

SauleBoard ☀️ Charge controller with MPPT and Bluetooth

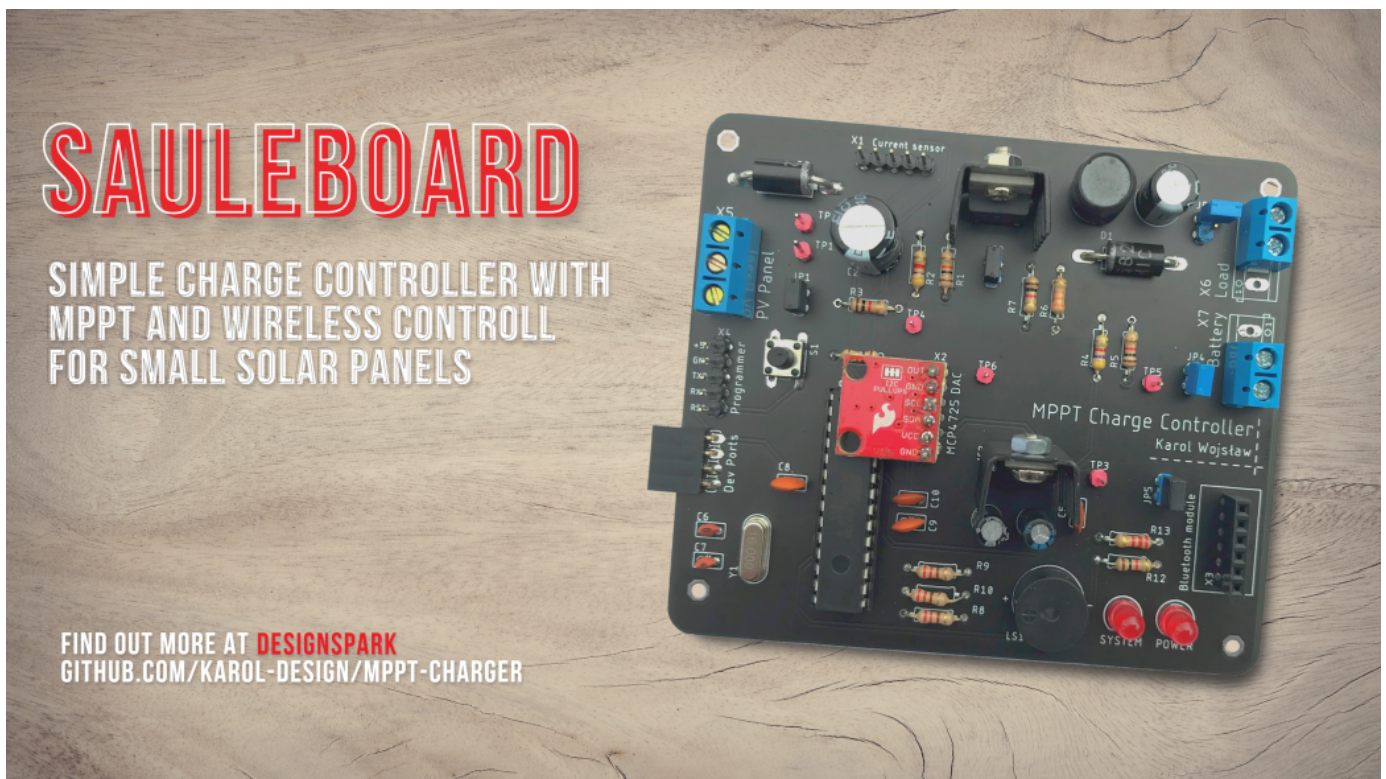


karol_poznan(/designspark/user/karol_poznan) 23 Jun 2021

0 ★

0

0 👍






Photovoltaics may be the future of energy generation though it requires additional electronics to work well. SauleBoard is an open-source project of a simple charge controller with an MPPT algorithm and wireless control for small solar panels.

Parts list

Qty	Product	Part number
-----	---------	-------------



Qty	Product	Part number	
1 x	LM2576T-ADJG Step-Down Switching Regulator	463-050 (/designspark/purchase-product/463-050?cm_mmc=en-ds_-web_-ds%3Alearn%3Astudent-articles%3Atest-54_bp_-463-050)	
1 x	Microchip ATMEGA328P-PU, 8bit AVR Microcontroller	131-0276 (/designspark/purchase-product/131-0276?cm_mmc=en-ds_-web_-ds%3Alearn%3Astudent-articles%3Atest-54_bp_-131-0276)	

See all

Each year renewable energy sources, with photovoltaic panels at the forefront, increase their contribution to the overall electricity production. The view of the solar installation on the roof of our neighbours does not surprise anyone anymore, but the way such a system works is not so widely known. The Solar charger project described below is, firstly, a technical challenge for a student like myself, but secondly, it's an attempt to understand the principles of operation of the technology that is shaping the future of energy generation. How to charge a battery from the solar panel? Why MPPT is important especially for bigger installations? And what actually is the MPPT algorithm? ...

A little bit of background

In September last year, I've started a bachelor degree in electronics at the University of Manchester. I'm from Poland, so I was excited about moving out to the UK and studying the subject, which I have always been fascinated with. Unfortunately quickly it turned out that there was no chance for any classes at the campus, so I was forced to move back to Poland and start studying remotely. I decided, however, that regardless of the situation, I will try to make the most of the theoretical content provided by the lecturers and complement this with my own practical projects. And that's what I've done.

Project definition – my approach to handle technical challenges

After the first semester and an introductory course on semiconductors, I've decided to try my hand out at solar energy and build my own basic system comprising of everything I need to generate, convert, store and use the energy provided by the Sun. As a project general requirement, I've also decided that I will try to keep it well organised and follow the simple process that I have developed based on the Agile / Lean management approach (figure 1).



Figure 1 – Hardware development process

Apart from following a predetermined plan I've also decided to use a version control system to back up all files regularly and keep track of any changes to the design. For this, I used GitHub with their desktop app. You can find the full repository and design files here.

I have started with general system requirements and pre-design research. After digging a bit in the online library resources I found a couple of great resources:

- "Photovoltaic Power System: Modeling, Design and Control" by Weidong Xiao (University of Sydney) [this was my main source of information about Photovoltaic technology]
- "Solar MPPT Battery Charger for the Rural Electrification System" by Microchip (36 p. application note) [I have used it a bit as an example of how such a system can be implemented]
- Other data sheets, application notes and online examples of similar projects (e.g. **Arduino MPPT Solar Charge Controller** (<https://www.instructables.com/ARDUINO-SOLAR-CHARGE-CONTROLLER-Version-30/>))

At this point, I have also specified general system requirements such as:

- The system will comprise of 10 W monocrystalline solar panel and 7 Ah (12 V) rechargeable sealed lead-acid battery (mainly for proof-of-concept purposes);
- The device should be able to act as a step-down converter with adjustable output voltage in the range of 13.0 – 15.0 V;
- There should be an option for sending reports on the state of the device wirelessly over Bluetooth;
- The charger needs to be able to track maximum power point (MPPT algorithm);

While I tried to make an iterative approach to designing and prototyping, I tried at the beginning to set the requirements in such a fashion that I won't have to change them during the project and I managed to do that.

...

The project of the charger can be divided into two main parts: hardware and software (or rather firmware). The hardware on the other hand comprises two sections: Switching step-down regulator and MCU & Bluetooth section.

It quickly turned out that if I want to charge the battery, especially from the Solar Panel the core of my hardware will be an adjustable buck converter. I have used such off-the-shelf modules in the past, however, I have never tried to build one myself. I started with the aim of building a converter from scratch, based on MOSFETs and fundamental electronic components but after few attempts, I discovered that having only very basic circuit analysis experience it's close to impossible for me to make it working and moreover efficient. Hence, I decided to use an IC manufactured exactly for that purpose.

I choose LM2576-ADJ with an output voltage of 1.23 to 37 V ($\pm 4\%$) and a guaranteed 3.0 A output current working at the frequency of 52 kHz. It requires a couple of additional components such as electrolytic capacitors, resistors, Schottky diode and an inductor however the datasheet guide the user quite well throughout the component selection process.

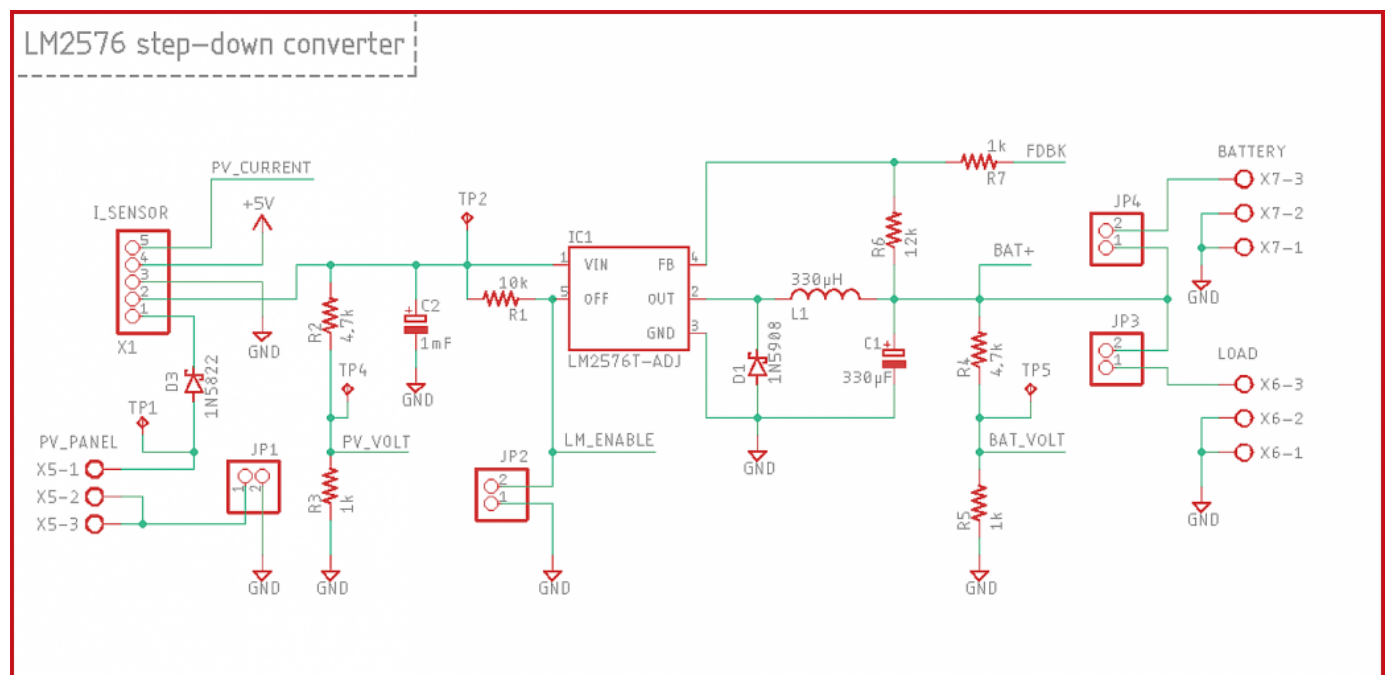


Figure 2 - LM2576 step-down schematics

Apart from the LM2576 chip and supporting components I've added also a reverse current protection diode to prevent current flowing back from the battery to the solar panel, e.g. during the night. There are also two voltage dividers for measurements, a current sensor header along with a couple of test points and jumpers (may also be used for connecting external switches).

For the buck converter to be really digitally adjustable I created a feedback loop that is connected to MCU (Microcontroller unit), such that by changing FDBK voltage output voltage can be controlled.

Microcontroller unit and Bluetooth module

While the charger is mainly the buck converter it has to be controlled by some external subsystem. In the case of this project, I decided to use the Atmega328P 8-bit AVR microcontroller, as I had experience with Arduino boards, which are in a lot of cases based on AVR MCUs. It has internal ADC (used for voltage measurements) as well as hardware and software UART serial ports (for debugging and Bluetooth communication).

One of the ideas of the device was to include in it some sort of wireless communication so that you can check the state of the charger without connecting a cable to the

For Digital-analog-converter I have used an external MCP4725 DAC board with an I2C interface. I tried to create an internal onboard DAC using a PWM signal from the Atmega microcontroller alongside with operational amplifier and RC filter however its performance was not satisfactory enough to control the feedback loop of the buck converter.

PCB design and soldering

After about two months of prototyping (Design --> Build --> Test loop iterations) and finding the most suitable components, modules and interconnections between all of them, it came time for finalising design files. I've made some last changes to the schematic, ask a couple of other hobbyist/engineers for any last pieces of advice online, and start creating a layout for the Printed Circuit Board (PCB).

As one of the last things, I added also a gold pin header connected to two unused I/O pins of Atmega328 along with +5 V & GND connections. These are marked as Dev Ports as they are intended to be used for future development purposes or external devices (like ESP32 with Wi-Fi capabilities) that may be connected to the board.

After making sure that all systems and hardware requirements should be met by the board and running the final ERC in schematic and DRC in board file I exported Gerber files for manufacturing. I ordered 5 test boards from JLC PCB, as their service was quite cheap and a good fit for the first batch of test boards. Below you can find some photos of the boards, soldering components in and the first try of connecting the charger to the power supply.

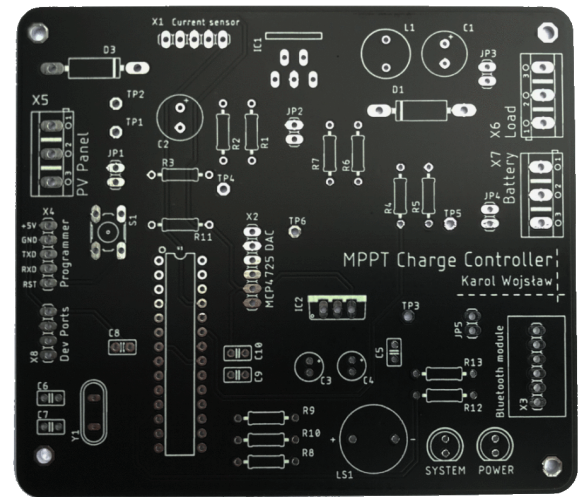
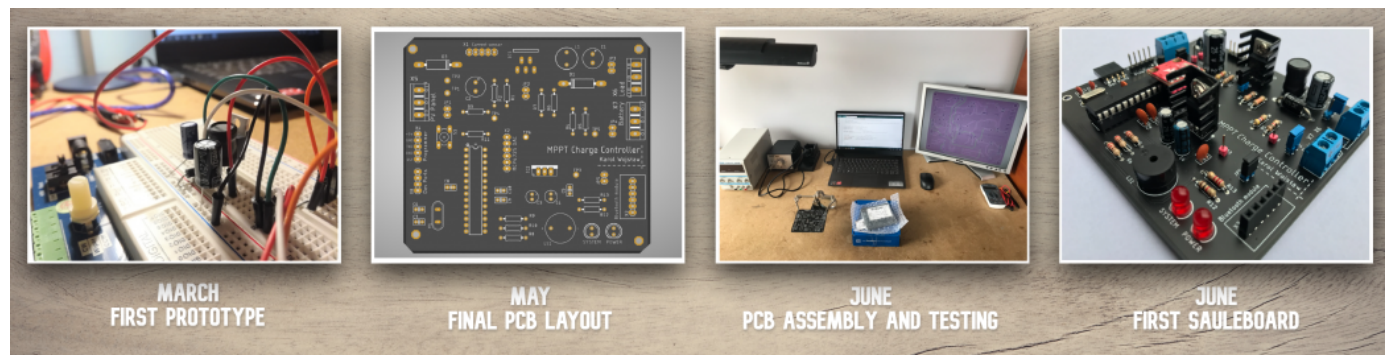


Figure 3 - SauleBoard PCB assembly animation



Software development – from first sketches to the board library

I didn't have much experience with "pure" C/C++ embedded software but I have been using the Arduino library for a long time, so I decided to stick to that option in this project as well (at least for the prototyping and first boards). I started with separate sketches for each functionality, and I tried to test them one by one before merging them into bigger files. Finally, I decided to try my hand at Object-Oriented Programming (OOP) and create a C++ library based on the Arduino library that would make the code more reliable, easy to read and edit.

There are several methods in the board class however, the two most interesting ones are `get()` and `setVoltage()`. The first one may be used to measure input and output voltage as well as the current (using Grove ACS70331 based module), while the following one is used to set some exact voltage at the output.

Taking voltage and current measurements

Measuring voltages, which includes also measuring current sensor output voltage is done by Atmega328 internal 10-bit ADC. The method (function in the class) takes as a parameter an int number indicating a command, however, by using defined in the library macros we can e.g. write `board.get(PV_VOLTAGE)`.

After being called it evaluates what is to be measured and what are the adequate voltage divider resistor values (values of which have been entered during the calibration process). Then it takes 100 measurements one after another with 5 ms breaks. It then calculates and returns the average of those readings.

Setting output voltage (Proportional controller)

The method that sets the output voltage is just a bit more complex as it uses the approach of a proportional controller. After we call it by e.g. writing `board.setCharging(12.54)` it will try to do its best to really set the output voltage to 12.54 V.

It starts with calculating the theoretical voltage that should be set on the feedback pin, and hence an adequate 12-bit value that should be passed to the DAC module. It sends it and then waits 200 ms for the output voltage to stabilise. After this time it takes a measurement (using `get` method) of the true output voltage and calculates the error.

If the error is bigger than ± 15 mV the board will try (for a maximum of 5 times) to decrease the error value. It firstly divides it by two, and then add/subtract it to the originally expected value. After first calibration and tests, it seems to work pretty well, as the algorithm was able to size down the error so far in all attempts, always with less than 5 iterations.

```

/* Method that change the charging voltage (LM2576 Vout) to a given value */
void Solar_charger::setCharging(float Voltage_V) {
    Adafruit_MCP4725 MCP4725_DAC; // Create MCP4725_DAC object
    MCP4725_DAC.begin(0x60);      // MCP4725 digital DAC I2C configuration (Address)

    /* Start by setting the feedback (and hence the output) according to the theoretical value
    uint16_t Vbat_desired_mV = (1000.0 * Voltage_V), Vfdbk_MC4725_mV, fdbk_MC4725;
    uint16_t error, Vbat_desired_mV_corrected = Vbat_desired_mV;

    Vfdbk_MC4725_mV = (1309 - (uint16_t) ((float) Vbat_desired_mV * 0.05988)); // Calculate the feedback value
    fdbk_MC4725 = map(Vfdbk_MC4725_mV, 0, Vin_mV, 0, 4095); // Map the value to the DAC range
    MCP4725_DAC.setVoltage(fdbk_MC4725, false); // Send the value to the DAC
    Solar_charger::LM2576(ON); // Turn on LM2576

    delay(500); // Wait for the voltage to stabilize
    error = Vbat_desired_mV - Solar_charger::get(BAT_VOLTAGE); // Calculate the error

    for(uint8_t i = 0; i < 5 && MOD(error) > 15; i++) { // For error > 15mV
        Vbat_desired_mV_corrected += (error / 2); // Add/subtract half the error
        if(Vbat_desired_mV_corrected > 6000) { // Protect the battery
            Vfdbk_MC4725_mV = (1309 - (uint16_t) ((float) Vbat_desired_mV_corrected * 0.05988));
            fdbk_MC4725 = map(Vfdbk_MC4725_mV, 0, Vin_mV, 0, 4095);
            MCP4725_DAC.setVoltage(fdbk_MC4725, false);
        }

        delay(500); // Wait for the voltage to stabilize
        error = Vbat_desired_mV - Solar_charger::get(BAT_VOLTAGE); // Calculate the error
    }
}

```

MPPT algorithm

MPPT algorithm is one of the cores of the SauleBoard. The need for any algorithm at all arise from the specific behaviour of solar cells. Their I-V characteristic is not linear like for e.g. an ideal resistor, but it's rather a curve that looks like the one shown on the left. You can notice that the voltage is constant

and independent of the current up to some point, after which it plummets and quickly reaches the x-axis, i.e. 0 V.

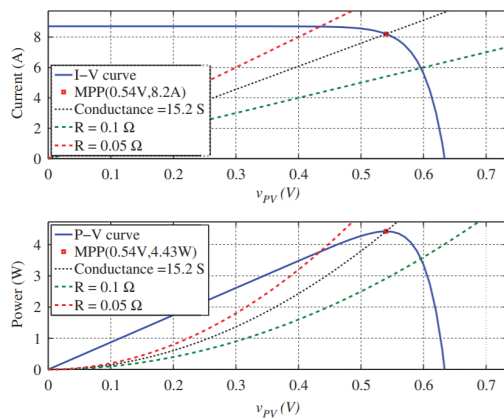


Figure 4 - Solar cell I-V and P-V characteristics

The second important graph is P-V characteristics, which illustrates Power (Solar cell current * voltage) against cell voltage. Now it becomes obvious that the power delivered by the photovoltaic panel depends on the voltage/current - the load connected to the panel. So depending on the impedance of the load there will be different voltage, current and hence power generated. Even if the Sun is at its zenith, there are no clouds nor wind, but we will connect a large value resistor to the panel, we will get only a fraction of the max panel power. The same is true for batteries. That's why it's often cost-effective to implement MPPT charge controllers that will alter the input conditions in such a way that the solar panel will be working at its maximum power point at all time. ...And that's where the name comes from, as MPPT stands for Maximum Power Point Tracking. :-)

There are several ways to implement such an algorithm, from the easiest ones up to much more complex ones, that can operate fast and with great precision, which can be crucial e.g. at the large solar farm generating Mega Watts of power.

The easiest to implement is the "hill-climbing" algorithm, which is an optimization technique. The idea is to alter a variable that changes the input (PV panel) condition (that can be a duty cycle in a switching step-down regulator), and check if the power delivered to the system is greater or smaller than the one before the change. This approach can easily track down the Maximum Power Point (MPP), however, as always, it can't be so simple. In fact, there are several key disadvantages of such an approach. One of them is the fact that after each change in the input condition the board has to give some time PV panel and charger to stabilise - because of this "hill-climbing" strategy may be a bit slow and unable to quickly react to changes. The other significant problem is the oscillation around MPP. The charger will never reach a steady-state as it will always try to go "down" or "up" the hill.

Nevertheless, it seems that the "hill-climbing" approach is a good choice for the beginning and for many applications it will "do the job" well. If it comes to the SauleBoard, I've just started experimenting with the MPPT algorithms as the software is still in the development process. However, I hope that in the next few weeks I will be able to release at GitHub the final version (at least for now of course).

Conclusion, future development plans and interesting facts

The name of the project and the board comes from a Saule, who is a solar goddess in the mythologies of Baltic countries.

Downloads

PCB layout and schematic (/designspark/download-attachment/4ecb2fc2-d42d-11eb-b409-42010a00001f_Design%20files.zip)
ZIP, 908.6 kB


SauleBoard PCB photo (/designspark/download-attachment/cb953d36-d42d-11eb-9898-42010a00001f_SauleBoard%20PCB%20photo.png)
PNG, 6.0 MB

SauleBoard_src.zip (/designspark/rel-assets/dsauto/uploads/attachments/6b6f2724-d42c-11eb-81d5-42010a00001f_SauleBoard_src.zip)
5.5 kB

Open source licences (/designspark/rel-assets/dsauto/uploads/attachments/e7e4a568-d42c-11eb-8a01-42010a00001f_licence.txt)



karol_poznan(/designspark/user/karol_poznan)

(/  / - **Edit profile**)

I'm a 1st-year Electronic Engineering student at the University of Manchester. Originally from Poznan, Poland. Interested in embedded hardware and software, IoT devices, RTOS and robotics. I also have some experience with space technologies and stratospheric probes.

Share this post

Facebook

Twitter

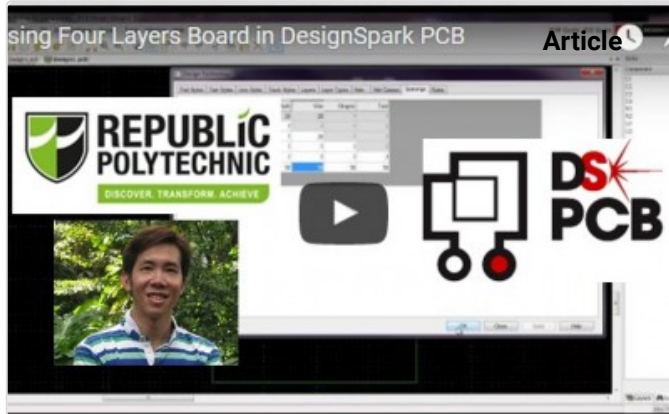
More

(/  / - **Edit content** / ? =)

Comments

Write a response...

Related Content



DesignSpark PCB Video Tutorials from Republic Polytechnic

👁 574 💬 1 👍 0

(/designspark/designspark-pcb-video-tutorials-from-republic-polytechnic)



SDRplay provides SDR-based "Radio Communications" undergrad teaching materials

(/designspark/sdrplay-provides-sdr-based-radio-communications-undergrad-teaching-materials)



Catch up with the Sussex Racing team in the first edition of our Newsletter #1


(/designspark/catch-up-with-the-sussex-racing-team-in-the-first-edition-of-our-newsletter-1)



In Conversation With... Elvis Tinago

👁 56 💬 1 👍 3

(/designspark/coffee-catch-up-with-elvis)

 English ▾

[About DesignSpark \(/designspark/about\)](/designspark/about) | [Cookie Policy \(/designspark/cookie-policy\)](/designspark/cookie-policy) |

[Privacy Policy \(/designspark/privacy-policy\)](/designspark/privacy-policy) | [Terms and Community Guidelines \(/designspark/terms-of-use\)](/designspark/terms-of-use) |

[Support Centre \(https://designspark.zendesk.com/hc/en-us\)](https://designspark.zendesk.com/hc/en-us)

Social Media: (<https://twitter.com/designsparkrs>) (<https://www.facebook.com/DesignSparkRS>)