Houston, the Ambilight project is reporting in! The project is still on track and our time plan seems to be working out pretty well. We managed to achieve all out goals for the last weeks, which focused primarely on the communication between the Arduino and the Android App.

**JSON for the win**

Like we explained in the previous post, the lighting profile consists of the following settings:

* At least one color represented by the RGB values
* 2 thresholds for the temperature and
* 1 threshold for the brightness regulation.

Since the user can set more than one color, the length of the data sent to the controller is variable, which would make it complicated to parse the string received via the TCP connection. We decided to use JSON as exchange data format to avoid having to write an own parser. Luckily, there is a very easy-to-use JSON library for Arduino.  
(<https://github.com/bblanchon/ArduinoJson>)

**RGB and the color temperature**A bigger challenge will be the color temperature regulation. Naturally, there are colors which are perceived warmer or cooler than others, as seen in this diagram:

<http://www.johnpaulcaponigro.com/blog/http://www.johnpaulcaponigro.com/blog/wp-content/themes/zinfandel-blue-10/images/hue_coolwarm.jpg>

However, we want to keep the shade of color set by the user in the activated profile yet still change the color temperature. We’ll have to try out how the RGB values change for warmer/cooler shades of a color and find an algorithm which works for every color.

For some tricky colors like blue, which is a cool color by nature, a temperature regulation could like this:

**What’s next?**Before tackling this core problem, we still have to implement a color gradient in case a user chooses more than one color. Another challenge will be to combine the color gradient with the temperature regulation, but we’ll probably think about that later. The app only needs some usability improvements and the design needs to be polished up.