FH Technikum Wien

Documentation Paper

University of Applied Sciences Technikum Wien
Biomedical Engineering and Sciences

Flickering Light Stimulator

By
Hirtl Rene, 1110228003
Mayer Thomas, 1110228014
Schwarz René, 1110228002

Supervisor: Dipl.-Ing. Christoph Veigl

Vienna, 06.03.2012

FACHHOCHSCHULE TECHNIKUM WIEN

Table of Contents

Attention Note

Using the Flickering Light Panels may cause a visual evoked epileptic shock!
DO NOT USE the panels before consulting a doctor!

1 Introduction

According to the AsTeRICS Project Framework a Flickering Light Stimulator is developed to do research in the affect of flickering light to a person's EEG and usage for Brain Computer Interfaces (BCI).

A visual steady-state evoked potential is a resonance phenomenon arising mainly in the visual cortex, when a person is focusing the visual attention on a light source flickering with a frequency above 4Hz. It consists of a periodic component of the same frequency as the flickering light source, as well as of a number of harmonic frequencies. A Visual Evoked Potential Detection Suite has to be developed within WP6 of AsTeRICS project. The Suite attains the online detection of Steady State Visual Evoked Potentials elicited by a light stimulus panel flickering at configurable frequency, duty cycle and intensity. (Starlab, 2011)

Before we started to work we tried to divide the whole project into several work packages. These are the development of the hardware, the microprocessor programming and the development of the graphical user interface in C#.

2 System

The Flickering Light Stimulator / Steady State Visual Evoked Potential (SSVEP) is realized with a LED Board including four LED panels controlled with an Arduino Evaluation Board with an ATmega 328 microcontroller. A PC application is used to vary the flickering frequency, intensity and duty cycle of the panels of the board. The communication with the microcontroller is done via RS232 serial interface. As you can see in the figure below (see Figure 2-1) the system consists of three general parts:

- PC with the GUI programmed in C# for configuration, starting and stopping the device
- Control Unit (CU) is the Arduino board with the ATmega328 microcontroller which communicates with the GUI, evaluates the data and controls the LED panels
- LED panel 1-4 which are used to create the SSVEP

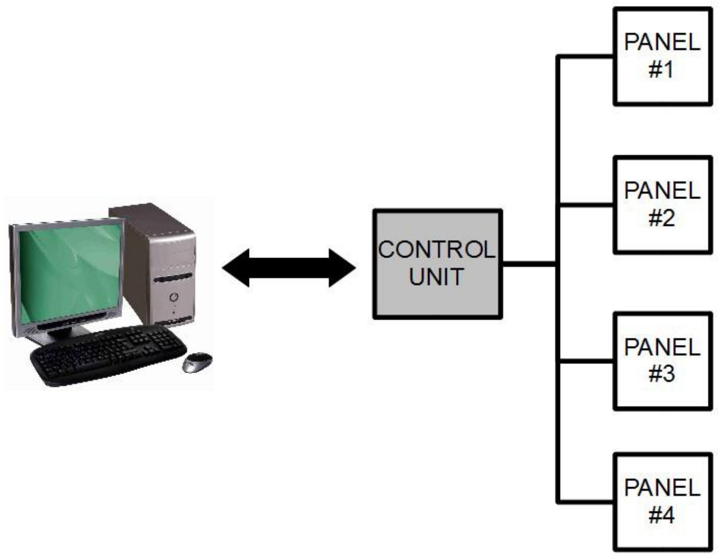


Figure 2-1 SSVEP Stimulation System Proposal (Starlab, 2011)

3 Software

This chapter focuses on the software of the control unit (see Figure 2-1), i.e. the Arduino board with the Atmel ATmega328 microcontroller. The software was written using the Atmel AVR Studio 4 (Version 4.18) development tool which is extended with the WinAVR tool that provides the GNU GCC compiler.

The control unit generally has to fulfil following features:

- communicate with the PC via RS232 interface
- evaluate data of the serial communication for the panel configuration
- control frequency, duty cycle and intensity of the LED panels

The following chapters will now describe how this software parts are realized.

3.1 Firmware Update

For updating the firmware of the microcontroller it has to be flashed with the new hex-file which includes the software updates. The Arduino UNO comes pre-loaded with a bootloader which is compatible to the Arudino software development IDE. Fortunately, the software tool “avrdude” understands various download protocols for firmware programming – among others also the protocol used by the Arduino IDE. The following manual describes how to download the updated hex-file to the microcontroller using the command shell of windows and the avrdude bootloader. Therefore the avrdude software has to be installed (download on http://www.mikrocontroller.net/attachment/69851/avrdude-5.10.zip)

First of all you have (see Figure 3-3) to change the path in the command shell to the directory in which the new hex-file is stored. The next step is done using the bootloader with the following command line:

```
avrdude.exe -pm328p -P COM3 -c stk500v1 -b 115200 -U mdba.hex –F
```

It is necessary to press the reset button of the Arduino board while confirming the command in the shell with ‘Enter’ and just after that you have to release the reset button again. It is a bit tricky to press the reset button when the Arduino is in the case of the device (see Figure 3-2). Therefore you may use a screwdriver to reach the button and hold it. The location of the reset button is shown in Figure 3-1.

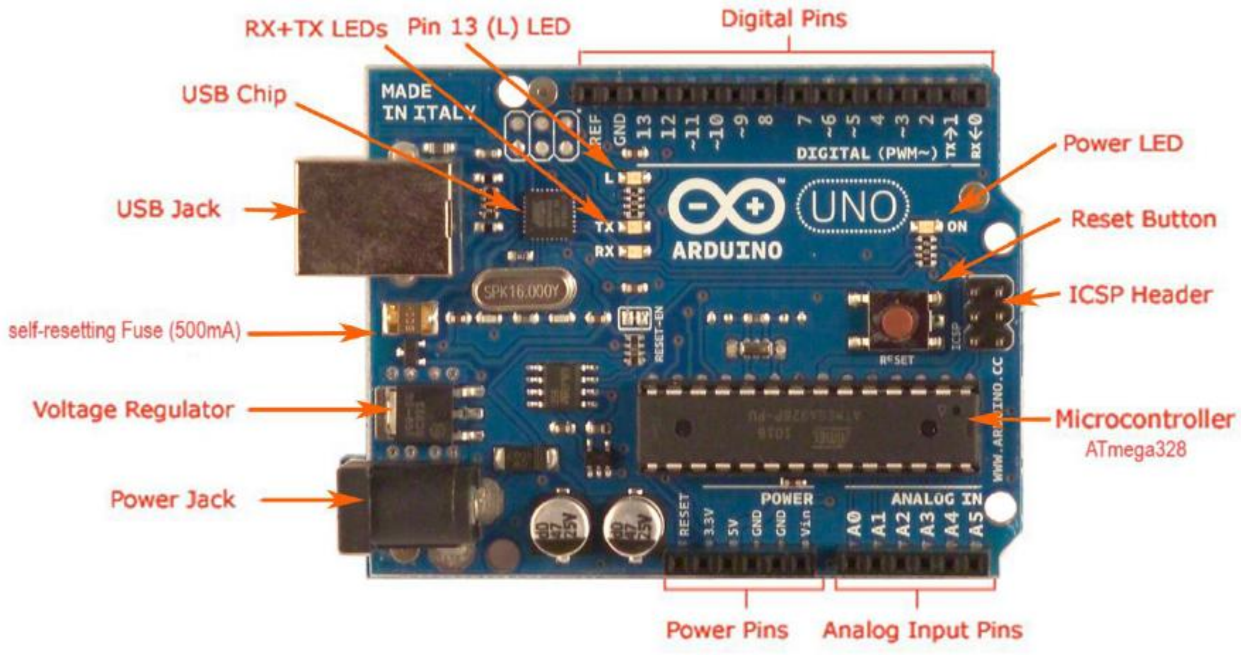


Figure 3-1 Top view of the Arduino UNO board

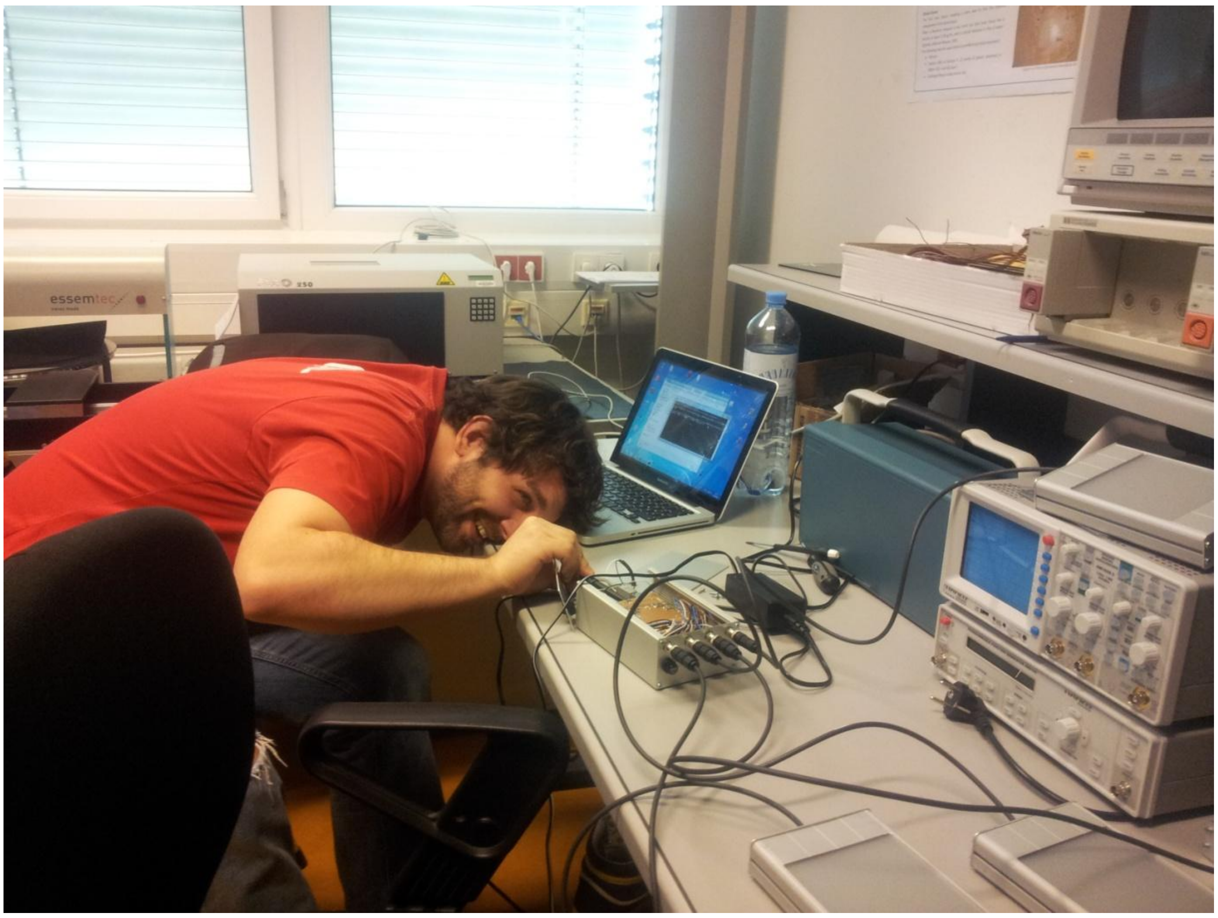
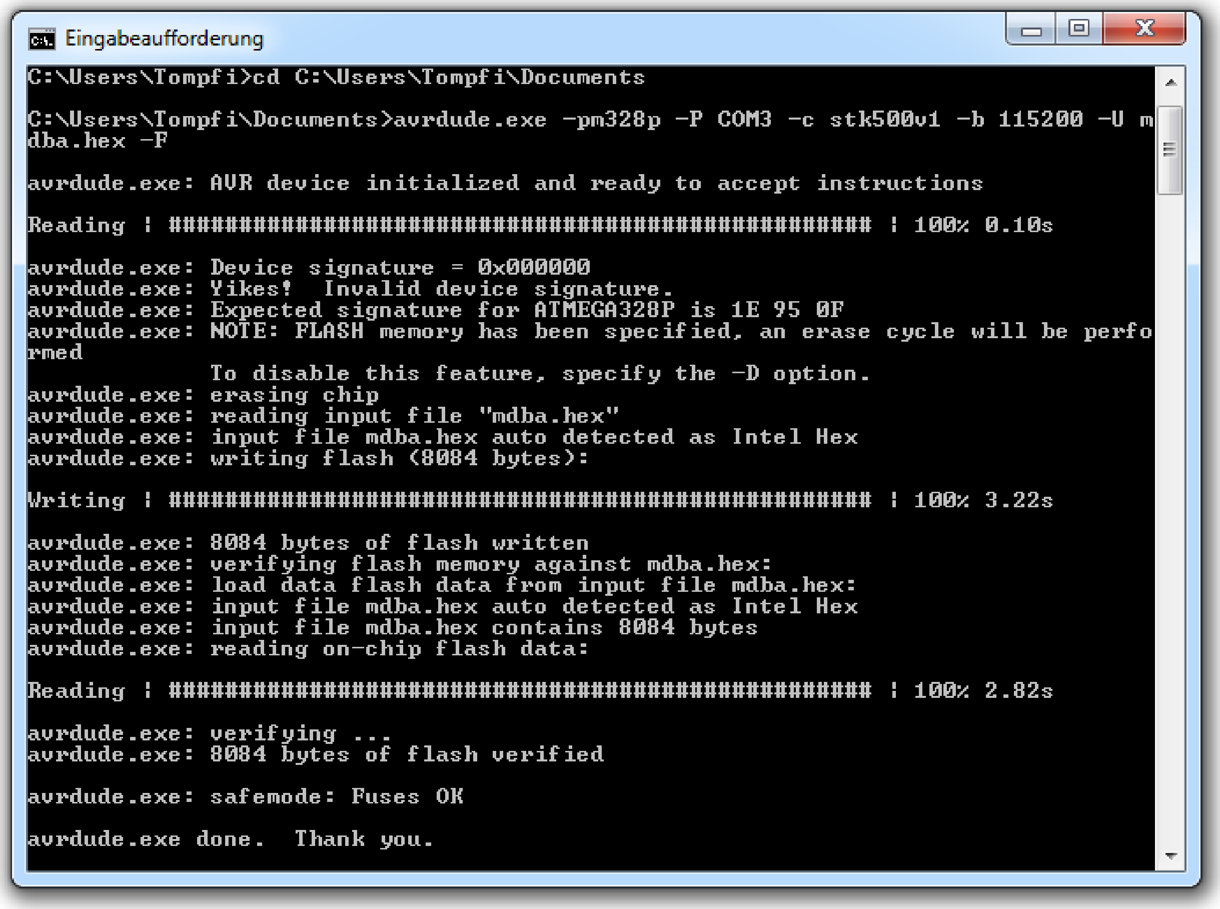


Figure 3-2 Pressing the Reset button of the Arduino within the case for flashing a new firmware

In the example above, the microcontroller part is selected as “m328”, “COM3” is the Virtual Com Port where the Arduino is connected, the protocol is called “stk500v1” and the baudrate is set to 115200. The last parameter is the name of the hex-file which should be downloaded to the microcontroller. Note that avrdude has to be in the system path otherwise you need to specify the path (e.g. \WinAVR-20100110\bin\avrdude.exe). For further information about avrdude please refer to the command line help.

After that the command shell views the downloading progress of the firmware. These steps are also shown in the figure below (Figure 3-3).



```
C:\Users\Tompfi>cd C:\Users\Tompfi\Documents

C:\Users\Tompfi\Documents>avrdude.exe -pm328p -P COM3 -c stk500v1 -b 115200 -U m
dba.hex -F

avrdude.exe: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.10s

avrdude.exe: Device signature = 0x000000
avrdude.exe: Yikes!  Invalid device signature.
avrdude.exe: Expected signature for ATMEGA328P is 1E 95 0F
avrdude.exe: NOTE: FLASH memory has been specified, an erase cycle will be perfo
rmed
             To disable this feature, specify the -D option.
avrdude.exe: erasing chip
avrdude.exe: reading input file "mdba.hex"
avrdude.exe: input file mdba.hex auto detected as Intel Hex
avrdude.exe: writing flash (8084 bytes):

Writing | ################################################## | 100% 3.22s

avrdude.exe: 8084 bytes of flash written
avrdude.exe: verifying flash memory against mdba.hex:
avrdude.exe: load data flash data from input file mdba.hex:
avrdude.exe: input file mdba.hex auto detected as Intel Hex
avrdude.exe: input file mdba.hex contains 8084 bytes
avrdude.exe: reading on-chip flash data:

Reading | ################################################## | 100% 2.82s

avrdude.exe: verifying ...
avrdude.exe: 8084 bytes of flash verified

avrdude.exe: safemode: Fuses OK

avrdude.exe done.  Thank you.
```

Figure 3-3 Updating steps using the command shell

If the download progress has finished successfully the device works with the new firmware. Otherwise there might be problems with the COM port the file name or with resetting the Arduino board during initialization of the download process.

3.2 LED Panel Signalling

The specification of the flickering light stimulator comprised the ability to modify some characteristics of the flickering light. According to that, the following parameters have an importance for generating the four LED panel signals:

• frequency	0.5 to 50 Hz, 0.5 Hz Resolution
• duty cycle	1 to 100%
• intensity	1 to 100%

3.2.1 Frequency and Duty Cycle

To create the frequency and duty cycle of the four LED panel signals the switching on and off timing of the LED panels has to be handled. Therefore the 8-bit Timer 0 of the ATmega328 microcontroller is used in the compare mode. Configuration of the prescaler and relevant registers was set to get a timer resolution of 4 µs. According to a compare value of 25, this leads to a compare interrupt for Timer 0 which is called every 100 µs. The following calculations show that this resolution is sufficient to meet the specifications.

t_{min} = (1 / $frequency_{max}$) * $dutycycle_{min}$ = (1 / 50 Hz) * 0,01 = 200 µs > 100 µs

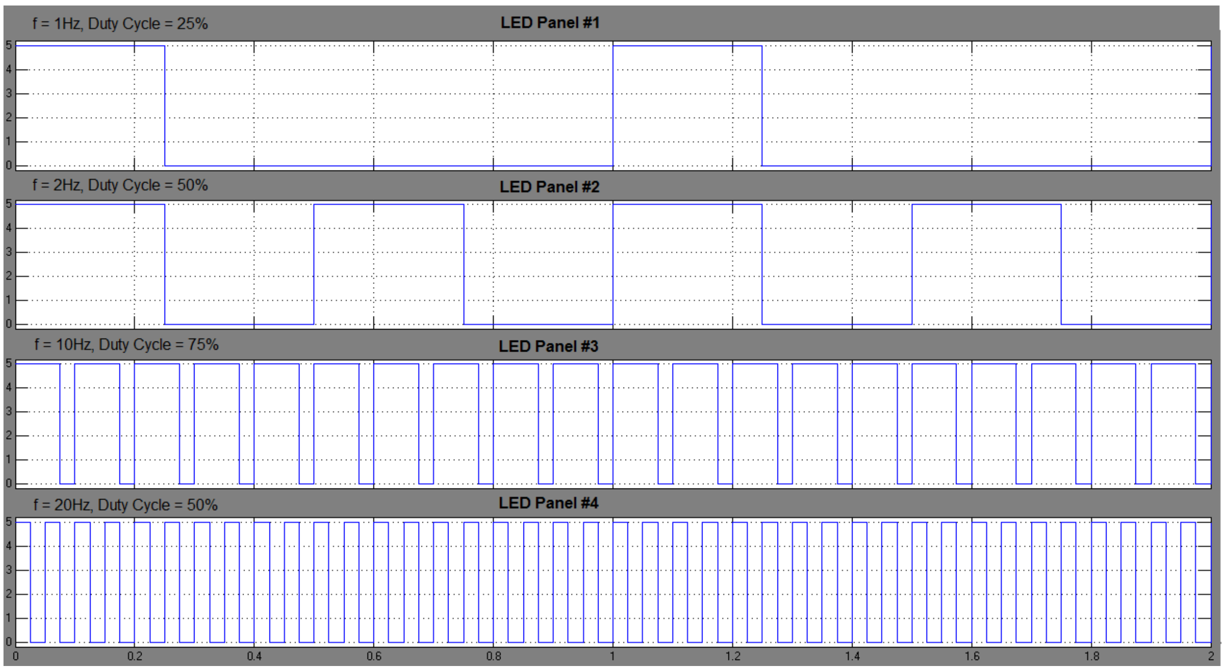


Figure 3-4 Example for control signals for the LED panels

The interrupt service routine (ISR, see Figure 3-5) contains a counter for each LED panel which is incremented on every interrupt. These counters are compared with the individual value for the duty cycle and the frequency for each led panel. Initially the outputs for the LED panels are set to low, which correlates to an on LED panel due to an inversion in the hardware. If a counter now reaches the compare value (variable, not compare register of Timer 0) of a panel's duty cycle, the corresponding output is set to high which means that the panel is switched off. Looking for an example on the calculation of the minimum appearing time above of 200 µs this would lead to a compare value of 2 for the duty cycle. After the counter reaches the frequency compare value of this panel, it is switched on again. The same process is done for each of the four LED panels. As you can see a variation in the frequency and duty cycle can be easily done by adapting the relevant compare values. To generate the signals (see example Figure 3-4) this procedure repeats continuously when the stimulation has started.

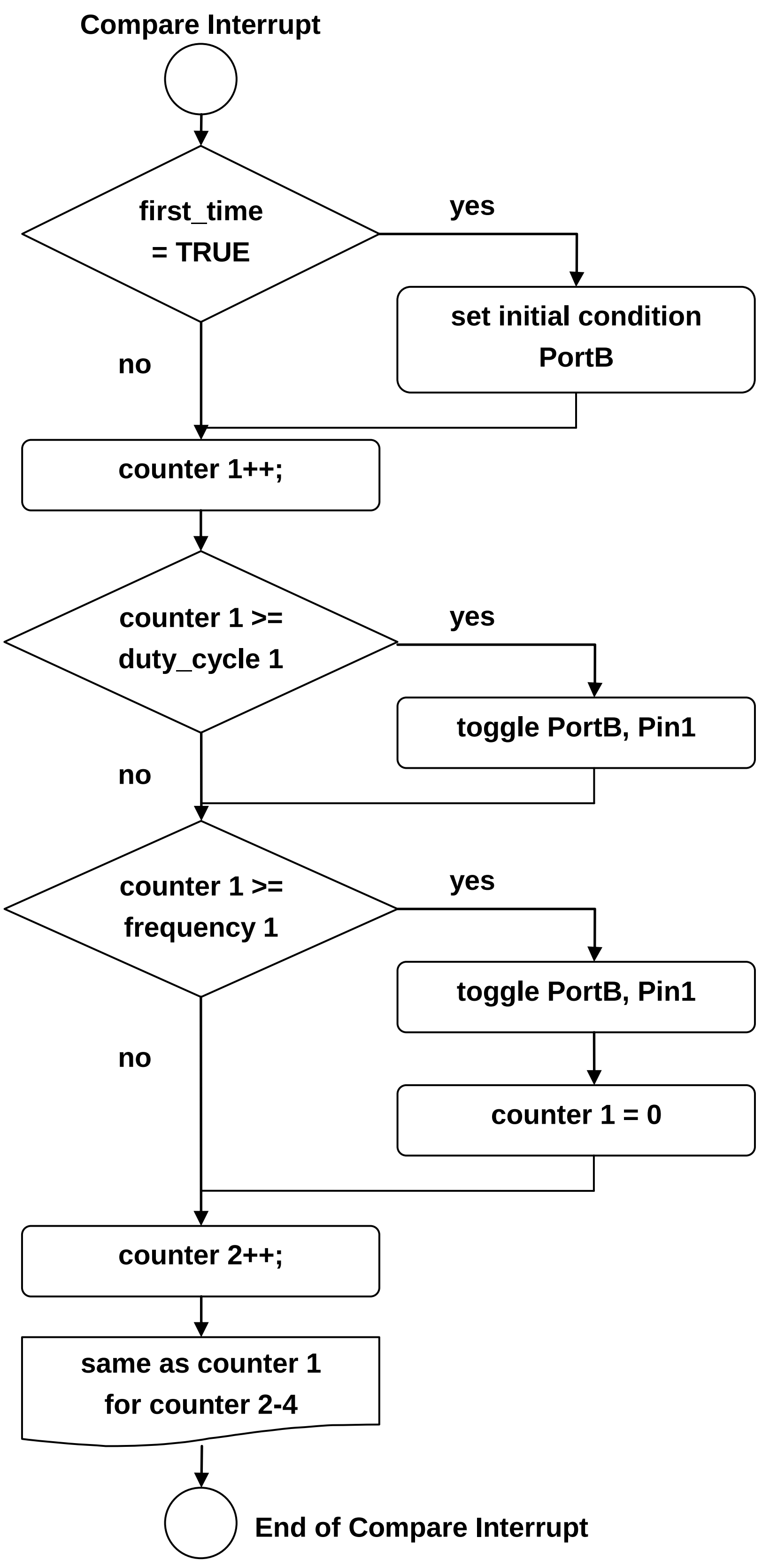
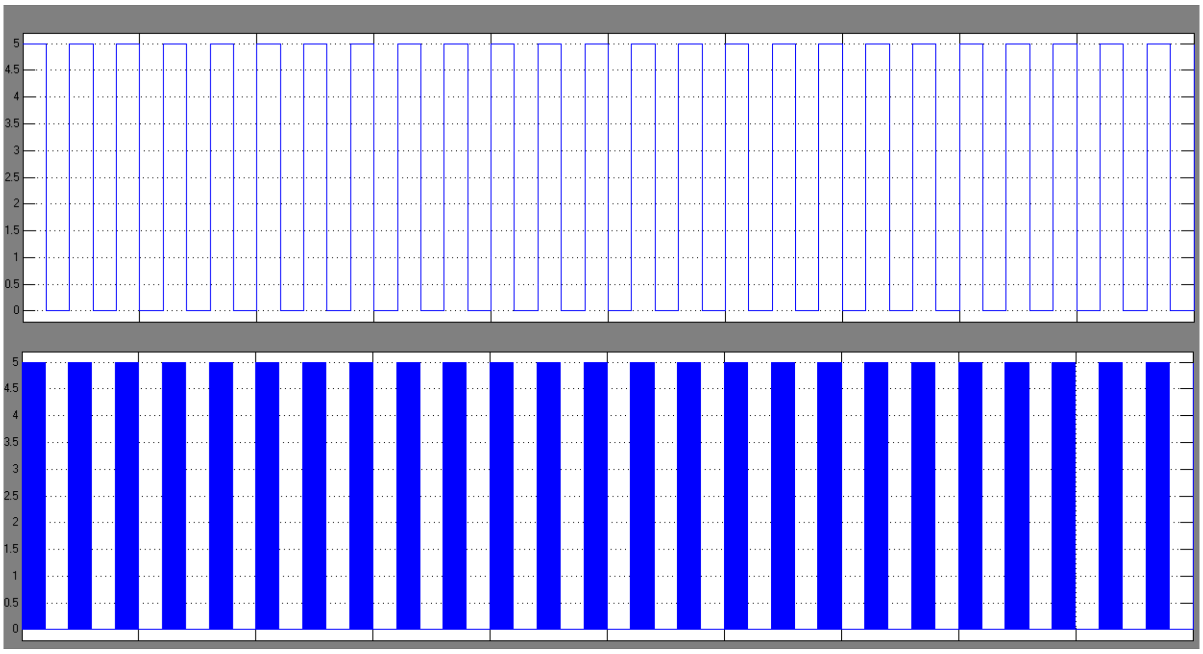


Figure 3-5 Flowchart for the LED panel signaling

3.2.2 Intensity

To enable the intensity control, the output of the LED panels is switched on with a high (for the eye non visible → 500 Hz) frequency during the on phase of the panel (see Figure 3-6). With the variation of the duty cycle of this high frequency signal, the intensity can be controlled. That is to say, the greater the duty cycle (the longer the high phase of the signal) of the intensity frequency, the higher the intensity of the LED panel. This is because the average power of the signal is influenced by this on/off proportion of the high frequency signal during the general high phases of the LED panel.



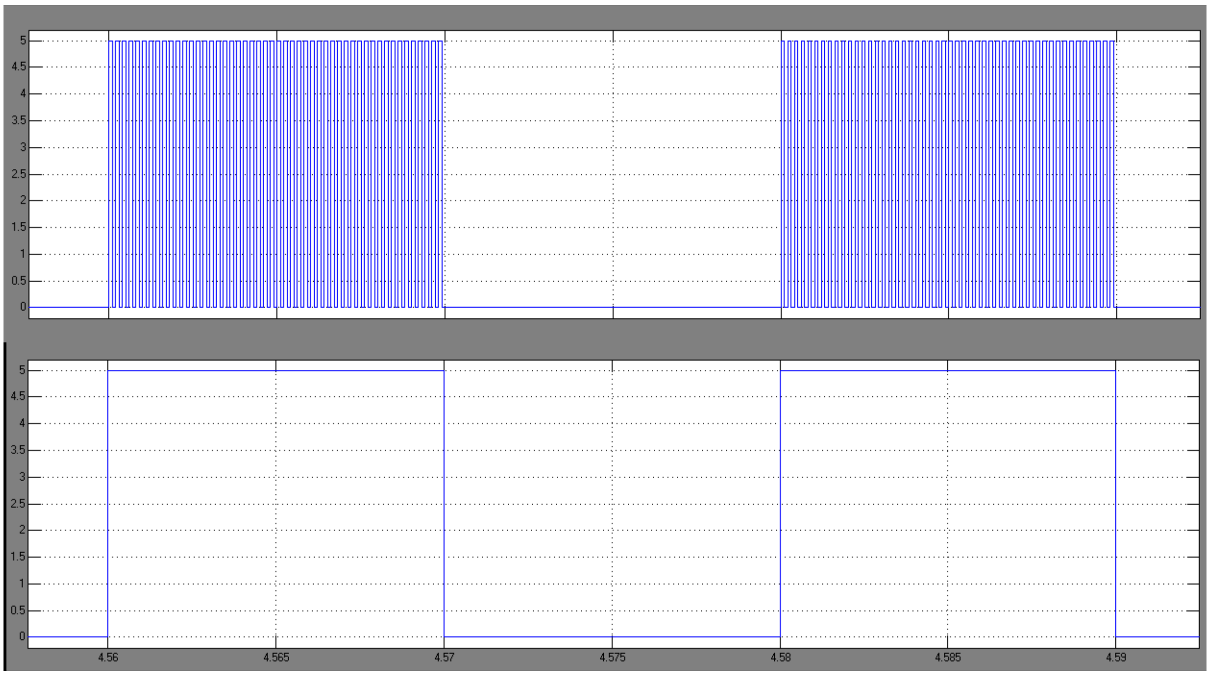


Figure 3-6 Generating Intensity by flickering with 500 Hz when LED panel is switched on

The functionality of the intensity is realized similar to the frequency and duty cycle (see Figure 3-6) by the use of Timer 2 in compare mode. With a timer resolution of 4 µs and a compare register value of 5 the Timer 2 compare interrupt service routine is processed every 20 µs. In the ISR there is again a counter for each LED panel which is incremented only if the panel was switched on by the other Timer (Timer 0).

To realize the intensity a frequency of 500 Hz is used to switch during the on phase of the LED panels. A frequency of 500 Hz results in a time of 2 ms, i.e. when the counter of a panel reaches the value 100 the output is switched to high again. The duty cycle of this frequency is now the proportion of the intensity, i.e. the greater the duty cycle, the longer the high phase and the higher the light intensity. In comparison to the fixed frequency compare value of 100 the compare value of the duty cycle differs with the intended light intensity.

For example a light intensity of 70% for a panel will lead to a compare value of 70. If the counter reaches the value the LED panel is switched off until the compare value of 100 (for the 500 Hz, 2 ms), when it is switched on again. This process happens with a too high frequency for the human eye and repeats continuously as long as the individual LED panel is switched on/high (output at low, because hardware inverted).

3.2.3 Colors

Although it was not required in the project specification an additional parameter was added to configure the color of the LED panels. This does not enable individual coloring for each panel but every panel has the same color supporting white, red, green, blue, yellow, cyan and magenta.

The coloration of the panels is realized with three additional outputs to choose the color binary. For example if every output pin is set to high, the LED panels are white. On the other hand if the second pin is set to high and the other two pins are low the light of the panel is green.

3.3 Serial Communication

Object of this part is the serial communication of the microcontroller and the GUI of the controlling PC. The serial communication is specified using fixed commands and configuration data which follow a defined protocol. In the following chapters this definition is described in detail.

3.3.1 Commands

The project specification defines following commands:

Command	**Action**
Status_Request	The computer requests the current configuration of each Stimulation Panel
Set_Panel_Configuration	The computer sets the configuration parameters of each Stimulation Panel including:stimulation frequency, duty cycle, and light intensity of the stimulus.
Start_Stimulation	The computer requests to the CU to start of the stimulation
Stop_Stimulation	The computer requests to the CU to stop of the stimulation
Status	The Control Unit reports the current configuration of each Stimulation Panel including: stimulation frequency, duty cycle and light intensity of the stimulus
ACK_Start	The Control Unit reports that the Stimulation has begun
ACK_Stop	The Control Unit reports that the Stimulation has stopped

Table 3-1 Specification for communication commands (Starlab, 2011)

PC-Command	**CU Response**
Status_Request	Status
Set_Panel_Configuration	Status
Start_Stimulation	ACK_Start
Stop_Stimulation	ACK_Stop

Table 3-2 Communication interaction of PC and CU (Starlab, 2011)

All commands of the tables above were adopted literally into the communication protocol. The ‘Status’ report contains the actual configuration for the different panels of the microcontroller, which is described in one of the following chapters. To realize the serial communication the UART of the microcontroller was configured with the following parameters:

- 9600 Baud
- 8 databits
- 1 stopbit
- no parity

If a connection between PC and microcontroller is build up, the received ISR of the microcontroller is processed, every time a character is received in the receive buffer. In this ISR the received character is compared with a null ('\0') which indicates the end of the command and additionally the received character is saved in an array. If a null termination is detected the characters of the array are compared with the defined commands and if a command fits, accordant program steps are done and the command response is transmitted. Regardless if a wrong command is received (typing error) or the command is correct, the buffer and array is cleared after the received null termination.

After a correct command is recognized the corresponding response is copied to a transmit array and the transmit interrupt enable is set. This processes the transmit ISR where the characters are copied to the transmit buffer until there is no character left. At the end of the transmitted ISR the interrupt enable is reset again. The transmission of the actual panel configuration after the status request and panel configuration command is realized with a routine which transmits the individual configuration parameters of the panels. The 'Set_Panel_Configuration' command will introduce the protocol for the panel configuration which is part of the next chapter. This is realized with a flag which is set after the command is received.

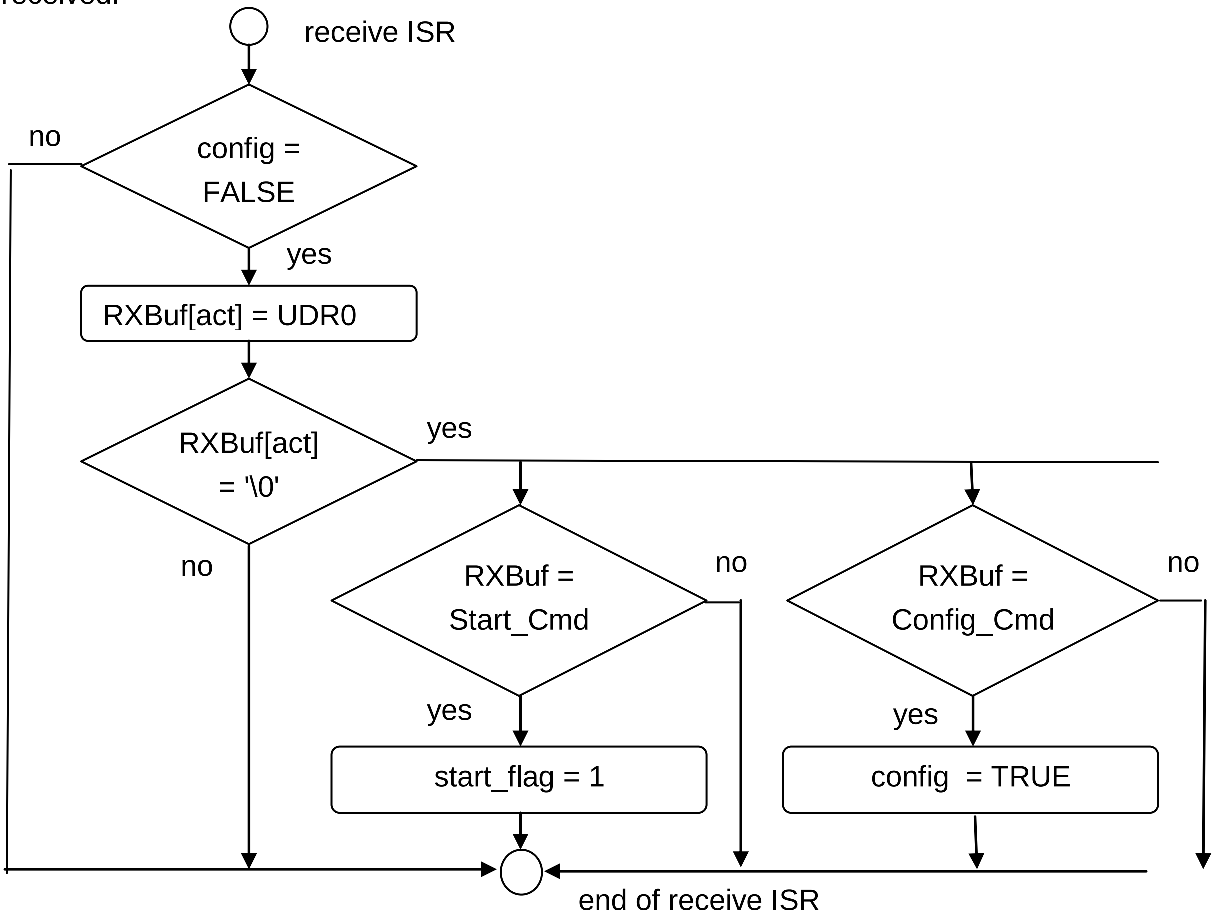


Figure 3-7 Flow Chart of the UART command communication

3.3.2 Configuration Data

After the correctly received ‘Set_Panel_Configuration’ command the following characters are treated as data for the configuration of the LED panels. The processing of this data is regulated with a protocol (see Figure 3-8).

LED Panel #1				LED Panel #2			
Frequency	Duty Cycle	Intensity	Checksum	Frequency	Duty Cycle	Intensity	Checksum
1 - 100	1 - 100	1 - 100	0 - 24	1 - 100	1 - 100	1 - 100	0 - 24
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

LED Panel #3				LED Panel #4			
Frequency	Duty Cycle	Intensity	Checksum	Frequency	Duty Cycle	Intensity	Checksum
1 - 100	1 - 100	1 - 100	0 - 24	1 - 100	1 - 100	1 - 100	0 - 24
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

Color Panel #1-4
0-7
1 Byte

Figure 3-8 Protocol for the LED panel configuration data

The figure above (see Figure 3-8) shows the structure of the protocol used for the configuration data transmission. Similar to the commands detection described in the former chapter, the received data is determined by the detection of a null termination. The received data before the null is converted to a numerical and saved into an array. First of all the frequency of the first LED panel is received. As already mentioned, the frequency can adopt values from 0.5 to 50 Hz. To communicate this value easier it is multiplied by 2 and transmitted, so the value of the first received byte can adopt values from 1 to 100. Duty cycle and intensity of LED panel 1 are determined by the second and third received byte. These bytes have a range from 1 to 100 which is directly the percentage value for these parameters. The last byte for the first panel contains of the checksum for the first three bytes, i.e. for the frequency, duty cycle and intensity of the first panel. The checksum is the number of ‘1’s found in the frequency-, duty cycle- and intensity byte. It is calculated by the PC program out of the transmitted data and also calculated by the microcontroller out of the received data. These two checksums are compared with each other and if they match, it is assumed that there is no failure in the received data. This protocol structure of the first panel, the first four bytes, is repeated for the other panels. The last byte of the protocol includes the color for the panels.

3.4 Data Evaluation

This chapter describes how the control data for the frequency, duty cycle and intensity are evaluated out of the received data.

3.4.1 Frequency

As mentioned in the communication chapter, the received value of the frequency can take values from 1 to 100. To calculate the compare value for the signalling of the frequency following equation is used.

$\text{compare}_{\text{frequency}}$ = 1 / received_frequency * protocol_factor * interrupt_resolution

Because of communication reasons the value received frequency is twice the intended panel frequency. Therefore the received value has to be multiplied with an protocol factor of 0,5. In general compare value represents how often the compare interrupt is called until this value is reached. Due to the fact that the interrupt is called every 100 µs (25 * 4 µs) this time is included into the equation. One divided by this term finally leads to the resulting compare value for the individual counter of the LED panel. In the following there is an example with a received value for the frequency is 20.

$\text{compare}_{\text{frequency}}$ = 1 / 20 * 0,5 * 0,0001 = 1000
T_{panel} = $\text{compare}_{\text{frequency}}$ * interrupt_resolution = 1000 * 100 µs = 0,1s
f_{panel} = 1 / T_{panel} = 10 Hz

3.4.2 Duty Cycle

Same as the frequency, the duty cycle range is from 1 to 100, which represents the percentage value of the on period in comparison to the whole period. That means a higher value for the duty cycle results in a longer time when the LED panel is bright and a correspondingly shorter time when the panel is switched off. The counter compare value for the duty cycle is calculated using the following formula.

$\text{compare}_{\text{duty\_cycle}}$ = received_duty_cycle * percentage_factor * $\text{compare}_{\text{frequency}}$

To get the mathematical value of the absolute percentage of the received duty cycle it is multiplied by a percentage factor of 0,01. The final result is simply the percentage of the compare value from the frequency, i.e. with a duty cycle of 100 the compare value for frequency and duty cycle are equally. This is illustrated below with the same example of the last chapter with a compare value from the frequency of 1000 and a duty cycle of 25.

$\text{compare}_{\text{duty\_cycle}}$ = received_duty_cycle * percentage_factor * $\text{compare}_{\text{frequency}}$ =
= 25 * 0.01 * 100 = 250

3.4.3 Intensity

The value range of the received frequency also can adopt values from 1 to 100, which should represent the percentage of the maximum light intensity of the LED panel. Apart from the frequency and duty cycle the intensity is based on another interrupt timing of 20 µs. It is necessary to include the general frequency of 5 kHz (200 µs) and the intended intensity, which is again mathematically transformed to a percentage value, into the calculation. This leads to following equation for the calculation of the intensity compare value for the counter of the Timer 2 compare interrupt.

$\text{compare}_{\text{intensity}} = \text{received\_intensity} * \text{percentage\_factor} * \text{compare}_{\text{frequency\_5k}}$

For a further description here is an example for a received intensity value of 80.

$\text{compare}_{\text{frequency\_5k}} = 1 / \text{intensity\_ frequency} * \text{interrupt\_resolution} = 1 / 5000 * 0{,}00002 = 10$
$\text{compare}_{\text{intensity}} = 80 * 0{,}01 * 10 = 8$

Here is advice for a possible improvement of the project. The realization of the intensity as it is done in the software up to know is relatively inaccurate and only allows steps of ten percent independent of received character values. This can easily be improved, for example by using float numbers for the counters in the compare interrupt of the timer for the intensity.

4 Hardware

As in Figure 2-1 visible, our system will consist of four LED panels and one central control unit, which is then connected to the PC. The LED panel only includes passive components (LEDs itself and the resistors), whereas the amplifier circuits are placed in the control unit. The reason is, to make a selection of the color with only four conductions possible. If we would have placed the amplifiers inside the panel casings, we would have needed a cable with five wires. Since our connecting plugs have only 4 poles, we decided to solve the problem in the way as described above.

4.1 Electronics

For the development of the LED- panel we started to design a SMD-print, but due to the fact that we were late for getting the print manufactured we decided to use those LED-strips which can be seen in Figure 4-1.

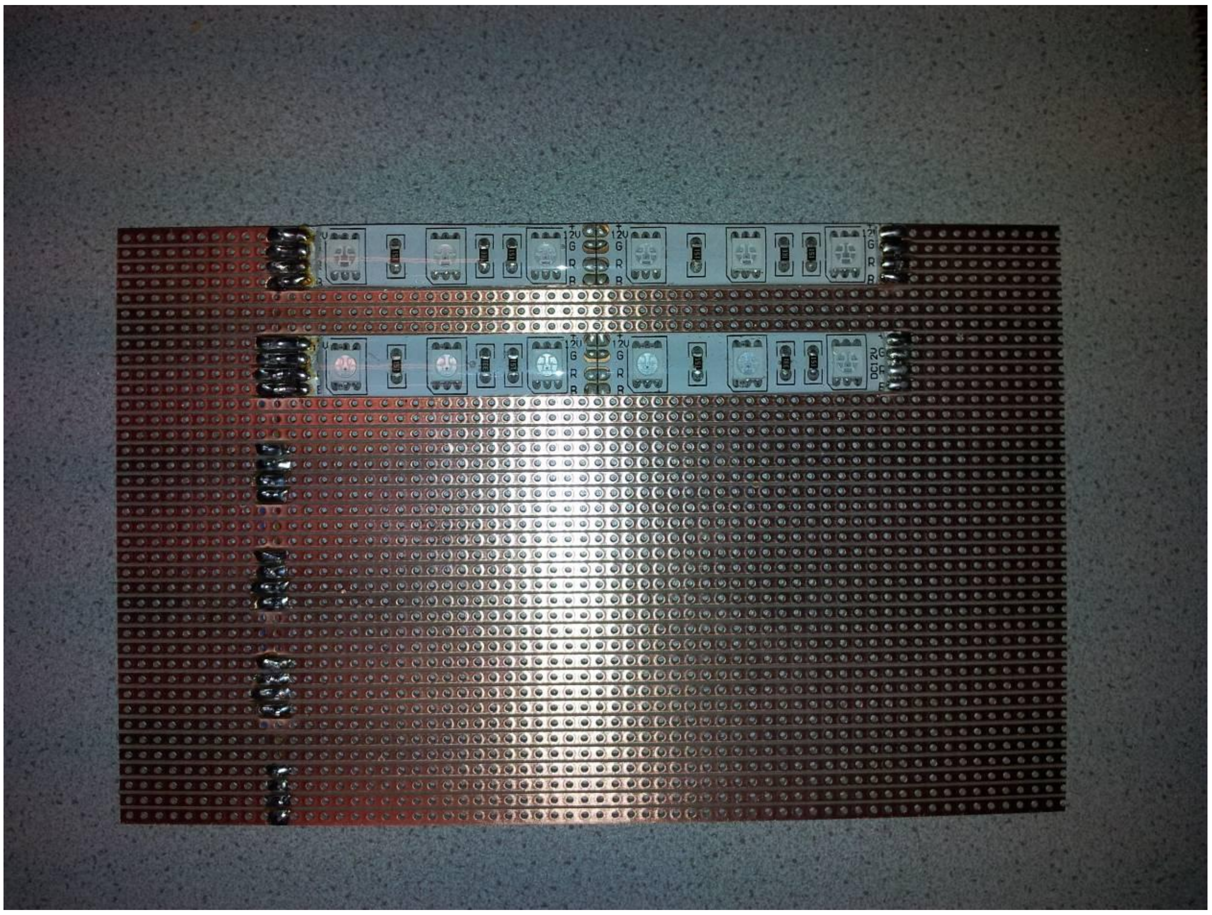


Figure 4-1 Soldered LED-stripes on the perfboard

This worked quite well and so we were able to solder all the perfboards. All the finished boards look like in Figure 4-1 and they are designed failure safety, which means that we soldered the power line twice.

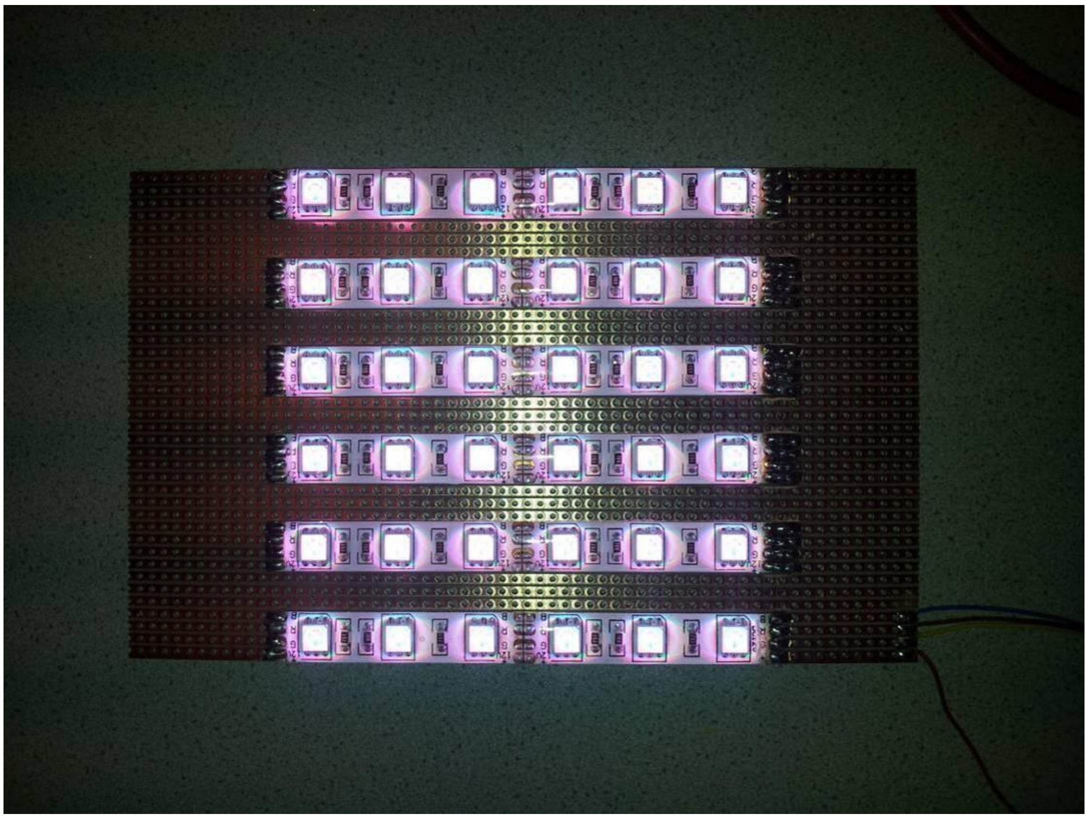


Figure 4-2 Finished perfboard with the LED-stripes

We also measured the power consumption as can be seen in Table 4-1.

Voltage Supply [V]	Color	Current Consumption [mA]	
		All 10 Units	Per Panel (6 from 10 Units)
12	White	930	558
12	Blue	310	186
12	Red	350	210
12	Green	340	204
11,5	White	800	480
11,5	Blue	250	150
11,5	Red	320	192
11,5	Green	280	168
11,95	White		640
11,95	Blue		200
11,95	Red		220
11,95	Green		220

Table 4-1 Power consumption of the LED stripes with different voltages and different colors.

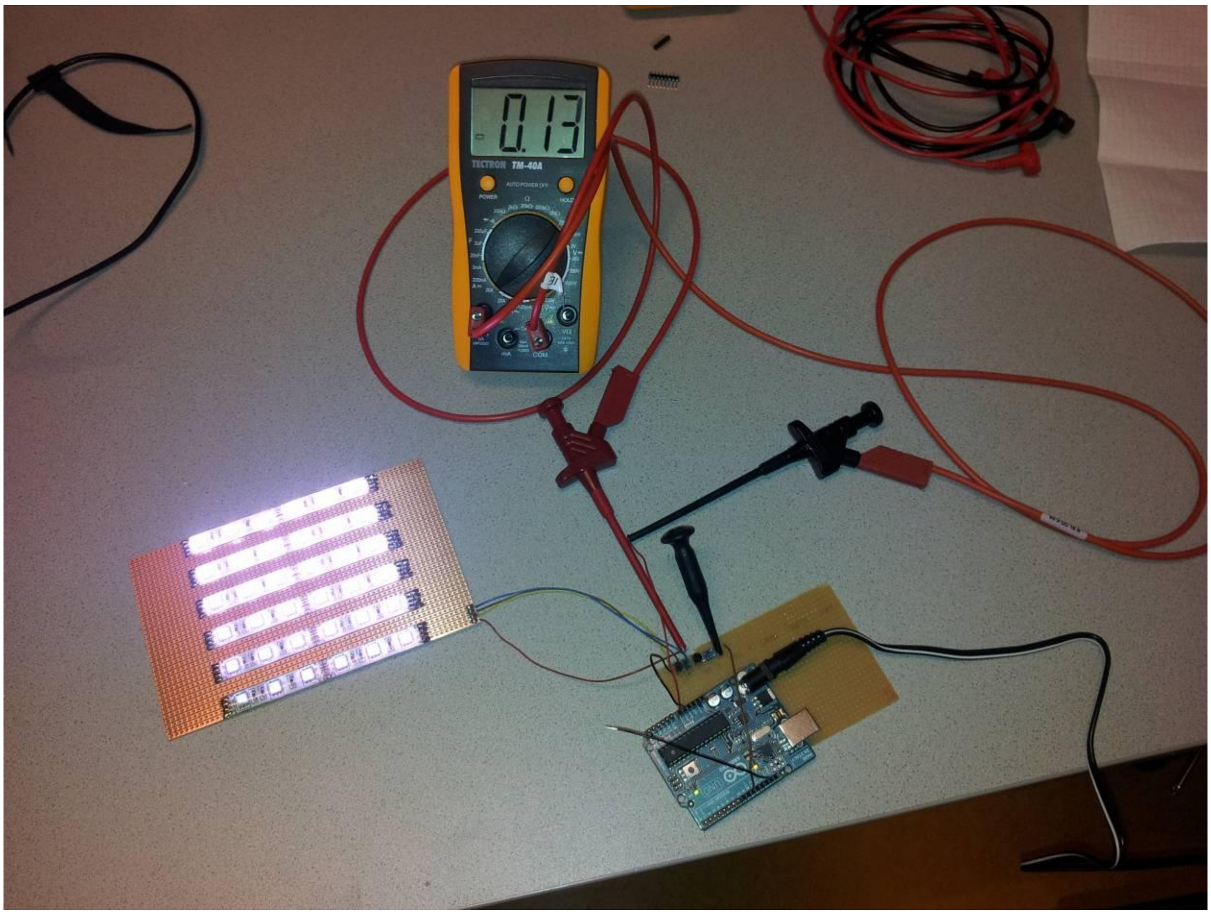


Figure 4-3 Measurement of the Power Consumption of the LED-panel

We also measured the temperature of the LED´s soldered on the panel which can be seen in Figure 4-4 and Table 4-2.

Temperature at U = 11,95V and $U_{Control}$ = 5V			
Frequency [Hz]	T_a [°C]	T_{LED} [°C]	Time [min]
0	22,6	46,1	90
30 (50:50)	-	33,3	30

Table 4-2: Measurement of the temperature with the voltage U from the power source and the Uctrl which equals the PWM signal from the µP-board

The measurement setup of measuring the temperature can be seen in Figure 4-4 as follows.

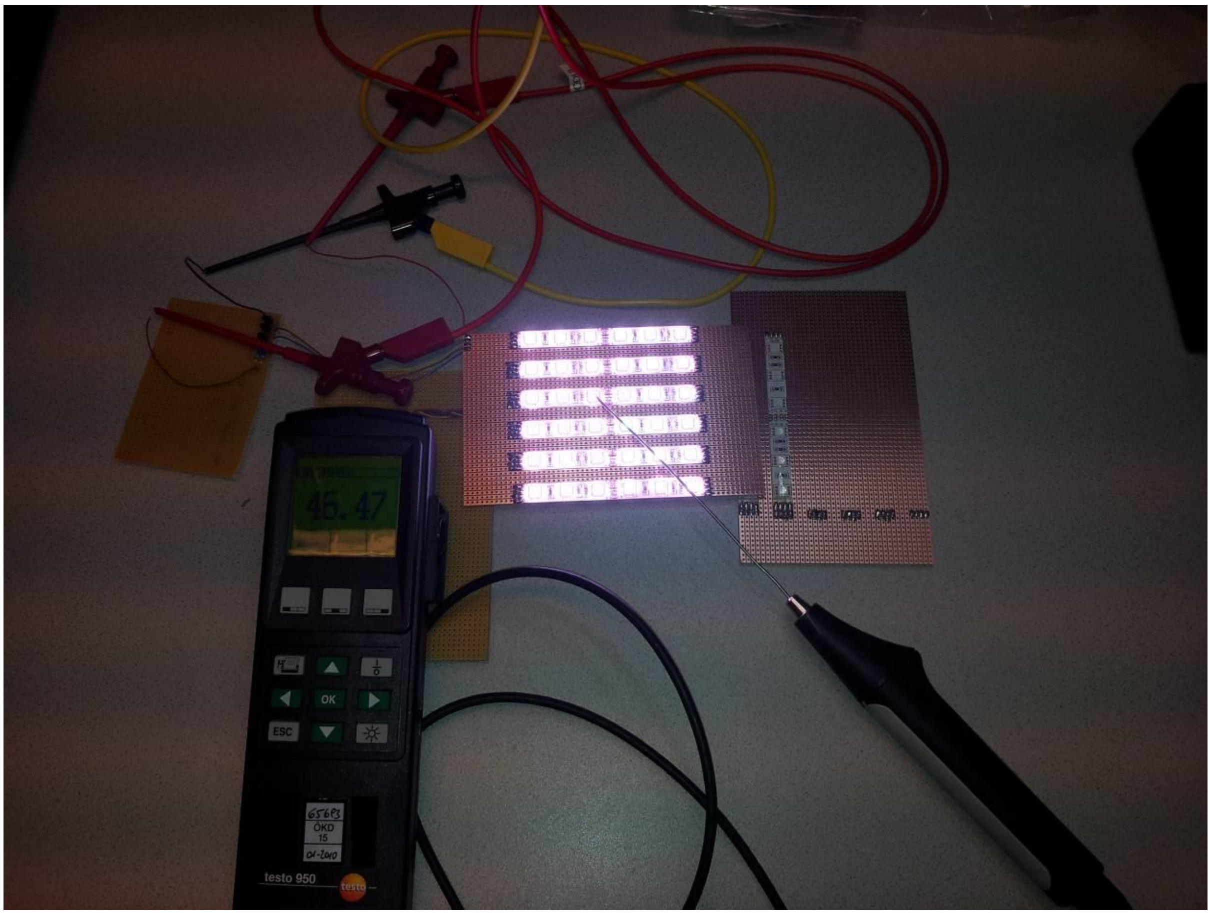


Figure 4-4 Measurement of the temperature with a flickering frequency of 0Hz

After the amplifier print was finished, we did some accurate measurements in order to ensure that the current amplification is enough (see Figure 4-5). You always have to be aware, that each color of each panel needs approximately 200mA. And once we decided to include also the possibility of changing the LED color by the GUI we had to add some logic hardware components, which normally can drive a current of maximum 10 to 15mA on each output. We just used a very common bipolar transistor and therefore a quite high base current is required for driving the transistor into saturation in order to keep the collector emitter voltage very low (the datasheet says the collector emitter voltage drops to 0,3V in saturation at a base emitter voltage of 1,1V, which we might not achieve with a current limited to 15mA).

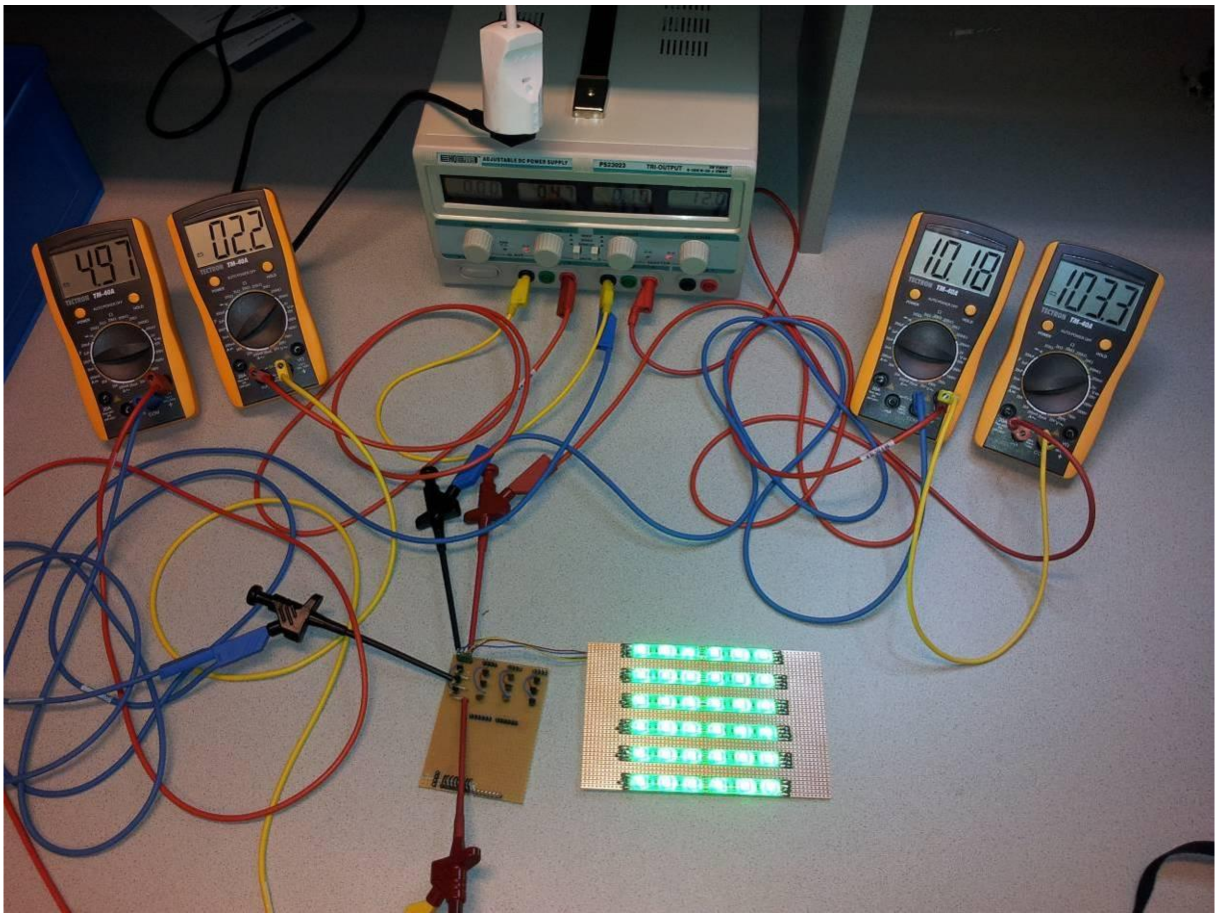


Figure 4-5 Measurement setup for identifying the current amplifiction of the amplifiers

With our measurement setup, to be seen in Figure 4-5, we found a maximum collector current of 182mA for the red LED’s.

	$U_{Control}$ [V]	U_{Power} [V]	U_{BE} [V]	I_B [mA]	U_{LED} [V]	I_C [mA]
Green	~5	~12	0,961	11,4	11,31	161
Red	~5	~12	0,977	11,4	11,19	182
Blue	~5	~12	0,951	11,4	11,39	148

Table 4-3 Results from measurement setup of Figure 4-5

$U_{Control}$ and U_{Power} wasn’t measured directly anymore. The important voltages are U_{BE} and U_{LED} (which can be calculated via U_{Power} and U_{CE} easily $\rightarrow$ $U_{LED} = U_{Power} - U_{CE}$) and is therefore an indicator of the level of saturation of the transistor. In our cases the collector emitter saturation voltage never dropped below 0,61V which is obvious, because we only have a maximum base emitter voltage of approximately 0,98V. To increase this voltage a higher base current would have been required, but we decided to keep it at 11,4mA. The logic gatter ICs we are using can drive a maximum of 50mA on all four outputs together, so 12,5mA is the maximum for each channel and color. Because the signal is pulsed, a higher current would have been also possible, but we decided to keep it at that level, to have some area of safety. Especially, because of the circumstance, that we don't have any information about maximum temperature inside the control unit after some hours of use.

4.2 Mechanics

Afterwards we had a look at the housing. We decided to buy alloy housings were we can mil out a window for the acrylic glass. In the beginning, to get a homogenous flickering light we wanted to sandblast two pieces of acrylic glass that can be glued together. But test measurements delivered the best results when we used a foil as a diffuser material. How we milled the window out of the housing can be seen in Figure 4-6 and Figure 4-7.

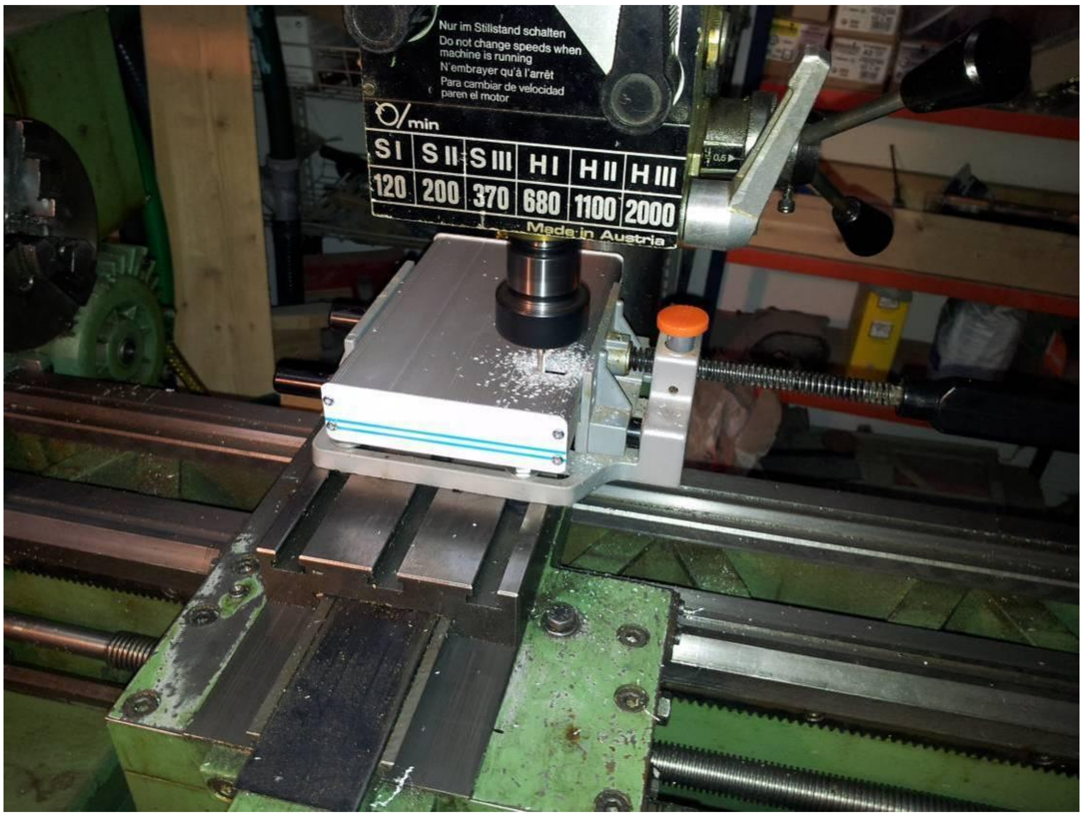

Nur im Stillstand schalten
Do not change speeds when machine is running
N'embrayer qu'à l'arrêt
Para cambiar de velocidad paren el motor
min
SI SII SIII HI HII HIII
120 200 370 680 1100 2000
Made in Austria

Figure 4-6: Milling out the window for the acrylic glass

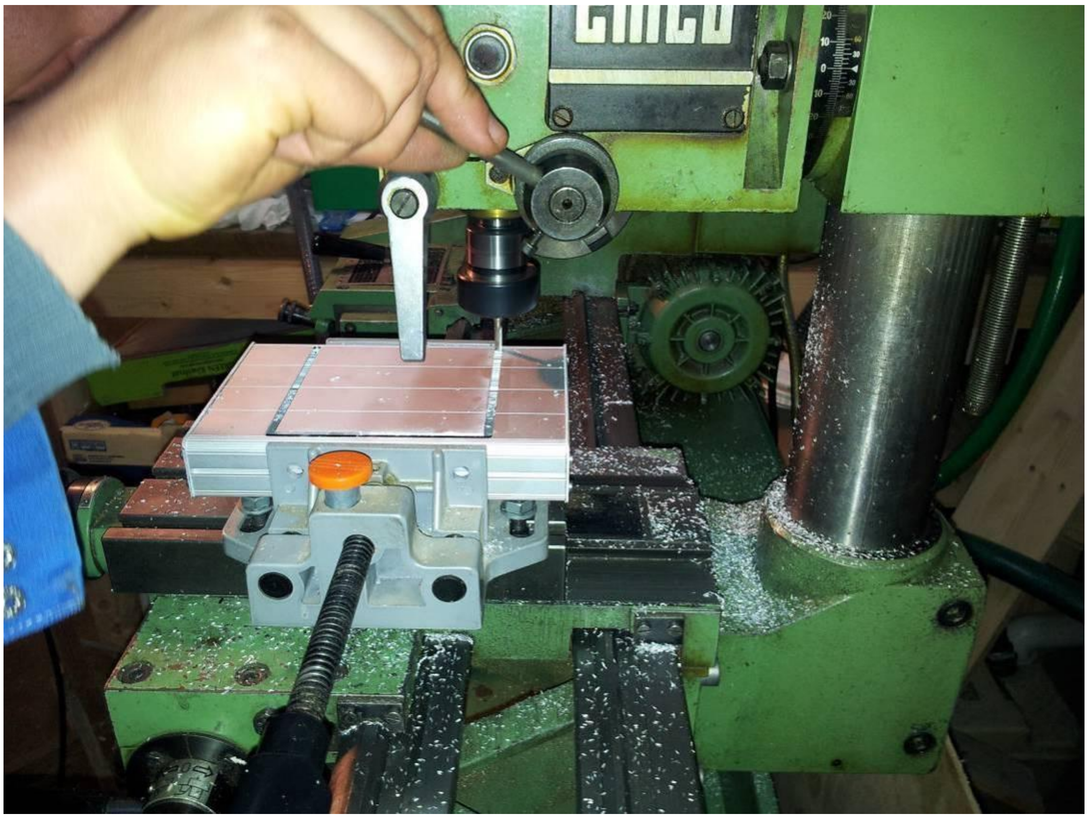


Figure 4-7: Milling out the window for the acrylic glass

4.3 Final setup of the hardware

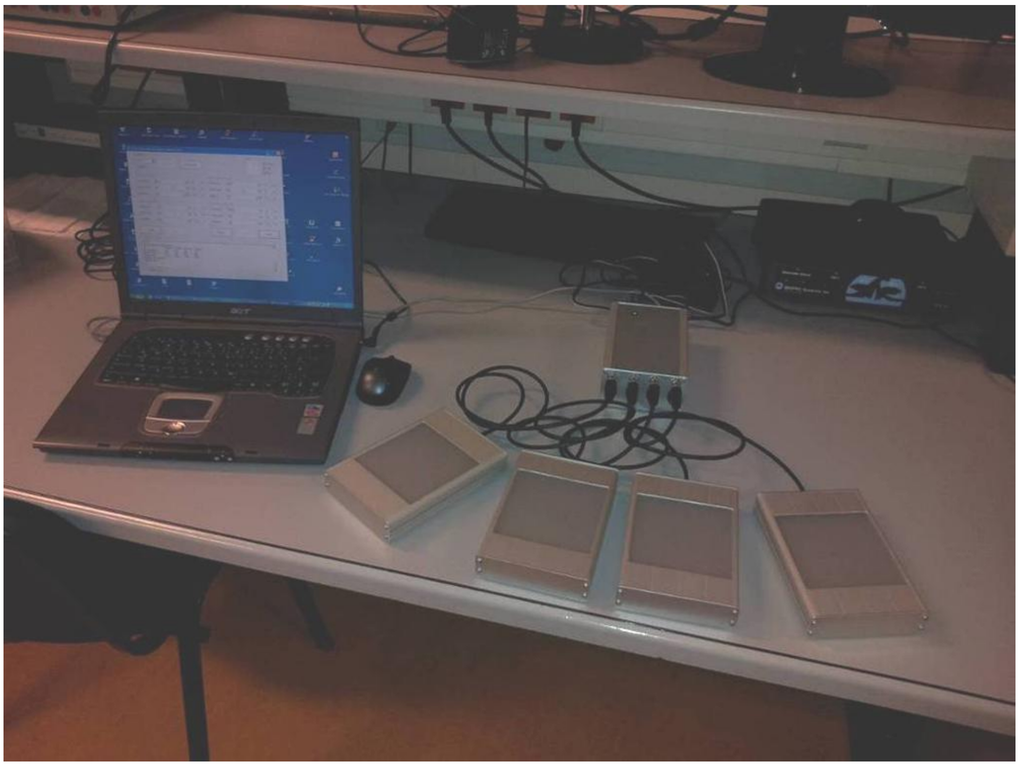


Figure 4-8 Final setup of the hardware – the control unit, 4 LED Panels, a power connection, a USB connection to the PC and the GUI running on the PC

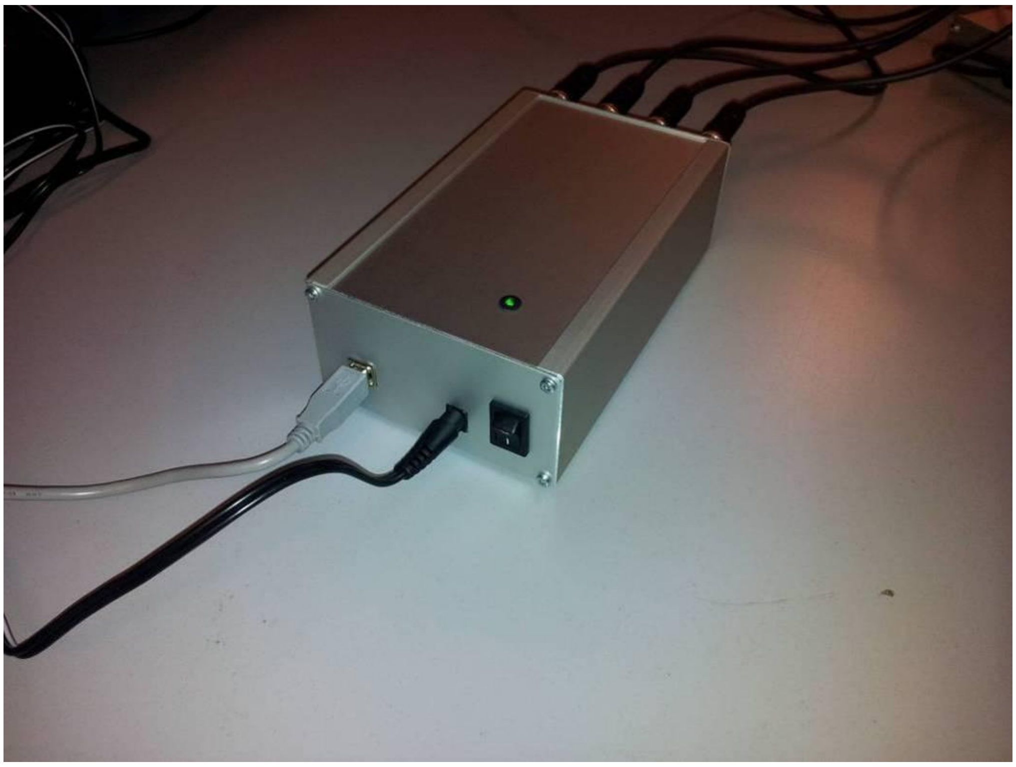


Figure 4-9 Control Unit of the flickering light panel

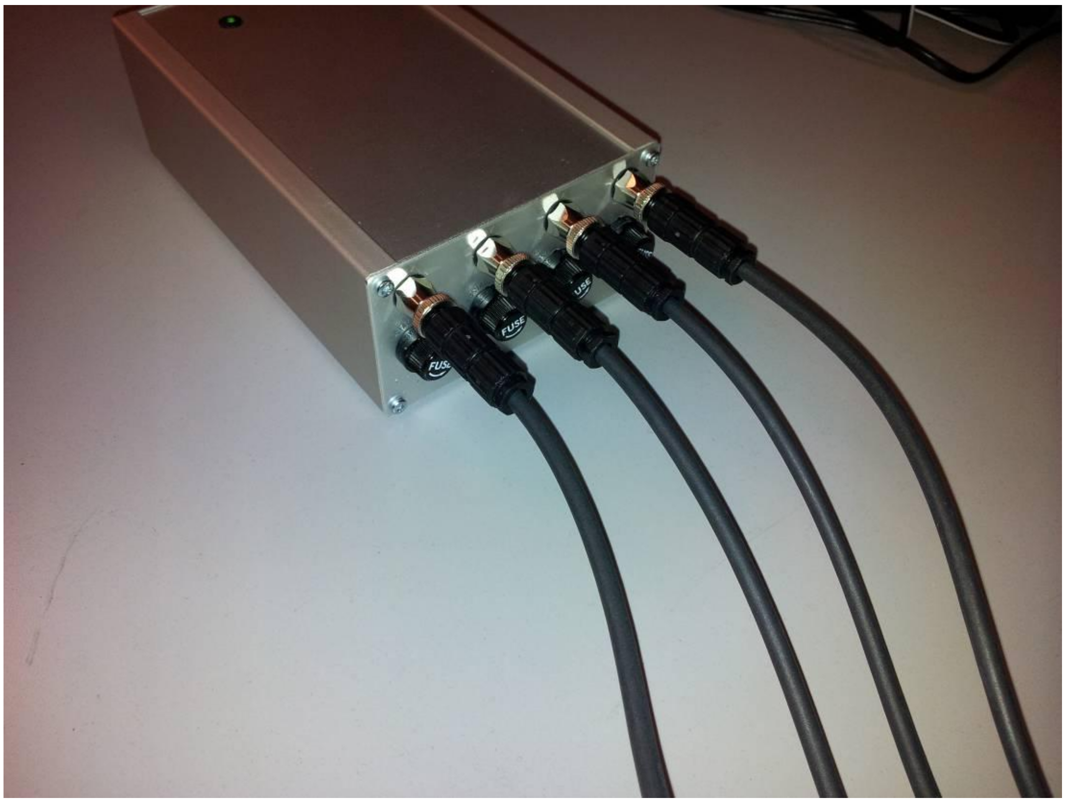


Figure 4-10 Connection side of the control unit

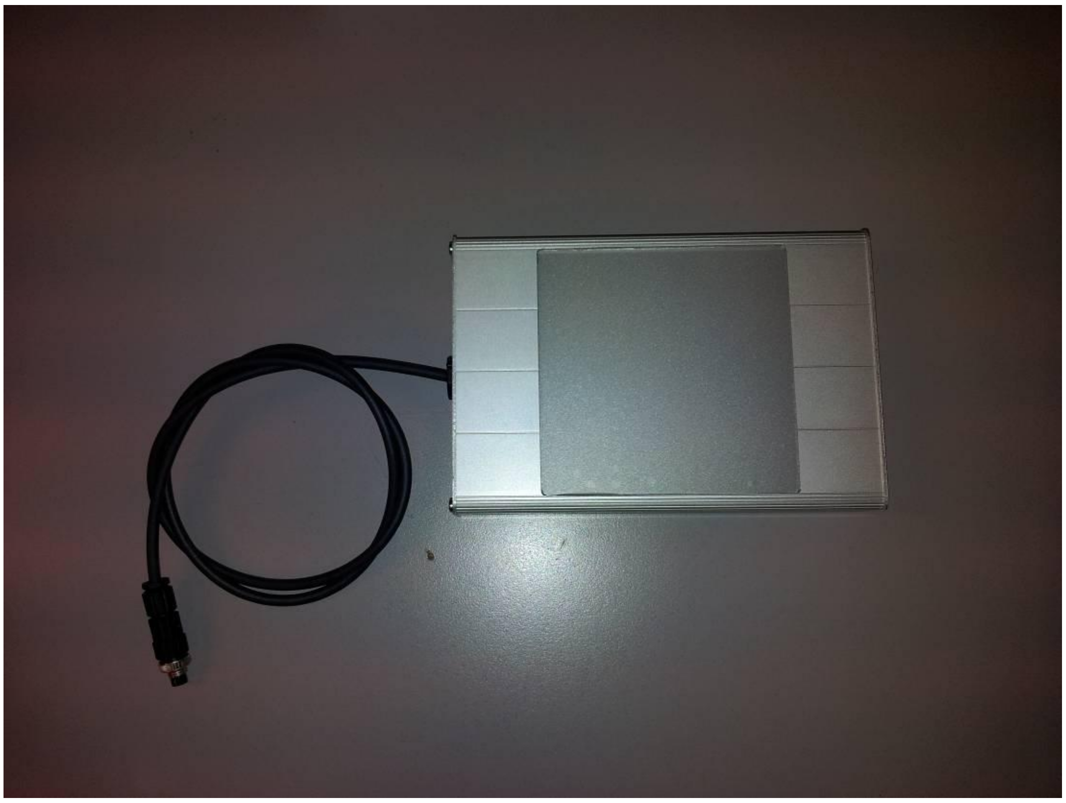
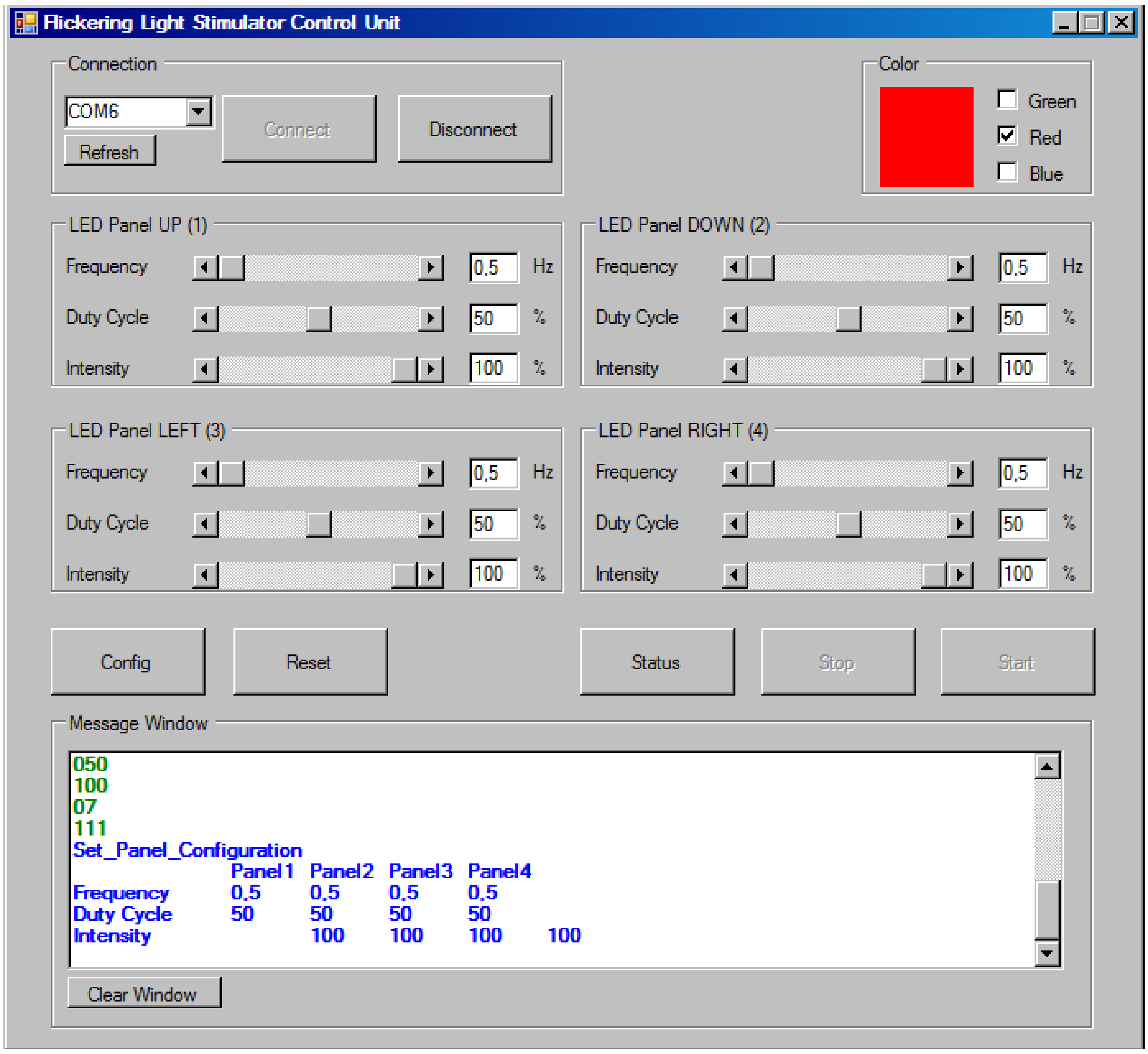


Figure 4-11 LED Panel with connection cable

5 GUI



COM6

Color: ☐ Green ☑ Red ☐ Blue

LED Panel UP (1)		
Frequency	0,5	Hz
Duty Cycle	50	%
Intensity	100	%

LED Panel DOWN (2)		
Frequency	0,5	Hz
Duty Cycle	50	%
Intensity	100	%

LED Panel LEFT (3)		
Frequency	0,5	Hz
Duty Cycle	50	%
Intensity	100	%

LED Panel RIGHT (4)		
Frequency	0,5	Hz
Duty Cycle	50	%
Intensity	100	%

```
050
100
07
111
Set_Panel_Configuration
                Panel1  Panel2  Panel3  Panel4
Frequency       0,5     0,5     0,5     0,5
Duty Cycle      50      50      50      50
Intensity               100     100     100     100
```

Figure 5-1 GUI of the Flickering Light Stimulator

5.1 Connection

The communication between GUI and the embedded platform is done with a serial port connection via USB. For making the right COM port available in the COM port list, you need to connect the control unit and PC with an USB cable. If the right COM port does not appear yet, press the Refresh button, which renews the list of available COM ports. After clicking the Connect button and the connection was successful, a message is displayed in the message window in bold black letters.

Now you are able to configure the Panels and starting it afterwards. When the Panels are running, it is possible to disconnect the COM port, without stopping the Panels. This means a PC is just for configuring, starting and stopping needed, but not for the operation itself.

<u>Important:</u> When you want to run the Panels without the PC, configure and start the Panels first. Afterwards **<u>press Disconnect</u>** and **<u>then you can disconnect the USB cable</u>**. The embedded part needs to know, that the connection is now closed. Unplugging the USB cable without closing the COM port before leads to unpredictable errors! If this happens, the embedded board needs a hardware reset. This is done through plugging off both, the DC power connector (alternatively you can turn off the device via the power on button) and the USB cable! It is necessary to plug off the USB cable too, because on the Arduino board, the micro processor is also supplied via USB.

5.2 Color

You have also the possibility to change the color of the LED-panel. Therefore you can use the checkboxes where you can choose between the basic colors, green, red and blue. By choosing two checkboxes you have the possibility to combine different colors. The colors are also displayed in a small box beneath the checkboxes.

5.3 LED-panel

You have the possibility to set different frequencies, duty cycles and intensity for each of the four LED-panels. Therefore you can slide the scrollbar or type in the numbers which you want.

5.4 Buttons

In the following we want to describe the functionalities of the different buttons on the bottom of the GUI from the left to the right as follows.

5.4.1 Config

After the successful connection of the control unit to the PC, you can configure the LED-panels as described in the chapter about the LED-panel. After you have chosen the configuration of the panels the data is stored on the PC and sent to the µP of the control unit.

5.4.2 Reset

When you press the Reset button the LED-panel configuration is set to the standard configuration as when you start the program. To deliver the configuration to the control unit, remember to press the Config button to store the configuration and the press start to start the stimulation.

5.4.3 Status

When you press the status button the actual panel configuration is displayed in the Message Window.

5.4.4 Start

To start the stimulation after connection and after panel configuration you have to press the start button so that the panels start to flicker.

5.4.5 Stop

To stop the stimulation, you have to press the stop button.

5.5 Message Window

In the Message Window you can get data about the connection, if it is successful or not. Also you get the information about the sent data. If the color of the font is green everything is ok. When it is red then you have got a connection error. The blue font is the response from the µP.

List of References

AsTeRICS - Requirement File: Flickering Light Stimulator [Buch] / Verf. Starlab. - Barcelona : [s.n.], 2011.

List of Figures

List of Tables