[0] = adc_v_val;
adc_i_vals[0] = adc_i_val;

// Get v_scale and i_scale
compute_v_and_i_scale(isc_adc, voc_adc, &v_scale, &i_scale);

// Calculate the minimum scaled adc delta value. This is the Manhattan
// distance between the Isc point and the Voc point divided by the
// maximum number of points. This guarantees that we won't run out of
// memory before the complete curve is captured. However, it will
// usually result in a number of captured points that is a fair amount
// lower than max_iv_points. The max_iv_points value is how many
// points there -would- be if -all- points were the minimum distance
// apart, -and- the the actual distance between the ISC point and the
// VOC point were equal to the Manhattan distance. But some points
// will be farther apart than the minimum distance. One reason is
// simply because, unless max_iv_points is set to a very small number,
// there are portions of the curve where the limiting factor is the
// rate that the measurements can be taken; even without discarding
// measurements, the points are farther apart than the minimum. The
// other reason is that it is unlikely that a measurement comes at
// exactly the minimum distance from the previously recorded
// measurement, so the first one that does satisfy the requirement may
// have overshot the minimum by nearly a factor of 2:1 in the worst
// case. And, of course, the actual IV curve is always shorter than
// the Manhattan distance.
min_manhattan_distance = (unsigned int) ((isc_adc * i_scale) +
                         (voc_adc * v_scale)) / max_iv_points;

// Proceed to read remaining points on IV curve. Compensate for the
// fact that time passes between I and V measurements by using a
```

```
  // weighted average for I. Discard points that are not a minimum
  // "Manhattan distance" apart (scaled sum of V and I ADC values).
  adc_i_val_prev = adc_i_vals[0];
  start_usecs = micros();
  while (num_meas < MAX_IV_MEAS) {
    num_meas++;
    //-----------------------------------------------------
    // Read both channels back-to-back. The current channel is first
    // since it was first in the reads for point 0 above.
    adc_i_val = read_adc(CURRENT_CH);  // Read current channel
    adc_v_val = read_adc(VOLTAGE_CH);  // Read voltage channel
#ifdef CAPTURE_UNFILTERED_POST_ISC
    //------------------------- Unfiltered ----------------------
    if (unfiltered_index < MAX_UNFILTERED_POINTS) {
      unfiltered_adc_i_vals[unfiltered_index] = adc_i_val;
      unfiltered_adc_v_vals[unfiltered_index++] = adc_v_val;
    }
#endif
    //---------------------- Current channel -----------------
    if (update_prev_i) {
      // Adjust previous current value to weighted average with this
      // value.  16-bit integer math!! Max ADC value is 4095, so no
      // overflow as long as sum of I_CH_1ST_WEIGHT and I_CH_2ND_WEIGHT
      // is 16 or less.
      adc_i_vals[pt_num-1] = (adc_i_val_prev * I_CH_1ST_WEIGHT +
                              adc_i_val * I_CH_2ND_WEIGHT +
                              AVG_WEIGHT) / TOTAL_WEIGHT;
    }
    //---------------------- Voltage channel -----------------
    adc_v_vals[pt_num] = adc_v_val;
    //------------------------ Deltas  --------------------
    adc_v_delta = adc_v_val - adc_v_vals[pt_num-1];
    adc_v_val_prev = adc_v_val;
    adc_i_delta = adc_i_vals[pt_num-1] - adc_i_val;
    adc_i_prev_delta = adc_i_val_prev - adc_i_val;
    adc_i_val_prev = adc_i_val;
    //---------------------- Done check  -----------------
    // Check if we've reached the tail of the curve.
    if (adc_i_val < done_i_adc) {
      // Current is very close to zero so we're PROBABLY done
      if (adc_i_prev_delta < 3) {
        // But only if the current delta is very small
        break;
      }
    }
    // We're also done if Isc polling timed out
    if (poll_timeout) {
      break;
    }
    //--------------- Voltage decrease check -------------
    // At this point we know that all preceding points are in order of
    // increasing voltage.  However, it is possible that one or more of
    // them are erroneously high due to relay "bounce". This is detected
    // when the voltage of this point is lower than the voltage of the
    // previous point. If that is the case, we search backwards through
    // the previous points until we find one that has a lower voltage
    // and replace its successor with the current point and rewind the
    // pt_num counter. While it is probably not possible for the bounce
    // to span more than two or three points, this covers the general
    // case of it spanning N points (and starting at any point).
    if (adc_v_val < adc_v_vals[pt_num-1]) {
```

```
      while (pt_num > 1) {
        if (adc_v_val < adc_v_vals[pt_num-2]) {
          pt_num--;
        } else {
          break;
        }
      }
      adc_v_vals[pt_num-1] = adc_v_val;
      adc_i_vals[pt_num-1] = adc_i_val;
      update_prev_i = true; // Adjust this I value on next measurement
      continue;
    }
    //------------------- Discard decision ---------------
    // "Manhattan distance" is sum of scaled deltas
    manhattan_distance = (adc_v_delta * v_scale) + (adc_i_delta * i_scale);
    // Keep measurement if Manhattan distance is big enough; otherwise
    // discard.  However, if we've discarded max_discards consecutive
    // measurements, then keep it anyway.
    if ((manhattan_distance >= min_manhattan_distance) ||
        (num_discarded_pts >= max_discards)) {
      // Keep this one
      pt_num++;
      update_prev_i = true;  // Adjust this I value on next measurement
      num_discarded_pts = 0; // Reset discard counter
      if (pt_num >= max_iv_points) {
        // We're done
        break;
      }
    } else {
      // Don't record this one
      update_prev_i = false; // And don't adjust prev I val next time
      num_discarded_pts++;
    }
  }
  if (update_prev_i) {
    // Last one didn't get adjusted (or even saved), so save it now
    adc_i_vals[pt_num-1] = adc_i_val;
  }
  elapsed_usecs = micros() - start_usecs;

  // Turn off relay (or SSR1)
  digitalWrite(RELAY_PIN, relay_inactive);

  // Turn on SSR2 (does nothing if this is not a module version SSR
  // IVS2)
  digitalWrite(SSR2_PIN, SSR2_ACTIVE);
  // Turn on SSR4 (does nothing if this is not a cell version SSR IVS2)
  digitalWrite(SSR4_PIN, SSR4_ACTIVE);

  // Report results on serial port
  //
#ifdef ADS1115_PYRANOMETER_SUPPORTED
  int16_t ads1115_val, retries;
  long ads1115_val_sum, ads1115_val_avg, ppm_error_from_avg;
  bool ads1115_present, tmp36_present, found_stable_value;

  // Pyranometer temperature (TMP36)
  ads1115.setGain(GAIN_TWO);  // -2 V to 2 V
  ads1115_val_sum = 0;
  ads1115_val_avg = 0;
  ads1115_present = true;
```

```
tmp36_present = true;
found_stable_value = false;
retries = 0;
while (!found_stable_value && (retries < 20)) {
  for (int ii = 0; ii < ADS1115_TEMP_POLLING_LOOPS; ii++) {
    ads1115_val = ads1115.readADC_SingleEnded(2);
    if (ads1115_val == -1) {
      // Value of -1 indicates no ADS1115 is present
      ads1115_present = false;
      found_stable_value = true;
      break;
    }
    if (ads1115_val < 4000) {
      // Values less than 250mV (-25 deg C) are assumed to be noise,
      // meaning there is no TMP36 connected to A2
      tmp36_present = false;
      found_stable_value = true;
      break;
    }
    ads1115_val_sum += ads1115_val;
  }
  if (ads1115_present && tmp36_present) {
    ads1115_val_avg = ads1115_val_sum / ADS1115_TEMP_POLLING_LOOPS;
    found_stable_value = true;
    ads1115_val_sum = 0;
    for (int ii = 0; ii < ADS1115_TEMP_POLLING_LOOPS; ii++) {
      ads1115_val = ads1115.readADC_SingleEnded(2);
      ppm_error_from_avg =
        (1000000 * abs(ads1115_val - ads1115_val_avg)) /
        abs(ads1115_val_avg);
      if (ppm_error_from_avg > MAX_STABLE_TEMP_ERR_PPM) {
        // If any value is more than MAX_STABLE_TEMP_ERR_PPM from the
        // average, we don't have a stable value
        found_stable_value = false;
        retries++;
        break;
      }
    }
  }
}
if (ads1115_present && tmp36_present && found_stable_value) {
  Serial.print(F("ADS1115 (pyranometer temp sensor) raw value: "));
  Serial.println(ads1115_val_avg);
} else if (ads1115_present && tmp36_present) {
  Serial.print(F("WARNING: TMP36 pyranometer temp sensor not stable"));
}
// Irradiance (PDB-C139)
if (ads1115_present) {
  ads1115.setGain(GAIN_EIGHT); // -512 mV to 512 mV
  ads1115_val_sum = 0;
  ads1115_val_avg = 0;
  found_stable_value = false;
  retries = 0;
  while (!found_stable_value && (retries < 20)) {
    for (int ii = 0; ii < ADS1115_IRRADIANCE_POLLING_LOOPS; ii++) {
      ads1115_val = ads1115.readADC_Differential_0_1();
      ads1115_val_sum += ads1115_val;
    }
    ads1115_val_avg = ads1115_val_sum / ADS1115_IRRADIANCE_POLLING_LOOPS;
    found_stable_value = true;
    ads1115_val_sum = 0;
```

```
      for (int ii = 0; ii < ADS1115_IRRADIANCE_POLLING_LOOPS; ii++) {
        ads1115_val = ads1115.readADC_Differential_0_1();
        ppm_error_from_avg =
          (1000000 * abs(ads1115_val - ads1115_val_avg)) /
          abs(ads1115_val_avg);
        if (ppm_error_from_avg > MAX_STABLE_IRRAD_ERR_PPM) {
          // If any value is more than MAX_STABLE_IRRAD_ERR_PPM from the
          // average, we don't have a stable value
          found_stable_value = false;
          retries++;
          break;
        }
      }
    }
  }
  if (ads1115_present && found_stable_value) {
    Serial.print(F("ADS1115 (pyranometer photodiode) raw value: "));
    Serial.println(ads1115_val_avg);
  } else if (ads1115_present) {
    Serial.println(F("WARNING: pyranometer photodiode not stable"));
  }
#endif
#ifdef DS18B20_SUPPORTED
  // Temperature
  if (num_ds18b20s) {
    sensors.requestTemperatures();
    for (int ii = 0; ii < num_ds18b20s; ii++) {
      Serial.print(F("Temperature at sensor #"));
      Serial.print(ii+1);
      Serial.print(F(" is "));
      Serial.print(sensors.getTempCByIndex(ii));
      Serial.println(F(" degrees Celsius"));
    }
  }
#endif
  // CH1 (current channel) ADC noise floor
  Serial.print(F("CH1 ADC noise floor (min):"));
  Serial.println(min_adc_noise_floor);
  Serial.print(F("CH1 ADC noise floor (max):"));
  Serial.println(max_adc_noise_floor);
  // Isc stable polling
  Serial.print(F("EMR Isc stable: "));
  Serial.print(emr_isc_stable);
  Serial.print(F("  SSR Isc stable: "));
  Serial.println(ssr_isc_stable);
  if (emr_isc_stable || ssr_isc_stable) {
    Serial.print(F("Isc stable point 1: "));
    Serial.print(isc_stable_adc_v_val_prev_prev);
    Serial.print(F(","));
    Serial.println(isc_stable_adc_i_val_prev_prev);
    Serial.print(F("Isc stable point 2: "));
    Serial.print(isc_stable_adc_v_val_prev);
    Serial.print(F(","));
    Serial.println(isc_stable_adc_i_val_prev);
    Serial.print(F("Isc stable point 3: "));
    Serial.print(isc_stable_adc_v_val);
    Serial.print(F(","));
    Serial.println(isc_stable_adc_i_val);
  }
  // Isc point
  Serial.print(F("Isc CH0:0"));
```

```
  Serial.print(F(" CH1:"));
  Serial.println(isc_adc);
  // Middle points
  for (ii = 0; ii < pt_num; ii++) {
    Serial.print(ii);
    Serial.print(F(" CH0:"));
    Serial.print(adc_v_vals[ii]);
    Serial.print(F(" CH1:"));
    Serial.println(adc_i_vals[ii]);
  }
  // Voc point
  Serial.print(F("Voc CH0:"));
  Serial.print(voc_adc);
  Serial.print(F(" CH1:"));
  Serial.println(adc_noise_floor);
#ifdef CAPTURE_UNFILTERED
  for (ii = 0; ii < unfiltered_index; ii++) {
    Serial.print(ii);
    Serial.print(F(" Unfiltered CH0:"));
    Serial.print(unfiltered_adc_v_vals[ii]);
    Serial.print(F(" Unfiltered CH1:"));
    Serial.println(unfiltered_adc_i_vals[ii]);
  }
#endif
  Serial.print(F("Isc poll loops: "));
  Serial.println(isc_poll_loops);
  Serial.print(F("Number of measurements: "));
  Serial.println(num_meas);
  Serial.print(F("Number of recorded points: "));
  Serial.println(pt_num);
  Serial.print(F("i_scale: "));
  Serial.println(i_scale);
  Serial.print(F("v_scale: "));
  Serial.println(v_scale);
  Serial.print(F("min_manhattan_distance: "));
  Serial.println(min_manhattan_distance);
  Serial.print(F("Elapsed usecs: "));
  Serial.println(elapsed_usecs);
  usecs_per_iv_pair = (float) elapsed_usecs / num_meas;
  Serial.print(F("Time (usecs) per i/v reading: "));
  Serial.println(usecs_per_iv_pair);
  Serial.println(F("Output complete"));

}

bool get_host_msg(char * msg) {
  bool msg_received = false;
  char c;
  int char_num = 0;
  int msg_timer;
  msg_timer = MSG_TIMER_TIMEOUT;
  while (msg_timer && !msg_received) {
    if (Serial.available()) {
      c = Serial.read();
      if (c == '\n') {
        // Substitute NULL for newline
        msg[char_num++] = '\0';
        msg_received = true;
        Serial.print(F("Received host message: "));
        Serial.println(msg);
        break;
```

```
      } else {
        if (char_num == (MAX_MSG_LEN - 1)) {
          msg[char_num] = '\0';
          Serial.print(F("ERROR: Host message too long: "));
          Serial.print(msg);
          Serial.println(F("...."));
          break;
        } else {
          msg[char_num++] = c;
        }
      }
      msg_timer = MSG_TIMER_TIMEOUT;
    } else {
      msg_timer--;
    }
    delay(1);
  }
  return (msg_received);
}

void process_config_msg(char * msg) {
  char *substr = NULL;
  char *config_type = NULL;
  char *config_val = NULL;
  char *config_val2 = NULL;
  int ii = 0;
  int num_args = 0;
  int exp_args;
  int eeprom_addr;
  int count, adc_v_val, adc_i_val;
  float eeprom_value;
  bool wrong_arg_cnt = false;
  const char CARRIAGE_RETURN = 0xd;
  substr = strtok(msg, " ");  // "Config:"
  while (substr != NULL) {
    substr = strtok(NULL, " ");
      if (substr != NULL &&
          *substr != CARRIAGE_RETURN) { // Windows phenomenon
      if (ii == 0) {
        config_type = substr;
      } else if (ii == 1) {
        config_val = substr;
        num_args = 1;
      } else if (ii == 2) {
        config_val2 = substr;
        num_args = 2;
      } else if (substr != NULL) {
        Serial.println(F("ERROR: Too many fields in config message"));
        Serial.println(F("Config not processed"));
        return;
      }
    }
    ii++;
  }
  if (strcmp_P(config_type, clk_div_str) == 0) {
    exp_args = 1;
    if (num_args == exp_args) {
      clk_div = atoi(config_val);
      SPI.setClockDivider(clk_div);
    } else {
      wrong_arg_cnt = true;
```

```
  }
} else if (strcmp_P(config_type, max_iv_points_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    max_iv_points = atoi(config_val);
    if (max_iv_points > MAX_IV_POINTS) {
      max_iv_points = MAX_IV_POINTS;
    }
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, min_isc_adc_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    min_isc_adc = atoi(config_val);
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, max_isc_poll_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    max_isc_poll = atoi(config_val);
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, isc_stable_adc_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    isc_stable_adc = atoi(config_val);
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, max_discards_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    max_discards = atoi(config_val);
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, aspect_height_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    aspect_height = atoi(config_val);
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, aspect_width_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    aspect_width = atoi(config_val);
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, write_eeprom_str) == 0) {
  exp_args = 2;
  if (num_args == exp_args) {
    eeprom_addr = atoi(config_val);
    eeprom_value = atof(config_val2);
    EEPROM.put(eeprom_addr, eeprom_value);
    if (eeprom_addr == EEPROM_RELAY_ACTIVE_HIGH_ADDR) {
      relay_active = (eeprom_value == LOW) ? LOW : HIGH;
```

```
      relay_inactive = (relay_active == LOW) ? HIGH : LOW;
      digitalWrite(RELAY_PIN, relay_inactive);
      digitalWrite(SECOND_RELAY_PIN, relay_inactive);
    }
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, dump_eeprom_str) == 0) {
  exp_args = 0;
  if (num_args == exp_args) {
    dump_eeprom();
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, relay_state_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    set_relay_state((bool)atoi(config_val));
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, second_relay_state_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    set_second_relay_state((bool)atoi(config_val));
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, do_ssr_curr_cal_str) == 0) {
  exp_args = 0;
  if (num_args == exp_args) {
    do_ssr_curr_cal();
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, read_bandgap_str) == 0) {
  exp_args = 0;
  if (num_args == exp_args) {
    read_bandgap(CMD_BDGP_READ_ITER);
  } else {
    wrong_arg_cnt = true;
  }
} else if (strcmp_P(config_type, read_adc_str) == 0) {
  exp_args = 1;
  if (num_args == exp_args) {
    count = atoi(config_val);
    for (ii = 0; ii < count; ii++) {
      adc_v_val = read_adc(VOLTAGE_CH);  // Read voltage channel
      adc_i_val = read_adc(CURRENT_CH);  // Read current channel
      Serial.print(F("ADC CH0 (voltage): "));
      Serial.print(adc_v_val);
      Serial.print(F(" CH1 (current): "));
      Serial.println(adc_i_val);
      delay(500);  // 0.5 seconds between reads
    }
  } else {
    wrong_arg_cnt = true;
  }
} else {
  Serial.print(F("ERROR: Unknown config type: "));
  Serial.println(config_type);
```

```
    Serial.println(F("Config not processed"));
    return;
  }
  if (wrong_arg_cnt) {
    Serial.print(F("ERROR: Expected "));
    Serial.print(exp_args);
    Serial.print(F(" args for config type "));
    Serial.print(config_type);
    Serial.print(F(", got  "));
    Serial.println(num_args);
    Serial.println(F("Config not processed"));
    return;
  }
  Serial.println(F("Config processed"));
  return;
}

void dump_eeprom() {
  int eeprom_addr, eeprom_valid_count;
  float eeprom_value;

  // Dump valid EEPROM entries
  EEPROM.get(0, eeprom_value);
  // Only dump if address 0 has "magic" value
  if (eeprom_value == EEPROM_VALID_VALUE) {
    // Second location has count of valid entries
    EEPROM.get(sizeof(float), eeprom_value);
    eeprom_valid_count = (int)eeprom_value;
    for (eeprom_addr = 0;
         eeprom_addr < ((eeprom_valid_count + 2) * (int)sizeof(float));
         eeprom_addr += sizeof(float)) {
      EEPROM.get(eeprom_addr, eeprom_value);
      Serial.print(F("EEPROM addr: "));
      Serial.print(eeprom_addr, DEC);
      Serial.print(F("  value: "));
      Serial.println(eeprom_value, 4);
    }
  }
}

char get_relay_active_val() {
  // The IV Swinger 2 hardware design calls for an active-low triggered
  // relay module.  Support has been added now for the alternate use of
  // an active-high triggered relay module. The host software writes
  // EEPROM address 44 with either the value 0 or 1 indicating
  // active-low or active-high repectively. At the beginning of setup()
  // this function is called to determine which type the relay is.  It
  // is possible that EEPROM has not been written yet, or that it was
  // written by an older version of the host software and does not have
  // a valid value at address 44.  In either of these cases, the default
  // value of LOW is returned.
  int eeprom_valid_count;
  float eeprom_value;

  // Check that address 0 has "magic" value
  EEPROM.get(0, eeprom_value);
  if (eeprom_value == EEPROM_VALID_VALUE) {
    // Second location has count of valid entries
    EEPROM.get(sizeof(float), eeprom_value);
    eeprom_valid_count = (int)eeprom_value;
    // Check that EEPROM contains an entry for the relay active value
```

```
    if ((eeprom_valid_count + 1) * sizeof(float) >=
        EEPROM_RELAY_ACTIVE_HIGH_ADDR) {
      EEPROM.get(EEPROM_RELAY_ACTIVE_HIGH_ADDR, eeprom_value);
      if (eeprom_value == 0) {
        return (LOW);
      } else {
        return (HIGH);
      }
    } else {
      // If EEPROM is not programmed with the relay active value, we
      // have to assume it is active-low
      return (LOW);
    }
  } else {
    // If EEPROM is not programmed (at all), we have to assume the relay
    // is active-low
    return (LOW);
  }
}

void set_relay_state(bool active) {
  if (active) {
    digitalWrite(RELAY_PIN, relay_active);
  } else {
    digitalWrite(RELAY_PIN, relay_inactive);
  }
}

void set_second_relay_state(bool active) {
  if (active) {
    digitalWrite(SSR6_PIN, SSR6_INACTIVE);
    digitalWrite(SECOND_RELAY_PIN, relay_active);
  } else {
    digitalWrite(SECOND_RELAY_PIN, relay_inactive);
    digitalWrite(SSR6_PIN, SSR6_ACTIVE);
  }
}

void do_ssr_curr_cal() {
  bool result_valid = true;
  int adc_i_val;
  bool keep_going;
  long adc_i_val_sum, adc_i_val_p1_avg, adc_i_val_avg_cnt;
  float adc_i_val_p2_avg;
  long start_usecs, elapsed_usecs;

  // Measure Vref (indirectly, by measuring bandgap)
  read_bandgap(CMD_BDGP_READ_ITER);

  // Activate SSR3/4
  digitalWrite(SSR3_PIN, SSR3_ACTIVE);  // module version
  digitalWrite(FET3_PIN, FET3_ACTIVE);  // module version
  digitalWrite(SSR4_PIN, SSR4_ACTIVE);  // cell version
  // Deactivate SSR2
  digitalWrite(SSR2_PIN, SSR2_INACTIVE);  // module version
  // Activate SSR1
  digitalWrite(RELAY_PIN, relay_active);

  // Loop for SSR_CAL_USECS microseconds
  //
  // This period is long enough for a human to read the measured value
```

```
  // on the DMM display.
  //
  // At the end of the loop, there are two periods of SSR_CAL_RD_USECS
  // in which the ADC current channel is read in a loop and the average
  // ADC is calculated. If the difference between the Pass 1 average and
  // Pass 2 average is more than 1%, the measurement is considered
  // "unstable".
  //
  start_usecs = micros();
  elapsed_usecs = 0;

  // Pre-Pass 1
  //
  // Just spin waiting for the elapsed time to reach 2 x
  // SSR_CAL_RD_USECS from the end (i.e. the beginning of Pass 1)
  //
  while (elapsed_usecs < (SSR_CAL_USECS - 2 * SSR_CAL_RD_USECS)) {
    elapsed_usecs = micros() - start_usecs;
  }
  //
  // Pass 1
  //
  // Loop until SSR_CAL_RD_USECS from the end. Accumulate sum of ADC
  // values and number of reads (for average ADC calculation). Bail out
  // if ADC saturated is seen.
  //
  // Pass 2
  //
  // Loop the rest of the way doing the same.
  //
  for (int pass = 1; pass <= 2; pass++) {
    adc_i_val_sum = 0;
    adc_i_val_avg_cnt = 0;
    keep_going = true;
    while (keep_going) {
      adc_i_val = read_adc(CURRENT_CH);  // Read current channel
      adc_i_val_sum += adc_i_val;
      adc_i_val_avg_cnt++;
      if (adc_i_val > (ADC_SAT - 10))
        result_valid = false;
      elapsed_usecs = micros() - start_usecs;
      if (pass == 1) {
        keep_going = ((elapsed_usecs < (SSR_CAL_USECS - SSR_CAL_RD_USECS))
&&
                      result_valid);
      } else {
        keep_going = ((elapsed_usecs < SSR_CAL_USECS) && result_valid);
      }
    }
    if (pass == 1) {
      // Compute the Pass 1 average
      adc_i_val_p1_avg = adc_i_val_avg_cnt ?
        adc_i_val_sum / adc_i_val_avg_cnt : 0;
    }
  }
  //
  // If the result is valid so far (ADC not saturated), compute the Pass
  // 2 average and check that it is within 1% of the Pass 1 average.
  // Print message with stability determination and both averages. Note
  // that Pass 1 average is integer (long) and Pass 2 average is
  // floating point at this point.
```

```
  //
  if (result_valid) {
    // Compute the Pass 2 average
    adc_i_val_p2_avg = adc_i_val_avg_cnt ?
      float(adc_i_val_sum) / float(adc_i_val_avg_cnt) : 0.0;
    if (abs(adc_i_val_p2_avg - adc_i_val_p1_avg)/adc_i_val_p2_avg > 0.01) {
      Serial.print(F("SSR current calibration ADC not stable. Pass 1: "));
      result_valid = false;
    } else {
      Serial.print(F("SSR current calibration ADC stable. Pass 1: "));
    }
    Serial.print(adc_i_val_p1_avg);
    Serial.print(F("  Pass 2: "));
    Serial.println(adc_i_val_p2_avg);
  } else {
    Serial.println(F("SSR current calibration: ADC saturated"));
  }

  // Deactivate SSR1
  digitalWrite(RELAY_PIN, relay_inactive);
  // Activate SSR2
  digitalWrite(SSR2_PIN, SSR2_ACTIVE);
  // Deactivate SSR3/4
  digitalWrite(SSR3_PIN, SSR3_INACTIVE);
  digitalWrite(FET3_PIN, FET3_INACTIVE);
  digitalWrite(SSR4_PIN, SSR4_INACTIVE);

  // If the result is valid, print result (average ADC value)
  if (result_valid) {
    Serial.print(F("SSR current calibration ADC value: "));
    // Round Pass 2 average to nearest integer
    Serial.println(int(adc_i_val_p2_avg + 0.5));
  }
}

void set_up_bandgap() {
  analogReference(DEFAULT);
  // Set the reference to Vcc and the measurement to the internal 1.1V
bandgap
  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(2); // Wait for Vref to settle
}

void read_bandgap(int iterations) {
  long result = 0;
  read_internal_adc();
  for (long ii = 0; ii < iterations; ii++) {
    result += (long)read_internal_adc();
  }
  Serial.print(F("Bandgap total ADC: "));
  Serial.print(result);
  Serial.print(F(" iterations: "));
  Serial.println(iterations);
}

int read_internal_adc() {
  // Read the Arduino internal ADC
  ADCSRA |= _BV(ADSC); // Start conversion
  while (bit_is_set(ADCSRA, ADSC)); // measuring
  return ADC; // ADCH, ADCL
}
```

```
int read_adc(int ch) {
  // This code assumes MCP3202.  MCP3302 would be slightly different.
  int ms_byte, ls_byte, cmd_bytes;
  cmd_bytes = (ch == 0) ?
    B10100000 :                          // SGL/~DIFF=1, CH=0, MSBF=1
    B11100000;                           // SGL/~DIFF=1, CH=1, MSBF=1
  digitalWrite(ADC_CS_PIN, CS_ACTIVE);   // Assert active-low chip select
  SPI.transfer (B00000001);              // START=1
  ms_byte = SPI.transfer(cmd_bytes);     // Send command, get result
  ms_byte &= B00001111;                  // Bits 11:8 (mask others)
  ls_byte = SPI.transfer(0x00);          // Bits 7:0
  digitalWrite(ADC_CS_PIN, CS_INACTIVE); // Deassert active-low chip select
  return ((ms_byte << 8) | ls_byte);     // {ms_byte, lsb}
}

void compute_v_and_i_scale(int isc_adc, int voc_adc,
                           int * v_scale, int * i_scale) {

  // Find integer scaling values for V and I, with the sum of the values
  // being 16 or less.  These values are used for calculating the
  // "Manhattan distance" between points when making the discard
  // decision. The idea is that the criterion for the minimum distance
  // between points on a horizontal part of the curve should be equal to
  // the criterion for the minimum distance between points on a vertical
  // part of the curve. The distance is literally the spacing on the
  // graph. The distance between points on a diagonal part of the curve
  // is overcounted somewhat, but that results in slightly closer points
  // near the knee(s) of the curve, and that is good. The two factors
  // that determine the distance are:
  //
  //       - The maximum ADC value of the axis
  //       - The aspect ratio of the graph
  //
  // The maximum value on the X-axis (voltage) is the Voc ADC value.
  // The maximum value on the Y-axis (current) is the Isc ADC value.
  // Since the graphs are rendered in a rectangular aspect ratio, the
  // scale of the axes differs. The initial scaling values could be:
  //
  //     initial_v_scale = aspect_width / voc_adc;
  //     initial_i_scale = aspect_height / isc_adc;
  //
  // That would require large values for aspect_width and aspect_height
  // to use integer math. Instead, proportional (but much larger) values
  // can be computed with:
  //
  //     initial_v_scale = aspect_width * isc_adc;
  //     initial_i_scale = aspect_height * voc_adc;
  //
  // An algorithm is then performed to reduce the values proportionally
  // such that the sum of the values is 16 or less.
  //
  // This function is only run once, but speed is important, so 16-bit
  // integer math is used exclusively (no floats or longs).
  //

  bool i_scale_gt_v_scale;
  int initial_v_scale, initial_i_scale;
  int lg, sm;
  int round_up_mask = 0;
  int lg_scale, sm_scale;
```

```
  char bit_num, shift_amt = 0;
  initial_v_scale = aspect_width * isc_adc;
  initial_i_scale = aspect_height * voc_adc;
  i_scale_gt_v_scale = initial_i_scale > initial_v_scale;
  lg = i_scale_gt_v_scale ? initial_i_scale : initial_v_scale;
  sm = i_scale_gt_v_scale ? initial_v_scale : initial_i_scale;

  // Find leftmost bit that is set in the larger initial value. The
  // right shift amount is three less than this bit number (to result in
  // a 4-bit value, i.e. 15 or less). Also look at the highest bit that
  // will be shifted off, to see if we should round up by adding one to
  // the resulting shifted amount.  If we get all the way down to bit 4
  // and it isn't set, the initial values will be used as-is.
  for (bit_num = 15; bit_num >= 4; bit_num--) {
    if (lg & (1 << bit_num)) {
      shift_amt = bit_num - 3;
      round_up_mask = (1 << (bit_num - 4));
      break;
    }
  }
  // Shift, and increment shifted amount if rounding up is needed
  lg_scale = (lg & round_up_mask) ? (lg >> shift_amt) + 1 : (lg >>
shift_amt);
  sm_scale = (sm & round_up_mask) ? (sm >> shift_amt) + 1 : (sm >>
shift_amt);
  // If the sum of these values is greater than 16, divide them both by
  // two (no rounding up here)
  if (lg_scale + sm_scale > 16) {
    lg_scale >>= 1;
    sm_scale >>= 1;
  }
  // Make sure sm_scale is at least 1 (necessary?)
  if (sm_scale == 0) {
    sm_scale = 1;
    if (lg_scale == 16)
      lg_scale = 15;
  }
  // Return values at pointer locations
  *v_scale = i_scale_gt_v_scale ? sm_scale : lg_scale;
  *i_scale = i_scale_gt_v_scale ? lg_scale : sm_scale;
}
```

Works Cited

Bojek, P. (n.d.). *Solar PV*. Retrieved from IEA: https://www.iea.org/energy-system/renewables/solar-pv

Peplow, M. (2022). Solar Panels Face Recycling Challenge. *ACS Central Science*.

Han, H. (2018). Degradation analysis of crystalline silicon photovoltaic modules exposed over 30 years in hot-humid climate in China. *Solar Energy*.

Khalifa, S. (2021). A Circularity Assessment for Silicon Solar Panels Based on Dynamic Material Flow Analysis. *IEEE*.

What is Voltage Follower? (n.d.). Retrieved from Learning about Electronics: http://www.learningaboutelectronics.com/Articles/Voltage-follower

Satterlee, C. (n.d.). *IV Swinger 2 - PCB (PV Module, SSR)*. Retrieved from Instructables: https://www.instructables.com/IV-Swinger-2-PCB-PV-Module-SSR/

(n.d.). *Photovoltaic (PV) module performance testing and energy rating - Part 1: Irradiance and temperature performance measurements and power rating.* NEN-EN-IEC 61853-1. Retrieved from IEC.

Meteorological data portal. (n.d.). Retrieved from TU Delft: https://www.tudelft.nl/en/ewi/over-de-faculteit/afdelingen/electrical-sustainable-energy/photovoltaic-materials-and-devices/dutch-pv-portal/meteorological-data