Contents

1 Introduction

This projects tries to use an Arduino[1] Uno to transfer the screen output of a Raspberry Pi to one or more addressable LED stripes. The Raspberry Pi screen is read out using boblight[2] and transfers a color array via UART to the Aruino, which creates the output for the LED stripe(s). In this project, the WS2811[3] LED-controller is used.

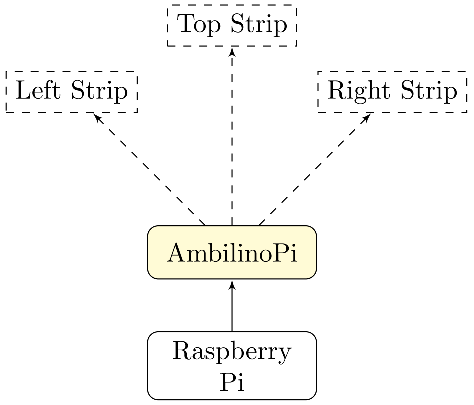
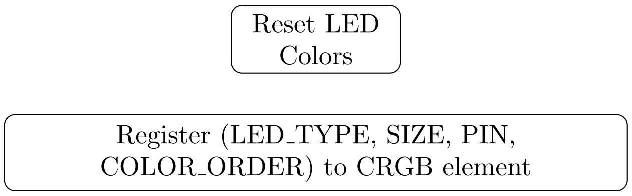


Figure 1: General Program Flow

1.1 Eclipse Arduino IDE

I used the Eclipse Arduino IDE [4] for creating the Arduino Software.



2 Setup

2.1 Hardware

2.1.1 Power consumption

The power consumption of an all white LED strip can be calculated using the equation

$$I_{LEDStrip} = 3 * 0.02\,\mathrm{A} * n_{LED} \tag{1}$$

[1] http://www.arduino.cc
[2] code.google.com/p/boblight/
[3] www.adafruit.com/datasheets/WS2811.pdf
[4] www.baeyens.it/eclipse/

So for 60 LEDs you would get a power consumption of 3.6 A! Inappropriate power sources result in LED flickering or other unexpected effects.

2.1.2 Board Schematic

The basic schematic of the circuit looks like this:

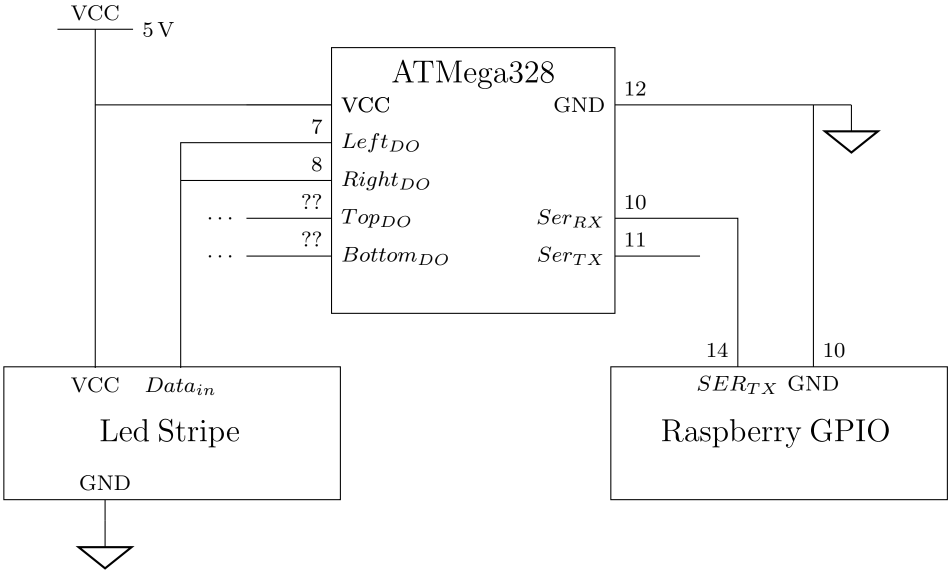


Figure 2: Schematic of Arduino, Led Stripes and Raspberry GPIO

2.2 Boblight

The Boblight software is used on the Raspberry inside the XBMC distribution like Openelec[5]. The LED setup has to be configured inside the configuration file which is loaded at startup of boblightd. You can use a configuration generator which can be downloaded at sedu-board.de. Refer to the boblight documentation[6]or details. In general, the file should contain the number of channels (= number of LEDs * 3), the update interval, prefix, postfix, and the baudrate of the system.

After the general setup, you can tweak the colors of your stripe with adjust, blacklevel and gamma. When you are finished, you can set the individual LEDs and their corresponding screen rectangles.

2.3 Arduino

The main purpose of the Arduino firmware is the translation of the UART output of the Raspberry to the desired LED strips. Another thing is the smoothing of incoming colors so that the LEDs do not look too 'agressive'.

[5] http://www.openelec.tv/

[6] f

2.3.1 Capture Rate

The time between two successive frames has an lower limit due to maximum serial readout. Currently it is used `8-N-1`[7]. So there are 10 bit needed to transmit 8 bit of usable data. This means, that only $P_{8N1} = 80\,\%$ of the transmission rate can be used. The minimum LED update frame time regarding the number of used LEDs and a specific Baudrate can be calculated:

$$T_{cap}(R_{Baud}, n_{LED}) = \frac{24\,\mathrm{bit} * n_{LED}}{R_{Baud} * P_{8N1}} \tag{2}$$

The 24 bit represent the size of a single color sent to one specific LED. Setting $n_{LED} = 60$ and $R_{Baud} = 38\,400\,\mathrm{bit/s}$, T_{cap} results in

$$T_{cap}(38\,400\,\mathrm{bit/s}, 60) = 46.88\,\mathrm{ms} \tag{3}$$

This results in a maximum theoretical frame rate of

$$R_{cap} = {T_{cap}}^{-1} = 21.33\,\mathrm{Hz} \tag{4}$$

It is recommended to guarantee $R_{cap} \geq 15\,\mathrm{Hz}$ for sufficiently dynamic screen capturing.

2.3.2 LED Smoothing

Putting the input colors directly on the LEDs can look quite aggressive and non-natural. It is recommend to use some kind of low pass filter on the incoming data. Figure 4 tries to explain the realization of the filter in this project for the red brightness in some frames.

The maximum change of brightness within one frame can be set via the maximum smoothing value combined with the maximum step size. The delay time between two smoothing frames can be set by the smoothing delay.

3 Contact

If you have any questions, feel free to contact me by mail: pfuhlert@web.de

[7] `http://www.modemhelp.net/faqs/8n1.shtml`

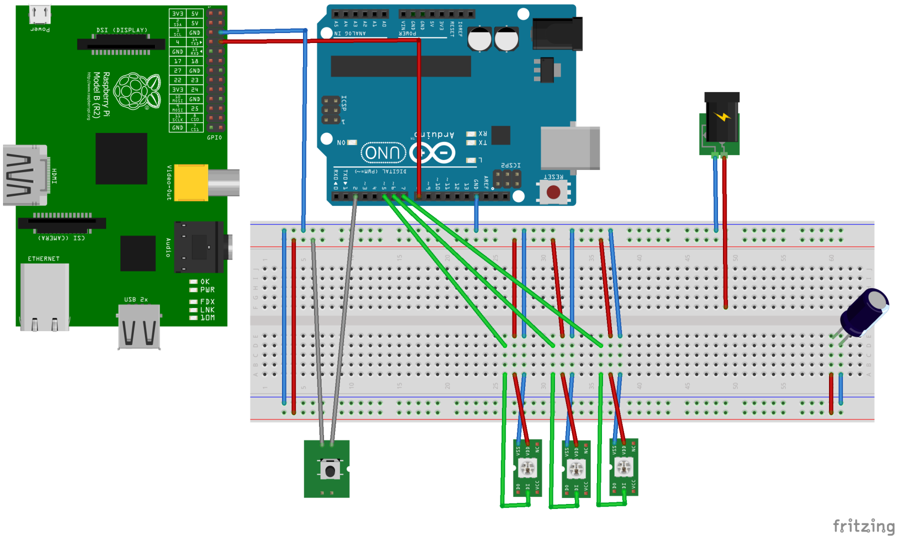


Figure 3: Realisation of the schematic on breadboard.

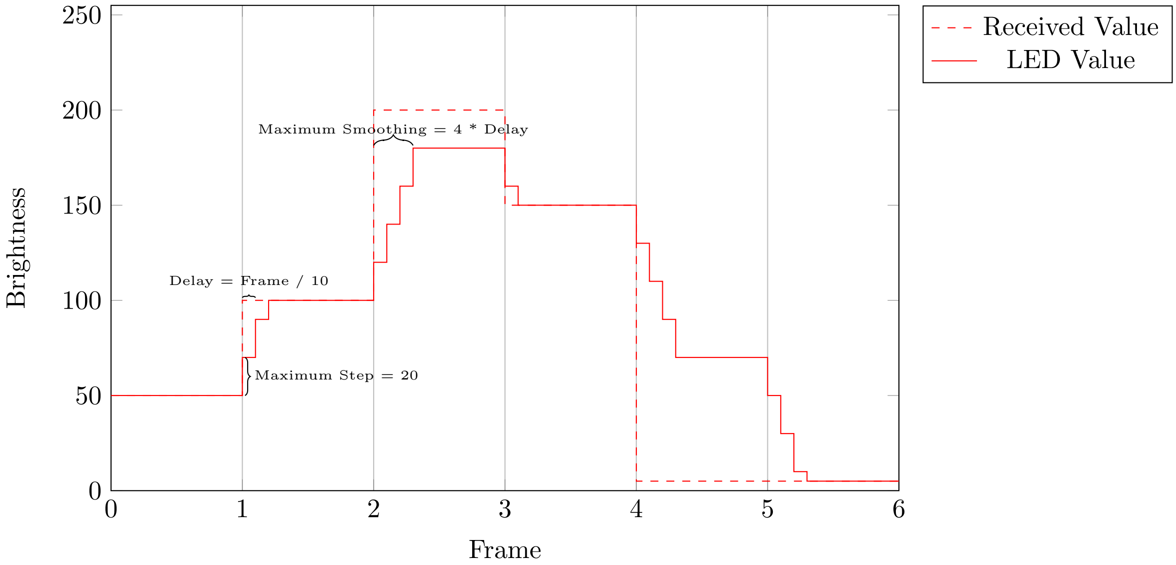


Figure 4: Graphical representation of smoothing algorithm using: