Developer Kit 30W Project

Lead acid battery charger

Project overview
The aim of this project is to build a hydrogen powered 12V lead acid battery charger. It is based on the Arduino, Developer Kit fuel cell shield and the XP Power 30W DC/DC converter. We will use a P channel mosfet to regulate the current supplied by the fuel cell to a safe limit. The mosfet will be controlled by a PWM output of the Arduino.

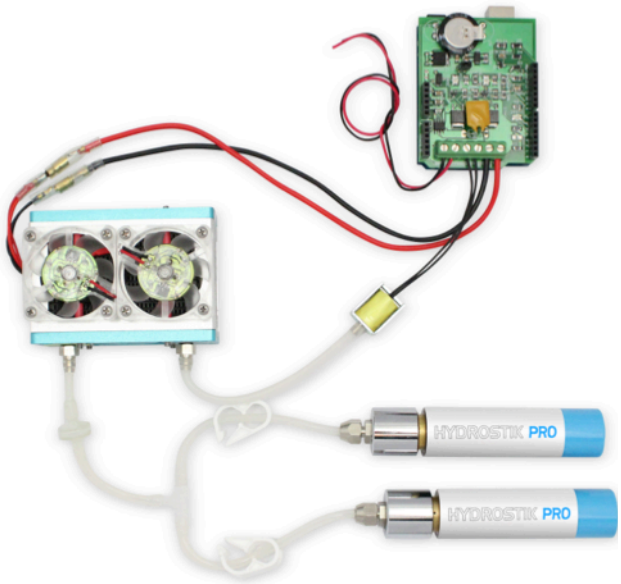

Table Of Contents

Hardware requirements

- 1 Arduino
- Developer Kit 30W fuel cell inventor kit
- 30W 15V output DC/DC converter (see below)
- Bread board
- Wires
- P channel mosfet that can handle 3A, for example FQP8P10
- NPN transistor, for example 2n3906
- resistors: 5k, 80k, 220k
- A 12V lead acid battery (capacity up to 7AH)
- A 10k or similar potentiometer
- A low ohmage power resistor (we used a 4ohm 25W)

DC/DC converter

The actual DC/DC converter used isn't so important. The features we need are:

- 30W
- Will boost from as low as 9V
- Output is 15V

An example of a suitable DC/DC converter is the XP Power 30W.

Battery charging theory

Say we have a project that is battery powered, and we want to extend the life of the battery. A fuel cell provides a neat way of doing this. However, if you look at the IV curve below, you can see that we don't get enough voltage under load to charge the battery. We want to provide 14 to 15V. This is why we need the DC/DC converter. The DC/DC efficiently boosts the lower voltage from the stack, high enough to charge the battery.

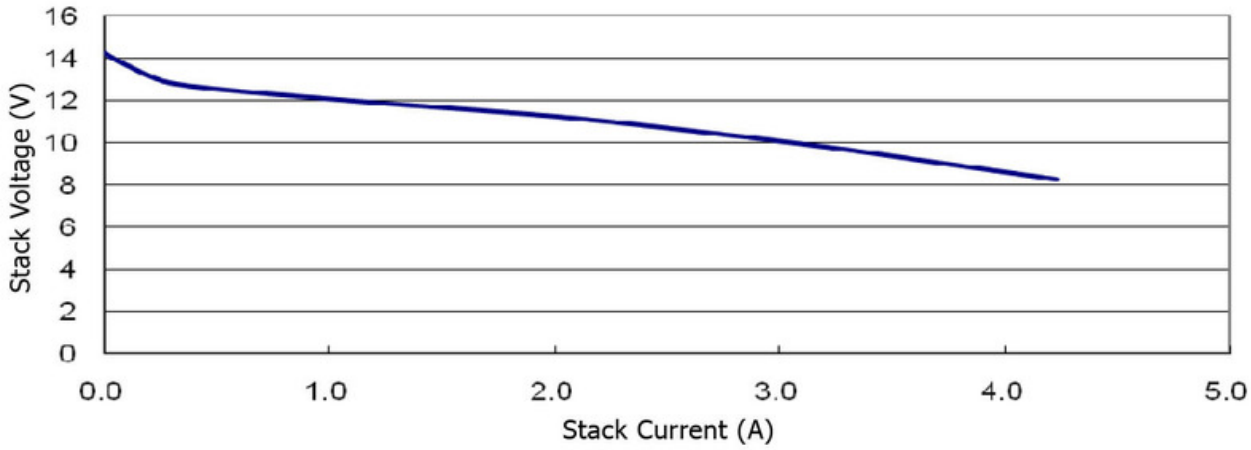

Figure 1. 30W IV curve

So we can choose our charging voltage with the DC/DC converter, but how do we choose our charging current? We have a balancing act; we want to run the fuel cell at maximum efficiency, which means close to its rated capacity, but if we try and draw too much power from the fuel cell, we can damage it.

A solution to this is to control the charging current with a PWM controlled mosfet. If the charging current is too high we can limit it by cutting back on the PWM amount. We can monitor total current drawn with the Developer Kit's on board current monitor, and we can monitor voltage with a simple potential divider.
To discover the correct PWM amount, we measure the current while adjusting the PWM with a potentiometer.

Software requirements

- Arduino IDE
- The Developer Kit library

Software preparation

Developer Kit library

Make sure you've got the latest version of the Developer Kit Arduino library installed by downloading and unzipping into your sketchbook/libraries folder.
Start the Arduino IDE and have a look at the file->examples->Developer Kit->batteryCharger sketch

Hardware preparation

Building the circuit

Build the circuit as shown in the diagram below except for now, don't connect the battery.

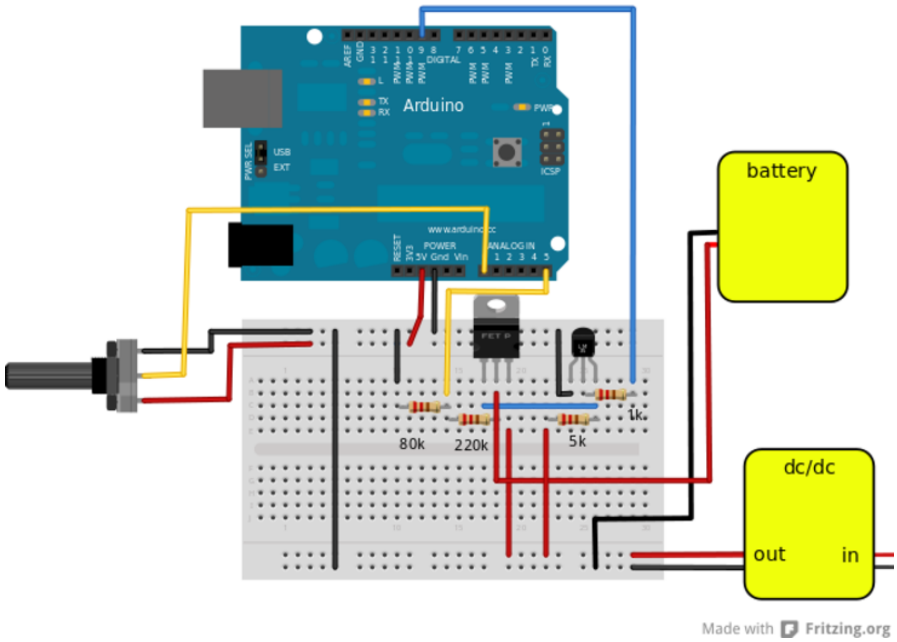

Figure 2. PWM battery charging circuit

Now load the examples->Developer Kit->batteryCharger sketch and upload onto the Arduino. Have a look close to the top of the file and you'll see this line:

```
#define SETUP
```

which tells the program that to start with we aren't charging a battery we're calibrating the system. Load the program onto the Arduino, and connect the hydrogen. Check the serial monitor. It should show something like this:

```
ArcolaEnergy fuel cell controller for 30W
bandgap mV set to 1100
Short-circuit: 100ms every 10 s
Purge: 50ms every 10 s
waiting for caps to charge
CHARGED
8.56V, 0.06A
charge PWM: 0
8.59V, 0.06A
charge PWM: 0
8.65V, 0.06A
charge PWM: 0
8.57V, 0.05A
8.56V, 0.05A
charge PWM: 0
8.61V, 0.05A
charge PWM: 0
```

If the charge PWM isn't 0 then adjust the charge knob until it shows 0. Connect the power resistor where the battery will go. We're going to calibrate the PWM charge rate using the resistor as this allows us to simulate a discharged battery. Connect the hydrogen (or use a bench PSU set to 10V current limit 2A). Slowly adjust the charge knob and as the PWM value increases you should see the current drawn from the fuel cell/PSU increase:

10.55V, 0.01A
charge PWM: 0
10.57V, 0.02A
charge PWM: 1
10.69V, 0.17A
10.58V, 0.51A
charge PWM: 17
10.49V, 0.70A
charge PWM: 28
10.55V, 0.80A
charge PWM: 34
10.54V, 1.09A
charge PWM: 47
10.51V, 1.06A
charge PWM: 47
10.12V, 0.82A
charge PWM: 47
10.46V, 0.93A
charge PWM: 47
10.55V, 1.09A
charge PWM: 51
10.57V, 1.07A
charge PWM: 57
10.51V, 1.26A
charge PWM: 65
10.46V, 1.15A
charge PWM: 79
10.52V, 1.29A
charge PWM: 85
10.55V, 1.17A
charge PWM: 107
10.43V, 1.15A
charge PWM: 115
10.42V, 1.08A
charge PWM: 116
10.06V, 1.18A
charge PWM: 116
10.37V, 1.04A
charge PWM: 116
10.39V, 0.94A
charge PWM: 116
10.44V, 1.11A
charge PWM: 123
10.42V, 1.03A
charge PWM: 140
10.43V, 0.97A
charge PWM: 152
10.39V, 1.12A
charge PWM: 159
10.45V, 1.00A
charge PWM: 169
10.39V, 0.95A
charge PWM: 172
9.60V, 1.18A
charge PWM: 173
9.40V, 1.20A
charge PWM: 172
9.34V, 1.29A
charge PWM: 172
9.18V, 1.53A

We've increased the PWM value to 172 to set a charge rate of about 1.5A. You can set the charge rate to anything you want, but ensure you're not running more than 3A for the 30W or 1.5A for the 15W. Find this line in the program:

```
const int chargeLimit = 100;
```

Change the value to what you want your charge rate to be. In our case it's 172:

```
const int chargeLimit = 172;
```

Then change this line

```
#define SETUP
```

to:

```
//#define SETUP
```

Which means the program will run in battery charging mode. Have a look at the values at the top of the file:

```
const int batteryStartChargeV = 11500; //mv
const int batteryEndChargeV = 14500; //mv
long maxChargeTime = 600000; //10 minutes expressed in ms
```

- batteryStartChargeV is the voltage (expressed in mv) when we start charging the battery,
- batteryEndChargeV is the voltage (expressed in mv) when we regard the battery as charged,
- maxChargeTime is the time (expressed in ms) when we give up charging the battery.

Now disconnect the load resistor and insert the battery. We recommend using a 3A quick blow fuse in series with the positive terminal. Recompile and load the program onto the Arduino, and you should see this in the serial monitor:

```
charging
battery voltage: 10464.90
10.45V, 0.95A
10.46V, 0.86A
charging
battery voltage: 10061.12
10.48V, 0.98A
```

Depending on what state your battery is in and what the values of the above variables are, the program will either be waiting to charge the battery, or be charging the battery.

This concludes the project, and shows the Developer Kit working to charge a battery as and when required.

Expansion ideas

- Add an LCD,
- Use a more sophisticated charging method (for example a PID controller)
- Combine with the remote sensor project to provide a warning when the battery is getting low.