Project Documentation

Project: Web-controlled LED light-suit

Mike Schoder

# Overall Description

## Product

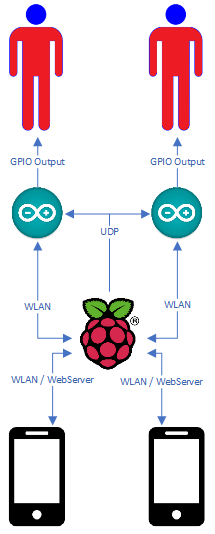


Figure 1: Product Overview with Interfaces and used Protocols

As the picture above shows us, we are hosting a webserver on a raspberry-Pi, over which the Arduino’s get the commands to switch the current animation. The webserver is available only on the Wi-Fi hosted directly on the PI. This Wi-Fi is hosted without a Broadcast, so with scanning for available Wi-Fi’s, it will not show up. Connections can only be made if you know the Wi-Fi, so it is relatively safe. Once connected, you get to open the browser on the device you just connected. There, type in the IP of the Raspberry, and you get a Website with many Buttons. By a push of a Button, the Raspberry sends a UDP-Packet to the Arduino that includes a code, which animation now should be played instead of the current.

## Product Features

### 1.2.1 GUI (Graphical User Interface)

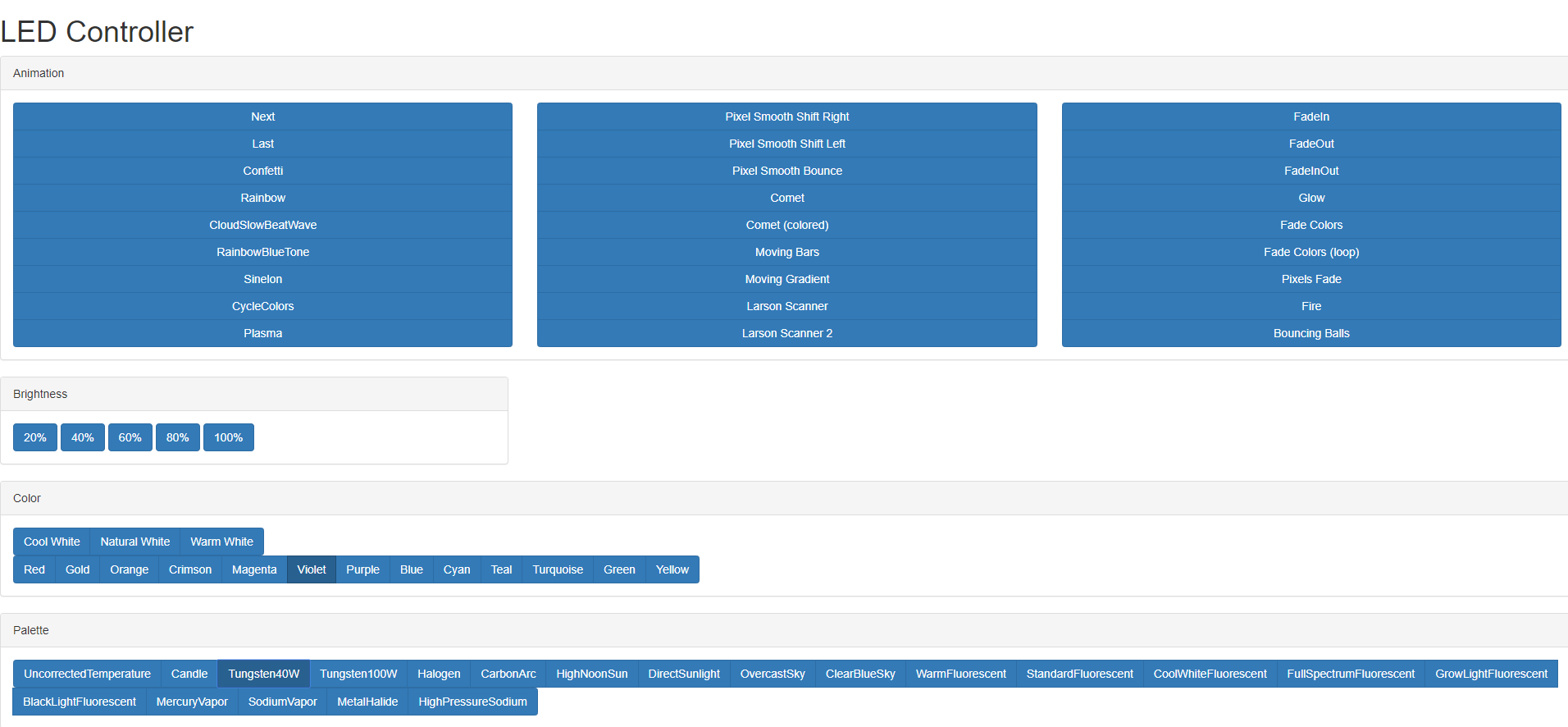


Figure 2: The GUI of the Webserver, divided into animations, brightness, color and color Palette

This is the GUI of the webserver that is hosted on the Raspberry-PI. It is created with a simple HTML-Page and a style guide from <https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css>. Flask is the chosen webserver, which hosts the webpage and listens for user interactions on the End-Device.

### 1.2.2 Communication

A major feature of the product is its communication between the devices, which allows fast changes on the visible part, the suit itself. It is implemented by web-sockets that use UDP as transfer protocol. At First, the hosted webserver listens for user interaction. If a button gets pressed, the webserver sends an UDP-packet over a web-socket to the broadcast-IP of the network. The Arduinos listen on incoming UDP-Packets and change their animation based on the content of the packet.

### 1.2.3 Customization

A significant function of the product is its customization. By using the GUI and the communication, you can rapidly change your look and react to outside influences. The suit’s Strobe-Effect gives someone epileptic seizures? Turn the brightness down or change the animation instantly to something more smoothly. The suit gathers too much attention? Turn it off instantly without damaging something, by just one click on a button.

## Operating Environment

The Product is built out of these components:

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Name | Quantity | Special |
| Microcontroller | Raspberry PI (newest Raspbian by 22.10.2019) | 1 | Installed Flask Webserver for Python 3 |
| Microcontroller | Arduino NodeMCU 1.0 with built-in ESP8266 | 2 | Installed Library is FastLED 3.0 |
| End-Device | Samsung Galaxy S9 | 2 | Fitted with a webbrowser |
| LED | NeoPixel Strip of LED | 2 x 5 metres |  |
| Cloth | Plain black Overall | 2 |  |

# Product

## GUI

<https://github.com/Mikscho/FastLEDpro/blob/master/gui.html>

The GUI, also visible under Point 1.2.1, is made of many buttons in different areas. Those Areas are split into their functionality, for example the buttons with the animations are on the top and not in the same category as the buttons for the brightness.

You can choose between different animations, different brightness, different color-palettes and different colors. Color-palettes are there to define the spectrum of colors. For example If you choose a halogen-palette, the colors range from blue to grey, no red and green colors are applied. The Colors are for animations that only use one color, like the animation Fade in. The specified color gets saved on the Arduino and defines the color of the next animation.

## Communication

The communication is made on the broadcast-IP of the network that is hosted on the Raspberry-PI. Basically, every device joined to this Network gets the UDP-packets from the web-socket, so you can maximally join 250 Arduinos, which means there can be 250 Suit’s that run the same animation at the same time.

<https://github.com/Mikscho/FastLEDpro/blob/master/index.html>

<https://github.com/Mikscho/FastLEDpro/blob/master/WebSrv.py>

The Communication on the side of the webserver is made with a python-script, that listens for access to sites that get called from the end-device. When you access the webserver, you can see the index.html. Every button has a link to its own site, that the end-device accesses. The listener gets the path and sends the digits over UDP to the broadcast-ip.

<https://github.com/Mikscho/FastLEDpro/blob/master/WebOverPISeq/WebOverPISeq.ino> Lines 91 - 108

The communication on the site of the Arduino is simply a listener, that saves the input of the UDP-packet sent to it. Based on this input, it changes its animation or color.

## Functions

<https://github.com/Mikscho/FastLEDpro/blob/master/WebOverPISeq/WebOverPISeq.ino> Lines 273 - 590

So far, there are 16 different animations, 5 brightness-settings, 14 Colors and 20 Color-Palettes implemented. But this product can be further expanded with new animations or colors.

## 2.4 Implementation

<https://github.com/Mikscho/FastLEDpro/blob/master/WebOverPISeq/WebOverPISeq.ino>

Arduinos are single-processor microcontrollers that are only capable of running one process at a time. But for this project, we want 2 different processes to run at the same time. Those processes are listening on new inputs, while at the same time it must output an animation on the light-strip. To tackle this problem, the listener is made into a function that gets called every now and then by the main process, the output of the animation. This means, that the animation you can see on the LED-strip gets interrupted to listen for new inputs, but this call is too fast to recognize it in the animation.