Off the Clock: Designing a Robot part 1

This is my second posting in the Würth-Midcom design challenge “Off the Clock”. I was tempted to arrogantly call the upcoming series of posts “How to Design a Robot”, because I’m pretty damn good at designing electronics and at the least decent at mechatronics. But I realized I’m not that arrogant, I’m just about young enough to still get away with it, but on the other hand after over twenty years of working (for money) in the fields this is about, I prefer erring on the experienced side. That means I realise that no one person can do everything exactly right and there is no single “How to” for anything as complex as designing a robot.

Instead, I’d prefer everyone to consider this a guide to how I prefer to tackle the design of a Robot. This all started while working on a Demo board for my local Würth reps, when I found a page on the <http://www.we-online.com> website while downloading the latest Altium libraries:

<http://www.we-online.com/web/en/electronic_components/termine_pbs/offtheclockdesignchallengelandingpage.php>

I decided that the contest advertised on this page, to make something cool with wireless energy, sounded like an interesting idea, so I submitted my proposal to make a small helper-bot that could use several attachments to help me around my own lab, and when the battery runs down, to locate its base and recharge wirelessly. Some reasoning behind wireless energy for the charging can be read in my previous post: <https://medium.com/@asmyldof/off-the-clock-3069af1cdd63>

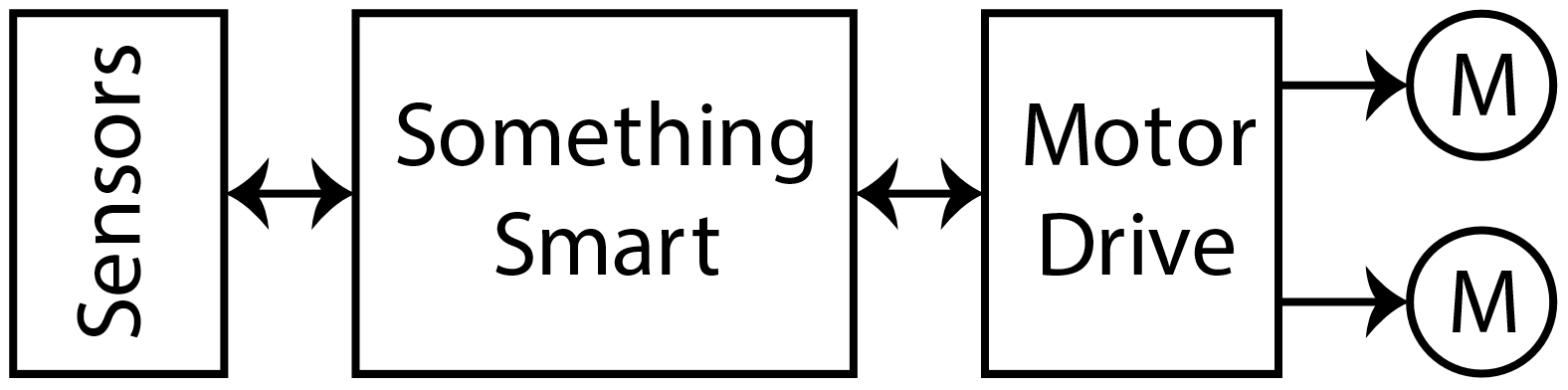
Of course, after that post I found some more good reasons:

1. It’s easy to put the bot onto its charger if it can’t find it: No orientation. No clicks. No latches.
2. The robot can “Prepare for Take-off” in a specific direction while still connected to the charge point.

Of course, the challenge I gave myself is to never need those two reasons, because I will try to make it smart and power efficient.

## Global Idea

Now that the decision is made to build a smart and energy efficient robot, we need to work out how it’ll work. What will the electronics look like? For the first parts, I’m going to leave the wireless electricity out of the equation, as Würth-Midcom have sent me a fully finished product for that (free of charge), which I can use as a “power plug” for my Robot. So, what does a robot need? A first thought would look something like this:

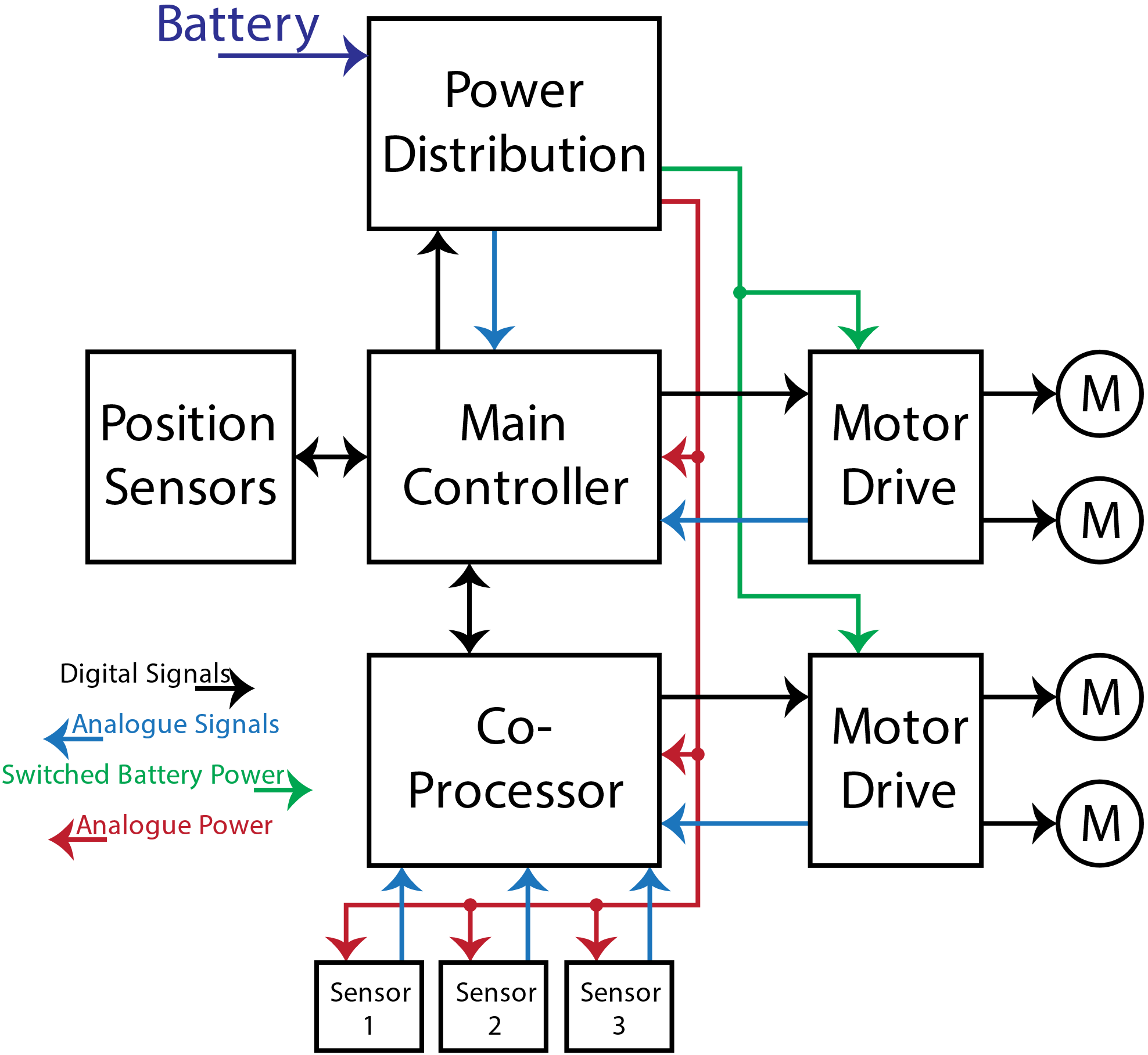


Which isn’t quite helpful yet, all we know is it’s smart, has sensors and has motors. I did make sure to note it has two motors. This can be a main motor and a navigational motor, like a propulsion engine and a steering servo, but it might be two propulsion motors working together like in a tank. The latter is more likely, since my lab always has cables and other obstacles, so we’re looking for an “all-terrain vehicle” type device.

## Expanding the idea

Now that we know the three main blocks we envision we can start to work out the “Something Smart” block and the “Sensors” a little bit. If we want the device to be super power efficient, it will need to be able to turn off large parts of its functionality when it decides to sleep, so it’ll need some power-control. If I want to have a lot of options with respect to expanding the device and having great options for communication, but also analogue, I’ll need some support to the main controller as well.

If I go for one powerful main controller with external ADCs, Amplifiers, and several external I/O expanders I’ll need quite a lot of chips on different voltages. Instead, being a long-time fan of the Atmel architectures, I decided to have a look at what was currently in the range of Arm-Controllers over at Microchip, who recently bought Atmel and quickly found that I could do all of that in just two affordable chips. Which makes the first step of fleshing out look a bit like this;

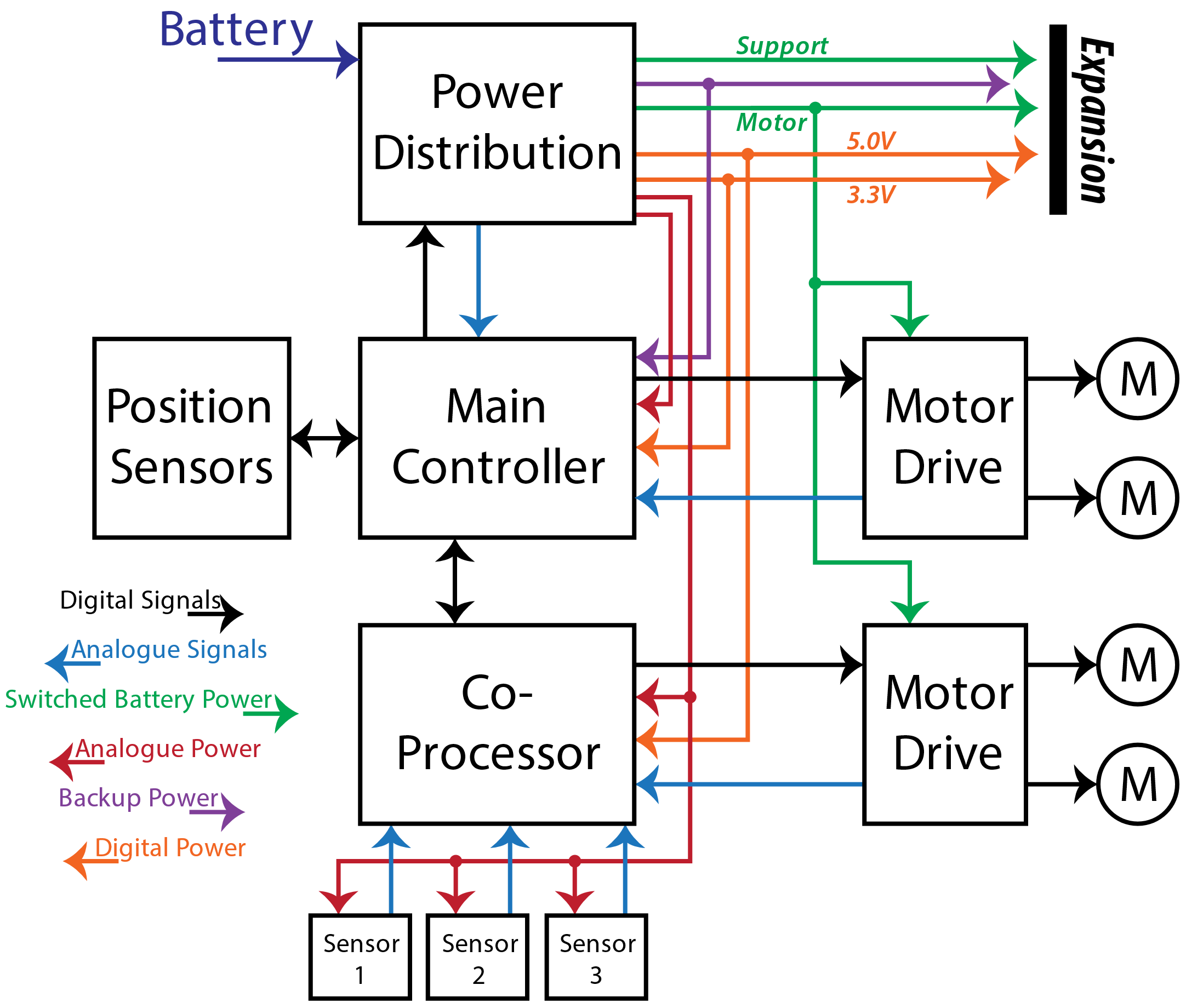


As can be seen I decided to give the Co-Processor two more motors to take control of. In principle, I’m saying: “I will probably put in two main motors and two auxiliary motors, or at least their drivers, and I’ll figure out what to connect where later.” The two extra motor controllers also already hint at a small fun idea I’m building in the back of my mind: Building a hopper. I may just equip my bot with a set of propellers so it can hop obstacles. It will probably not fly large distances, since that takes a lot of power, though at this point I don’t even know if I’ll get to flight at all.

## Designing the power paths

A possible next step is to add some more detail to the plans for the power-system. Before we go into that, it’s important to know that I plan to make a basic robot that navigates, which can attach expansions to perform interesting tasks. I want, if possible, to make the basic system compatible with 3.3V, for my own work, but also with 5V, so that other people can hook up any type of Arduino or Raspberry they have laying around.

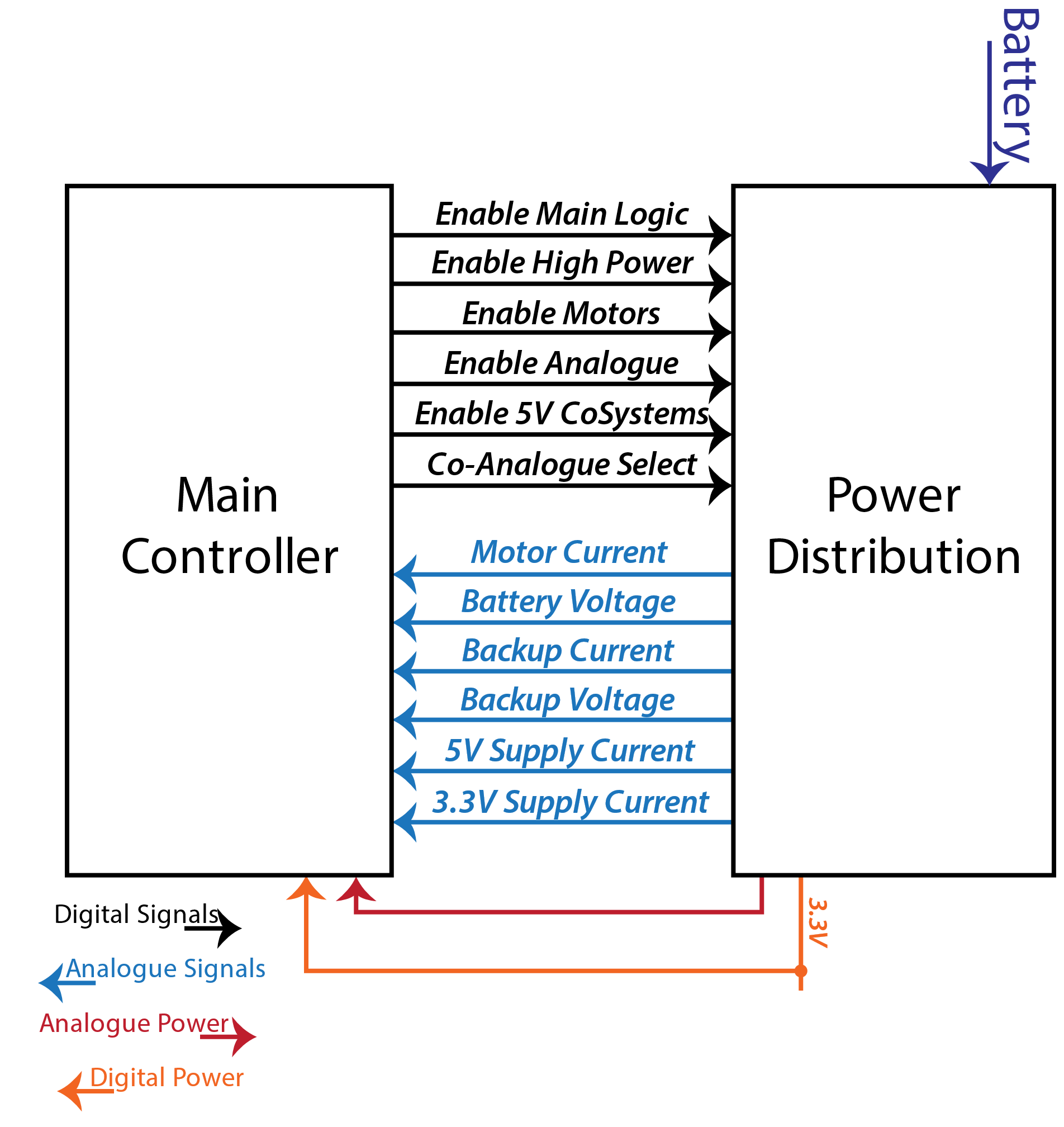
So, looking back at the Microchip site, I decided to order samples for the Atmel ATSAMG55 and the ATSAMC21. The first one as the main processor, featuring USB2.0, up to 120MHz of core speed, floating point numbers and SRAM high speed code execution. The second one as a co-processor, which can run on voltages up to 5V and features a richer set of analogue and timer options. A match made in heaven, for only $6-ish at volume. Which makes the power-path look as below:



Now there will be a switchable 5V, which powers the expansion and Co-Processor, a switchable 3.3V which powers the expansion and the Main Controller, a switchable Support Power for anything running at battery voltage and one special switchable line for Motor Power. I also chose to explicitly draw the two analogue supplies separate, since we may want to keep an eye on battery voltages with the main processor, while we leave the auxiliary sensor grid powered off. In this design, the Co-Processor can host all the 5V data busses, while the main processor hosts all the 3.3V data busses and I’ll only need to put a level translation in the link between the two processors.

## Power Control in Detail

The last part of this design session I’ll quickly zoom in to the power-control. If only to make all of you start wondering about all the other signals, as I’ll work on the images and blog posts detailing those later.



This image simply shows what kinds of control signals the Main Controller sends to the Power Distribution and what is measured back. Some of the analogue signals may be processed by hardware comparators or dedicated charge control chips at a later stage in the design, but for now we assume all that is done in Firmware, through the internal ADC. Since the backup voltage is also supplied to the expansion connection it is assumed for now we keep an eye on power consumption in that as well in some manner. Maybe the thing will trip a shut-off of the expansion at more than a few mA, or possibly there will be several alarm levels. This will also depend on the later choice of whether or not to add an actual backup battery or not. Right now, the backup power could also be a very-low-power linear regulator from the main battery, allowing the main processor to only “leach” a few dozen uA from it while sleeping. But I want expansions to also have the option to “stay ready” in a low power state when all main power is switched off, so the expansion will get a backup power wire, switched or not.

## Final notes

You can keep up to date with the main design of this project, once real content starts to get generated at any or all of the following GitHub projects:

* <https://github.com/Asmyldof/OffTheClock-HW>
* <https://github.com/Asmyldof/OffTheClock-FW>
* <https://github.com/Asmyldof/OffTheClock-SW>

Or you can look at the source files for this blog post and the revisions I made before publishing at the following link:

* <https://github.com/Asmyldof/OffTheClock-Docs>