Concept Report

Moodlight

I. Krajinovic, L. Müggler

Lecturer: M. Nussberger

ZHAW School of Engineering ET18

Winterthur, 28.03.2020

Abstract

This report should describe the ideas and the corresponding requirements for the hardware and software of the Moodlight project. This project is part of the ETP2 course at the ZHAW.

The goal of this project is to develop four current drivers for the LED strings red, green, blue and white. Furthermore these LED strings should be controllable via a smartphone app. At least one extra feature is required in addition. Finally, the whole hardware should be covered by a lampshade or mounted in an enclosure.

To control the brightness of the different LED strings, a PWM signal per string is used to adapt the current through the LEDs. The PWM signal is controlled by the touchslider on the Gecko DevBoard or the smartphone app, which communicates via bluetooth. The Moodlight is additionally equipped with an ambient light and a temperature sensor. The sensor data should allow two functions. The first function is the temperature mode, which allows to convert the ambient temperature to a RGBW-value. This leads to a temperature-dependent colour output on the Moodlight. The second function controls the brightness of the LED strings by the illumination data. This results in a ambient light-dependent brightness output on the Moodlight.

Hardware Concept

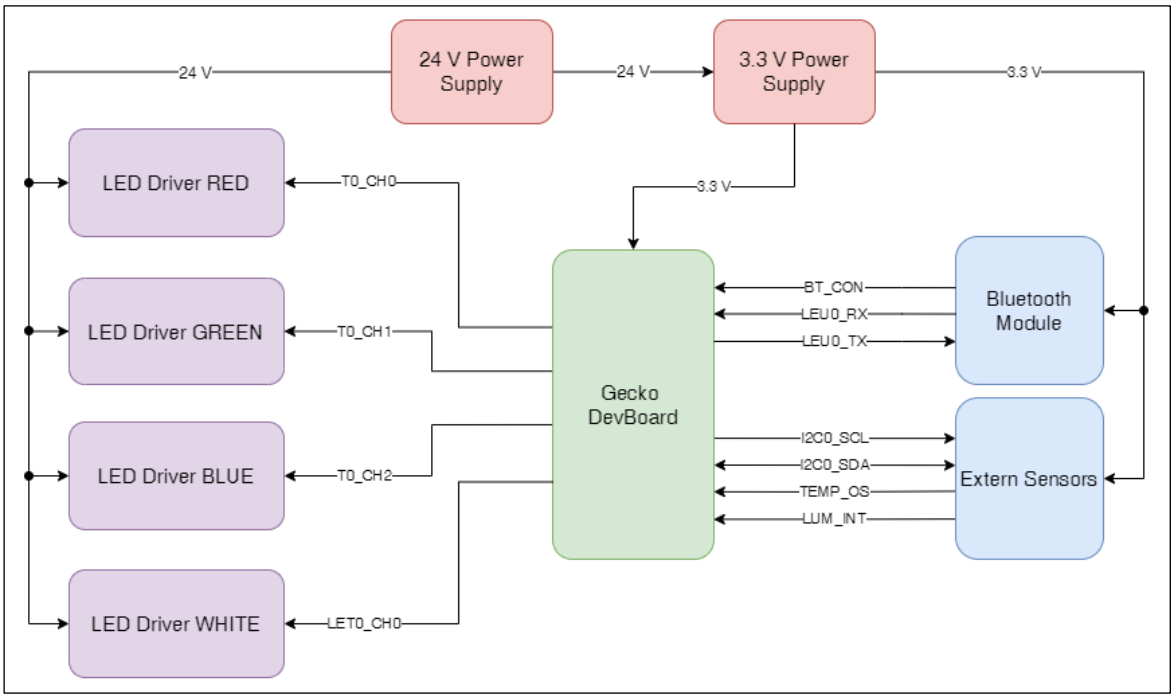

Figure 1: Moodlight blockdiagram

Because the Moodlight is used as an embedded system, a power supply of 3.3 V is needed for the bluetooth module and the Gecko DevBoard. Four PWM signals are used to control the individual LED strings. As shown in Figure 1 the PWM signals of the red, green and blue strings are controlled by TIMER0. The signal for the white string is generated by LETIMER0. The LED driver *ZXLD1350* is operating in low frequency mode and therefore the PWM frequency is set to 400 Hz.

The bluetooth module is connected with the Gecko DevBoard via LEU0 interface (UART). With the signal BT_CON the Gecko DevBoard gets informed when a bluetooth device is connected. The I2C0 interface of the Gecko DevBoard is responsible for the communication with the extern sensors. For

additional functions the signals LUM_INT and TEMP_OS are routed to interrupt-capable GPIOs of the Gecko DevBoard.

Gecko pin	Signal name	Signal type
PD1	TIMER0_CH0	PWM
PD2	TIMER0_CH1	PWM
PD3	TIMER0_CH2	PWM
PC4	LETIMER0_CH0	PWM
PD4	LEU0_TX	UART
PD5	LEU0_RX	UART
PD7	I2C0_SCL	I^2C
PD6	I2C0_SDA	I^2C
PD0	TEMP_OS	Digital Input
PC5	LUM_INT	Digital Input
PE1	BT_CON	Digital Input

Table 1: Pin to signal distribution

Requirements for µC:

- I^2C interface
- UART interface
- 4 x Timer outputs
- 3 x Interrupt capable GPIOs

Microcontroller Software Concept

Each mode of the user interface corresponds to a status in the finite state machine. With the pushbuttons the modes can be changed manually. The smartphone app uses only the two modes *RGBW* and *TEMPERATURE* (green in Figure 2).

When a bluetooth connection is detected, the state machine will automatically switch to the state *RGBW*. If the Bluetooth connection is lost or disconnected, the state machine will change to the *IDLE* state. As long as a Bluetooth connection is active, the pushbuttons do not impact the state machine.

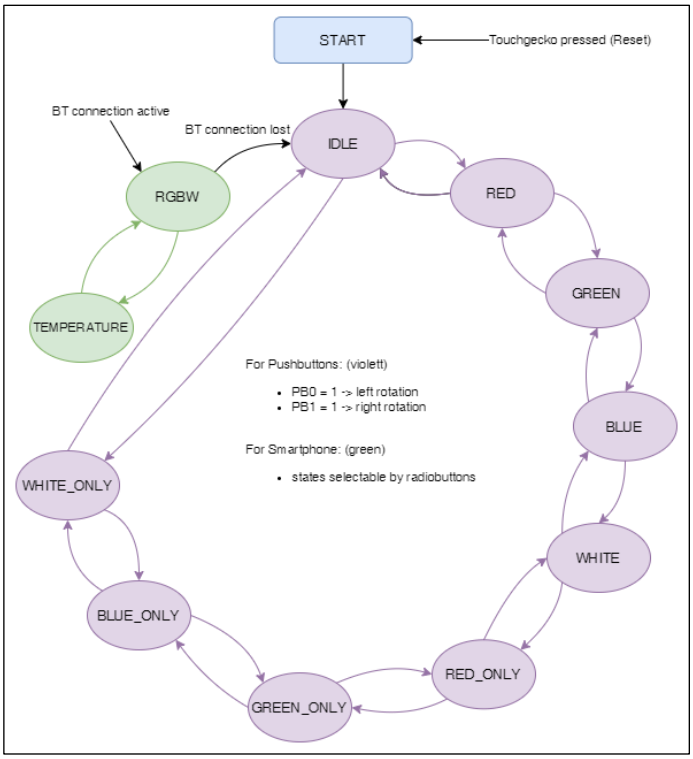

Figure 2: Finite state machine

Custom made interrupts of the UART and the touch interface generate flags which are polled by the software as shown in Figure 3 and Figure 4. If a flag is set, the corresponding function is called, which changes the PWM values or the states. Regardless of the mode, the temperature of the device is sent to the smartphone after each loop. The PWM duty cycle is depending on the maximal allowed forward current through the LEDs and the app or touchslider values.

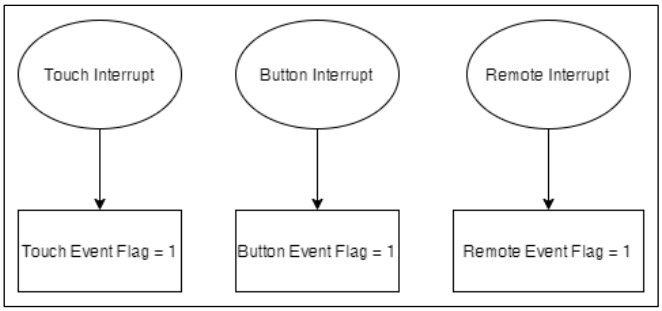

Figure 3: Custom made interrupts

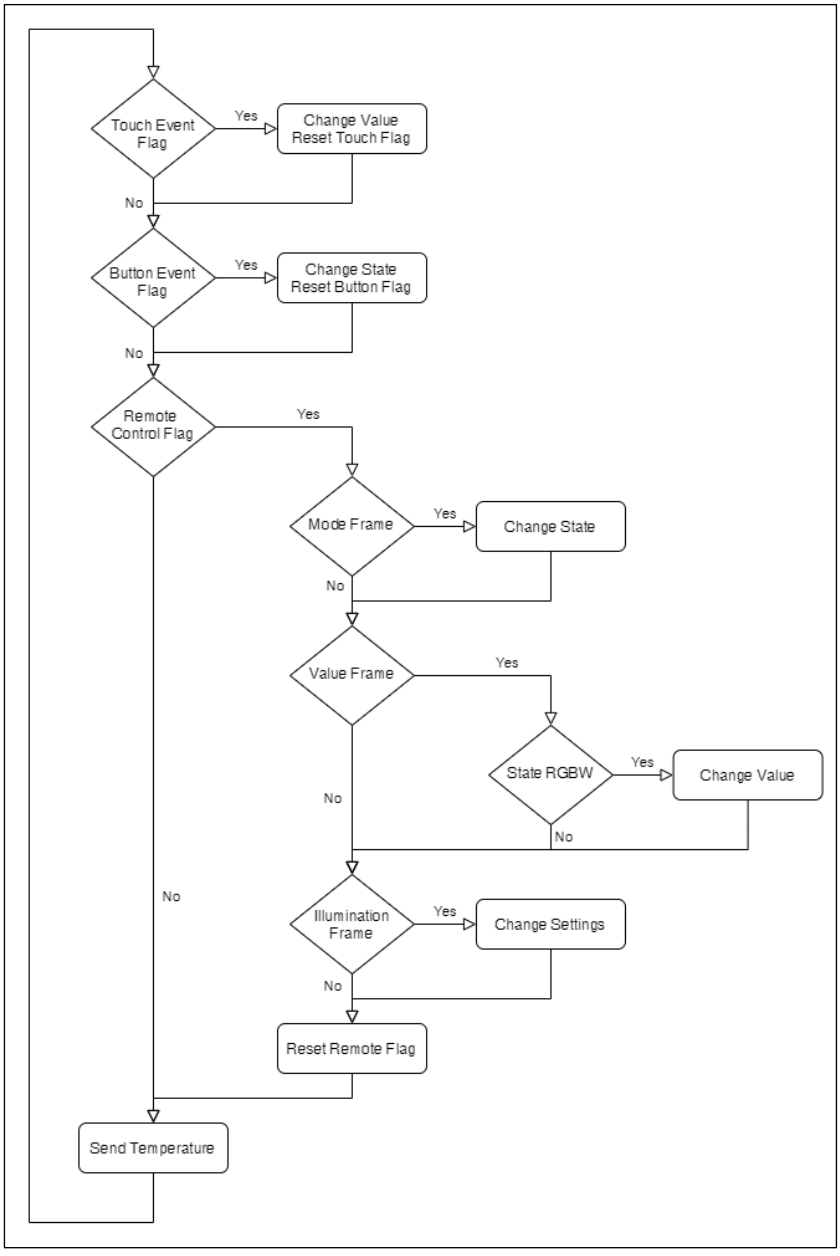

Figure 4: Gecko software flowdiagram

UART-Communication protocol

The length of the UART data frame is 15 bytes.

When switching the mode via smartphone app, the data frame starts with the character *M* followed by the mode number.

$$M0000000000000X$$

When changing the RGBW values in *RGBW* mode, the frame is set up as followed whereas each frame section represents a colour value.

$$RRR\backslash nGGG\backslash nBBB\backslash nWWW$$

In the software the frame gets separated by "\n" to get the values for the different colours. The following frame is necessary to change the settings of the illumination function:

$$L\backslash n00000I\backslash n00000S$$

To send the temperature to the smartphone, the frame gets structured as followed:

$$T\backslash n00000000000ML$$

Frame structures

M0000000000000X	Mode • X = 0: Mode RGBW • X = 1: Mode TEMPERATURE
RRR\nGGG\nBBB\nWWW	Value • RRR: Red value 0 to 255 • GGG: Green value 0 to 255 • BBB: Blue value 0 to 255 • WWW: White value 0 to 255
L\n00000I\n00000S	Illumination • I = 0: if Room illumination ↑ then LED brightness ↑ • I = 1: if Room illumination ↑ then LED brightness ↓ • S = 0: Output colour is not ambient dependent • S = 1: Output colour is ambient dependent
T\n00000000000ML	Temperature • M: Temperature integer places °C • L: Temperature decimal places °C

Table 2: Frame structures

Additional features

Temperature sensor (TEMPERATURE mode)

The idea is to display colours as a function of the temperature. This means that at a temperature of 20°C the output light should be blue. On the contrary at a temperature of 30°C the light should be red. In between these limits should occur a smooth transition. The smartphone app compares the internal battery temperature with the Moodlight temperature. If the difference between these temperatures is too big, the current through the LEDs gets switched off and a message appears on the smartphone.

Ambient light sensor (Illumination function)

The brightness of the Moodlight can be regulated automatically with illumination data. This means the greater the illumination, the higher the brightness of the LEDs and vice versa. Additionally this logic can be inverted. These settings can be configured on the smartphone app. This function is global and does not belong to a state. Therefore it is possible to enable this function, no matter if in *RGBW* or *TEMPERATURE* mode.

Android Software Concept

When starting the smartphone app, all available Moodlight devices get listed as shown in Figure 5. If a device is connected, the first tab is displayed (Figure 6). On the first tab it is possible to adapt the colour with a colour circle. Two radio buttons are available to change the modes and two switches to change the settings for the luminosity function. Figure 7 shows the second tab, where four sliders are displayed to adapt the output colour. Each colour string can be enabled and disabled with a checkbox. On the one hand the last tab is used when sending custom strings via an input window and on the other hand to display the received data like in Figure 8. For security reasons the temperatures of the battery and the Moodlight get observed and displayed on each tab.

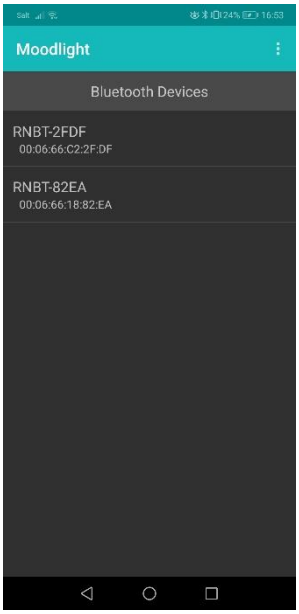

Figure 5: Startscreen

Figure 6: Tab 1 colour circle

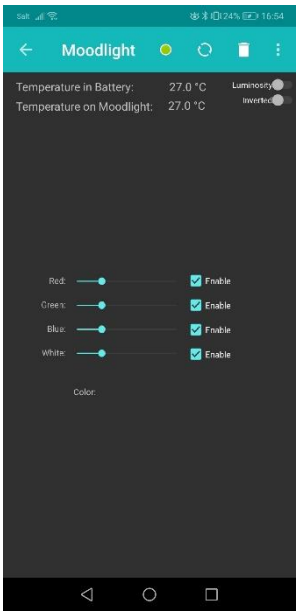

Figure 7: Tab 2 slider

Figure 8: Tab 3 input window