VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

LIGHTING SYSTEM FOR AUDITORIUM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MAREK NOVÁK
AUTHOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

LIGHTING SYSTEM FOR AUDITORIUM

OSVĚTLOVACÍ SYSTÉM PRO AUDITORIUM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE AUTHOR	MAREK NOVÁK
VEDOUCÍ PRÁCE SUPERVISOR	Ing. MARTIN FRIEDL

BRNO 2014

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student:	Marek Novák	***ID:***	147684
Ročník:	3	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Osvětlovací systém pro auditorium

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s protokolem DMX, zhodnoťte jeho možnosti pro použití při velkém počtu koncových prvků. Proveďte rešerši o průmyslových sběrnicích a komunikačních protokolech, které by mohly být použity pro realizaci osvětlovacího systému. Na základě rešerše vyberte vhodné řešení nebo navrhněte vlastní komunikační protokol. Systém navrhněte a realizujte. Prototyp koncového prvku, který bude přijímat data pro osvětlovací jednotku oživte. Proveďte detailní testování celého zařízení a diskutujte jeho vlastnosti.

DOPORUČENÁ LITERATURA:

[1] MOBSBY, N. Practical DMX. Entertainment Technology Press, Limited, 2005

[2] AXELSON, J. Embedded Ethernet And Internet Complete: Designing and Programming Small Devices for Networking. Lakeview research llc, 2003

Termín zadání:	10.2.2014	***Termín odevzdání:***	30.5.2014

Vedoucí práce: Ing. Martin Friedl
Konzultanti bakalářské práce:

doc. Ing. Tomáš Kratochvíl, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRACT

This bachelor project details design of a lighting system, which exceeds the capabilities of the widely spread protocol DMX512 over RS485. In this document, you will learn about the limits of the protocol followed by a presentation of CAN bus protocol and 100BASE-TX protocol and its own limits. Having considered the above protocols, a topology which scales with the number of end devices will be designed along with the light fixture, which represents the end device in the system.

KEYWORDS

DMX512, RS485, ArtNet, CAN, Fast Ethernet, 100BASE-TX, PWM, LED driver, PIC32, LabWindows, QLC+

ABSTRAKT

V této bakalářské práci je uveden návrh a realizace osvětlovacího systému, který svou specifikací překonává možnosti široce rozšířeného protokolu DMX512/RS485. Dále jsou zde popsány omezení tohoto protokolu a také protokolů CAN bus a 100BASE-TX. V závěru je pak navržena síťová topologie, která je vhodná pro řízení daného počtu koncových prvků, spolu se samotným koncovým prvkem (řízeným světlem).

KLÍČOVÁ SLOVA

DMX512, RS485, ArtNet, CAN, Fast Ethernet, 100BASE-TX, PWM, buzení LED, PIC32, LabWindows, QLC+

NOVÁK, Marek *Lighting system for auditorium*: bakalářská práce. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Radioelectronics, 2014. 73 p. Supervised by Ing. Martin Friedl

DECLARATION

ACKNOWLEDGEMENT

I would like to thank to my supervisor Ing. Martin Friedl for his constant support, I would like to thank also to Ing. Josef Stastny for his analytic insights during our consultations.

Brno . . . .

(author’s signature)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Since the moment we can call the existence of human kind a civilization, people have been resolving problems of daily life using the results of their ancestors. That is a quick and efficent way we can advance in our understanding of the world.

It is possible to resolve an engineering problem using standardized protocols and methods or by inventing a new method or a system. DMX512 is a standardized protocol used in the domain of lighting, however there may exist situations where its limits on multiple layers make it unusable or not practical to use. In this document we place ourselves in such a situation:

The task of this thesis is to consider the usage of the DMX512 protocol for a given number of end devices, which will realize the actial dimming. The goal is to deliver the dimming data safely and in time to the destination end device. In this document, we will select the best topology, which will fulfill this role and which will scale with the number of end lighting fixtures.

Requirements for the end application is to be able to control 256 lighting fixtures each using the RGBW (Red, Green, Blue, White) color scheme with up to 16 bit color depth per color. The input interface to the lighting network will be done by the DMX512 protocol.

In the first part of this thesis, we will describe and analyze the DMX512 protocol, in the second one, we will explore other alternatives in the domain of industrial communication, in the fourth one, the selected topology and protocols will be described and in the last part, the actual design and production of network end devices will be described and discussed along with their parameters and testing.

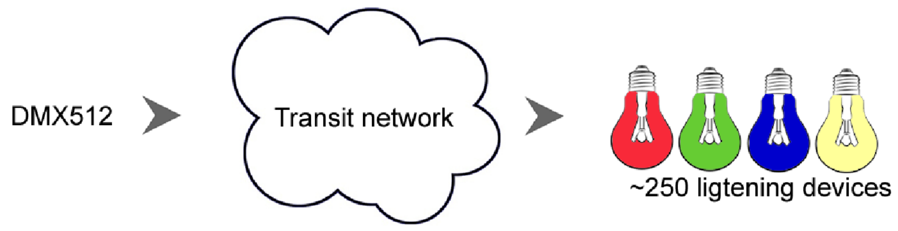


Fig. 1.1: Issue to be resolved

2 DMX512 PROTOCOL

2.1 Physical layer of the DMX512 protocol

DMX512 protocol uses for its physical layer the TIA-485 protocol (formerly known as RS485), which defines the electrical characteristics of the bus. This standard allows communication either at low speed (˜100 kBd) over higher distance (up to 1000-1200 m depending on used cable quality and its characteristic impedance) or at high speed (up to 10 MBd) over short distance (˜10 m). The communication is half-duplex, which is not limiting for the DMX protocol running on it, since in only sends dimming data to the remote lighting devices. (the communication is unidirectional)[8]

2.1.1 Cabling

As already mentioned, the cabling should have a well defined characteristic impedance of 120 Ω and for better performance, a termination resistor network is needed at the end of the twisted pair cable. Cat5 (Cat5e) UTP ethernet cables can be used in certain circumstances for the signal transport, especially when a full-duplex communication is required. The Cat5 UTP cable contains 8 wires, which enables the user to implement for example another control bidirectional communication channel running on a separate twisted pair within the same Cat5 cable.

2.1.2 Connectors

TIA-485 itself does not specify what type of connector should be used in the network, but this is well defined in the DMX512 protocol, which allows only 5-pin XLR or RJ45 connectors. (For the latter one, a specific pinout must be respected) In practice, the 3-pin XLR connector is sometimes used as well, but its usage is not recommended, because it can be easily confused with a microphone cable and this may electrically damage the DMX512 network controllers.

2.1.3 Baud rate

The communication speed is fixed by the DMX512 to 250 kBd and the serial data format is 8N2. (8 data bits, no parity and 2 stop bits). Two stop bits assure a better synchronization than the more known and used format of 8N1 (one stop bit), which is crucial for upper layers of DMX512 protocol.

The reason why the communication speed is fixed is to create a standard and robust protocol. Indeed, 250 kBd is a compromise between a long distance between

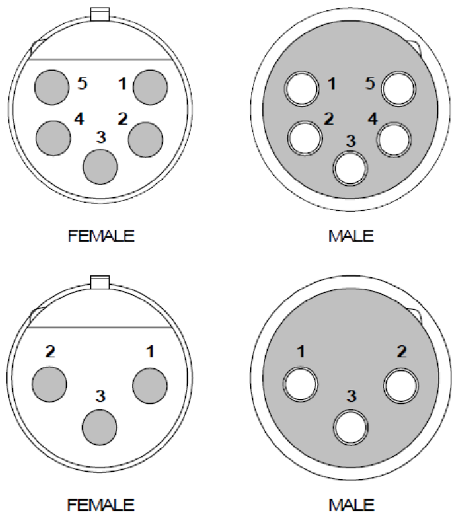


Fig. 2.1: XLR3 and XLR5 connectors [8]

Pin	Wire	Signal
1	Shield Drain	Ground / 0 V
2	Inner Conductor (Black)	Data -
3	Inner Conductor (White)	Data +
4	Inner Conductor (Green)	Data - (Spare)
5	Inner Conductor (Red)	Data + (Spare)

Tab. 2.1: XLR3 and XLR5 connectors DMX512 pinout [8]

the nodes and the network throughput. To understand better this decision, it is important to keep in mind, that RS485 protocol cannot support more than 32 devices on the same bus. The reason for this is to limit the intensity of current, which must be sourced or sunk by a single RS485 driver if there is the maximum number of devices. If we divide this actual speed of 250 kBd by the maximum number of devices on the same bus, we get a baud rate of 7.8125 kBd or 710 B/s, which can be delivered to a single device. (note: one byte on the DMX512 network consist of a start bit, 8 data bit and 2 stop bits) If we give ourselves the goal of refresh frequency of for example 50 Hz, which is enough considering human eyes' vision persistence, we can have up to 14 B of data per one refresh delivered to a single dimming/lighting device. This enables us for example to make the device dim 4 colors (red, green, blue and white scheme) and to send some special control data during each refresh. Thus, 250 kBd is quite a generous speed for 32 dimming devices.

2.1.4 Robustness

One of the reasons, why the DMX512 protocol got accepted as a lighting communication protocol standard is its robustness. The TIA-485 is a differential (it uses a differential balanced line for the transmission) and thus is resistant to electromagnetic interference from the outside world. When the difference voltage between the positive and the negative differential input (positive minus the negative) greater than 200 mV a logic 1 is received and when this voltage is less than -200 mV a logic 0 is received. The interval in between these two values is undefined and will act as a hysteresis. [2]

2.2 Upper layers of the DMX512 protocol

2.2.1 Packet format

The standard packet consists of several elements from which some are optional and some are not:

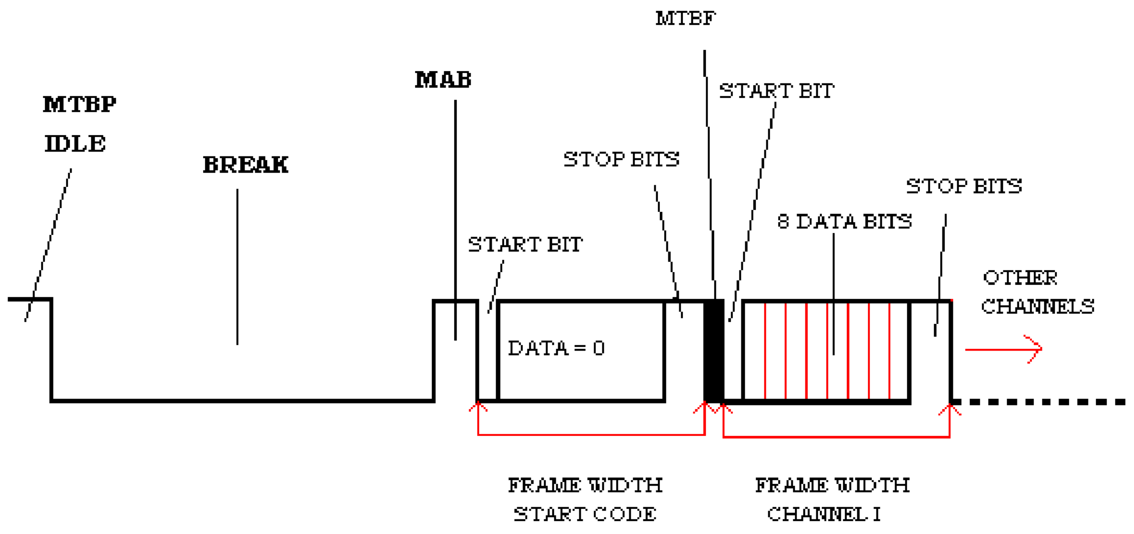


Fig. 2.2: DMX512 Timing diagram [10]

Break

Break impulse is the first required timing element of the DMX512 packet, since it initiates the packet transmission. It needs to be longer than 88 µs (in other words 22 "low" bits), to let the packet be accepted by the remote receiver. In practice, a slightly longer period of time is used for the BREAK impulse (˜100 µs) to make sure this condition is met. In fact, this is what makes the DMX512 protocol robust: The remote receiver is latching last valid data, if an erronous BREAK impulse is detected and so the synchronization is never lost. This implies also another propriety of the protocol – no address is sent to make the receivers know what is the meaning of the data. Instead, the receivers count the sequence number of bytes to drop all data they don't need and to pick only the data that are really for them. The sequence number of each byte is called DMX channel.

Mark After Break (MAB)

When the BREAK impulse ends, the line goes high for a minimum time of 8 µs (or 2 "high" bits), which is called "Mark After Break". The duration of the Mark is limited also from above by 1s and if this condition is not met, the packet is dropped as incorrect. In older standards, the minimum duration is set to 4 µs , which may make an old lighting equipment not work with the new one.

Start Code (SC)

Next packet element is called “Start Code”, whose role is to define who is the data destined for. In fact, it is the first byte of data stream, which can take up to 513 B (1 to 512 + 1). For dimmers and the majority of devices, this byte is set to 0. The format of the byte is as mentioned before 8N2, which means each byte is composed of the following: 1 start bit, 8 data bits and 2 stop bits. The duration of a bit is 4 µs , so one byte on the bus takes 44 µs to be transmitted. Notice that no parity bit is being sent: Indeed, DMX512 is an unreliable data transfer protocol, which means that the real-time capabilities are prioritized and possible data corruption on the media is acceptable. If we added a checksum or at least the parity bit in DMX512, the transfer would become more reliable, but the data transmission would be slower and each device would have to process each byte to calculate the checksum.

Mark Time Between Frames (MTBF) and Mark Time Between Packets (MTBP)

These two timing elements are optional, which means their duration can be as low as 0 s . MTBF represents an additional prolongation of the stop bits at the end of each byte. This prolongation may be added for example if the DMX512 bus master must load next byte in the transmit FIFO of its UART interface and is too slow and thus, this is generally an unwanted effect. MTBP is an additional pause between whole DMX512 packets – between the last channel data of one packet and the break of the next packet. This additional time may be used to unload the DMX512 bus master and to reduce (if it is needed) the refresh rate. [16]

2.3 Limits of the DMX512 protocol

In order to better judge, if the DMX512 protocol can be used for a big number of target devices, let’s sum up its limits.

2.3.1 Limits of the physical layer

First, TIA-485, which is used as the physical layer of the DMX512 limits the maximum number of devices on the bus to 32 (in case no repeater is present). This implies DMX512 cannot be directly used without the use of repeaters or “daisy chaining” of devices with signal retransmission at each node. Second, TIA-485 does not impose galvanic isolation between nodes and the bus, which brings problems considering a higher number of devices.

2.3.2 Limits of upper layers

First, the largest possible frame, which can be transmitted over the DMX512 network contains only 512 B . This is indeed limiting if we want to have many devices which each of them need at least 8 DMX channels (e.g. lighting device with RGBW scheme, each color with the depth of 16 bit). This would allow us to have at maximum 64 of such devices. Second, if we wanted to increase the number of DMX channels, we could use some non-standard Start Codes (other than 0) to have up to 131072 channels, which would let us have up to 16384 devices requiring 8 DMX channels, but the refresh rate would be unacceptable (˜0.2 Hz). If we wanted only 1024 such devices, the refresh rate would be about 3.5 Hz , which is still unacceptable. Whats more, such settings is not standard and only proprietary lighting devices could be used in such a configuration.

2.4 DMX512 - Conclusion

DMX512 is not suitable for a network, where an extensive number of lighting devices, which each require multiple bytes (8 B) for one refresh cycle is used. The only possible option would be to use multiple buses, with each of them controlling several target devices. This is an option, which will be discussed against other solutions later.

3 EXISTING INDUSTRIAL PROTOCOLS

In previous section we showed, that DMX512 protocol is not suitable for a lighting network, where a higher number of devices, which require much data is present. Therefore, in this section, we will make a brief review of existing industrial protocols in order to see, whether there is any better solution or if a new, proprietary approach is needed.

3.1 Fast Ethernet (100BASE-TX)

Fast Ethernet is a new revision of existing Ethernet standard, which brings a faster speed of 100 Mbit/s (Compared to 10 Mbit/s). Nowadays, the “-TX” version is the most widely spread and it uses two twisted copper pairs of Cat5 cable as the physical medium. Thanks to the popularity of this technology, it has become a cost-effective alternative for high-speed backbone networks. [19]

3.1.1 Ethernet and ISO/OSI model

Talking about communication between different systems, one should not forget to mention the ISO/OSI model, which clearly defines the interface between each protocol of each layer of the communication. Even in the previous section, the DMX512 protocol could be explained using the ISO/OSI model, but having in mind the simplicity of the protocol, I did not find it necessary.

So what is the reference model ISO/OSI? It is a theoretical model, which splits the process of communication in several layers.

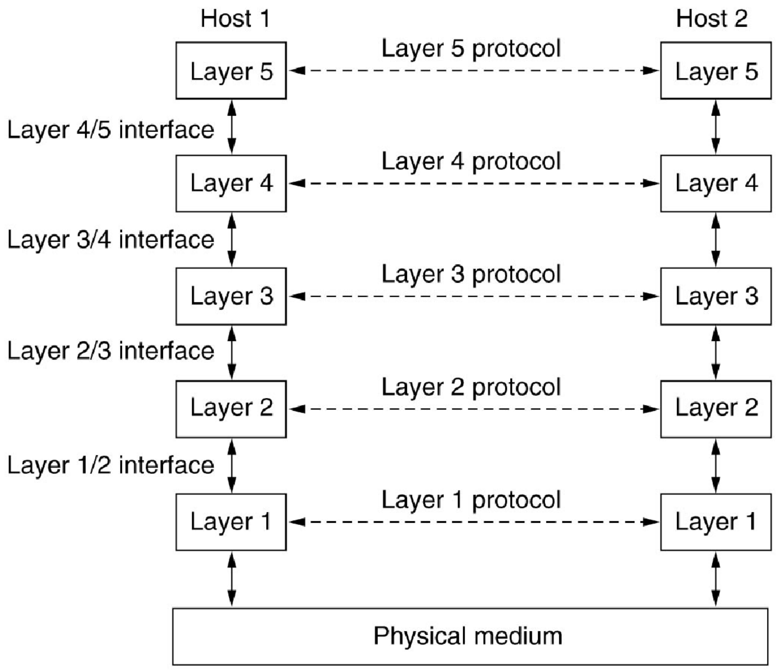


Fig. 3.1: ISO/OSI model - interlayer communication [9]

As shown on the image, each layer communicates in a horizontal manner with its correspondent “on the other side”. This is how the protocols are programmed, using abstraction over all lower layers. The reality is, however, that the data flow exists always between two neighbor layers within one single device and when the lowest layer is met, it handles the actual transmission to the destination device. We are talking about the encapsulation mechanism. The process is inverted on the other side and the data are de-encapsulated or let’s say passed-through each layer starting from the lowest one (the actual reception of the signal) to the highest one (the message is processed by the application logic).

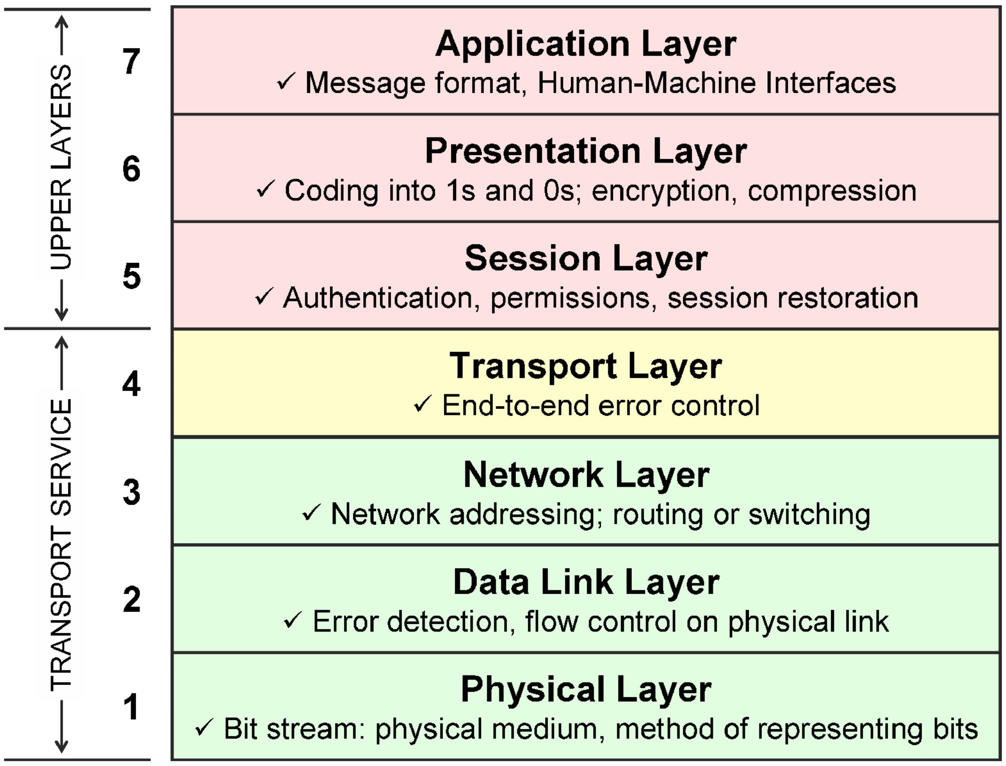

UPPER LAYERS
TRANSPORT SERVICE
7
Application Layer
✓ Message format, Human-Machine Interfaces
6
Presentation Layer
✓ Coding into 1s and 0s; encryption, compression
5
Session Layer
✓ Authentication, permissions, session restoration
4
Transport Layer
✓ End-to-end error control
3
Network Layer
✓ Network addressing; routing or switching
2
Data Link Layer
✓ Error detection, flow control on physical link
1
Physical Layer
✓ Bit stream: physical medium, method of representing bits

Fig. 3.2: ISO/OSI model - layers' names [17]

Ethernet is a protocol present on layer 1 (Physical Layer), where it is represented by the definition of cabling, bit encoding and electrical parameters of the media, but it is also a protocol of layer 2 (Data Link Layer), where it is represented by a standardized Ethernet frame format, which is used for basic addressing and first error-checking of the received data. Interface between layer 1 and 2 is done by MII (Media Independent Interface) protocol. This will be discussed in detail later.

3.1.2 Physical Layer of Ethernet – cabling and connectors

100BASE-TX Ethernet physical medium is a twisted pair cable known as Cat-5 or Cat-5e (Category 5 cable). Nominal characteristic impedance of this cable at 100 MHz is 100 Ω and the bandwidth is 100 MHz. Each cable contains 4 twisted pairs and each of these pairs is twisted differently (with different number of twisting per distance) in order to reduce the cable crosstalk.

Maximum length of a point-to-point connection using this type of cable is 100m. In order to interconnect two devices, different wiring schemes can be used. The most common are crossover and straight-through cables which use T568B wiring

	Pair color	[cm] per turn	Turns per [m]
🟩	Green	1.53	65.2
🟦	Blue	1.54	64.8
🟧	Orange	1.78	56.2
🟫	Brown	1.94	51.7

Tab. 3.1: Category 5 cable twisted pair twisting distances [25]

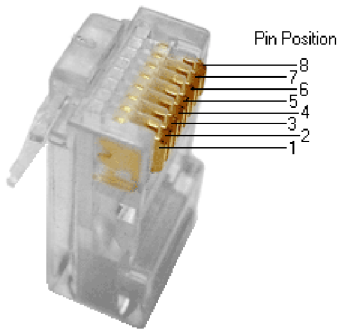


Fig. 3.3: Category 5 cable connector (8P8C/RJ45) [26]

on both side (straight-through) or T568A on one side and T568B on the other one (crossover).

The connector depicted in the previous table is called RJ45 or 8P8C and is the standard connector to be used with Cat-5 cables for the Ethernet physical layer, as defined by the TIA-568 standard.

Since the Ethernet is really thoroughly defined and well specified on each of its layers, the physical layer must also define a pinout for the cabling. There are two standards defining the correct pinout to use in 100BASE-TX Ethernet networks, T568A and T568B as mentioned before:

Pin	Pair	Wire	Color
1	3	1	white/green
2	3	2	green
3	2	1	white/orange
4	1	2	blue
5	1	1	white/blue
6	2	2	orange
7	4	1	white/brown
8	4	2	brown

Tab. 3.2: T568A Wiring [25]

Pin	Pair	Wire	Color
1	3	1	white/orange
2	3	2	orange
3	2	1	white/green
4	1	2	blue
5	1	1	white/blue
6	2	2	green
7	4	1	white/brown
8	4	2	brown

Tab. 3.3: T568B Wiring [25]

3.1.3 Physical Layer of Ethernet – encoding

If we want to send some data over a medium, we need to use an encoding. First of all, let's briefly present the basic types of encoding:

- NRZ (Non-Return to Zero): An obvious technique of transmitting bits. If a bit value 0 is to be transmitted, a low logic level is output on the bus and the same for bit value 1. This simple technique may cause problems if a long string of zeroes or ones is to be transmitted, because in such case, the transmitter and the receiver can get out of synchronization.
- NRZI (Non-Return to Zero Inverted): This technique represents a bit value 0 by leaving the logical level of the bus the same as it was before and a bit value 1 by inverting the logical level of the bus. This solves the problem partially for long strings of bits with the same value, since the ones are represented by an alternating signal, but does not solves the problem if a string of zeroes is to be sent.
- Manchester: In order to be able to send any data without risking bad synchronization, a technique called Manchester can be used. This encoding is done by applying and exclusive-OR operation on the clock signal and the data to be sent and the resulting signal is then output on the bus. This way is used by the former 10 Mbit Ethernet standard, but cannot be reasonably used with the new 100BASE-TX, since it doubles the baud rate of the signal as it can be seen on the following figure. As we mentioned earlier, Cat-5 cables have a bandwidth of 100 MHz. This cable would obviously not be compatible with a 200 MHz signal.

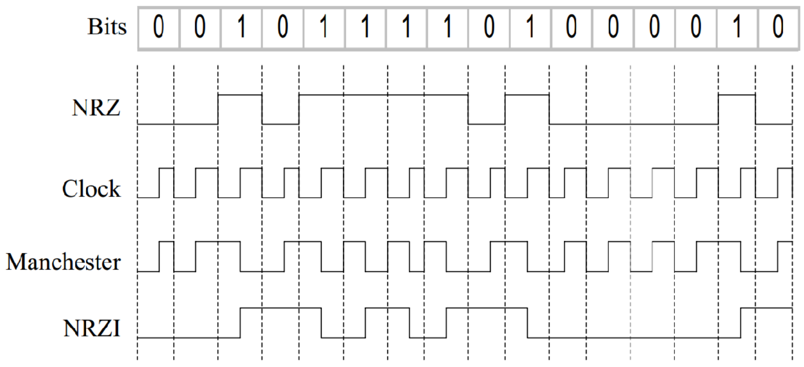


Fig. 3.4: Example of encodings [18]

- 4B5B: To reduce the required bandwidth (economic problem with adequate cabling), a 4B5B encoding was invented. The idea is that a short string of consecutive zeroes or ones does not cause a bad synchronization if the clock sources of both asynchronously communicating devices are reasonably stable and accurate. It's all about a compromise. This is realized by representing a nibble (four bits) by a sequence of 5 bits transmitted using the NRZI technique (strings of ones disappear completely). There is another advantage of this encoding: There are some of the codes, that remain free for some control messages, since there is only 16 bit patterns to be encoded using 5 bits, which leaves about a half of unused codes. Furthermore, the resulting spectrum is not so rich to higher harmonics, since the resulting signal does not contain more than 3 consecutive zeroes. (reader's deeper knowledge of signals is required for a complete understanding) This reduces the crosstalk between the cable wires and improves the EMC characteristics of the device using such an encoding instead of a simple NRZ encoding. Finally, the DC equalization is introduced when 4B5B encoding is used, which means that the DC component of the signal is equal or very close to 0 V.

However, 100BASE-TX goes even further in the encoding and uses not only 4B5B encoding, but inputs the output of 4B5B to MLT-3 (Multi-Level Transmit) encoding mechanism, which reduces the fundamental frequency of the 4B5B signal to one fourth by adding another level in the output signal. With MLT-3, the bus can be in 3 states: high (positive voltage), low (negative voltage) and middle (0 V). This enables the bit value 1 to be transmitted as a transmission either from high to middle, or middle to low or low to middle or middle to high. If a bit value 0 is to be transmitted, the bus "memorizes" its last state for a period corresponding to one bit transmission time. Thus, the fundamental frequency is reduced from 125 MHz (output from 4B5B is 125 MBd to have 100 Mbit/s effectively) to 31.25 MHz, which is more amenable for transmissions over copper wire. [18]

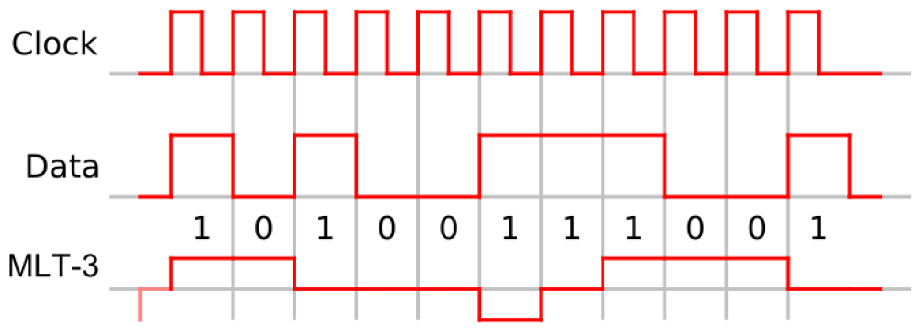


Fig. 3.5: MLT-3 encoding [18]

3.1.4 Physical Layer of Ethernet – CSMA/CD

CSMA/CD (Carrier-Sense Multiple Access / Collision Detection) is a principle of accessing the medium in Ethernet networks. This technique appeared with the old 10BASE5 Ethernet over coaxial cable, where there were physically multiple devices (more than two) capable of accessing the same medium at the same time (shared medium). Let's explain its operation: A device on the bus is constantly listening to the traffic and only transmits if there is no traffic detected. It a collision occurs a signal called jam signal is transmitted on the bus to make all devices know, that a collision occurred and all transmitting devices stop their operation for a random period of time. After this time, which is not completely random (there is a fixed part, that is increasing with the number of successive bus collisions and a random part), the transmitter whose period of time was the shortest starts to transmit. Others detect that the bus is occupied and wait for their turn.

This principle is not much needed nowadays, since 100BASE-TX uses twisted pairs as medium and the connections are mostly made point to point (there are only two devices on the bus). An exception to this rule is made by devices called hubs, which are not often used. These devices literally repeat the signals they receive on their ports to all other ports, thus extending the collision domain. The use of such devices is not advisable, only for diagnostic purposes. Last thing to point out is, that the active usage of CSMA/CD technique significantly decreases the effective bit rate, since during the collisions, time is wasted by waiting for the end of the collision state.

3.1.5 Data Link Layer of Ethernet

Ethernet is a bus, which means, that the traffic is forwarded to all devices present on the network. Who is the data contained in the traffic destined for is determined by what we call a MAC address, which is included in the header of Layer 2 data unit. Each Layer has its own data unit to be able to work with the data in an efficient way. The data unit of the Data Link Layer is called a frame.

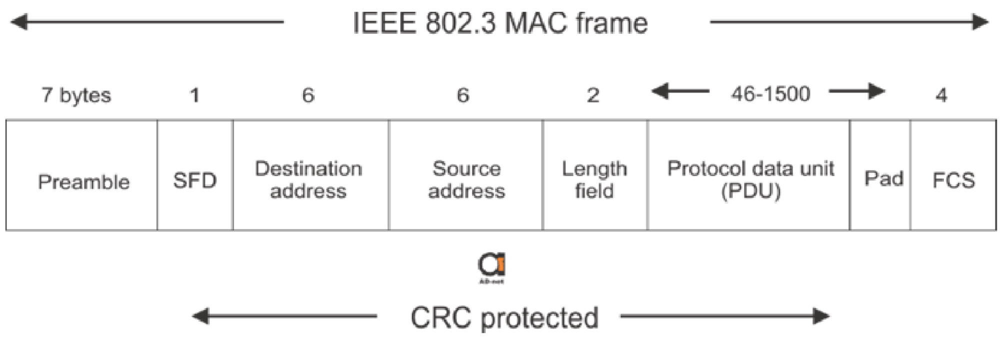


Fig. 3.6: IEEE 802.3 MAC frame format [1]

On the previous figure, you can see the structure of an Ethernet frame, let's describe its components:

- Preamble: It consists of 7 bytes, each containing 10101010 pattern, which is used to synchronize receivers local clock signal with the transmitting node one.
- SFD (Start Frame Delimiter): This byte, which is a part of the Preamble, is equal to binary 10101011 and it makes the receiver know, that the actual frame is following.
- Destination address: Six byte long destination MAC (Media Access Control) address of the Ethernet interface to which the frame is to be sent. MAC address consists of two parts, the first 3 bytes of the address identifies the vendor and last 3 bytes identify the device address within that vendor's production.
- Source address: Six byte long MAC address of the interface, which sent this frame. (Address of the interface, which last forwarded the frame).
- Length field / EtherType: This 16-bit field either identifies the higher protocol used in the Protocol Data Unit part of the frame or, if the value is in the range of 46-1500, it contains directly the length in bytes of the PDU.
- PDU (Protocol Data Unit): The actual payload of data, which contains higher protocols' data.

- Pad: In case the PDU is not aligned (to 2 or 4 bytes for example), an extra padding can be inserted to allow faster processing by all network devices.
- FCS (Frame Check Sequence) or CRC (Cyclic Redundancy Check): Four byte field consisting of a mathematical operation result, which is made on Destination Address, Source Address, Length field and PDU. This value is compared to the value calculated locally by the receiver using the same algorithm to check the integrity of the frame.

3.1.6 Upper layers of Ethernet

Upper layers of Ethernet consist of stacks such as IPv4, IPv6 or EtherCat or others. Nevertheless, the choice of the stacks does not significantly influence the parameters such as speed of transmission (bitrate), maximum length of a point-to-point connection or maximum number of devices on a bus, which are crucial for us. Therefore, we will not discuss them in this section.

3.1.7 Ethernet - Conclusion

Ethernet is a network standard, which was primarily designed for local area networks. Nevertheless, its usage is ubiquitous in today's networks and its development was not yet stopped, which shows it is a well designed general purpose network protocol. There exist even industrial implementations of Ethernet such as EtherCat or more common implementations used in Internet networks such as IPv4 or IPv6, which shows its versatility. In our particular case, 100BASE-TX would be well suitable for a backbone data transfer of lighting data.

3.2 CAN bus (Controller Area Network)

CAN bus is a communication protocol, which was designed by Bosh in 1980's - 1990's to form a standard in the automotive industry. Since its first specification in 1983, it has evolved thanks to the demandings of the industry it was designed for. In 1991 CAN 2.0 specification was published by Bosh and even in 2012, Bosh has specified an improved CAN data link layer, called CAN FD. [4]

3.2.1 Physical layer of CAN

The physical layer defines electrical characteristics of the bus and of the drivers to be used to transmit on the CAN bus. Signaling rate, bus length, cabling and bus termination is also defined in this layer. The specification is indeed too detailed to be completely defined in this document, therefore only the most important aspects and rules are goind to be explained.

Signaling rate

In order to implement a multimaster bus, Bosh had chosen to make the bus drivers active only in one state - the logic zero. Thus, if multiple masters want to transmit, no short circuit or bus driver overloading can occur, since the drivers use the open drain/collector configuration. This mechanism enabling CAN for real-time transmission and bitwise prioritization of messages based on the leading bits of each message has however a big disadvantage: If a logic one is transmitted, the bus is in high impedance state with respect to both power supply rails and if the overall capacity of the bus is too high, an unacceptable slew rate is achieved at a too high signaling rate. This is why there is a quasi-linear depence between the maximal applicable bit rates and the total bus length as given by the following table:

Bus Length [m]	Signaling Rate [Mbps]
40	1
100	0.5
200	0.25
500	0.10
1000	0.05

Tab. 3.4: CAN signaling rates [5]

The rule of thumb is that the signaling rate in Mbps multiplied by the bus length in meters should not exceed 50.

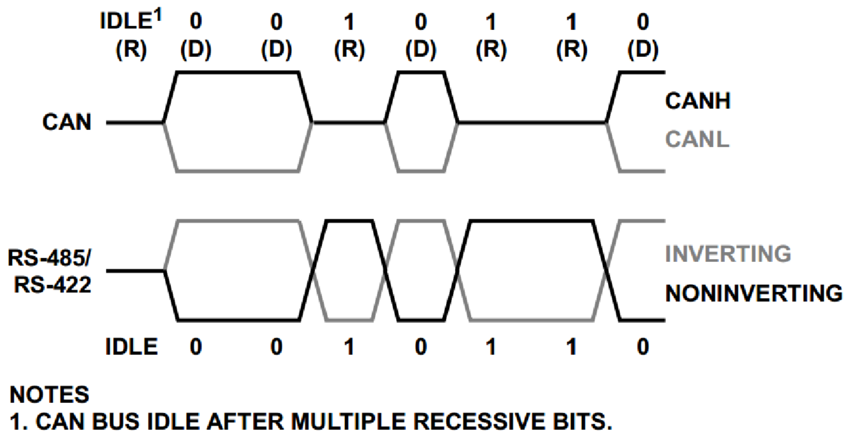


Fig. 3.7: Comparison of differential signaling for CAN and RS-485/RS0422 [24]

Cabling and connectors

Standard CAN implementation uses for node interconnection single balanced twisted pair with characteristic impedance of 120Ω. If CAN is running at its highest specified signaling rate of 1 Mbps, the length of the line is limited to 40 m and the number of connected nodes to 30. ISO 11898 defines as well, that any unterminated stub connected to the main bus should not exceed 0.3m in length to avoid reflections. It should be emphesized, that CAN uses differential signaling, since thanks to this, it has its robust noise immunity and fault tolerance.

Even if the standard does not strictly define the type of connectors to be used, there are several common variants, which can be used. Among these 9-pin DSUB, RJ10, M12 can be cited. [6]

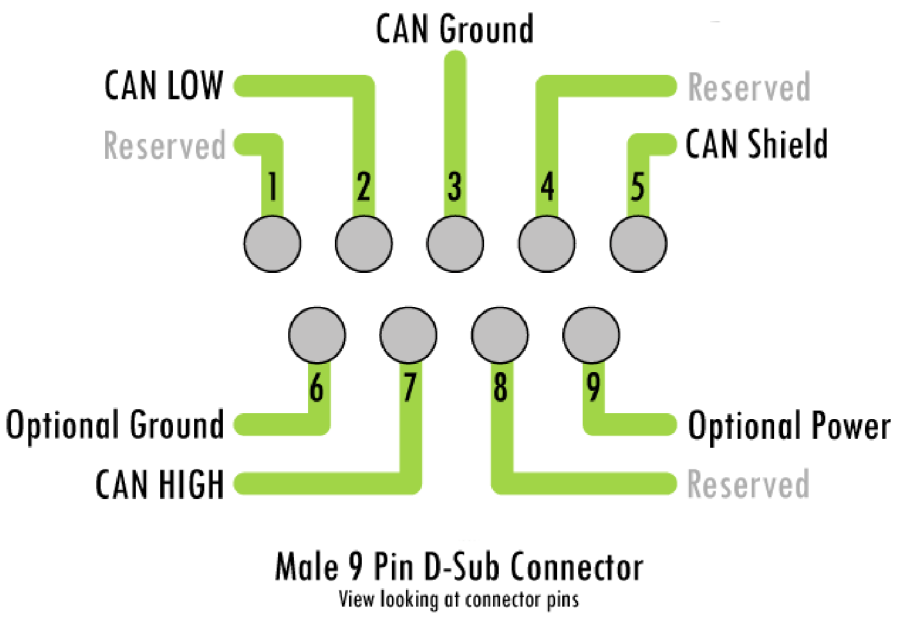


Fig. 3.8: CAN Bus 9 Pin D-Sub Pinout [3]

Line termination [15]

In order to eliminate refrections on the bus, a termination circuit must be connected at each end of the distributed bus topology network. The value of the terminating impedance must be equal as exactly as possible to the characteristic impedance of the bus. If any other value is selected, the reflections do not disappear completely. There are three basic methods to terminate properly a line:

- Standard termination: The most simple termination method, yet a sufficient one for the majority of CAN networks.

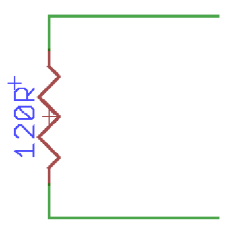


Fig. 3.9: Standard termination circuit

- Split termination: To increase EMC performance, one can use two resistors of 60 Ω instead of a single resistor of 120 Ω and short circuit the middle using a capacitor of about 4.7 nF . This results, considering the parallel combination of two capacitors at each end of the bus, in a 3 dB loss point at approximately 1.1 MHz . The maximum speed of CAN is 1 Mbit, which corresponds to 1 MHz square wave signal, since CAN uses a simple NRZ encoding.

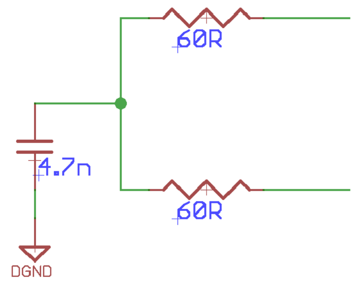


Fig. 3.10: Split termination circuit

- Biased split termination: As its name says, this termination method is based on the preceding one, but an aditional biasing resistor network is added to maintain the voltage level on the capacitor to be equal to the half of VDD. Thus if there is no transmission over a long period of time, the common mode voltage on the bus is kept stable.

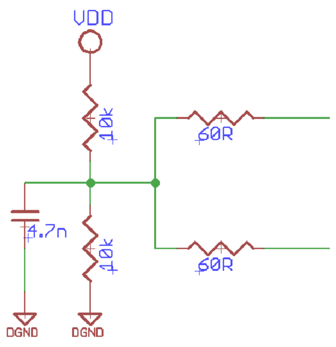


Fig. 3.11: Biased split termination circuit

3.2.2 Data link layer of CAN

Transfer layer

Transfer layer is where the majority of CAN specific principles are applied such as Arbitration, Transfer Rate and Timing handling, Acknowledgment of received messages, Error Detection and finally Message Framing. Let's discuss the most important ones.

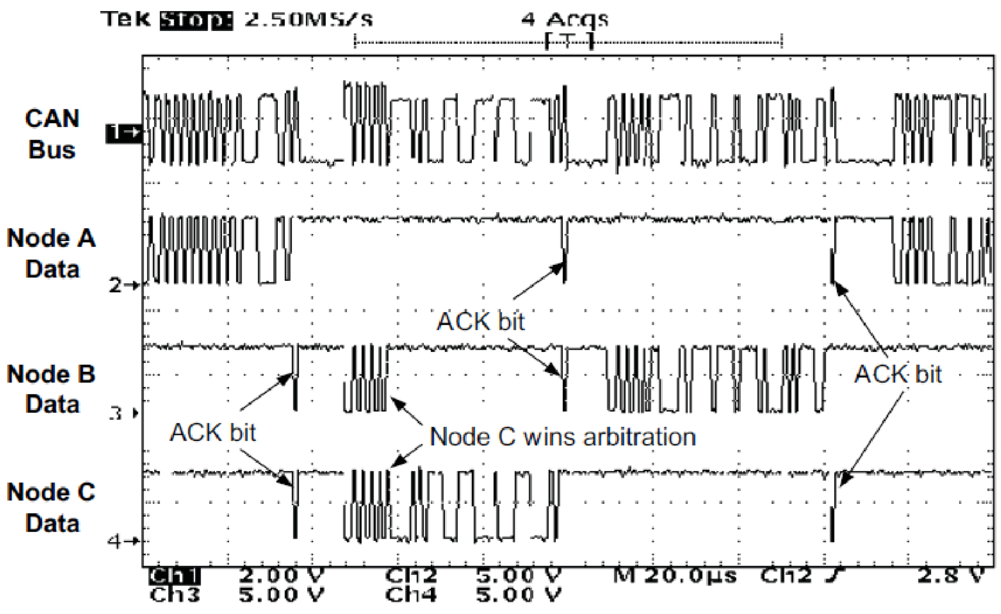


Fig. 3.12: CAN bus traffic - arbitration [6]

Arbitration - CSMA/BA [21] CAN bus uses a shared medium to deliver the data from one node to another one, therefore a principle which avoids intermittent collisions is required. As mentioned before, CAN was developped for the automotive industry, where a precise timing may become very critical in certain circumstances. Along with these potentially critical messages, which can be output to or destined for a motor control unit for example or any other device, which requires a precise timing, in every car, there are also messages associated with uncritical systems such as front light inclination control and so on. CAN was designed to allow coexitence of these two types of nodes.

CSMA/BA stands for Carrier Sensing Multiple Access / Bitwise Arbitration. This quite well explains how this method of collision handling works: A node which wants to transmit is sensing the shared bus for any other node's transmission and if there is no transmission taking place, it starts to send its own data. This clarifies the first part of the abbreviation, the same method is used by the way by the Ethernet. As for the second part - Bitwise Arbitration, it is applied only in a situation, where more than one node starts to transmit at the same time. In such case, a node which transmits a message which starts with logic zeroes wins the arbitration and will continue to transmit with no interruption. The other one detects, that the data it output on the bus are not the same as the data it received and stops its transmission and retries automatically after the node which won the arbitration finishes with its own message. This is possible thanks to the fact, that logical zero overwrites the logical one, if both are output on the bus at the same time. We are talking about a recessive bit (logical 1) and a dominant bit (logical 0).

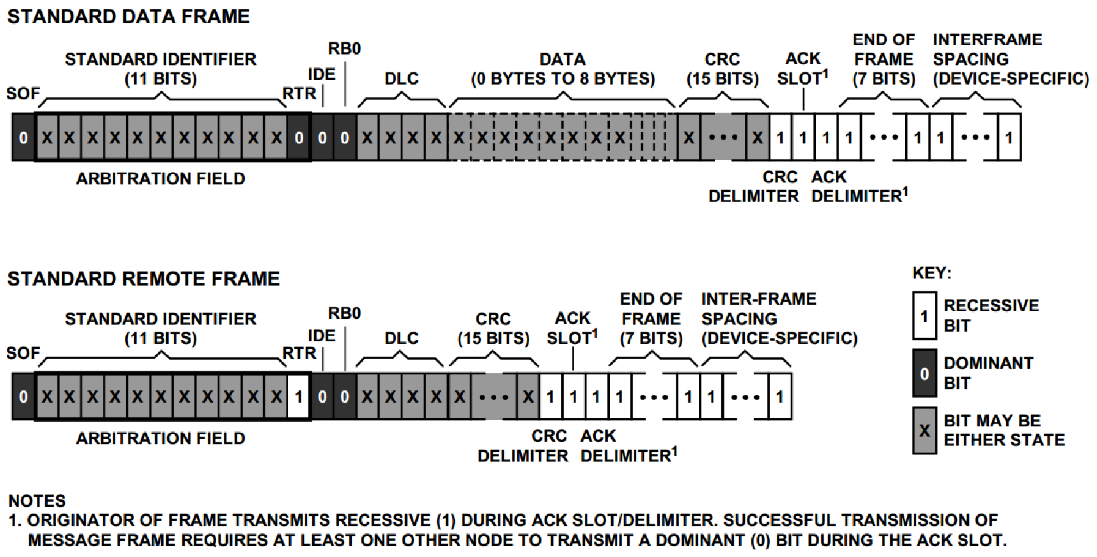

STANDARD DATA FRAME
STANDARD IDENTIFIER (11 BITS)
SOF
RTR
IDE
RB0
DLC
DATA (0 BYTES TO 8 BYTES)
CRC (15 BITS)
ACK SLOT1
END OF FRAME (7 BITS)
INTERFRAME SPACING (DEVICE-SPECIFIC)
ARBITRATION FIELD
CRC DELIMITER
ACK DELIMITER1
STANDARD REMOTE FRAME
STANDARD IDENTIFIER (11 BITS)
SOF
RTR
IDE
RB0
DLC
CRC (15 BITS)
ACK SLOT1
END OF FRAME (7 BITS)
INTER-FRAME SPACING (DEVICE-SPECIFIC)
ARBITRATION FIELD
CRC DELIMITER
ACK DELIMITER1
KEY:
RECESSIVE BIT
DOMINANT BIT
BIT MAY BE EITHER STATE
NOTES
1. ORIGINATOR OF FRAME TRANSMITS RECESSIVE (1) DURING ACK SLOT/DELIMITER. SUCCESSFUL TRANSMISSION OF MESSAGE FRAME REQUIRES AT LEAST ONE OTHER NODE TO TRANSMIT A DOMINANT (0) BIT DURING THE ACK SLOT.

Fig. 3.13: CAN Standard Message Frame Fields [24]

Message Framing When a node wants to send its data, it must split it into unified data units, which are called frames. Each frame may contain up to 8 data bytes, a Cyclic Redundancy Check, Standard Identifier and other control fields.

SOF (1bit) SOF stands for Start Of Frame. Since the idle state of the bus is read as a logical 1 (or a recessive bit), each frame transmission starts with a logical 0 bit.

Standard Identifier (11bits) Standard identifier is used for message prioritazation and in higher layers, it is also used to filter out the data or to clasify them.

RTR (1bit) RTR stands for Remote Transmission Request. This bit is used to decide, whether the frame is a data frame or a remote frame. Data frames contain the actual data, whereas remote frames contain only requests destined for a remote nodes to ask them to send some specific data frames. There is no master or slave on the bus. Every node acts and decides about transmitting completely independently.

IDE (1bit) IDE stands for Externded Identifier. If a further subaddressing is required for a specific network, this bit is set to logic 1 and followed by another 18bits which contain the Extended Identifier. Nodes using extended and standard identifiers may therefore coexist on the same bus.

RB0 (1bit) RB0 is a Reserved Bit 0. It is reserved for future CAN bus expansion.

DLC (4bits) DLC is the Data Length Code. It contains the length of the following data. Valid values go from zero to eigth.

Data (0 ˜ 8bytes) Data field contains application specific data.

CRC (15bits) and CRC Delimiter (1bit) CRC is a Cyclic Redundancy Check and it serves to verify the data integrity.

ACK slot (1bit) and ACK Delimiter (1bit) This bit is transmitted by the message originator as a logic 1 and it is used for frame transmission acknowledge. Any other node present on the bus will automatically transmit a logic 0 during this bit time slot if it calculated the same CRC as it receives in the frame. The acknowledge is sent even if the frame is later on filtered out in the receiving node by its higher CAN layers.

End of Frame (7bits) This field terminates the frame leaving the bus in recessive state for a period of 7bits. After this time, another frame can be transmitted on the bus.

Object layer

Object layer takes care of Message Filtering based on the Standard Identifier and if IDE bit is set to one also based on the Extended Identifier. Messages may be this way input in some implementation specific FIFOs (First In First Out buffers).

3.2.3 Upper layers and stacks

Different stacks can be run on the top of the CAN protocol. This is not required, but recommended especially for more complicated systems. It is however important to consider, that any higher layers are mostly always implemented in software, which means that effects known as jitters, time delays and other time related effects may occur. The whole system may therefore lost its real time capability. Among the most known stacks, we can cite CANopen, CANbedded, DeviceNet and others. To mention also the advantages of these stacks: User applications running on different microcontrollers get an easy to use interface to share application data between them. Things as shared application memory/registers or application input and output FIFOs may be implemented. The user programming such communication can adopt a high level of abstraction and so, even if he doesn't understand all principles of the CAN bus, he can design a working communication in his system. The abstraction in the system design process is a must, especially in more complicated systems, since it enables the programmer to focus on the problem to be solved and not on implementation of details, such as how to send a message over the CAN bus.

3.2.4 CAN - Conclusion

As it was shown in this section, CAN bus is a ready to use network solution, which was standardized and is widely used in automotive idustry. It is made primarily for multimaster communication purposes and thus, it is well suitable for systems, where asynchronous events may occur often and a critical timing is required. In our case, the end ligtening device may be treated almost as a slave device, since its role is to receive the dimming data and to output it in form of the light and color. It is to be discussed if we want or if we need a communication in the other direction for diagnostic purposes for example. Nevertheless, CAN bus can handle this. Another aspect to consider is if there is a cost effective solution on the market (a microcontroller), which integrates the CAN bus layer 2. The layer 1 is nearly always implemented in a separate bus driver. The standard number of nodes on a single CAN bus without the use of repeaters is 30 (it can be extended to 64 if a carefull bus design is applied [22]), this is another aspect to consider.

4 CHOOSING PROTOCOLS AND A TOPOLOGY

4.1 Minimum requirements

4.1.1 Detailed description of the task specification

The goal of this thesis is to deliver the dimming data to lighting fixtures, which will be situated in the Scottish Rite auditorium in Oakland, California.

Fig. 4.1: Scottish Rite Center auditorium

The lights to control will be physically positioned in a circular topology in order to deliver the light equally to entire lightened area. The radius of the circular topology is approximately 50 m. The estimated number of fixtures, which will be necessary to lighten such area goes from 250 to 350, depending on the end lighting device and the color scheme to use will be RGBW (Red, Green, Blue, White) with either 16 bit per color or 8 bit per each red, green and blue channel and 16 bit for the white channel. The desired refresh rate of light level of each light should be 40-50 Hz minimum. As an lighting industry standard, the input to this network is asked to be the DMX512 protocol. These are the main requirements to consider, when selecting communication protocols to use and the right topology to use for the network interconnecting the end devices.

4.1.2 Specification analysis

To better understand, what demands are implied by the specification, let's calculate some figures.

Total Required Data Rate

Considering the ”worst case” situation, there will be 350 light fixtures each needing 4x16 bit with the rate of 50 Hz, which gives us a required throughput of:

$$Throughput = N \cdot C \cdot B \cdot f = 350 \cdot 4 \cdot 16 \cdot 50 = 1.12\, Mbit/s \tag{4.1}$$

Where N represents the number of light fixtures, C stands for the number colors per fixture, B stands for the bit depth of each color and f is the required refresh rate.

In the equation, the byte justification, which is required by the majority of communication protocols and by all communication protocols mentioned in this document is not considered. In case for example 12 bit color depth is asked, 16 bit must be sent to the network, which adds an additional overhead.

Minimum length of the bus

In this calculation, let’s ignore any stubs required to connect a device to a bus and let’s calculate only the minimum length of the backbone bus knowing that the radius of the amphitheatre is around 50 m.

$$l = 2\pi\cdot\mathrm{r} = 2\pi\cdot 50 = 314.2\ \mathrm{m} \tag{4.2}$$

Where r is the radius.

Indeed, 314.2 m is a significant lenght and must be considered when selecting the appropriate communication protocol.

Price minimization consideration

Since there may be up to 350 end devices, the selection of the communication means will considerably change the cost of the whole lighting system.

4.2 Considering the studied protocols

In order to make a selection, let’s compare the protocols mentioned before in this document in a table:

Communication protocol	DMX512	CAN	Fast Ethernet
Max length of the bus [m]	1000	1000	100
Max speed for this length	fixed 25 kB/s	6.25 kB/s	12500 kB/s
Maximum number of nodes on a bus	64	32	2 (no repeater)
Relative price per a node	low	medium	high
Standard in the lighting domain	yes	no	yes (e.g. Artnet)

Tab. 4.1: Comparison of communication protocols

From a quick study of the table one can deduce, that not a single protocol mentioned in this document can fulfill the requirements of the specification. Therefore a subnetting is needed and an appropriate topology must me chosen, which will be discussed in the following section.

4.3 Choosing the topology

Our network's goal is to deliver mainly streamed data to a lot of devices. The network should allow also at least a half-duplex communication with each end device, to be able to transfer some diagnostic data (voltage levels, temperature, etc.) and to update the devices via a bootloader, nevertheless the majority of the data will be sent from control PC to the end device. Another goal is to respect or make use of the physical circular topology of the auditorium.

To reach these goals, Fast Ethernet was chosen as the backbone protocol with a circular physical topology. To interface the backbone networks to the following subnetwork, Ethernet-to-X converters will be used. The X stands for a network protocol(s), which will be chosen in the next section and which will interconnect this converter and the end devices. Since Fast Ethernet will not directly interconnect many devices, but only several Ethernet-to-X converters, the total price of this solution will not be high even if relative price per a single connected node is higher compared to DMX512 and CAN.

What's more, there is a protocol in the family of Ethernet related protocols, which will handle the redundancy of the circular topology of the backbone network. This protocol is called the Spanning Tree Protocol or STP and it will be present at each Ethernet-to-X converters, since each of them will contain a 3-port embedded switch (such as KSZ8863 from Micrel).

Moreover, Ethernet is suitable for a real time communication using for example UDP/IP or EtherCAT on one hand and also for diagnostic data and bootloading using the TCP/IP on the other hand.

Another advantage of this solution is, that there is no need for any other converters to connect a standard PC to this network to control the lights, because Fast Ethernet network card is present in all today's PCs.

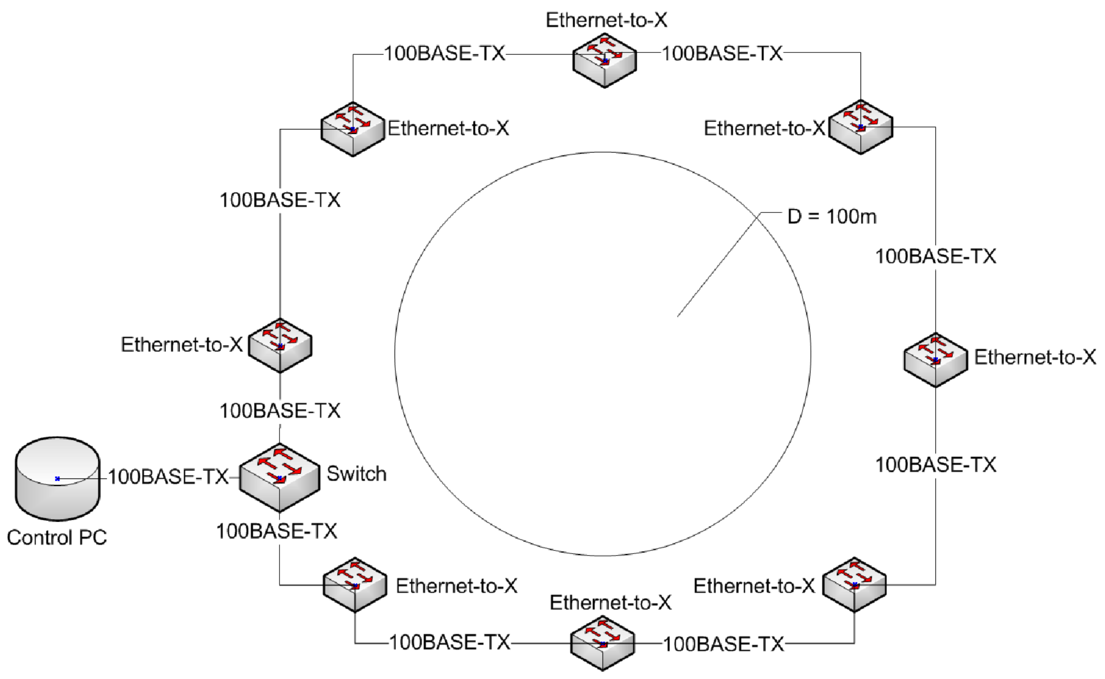


Fig. 4.2: Backbone network

4.4 Designing last mile bus

The goal of this section is to design the bus, which will interconnect an Ethernet-to-X converter and end devices. There will be mainly streamed data, which will need to be delivered in time. If an error occurs durring the delivery, there should be no retransmission, it is better to latch the old data until the new data arrive successfully. These characterstics are met with User Datagram Protocol or UDP. What's more, there is a UDP based lighting protocol called Artnet, which is a standard DMX512 encapsulation for IPv4 networks. Therefore, DMX512 will be used in the last mile bus. Nevertheless this is not the only reason to use DMX512: If one compares DMX512 with another candidate, e. g. CAN bus, the advantages main advantages are:

- It is standard in the lighting industry, which means, that theoretically any other lighting end device may be connected to DMX512 network, compared to a CAN bus network.
- DMX512 was designed for streamed simplex communication and its timing is tuned up to allow successful delivery over higher distances at a constant bus

Pin	Name	Purpose
1	DMX+	DMX512 positive or ”A” wire
2	DMX-	DMX512 negative or ”B” wire
3	RS485+	RS485 half-duplex positive wire
4	12V	Control logic alimentation
5	12V	Control logic alimentation
6	RS485-	RS485 half-duplex negative wire
7	GND	Common ground
8	GND	Common ground

Tab. 4.2: DMX512 RJ45 modified pinout

speed (refresh rate of the lights may be therefore constant).

- There are no retransmissions if a packet gets lost, the bus functionality is determined only by the right timing of each packet.
- There is a standardized wiring using RJ45 connectors and Cat5 cable, which is ubiquitous and therefore cheap.

Nevertheless there is a problem, when one wants a half-duplex communication to be run over the same bus, since DMX512 is simplex only. The solution is to use spare twisted pairs in the Cat5 cable and to add another RS485 half-duplex bus to it.

What’s more, this half-duplex communication bus may be used as a secondary DMX512 bus, if the primary DMX512 bus fails to allow the dimmers to work. Such a network collision would be then detected and technical support would be called to fix it, nonetheless the dimmers could continue to work in an ”emergency state” with no other diagnostic data available.

To finish, Cat5 cable can also be used to power the control electronic of each end device. This is not a neglectable advantage, since the power for the LEDs will go most likely from 36 V to 48 V and it is indeed expensive to transform such a high voltage to a standard 3.3 V or 5 V logic alimentation voltage level to power the control logic.

4.5 Mutual galvanic isolation of the network devices

To avoid, that a single failing light on the bus of 64 end devices causes the whole bus to fail, it is crucial to split the bus in several electrically isolated segments. In such case, if a single device fails, only lights which are not galvanically isolated may fail as well, but other lights will continue to work. In an ideal situation, each light fixture

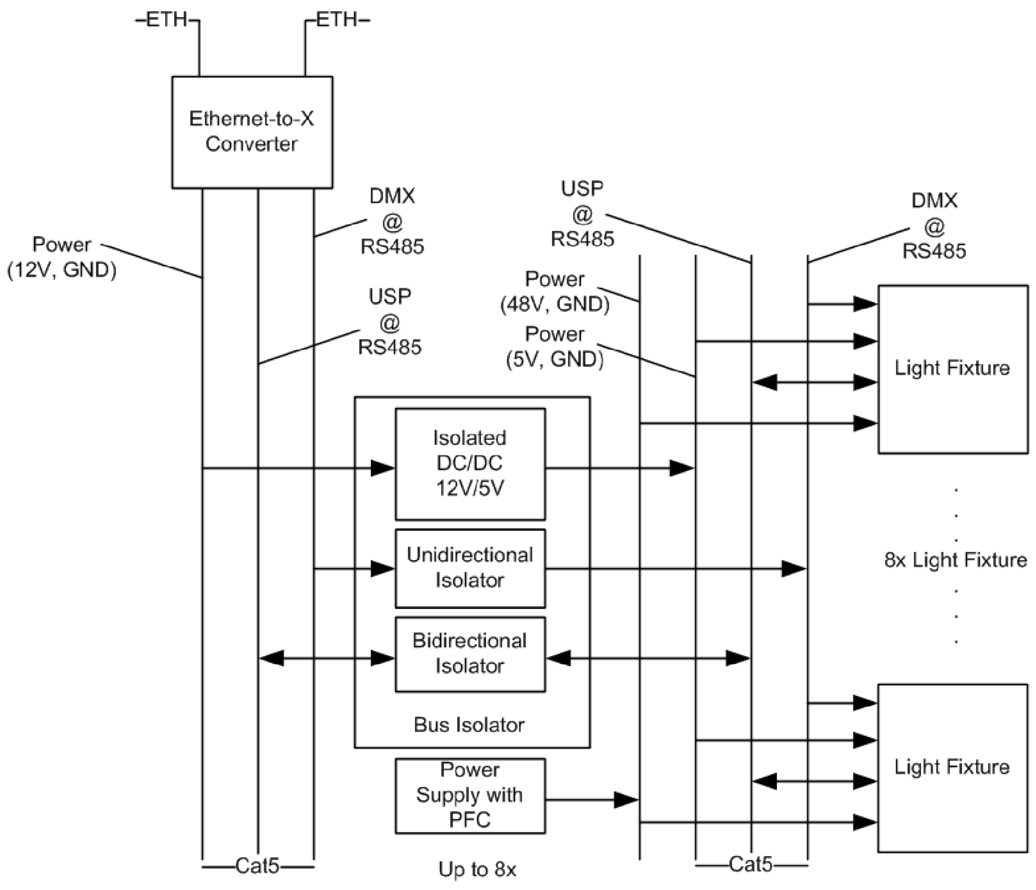


Fig. 4.3: Last Mile Bus

would be isolated from another, but this would increase considerably the costs of the whole system. (Separated power source for each fixture, isolation of RS485 at each fixture etc.) Therefore a compromise was made and the bus was segmented in segments of 8 fixtures mutually galvanically connected and powered from the same power source. The situation is explained on the figure 4.3. From the figure, one can deduce, that there can be up to 64 light fixtures organized in 8 galvanically isolated segments.

4.6 Conclusion

To sum up, there will be four different devices in the whole network:

4.6.1 Ethernet switch

This device may be inserted anywhere in the ring backbone bus. The PC controlling the whole system will be connected to the network through it.

4.6.2 Ethernet-to-X converter

This converter will contain an on board 3-port switch for 100BASE-TX Ethernet and an microcontroller with TCP/UDP/IP stack. This microcontroller will realize the Artnet to DMX512 translation and will act also as a USP TCP/IP to RS485 gateway. USP[23] is a propriatary communication protocol running on RS485. This unit will have a separate alimentation and will be galvanically isolated one from another (Ethernet is isolated). Since all light fixtures will be connected to the bus through an isolation, this unit will also be isolated from the light fixtures.

4.6.3 Last mile bus isolator

This simple device will act as a repeater for DMX512 and USP. There will be also an isolated DC/DC step down power supply, that will transform 12 V to 5 V to power the control electronic of the light fixtures.

4.6.4 Light fixture

This is the device, which will be further discussed in the following chapter. It contains DMX512 receiver and USP node implementation running on a microcontroller along with a voltage level and temperature sensing to provide basic diagnostics. It controls a LED driver using 16 bit PWM.

5 END DEVICE

5.1 Overall description of function

The end device (a.k.a light fixture) receives the dimming data over DMX512 which runs on RS485. The RS485 signals must be first converted to 3.3V logic using RS485 receiver and are consequently received by a microcontroller, which contains the control firmware. The microcontroller then produces a PWM signal for each color (red, green, blue and white), which is proportional to the luminosity of concerned color. If needed by the LED used in the circuit, a linearization of the DMX channel data to luminosity characteristic may be made by the microcontroller. For modification of each parameter in the system, an EEPROM communicating with the microcontroller via I2C is introduced as well. The PWM signal produced by the MCU is filtered by an RC filter and then output to the power board, which contains a current source integrated circuit, which in analog implements a high-speed (˜600 kHz) PWM to control the power LEDs. All LEDs are placed in series, so shunt protectors are used to bypass groups of LEDs in case of a failure in order to enable the remaining LEDs of the color in question to continue to work. An NTC thermistor is soldered close to the LEDs to enable the control board MCU to monitor the temperature (for protection and failure diagnostics). The MCU monitors also the power board alimentation voltage level and its own alimentation, which is supplied via the last-mile bus. The alimentation of the control logic is converted from last-mile bus 5V to 3.3V by an LDO. The signals, which go from the power board to the control board (temperature and power board alimentation monitoring) are not directly connected to the input of MCU's A/D converter, but they are instead connected to and operation amplifier, which acts as a buffer. The reason for this is to protect the MCU and not to overpass the maximum impedance of the signal source, which is allowed by the A/D converter.

Until now, the second RS485 bus was not mentioned, let's fix this: This second RS485 bus is used to reprogram the on board microcontroller after a bootloader was programed to it in production. This may be useful if for example another power board is to be used with the microcontroller and the linearization data must be changed or the processing of the temperature and voltage level monitoring must be changed. Another use of the bus is for system diagnostics and central monitoring. The last-mile bus master may collect the diagnostic data and generate and email describing which unit has a problem, such email would be then typically sent to the system administrator. The diagnostics may be also used for light state logging in order to tell the system administrator, that the life cycle of a light (LED) was reached and that the performance may drop and an replacement is suggested. The last purpose of this bus is to work as a backup DMX bus in case the main DMX bus gets somehow broken. The protocol used on the physical layer represented by the RS485 is the USP protocol, which specification may be found on the web site of Ultman Inc. or in the firmware source code. All this allows me to say, that the control board will be an universal DMX controller, which may be used with a multitude of light front ends.

5.2 Control Board

5.2.1 Selected microcontroller

The crucial requirements on the MCU were to contain at least two UART interfaces and to be able to generate at least four 16bit PWM outputs in hardware. Other parameters were the price and availability for the potential mass production. After a research, the PIC32MX110F016B from Microchip Technology was selected. To be precise, the exact part number is PIC32MX110F016B-I/ML with -I/ML meaning a QFN28 package and industrial temperature range. Between the main benefits of this MCU belongs the remappable pins, which enables the PCB designer to modify the position of peripherals inputs and outputs. This feature simplifies considerably the whole design, if it is wisely used. Let's sum up the main characteristics of this microcontroler in a table:

Parameter	Value	Parameter	Value
Architecture	MIPS32® M4K®	I^2C	2
Program Memory	16 kB + 3 kB	DMA	4
Data Memory	4 kB	RTC	Yes
16bit Timers	5	Package	QFN28
UARTs	2	A/D Converter	9ch, 10 bit, 1 Msps
SPI	2	Price	approx. 1.82$/pcs

Tab. 5.1: PIC32MX110F016B characteristics [13]

The description of each part of the microcontroller would be indeed exhaustive, since it's datasheet has something aroung 328 pages and there are also many application notes dedicated to each of its peripherals. This subsection therefore gives only the most important aspects, which were used for the selection.

5.2.2 TIA-485 transmitter and receiver

DMX512 RS485 receiver

Since DMX512 is a simplex communication protocol only a RS485 receiver is needed to connect the light fixture to the bus. I have selected MAX3280E from Maxim, since it has a fail-safe operation support (in case the RS485 bus is shorted or disconnected a logic 1 output is guaranteed) and only 1/4-unit-load input circuitry, which multiplies by four the maximum number of nodes on the same bus. Another advantage of this integrated circuit is its small footprint: It is available in SOT23-5 package, which is handy for the PCB layout. Again, let's sum up these parameters in a table:

Parameter	Value
Supply voltage	3 - 5.5 V
Maximum number of nodes	128
Bus load	1/4 of nominal load
Package	SOT23-5
ESD protection	$^{+}$-15 kV
Price	approx. 1.18$/pcs

Tab. 5.2: MAX3280E characteristics[11]

USP RS485 transceiver

As mentioned in the beginning of this chapter, the second RS485 bus is used for a half-duplex diagnostic communication and so a RS485 transceiver is necessary to connect a node to the bus. After a research, I selected ST1480AC from STMicroelectronics. Since the demands on this transceiver are not significant in this projects, because the bus is segmented and there are not many nodes on a single isolated bus (up to 8), the price and availability were main parameters when selecting this device. This however does not mean, that this device represents the low end between the RS485 transceivers. Let's see its parameters in a table:

Other benefits of this integrated circuit are for example that it has the Industry Standard 75176 pinout, which alows it to be directly replaced by many other RS485 drivers if this one gets obsoled or sold out for a longer amount of time. When working in the receiver mode, its operation is also fail-safe as it is for the MAX3280E.

Parameter	Value
Supply voltage	3 - 7 V (max)
Maximum number of nodes	64
Bus load	1/2 of nominal load
Package	SO8 (SOIC8)
ESD protection	$^{+}$-15 kV
Driver output protection	overcurrent, thermal
Price	approx. 1.62$/pcs

Tab. 5.3: ST1480AC characteristics[20]

5.2.3 Power management

The power management for this application was not as evident as it seemed for me to be. My first approach was to use a DC/DC converter from 48 V, which is the supply voltage for the power board, but this is not possible, if we want to stay in an acceptable interval of price. Another approach would be to reduce the supply voltage of the power board to something like 36 V, but the price for this would be the reduction of the number of series LEDs and since this device purpose is lighting, it is not a good way to go.

After a reflection, the solution to have a single DC/DC converter at the bus segment isolator was selected and thus the end device receives the alimentation for the control logic directly from the bus. The voltage of 5 V was selected as sufficient, which allows the bus segment with 8 end devices to be up to around 30 meters long if the consumption of a single device control logic is around 80 mA and an LDO with 200 mV dropout voltage is selected. Nontheless, let's calculate the maximum length of the bus for the dropout voltage of 300 mV to get the worst-case figure:

$$l = \frac{U_{bus} - 3.3 - U_{drop}}{\rho \cdot I_{device} \cdot N} = \frac{5 - 3.3 - 0.3}{0.094 \cdot 0.080 \cdot 8} = 23.3\, m \tag{5.1}$$

Where U_{bus} is the voltage supplied by the bus, U_{drop} is the maximum dropout voltage of used LDO, ρ is the resistivity of two pairs of Cat5 cable, I_{device} is the average current of a single device control electronic and finally N is the number of devices on the same bus segment.

Furthermore, if the bus alimentation is placed in the middle of the segment, which is possible in our configuration, the total length of the bus can be the double, so up to 50meters approximately, which is sufficient.

As for the LDO, I have selected the MCP1824T-3302E from Microchip Technology, which has the maximal dropout voltage of 320 mV, the typical value is 200 mV. The following table describes the selected fixed voltage LDO.

Parameter	Value
Dropout voltage	200 mV(typ)/320 mV(max)
Maximum input voltage	6 V
Maximum continuous output current	300 mA
Maximum pulsed output current	500 mA
Package	SOT23-5
Price	approx. 0.50$/pcs

Tab. 5.4: MCP1824T-3302E LDO characteristics[12]

5.3 Power Board

Since the power part of the device will require a heatsink on the bottom side of the printed circuit board and since I wanted to conserve the modularity of the whole solution, LEDs and LEDs drivers are situated on a separate PCB, which is interconnected with the control board via a 10-pin connector. This design will facilitate the hardware debugging, since the control board and the power board can be tested and developped separately.

5.3.1 LED driver

The selected LED driver is from Diodes Inc. and its name is ZXLD1356. It is a continuous mode inductive step-down converter designed for driving LEDs connected in series. It can operate from an input voltage of up to 60 V and can provide an adjustable current of up to 550 mA.

Let's describe briefly the principle of operation of this integrated circuit, whis is depicted on the figure 5.1:

On power up, the Q_1 FET is disconnected and no current is flowing across the LEDs and a constant Adj voltage appears on the non-inverting input of the first comparator. This situation makes the inverting input of the same comparator to be grounded and thus a logic level 1 appears on its output. This turns on the Q_1 FET, which makes the current start to flow throught the R_s resistor, LEDs and L_1. This makes the drop voltage on the resitor rise and the FET Q_2 is getting open more and more. This forces more current to flow through R_1, R_2 and R_3/Q_3 branch and the voltage level at the inverting input of the first comparator is rising. When the comparation level is reached, the Q_1 is disconnected and current continues to flow through D_1 back to the source. This produces a falling ramp at the inverting input of the first comparator and when the comparation level is reached again, the cycle repeats. Resistor R_3 along with Q_3 work as a 15 % hystheresis. To adjust

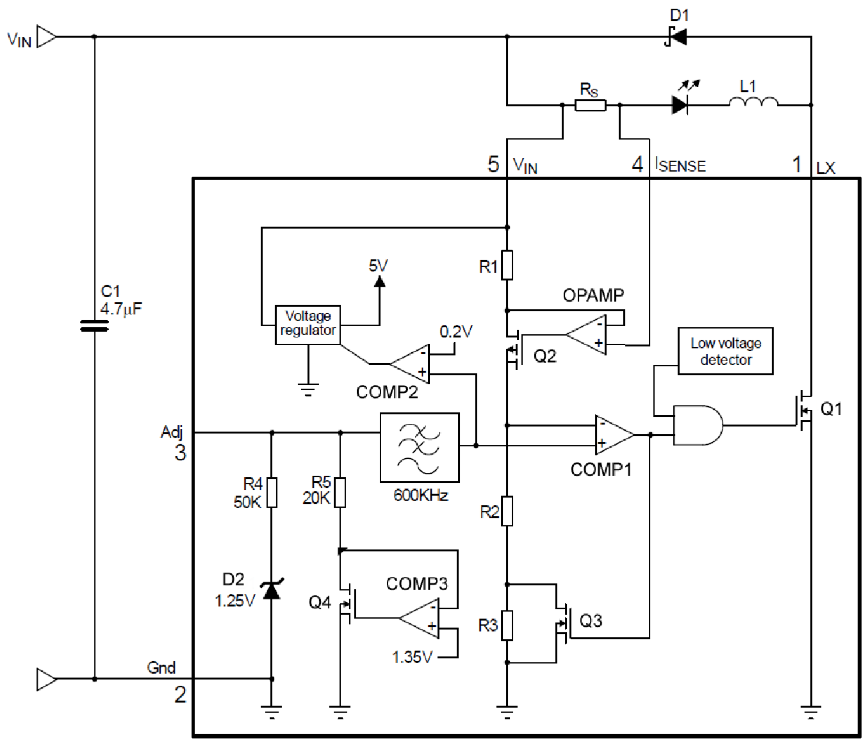


Fig. 5.1: ZXLD1356 reference design/principle of operation[7]

the average current flowing through the LEDs, the voltage level at Adj pin can be changed from 0 to 2.5 V. The switching frequency of this circuit depends on the value of L_1, supply voltage and the number of series LEDs and is given in the datasheet. For our circuit, it should be close to 300 kHz and will differ for each color. (Refer to the Power Board schematics for details)

5.3.2 Selected LEDs

The selected LEDs are LRTB C9TP from Osram for the RGB LEDs and XBDAWT-00-0000-00000LBE7 from Cree Inc. for the white LEDs. During the selection the main aspect was the optical efficiency. The LEDs parameters and U_{ADJ} vs. luminosity curve will be measured to make the linearization in firmware. This will be described in the following chapter.

6 FIRMWARE

6.1 Used development environment

In order to implement the firmware part of the functionality described on the page 46 of this document, Microchip MPLAB IDE v8.91 was used along with the C30 proprietary Microchip compiler. There also exists a newer version of the IDE called MPLABX, nevertheless the v8.91 was more throughroully tested and used by Ultman Inc. in the past, so I have selected this IDE. Optimization level used by the firmware was set to -Os, since for cost minimization, a microcontroller with relatively small flash memory (16 kB) for such an application was selected.

6.2 Analytic design of the application

Before writing a line of source code, I have performed an analytic and modular design of my firmware application. Of course, this should be the most important part of every firmware writting, but it is often skipped by many programmers. First of all, I have split my application in modules (named module.c and module.h). Each of these modules consist of an abstraction layer which handles certain part of the application. In order to interact with other modules, selected functions and global variables are exported in the .h file of the module. As an example of a modular design, the USP and DMX protocol implementation can be analyzed: DMX and USP modules are in my application though of as application layer protocols and thus they interact with a lower level module called RS485, which handles the basic serial data inputs and outputs. Furthermore, USP and DMX modules both use the main module, which implements a milisecond system timer, which is consequently used to handle communication timeouts.

To grasp better the idea of modularity, please observe the following include diagram:

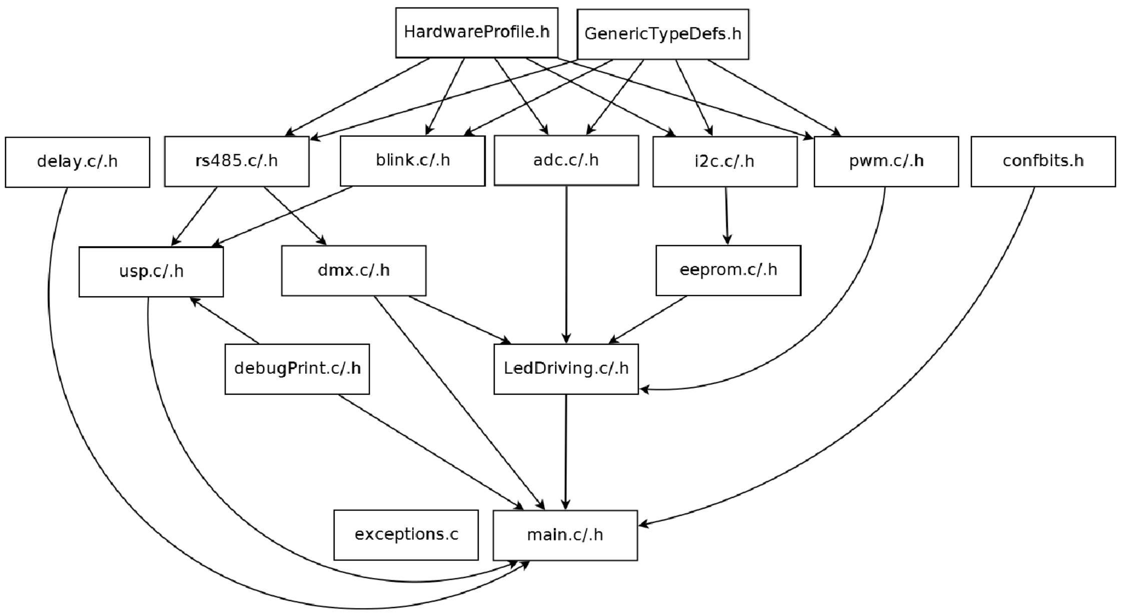


Fig. 6.1: Module overall include diagram

6.3 Description of important modules

The overall include diagram was created manually in the open-source software called Dia, but to be more practical, I have used another open-source software called Doxygen to generate such include and call diagrams for me automatically using special tags in the source code, which also serve as the code documentation. Pictures used in this section are automatically generated by Doxygen software. Interactive documentation generated by Doxygen of the entire project is available on the CD-ROM attached to this document. If you open the source codes in any text editor, you will find Doxygen tags such as the following example:

```
/*********************************************************//**
 * @brief       This function is used to output txbuff to I2C
 * @param[in]   uint8_t* txbuff  buffer to send
 * @param[in]   uint8_t addr  remote device address
 * @param[in]   uint16_t len      length of the tx buffer
 * @return      int  0 on success, -1 if sending failed
 **********************************************************/
int I2CWriteBuffer(uint8_t* txbuff, uint8_t addr, uint16_t len)
{
  ...
```

6.3.1 Module main.c/.h

This module contains the C language entry point, which is a line, where the program starts right after the stack definition and some low level core initialization. In the function main(), all necessary initialization is done and then the program enters an infinite loop, where it calls state machine functions of the DMX module, LedDriving module and USP module.

Moreover, this module contains firmware revision definition and functions to determine the current system time in miliseconds and to calculate a time difference between a time stamp and current system time.

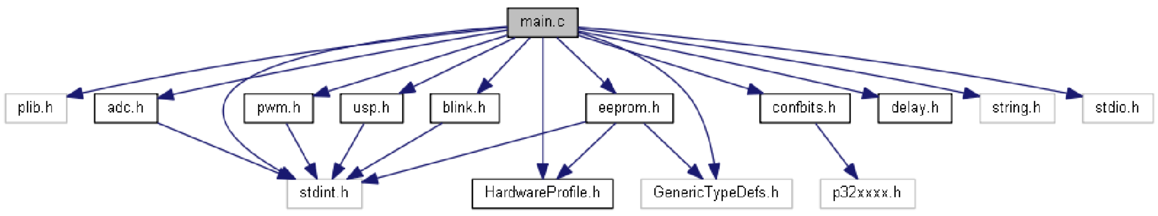


Fig. 6.2: main.c/.h include diagram

6.3.2 Module dmx.c/.h

The module DMX contains an implementation of a DMX receiver. The number of consecutive channels to receive is defined to be 4, but this can be easily changed in dmx.h if necessary. The address of the DMX receiver is set at runtime by writing to the exported variable DMXaddr. In order to use this module, after the low-level init, call the function DMXInit() to initialize necessary hardware used by the DMX and then call periodically in a non-interrupt context the function DMXProcess(). Everything else concerning the DMX receiver functionality is handled inside the module (detection of the break signal, setting the level of luminosity of each channel, timeout handling etc.).

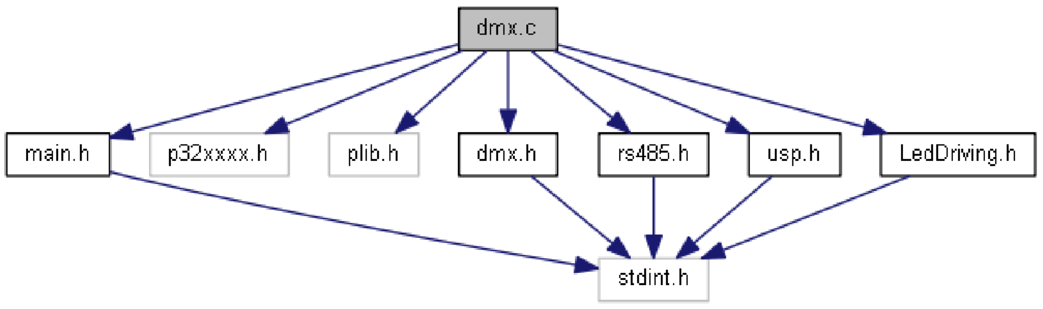


Fig. 6.3: dmx.c/.h include diagram

6.3.3 Module usp.c/.h

The module USP implements a slave node of the USP protocol based network. Detail description of USP protocol is available on the website of Ultman Inc.[23]. In usp.h, there is a register map defining each diagnostic command available. This protocol is also used to program the on board EEPROM with the luminosity vs. control word look-up table, which is used to linearize the controlled light characteristic if necessary.

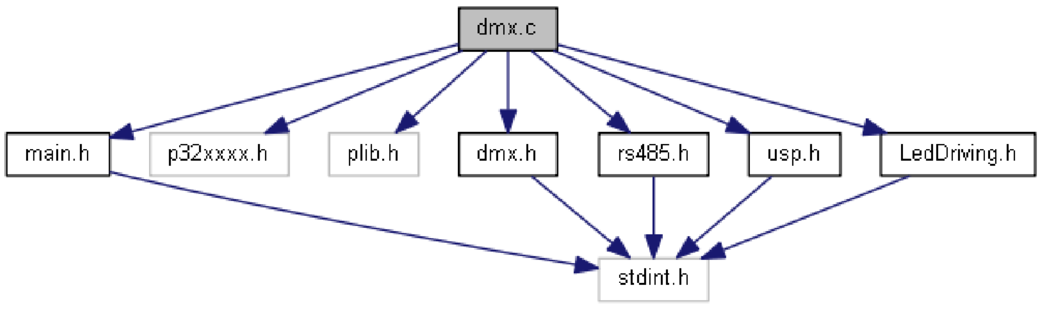


Fig. 6.4: usp.c/.h include diagram

6.3.4 Module LedDriving.c/.h

This module implements functions to set and to get current luminosity level in relative unit with the step of 1/255 and the overtemperature protection which shuts down the lights if the measured temperature goes higher than a defined level. After the power module cools down, the last written luminosity level is set again. Function LEDDrivingCheckTemperature() which handles the temperature protection is to be called periodically in the main loop of the application.

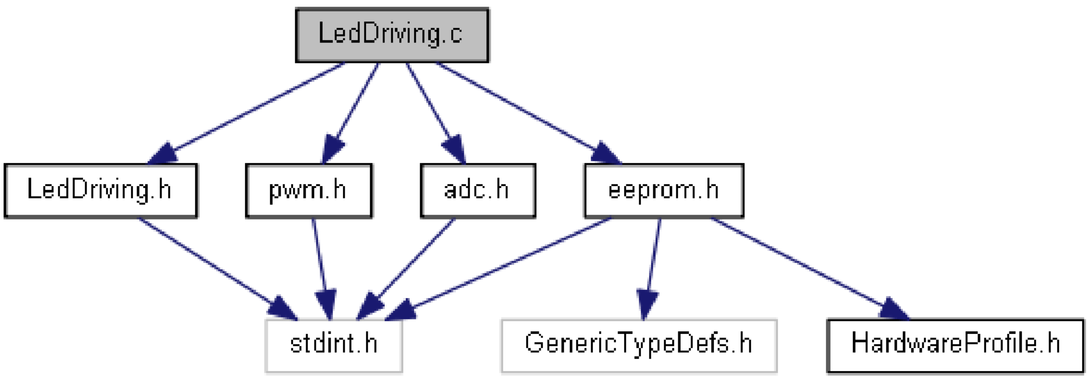


Fig. 6.5: LedDriving.c/.h include diagram

7 LED LINEARIZATION

7.1 Luminosity data transfer chain

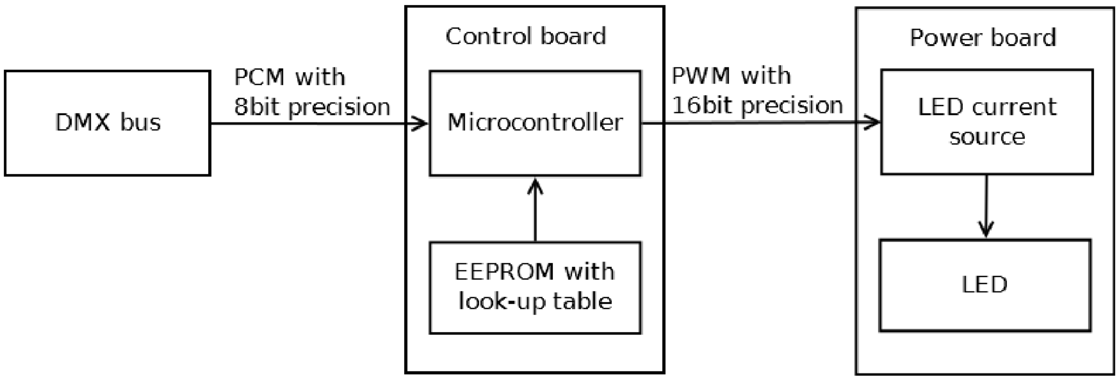


Fig. 7.1: Luminosity information system propagation diagram (for each color)

To facilitate the design of the power board and especially considering non-linearities introduced by the non-constant input impedance of the current source integrated circuit control pin (refer to the description of ZXLD1356), the resolution of the PWM signal serving as interface between the control board and the power board was selected higher than the resolution of the PCM data received via the DMX512 bus. This configuration enables the control board to adapt to a lot of possible modifications concerning the power board such as the replacement of selected LEDs or even replacement of selected current source. What's more, any non-linearities may be compensated by rewriting corresponding look-up table constants in the on board EEPROM using the USP protocol and the GUI, which will be described later on.

7.2 Used measuring hardware and software

As the linearization method consist of finding 16 bit values of PWM control word corresponding to all of 256 possible PCM data from the DMX bus. Several thousand measurements would have to be done. This could indeed be done manually, but it would be extremely impractical. Moreover, the loading process needs to be automated as well, since even having every 1024 constants ($256[16\mathit{bit\ constants}] \cdot 4[\mathit{colors}] = 1024$), it would be very long to load them one by one to each device. This process will be briefly discussed in this section.

7.2.1 Used luxmeter

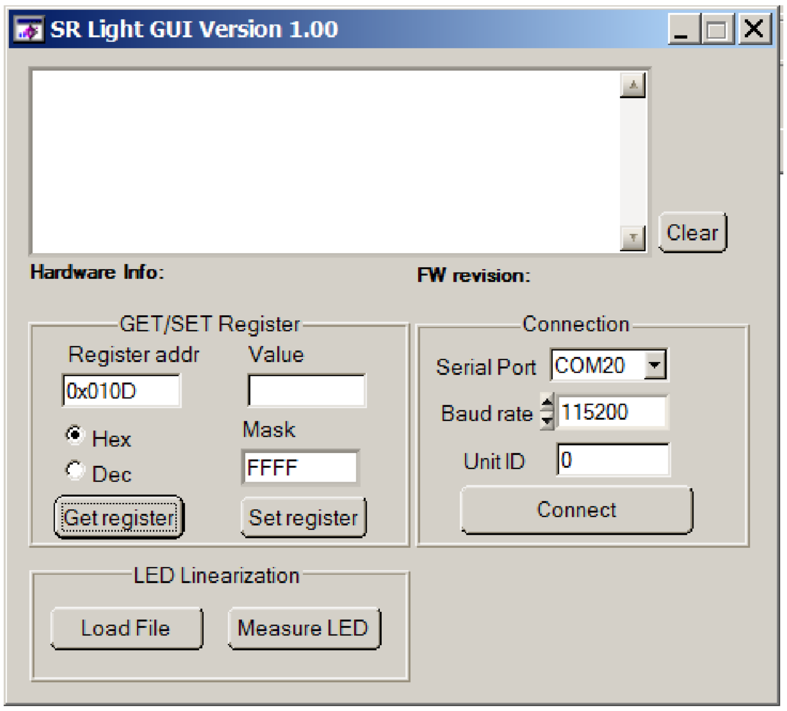
The luxmeter used for our measurements is called Vega and it is from Ophir Optics. The instrument consist of a main platform and of a set of sensors. During the measurements, I have used a non-selective high-bandwidth sensor, since for our purposes, we do not need the exact value of the optical power, but only a relative value normalized by the maximum attained value per each color.

To be able to automate the process, I have used the RS232 interface of Vega power meter and manually setup the instrument before the measurements to auto-scale with the measured value. In my calibration software, I only needed to send "$SP\r" and the power meter sent me a measured value in watts in the following format: "* 1.3E-003", which means 1.3 mW.

Fig. 7.2: Vega power meter[14]

7.2.2 Used GUI

The linearization GUI is based on a LabWindows project written in C called USP Debug, which is used to communicate with any device implementing the USP communication protocol. I have modified this GUI and added two panels to allow first the measurement of the characteristics and second data mining of the look-up table from these characteristics and its loading to the control board EEPROM.



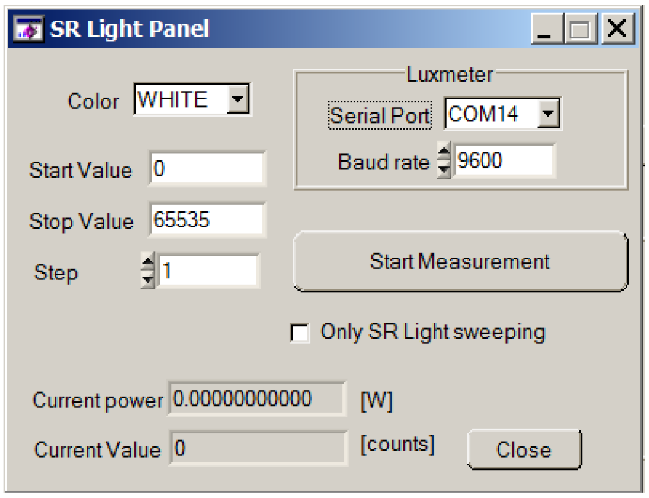
Hardware Info:

FW revision:

Register addr: 0x010D
Value:
[✓ Hex] Dec
Mask: FFFF

Serial Port: COM20
Baud rate: 115200
Unit ID: 0

Fig. 7.3: SR Light GUI - Main panel



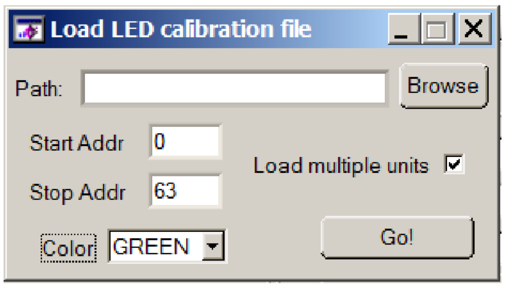
Color: WHITE
Start Value: 0
Stop Value: 65535
Step: 1

Serial Port: COM14
Baud rate: 9600

☐ Only SR Light sweeping

Current power: 0.00000000000 [W]
Current Value: 0 [counts]

Fig. 7.4: SR Light GUI - Measurement panel



Start Addr: 0
Stop Addr: 63
Load multiple units: ✓
Color: GREEN

Fig. 7.5: SR Light GUI - Loading panel

7.3 Results of measurements

7.3.1 Data mining

As mentioned above, since the measurements of the optical power are done only to linearize the relation between the input 8 bit PCM data from DMX and the resulting optical power, we are not really interested in the value of the power itself, but rather in the normalized value going from 0.0 to 1.0. Having done this normalization, the program selects the PWM 16 bit constant corresponding to 0/255th, 1/255th, 2/255th up to 255/255th value of the optical power measured. These constants are consequently loaded to a range of devices as you can see in the figure of the loading panel of the GUI.

7.3.2 Graphs

In the figure above, the orange line represents the tangent to the measured curve (blue) at point [1000, 0.18]. There are two major non-linearities: First occurs at the point [0,0] where there is an initial offset and the second occurs at the point [2550, 0.46], where there is a significant diminution of the measured curve slope.

Graphs for the other colors (red, green and blue) look very similar, so it seems like the non-linearities are inserted by the control input of the current source.

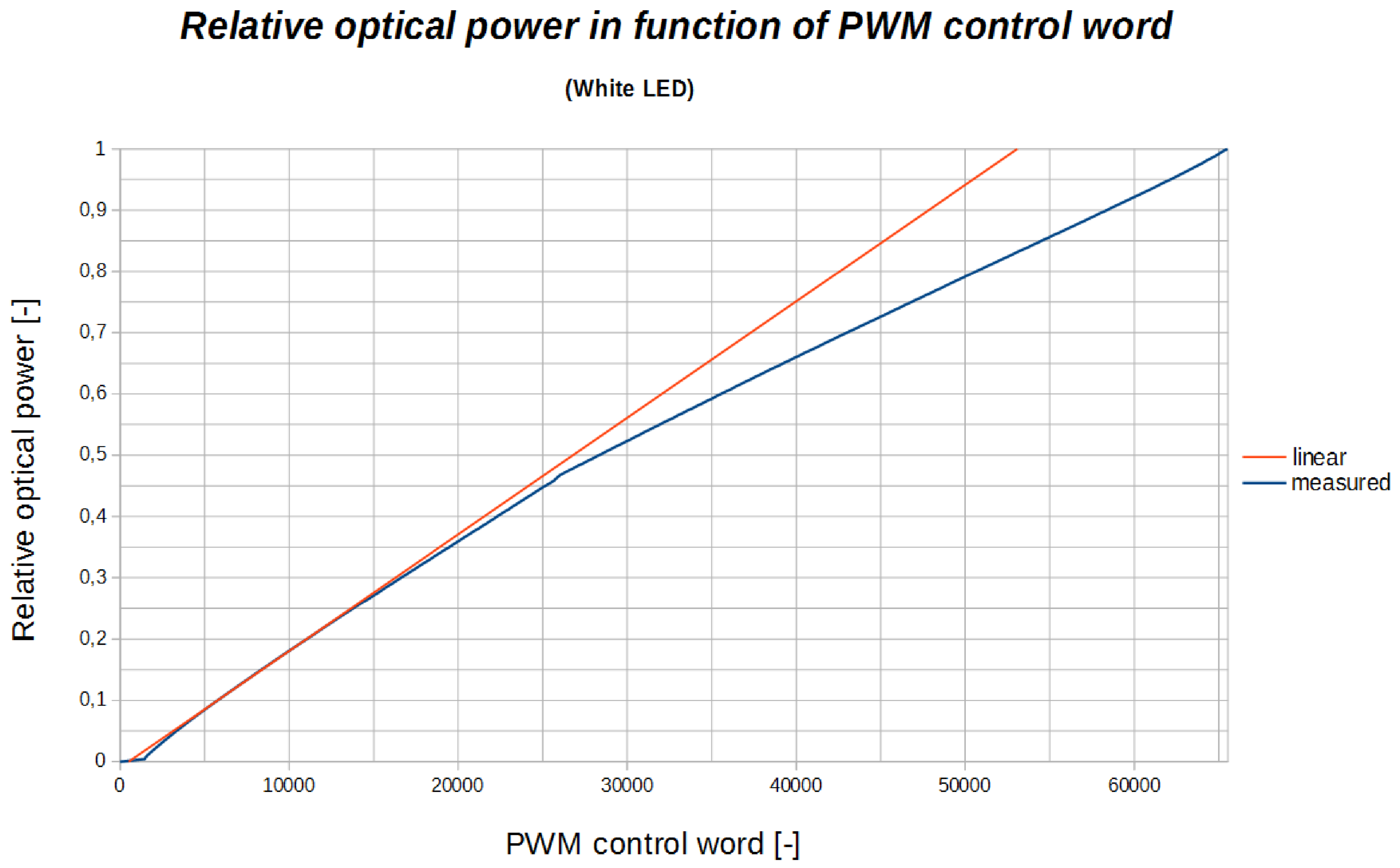


Fig. 7.6: Relative optical power in function of PWM control word (white LED)

8 DEVICE TESTING

8.1 Overtemperature testing

To test the feature implemented in the LedDriving.c/.h module, I have set the luminosity of all four colors to its maximum value and measured the temperature via the on-board termistor using USP protocol and SR Light GUI described above.

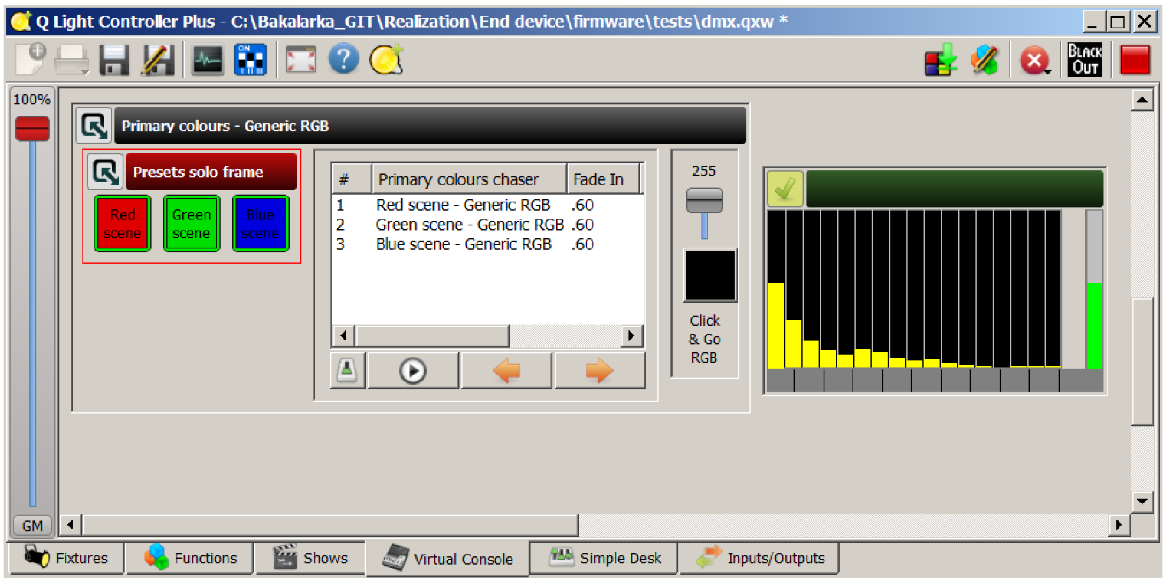
First of all, I have disabled this protection and tried to reach a maximum temperature reading out manually the current value of the temperature. When the active cooling was not enabled, the temperature growing stopped at about 65 °C. When it was enabled, the temperature stabilized at about 38 °C. Then, I have enabled the protection feature with the maximum allowed temperature set to 55 °C and tried to turn on and off the cooling fan. The protection worked as expected. The cooldown temperature was set to 40 °C to avoid intermittent blinking of a light unit in case of bad cooling.

8.2 DMX testing and demonstration application

For overall device testing and to prove its compatibility with the standard DMX512 lighting protocol, I have established a RS485 link between a notebook running QLC+ open source DMX generation software and the prototype of the end device. QLC+ software alows its user to create a simple GUI environment asociating each button, slider, checkbox and other GUI elements a predefined function. If a user needs an advanced function to be executed upon a button press, he can program its own function in javascript, which can be consequently linked with the button.

As for the demonstration application, I have used a predefined FFT component, which processes the sound recorded by the computer microphone and triggers an event (similar to a button press) if the sound level in a certain frequency band exceeds a selectable trigger value.

The demonstration application is attached on the included CD and may be simply opened by the QLC+ software, which can be found at http://qlcplus.sourceforge.net/.



#	Primary colours chaser	Fade In
1	Red scene - Generic RGB	.60
2	Green scene - Generic RGB	.60
3	Blue scene - Generic RGB	.60

Fig. 8.1: QLC+ and demonstration application

9 CONCLUSION

In this bachelor project, several industrial protocols were considered such as CAN, 100BASE-TX, DMX512, LIN and EtherCAT, from which three were chosen for a further description and a finer selection of the right one for the implementation of the resulting topology. The design of the end device was done not only by considering the electrical requirements, but also practical requirements such as availability of parts, the possibility of a drop-in replacement and the price. The graphical user interface written in LabWindows/CVI was used to measure the relative optical power produced by a LED as a function of the PWM control word. The data were later used to linearize the relation between the input PCM data received via the DMX512 protocol and the desired output luminosity.

Designing the PCB in Eagle for both control board and test power board have improved my skills in this domain, which I will indeed use in my future projects.

Moreover, the creation of the firmware for used microcontroller, which consisted of a analytic design of functional modules followed by their implementation in C language, have tested my capabilities in firmware programming.

Furthermore, the implementation of the automation GUI have improved my skills in the domain of GUI programming and measurement automation.

Finally, a demonstration application was created in QLC+ in order to test the whole unit in a real operating conditions.

To conclude, I hope that I will be able to continue this project and implement the entire system including the Ethernet-to-DMX converters. I would like to see it one day fully operational and fulfilling its intended purpose.

BIBLIOGRAPHY

[1] Ad-net. EPON Architecture, 2005. Available from: `http://www.ad-net.com.tw/index.php?id=594`.

[2] Zdeněk Bartoň. Analyzátor protokolu DMX512 s připojením na USB rozhraní. Technical report, Brno University of Technology, 2008.

[3] Chris Smith CANGuru. CAN Bus 9 Pin D-Sub Pinout, 2008.

[4] CiA GmbH. CAN history, 2007. Available from: `http://www.can-cia.de/index.php?id=161`.

[5] Steve Corrigan. Controller Area Network Physical Layer Requirements. Technical Report January, Texas Instruments, 2008.

[6] Steve Corrigan. Introduction to the Controller Area Network (CAN). Technical Report August 2002, Texas Instruments, 2008.

[7] Diodes Incorporated. ZXLD1356 Datasheet. Technical Report October, Diodes Incorporated, 2012.

[8] Elation Professional. *DMX 101 : A DMX 512 HANDBOOK*, volume 20. 2008.

[9] Petr Grygárek. Architektury komunikujících systémů, 2005.

[10] Ujjal Kar. The DMX512 Packet, 2010. Available from: `http://www.dmx512-online.com/about.html`.

[11] Maxim Integrated. MAX3280 Datasheet. Technical Report December, Maxim Integrated, 2013.

[12] Microchip Technology. MSP1825 LDO Regulator Datasheet. Technical report, Microchip Technology, 2007.

[13] Microchip Technology. PIC32MX1XX/2XX Datasheet. Technical report, Microchip Technology, 2012.

[14] Ophir photonics. Power meter - Vega. Technical report, 2014. Available from: `http://www.ophiropt.com/laser/pdf/vega_catalog.pdf`.

[15] Pat Richards. A CAN Physical Layer Discussion. Technical report, Microchip Technology, 2002.

[16] Freescale Semiconductor. DMX512 Protocol Implementation Using MC9S08GT60 8-Bit MCU. Technical report, 2006.

[17] Prashant Sharma. OSI Seven Layer Model, 2009.

[18] Joseph Spring. Direct Link Networks. Available from: `http://www.mathcs.emory.edu/~cheung/Courses/455/Syllabus/2-physical/1-Others/CNPA09.pdf`.

[19] Charles E. Spurgeon. *Ethernet: The Definitive Guide*. 2000.

[20] STMicroelectronics. ST1480AB Datasheet. Technical Report July, 2013.

[21] Youxian Sun. *Control and Scheduling Codesign: Flexible Resource Management in Real-Time Control Systems*. Springer, 2008.

[22] Texas Instruments. Isolated CAN Transceiver - ISO1050. Technical Report June 2009, Texas Instruments, Dallas, 2009.

[23] Ultman Inc. USP protocol specification, 2007. Available from: `http://projects.ultman.com/doku.php?id=usp:usp_spec`.

[24] Conal Watterson. Controller Area Network (CAN) Implementation Guide. Technical report, Analog Devices, 2012.

[25] Wikipedia. Category 5 cable. Available from: `http://en.wikipedia.org/wiki/Category_5_cable`.

[26] Wikipedia. RJ45. Available from: `http://cs.wikipedia.org/wiki/RJ-45`.

[quoted on 13/12/2013]

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

DMX512 Communication protocol used in lightening domain.

MBd, kBd Megabaud, kilobaud, unit of communication speed. 1Bd is one symbol per second.

TIA-485, RS485 Communication protocol using twisted copper pair and differential signaling.

UTP Unshielded Twisted Pair.

CAN Controller Area Network, networking protocol interconnecting multiple control units.

CSMA/BA Bus access method used by CAN.

EMC Electro-Magnetic Compatibility.

FIFO First-In First-Out, queuing technique.

UART Universal Asynchronous Receiver Transmitter.

100BASE-TX Fast Ethernet standard.

T568A, T568B Wiring standard for Cat5 cabling using RJ45 connectors

CSMA/CD Bus access method used by Ethernet.

MAC Media Access Control(-er).

IPv4, IPv6 Internet Protocol version 4(6).

EtherCAT Networking standard running on layer 3 of the ISO/OSI model.

TCP, UDP Networking standard running on layer 4 of the ISO/OSI model.

Artnet Application layer networking protocol allowing to send DMX512 frames over Ethernet.

PWM Pulse-Wide Modulation.

USP Ultman Serial Protocol, proprietary layer 2 communication protocol.

MCU Microcontroller Unit.

EEPROM	Electrically Erasable Programable Read Only Memory.
NTC	Negative Temperature Coefficient, thermistor type.
PCB	Printed Circuit Board.
ESD	Electro-Static Discharge.
LDO	Low Drop voltage regulator.
FET	Field-Effect Transistor.
RGB LED	Red Green Blue Light Emitting Diode.
IDE	Integrated Development Environment
CVI	C (language) for Virtual Instrumentation
QLC+	DMX lighting data generation software

10 APPENDICES

10.1 Contents of the included DVD

The included DVD contains schematics and the layout of the control board and the power board, installation of the calibration software, doxygen documentation for the firmware of the on board PIC32, demonstration application, which can be opened in the QLC+, which is also included and finally, a demostration video with pictures of the final device.

10.2 Schematics, layout and pictures of assembled device

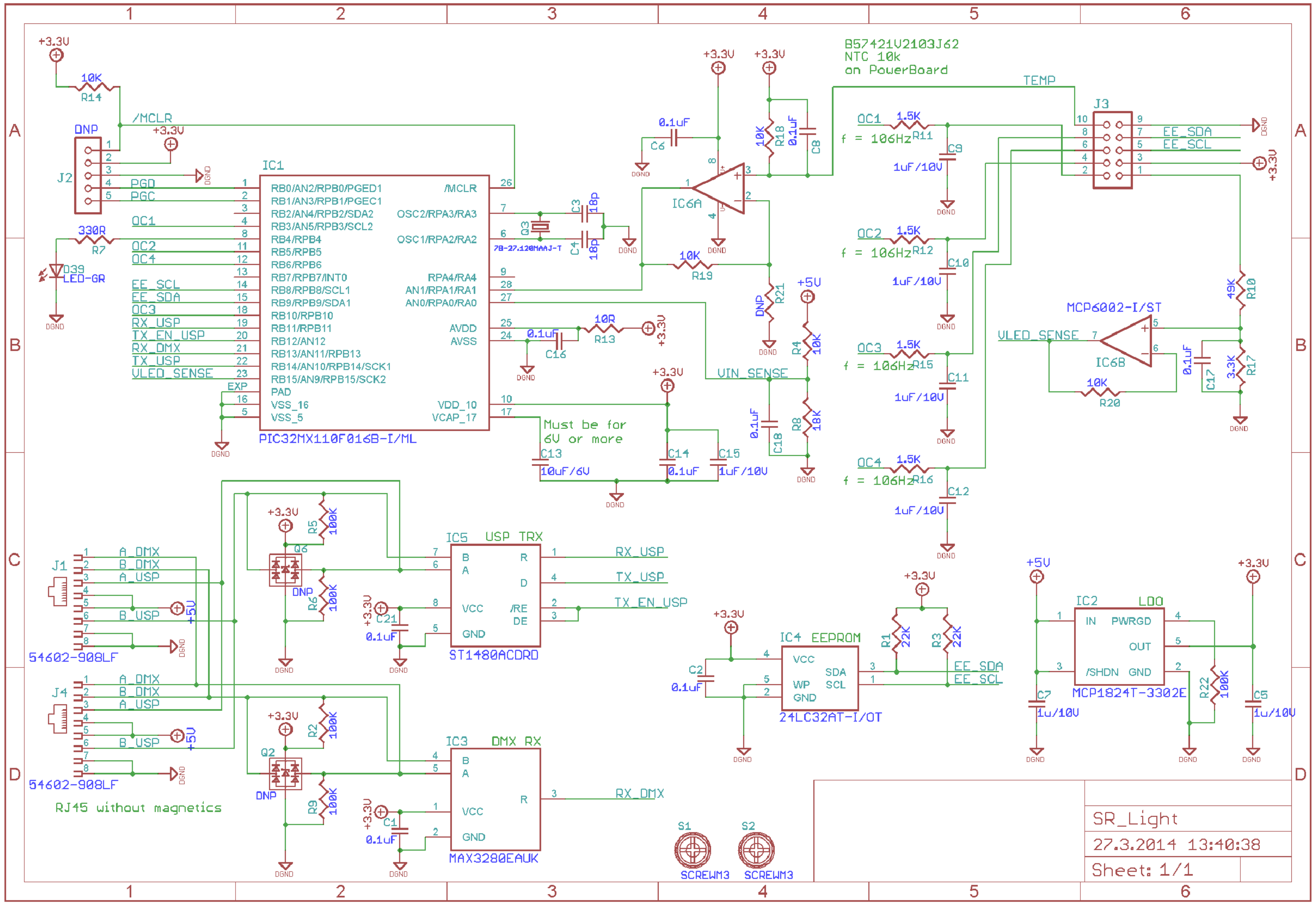


Figure 10.1: Schematics of the Control Board

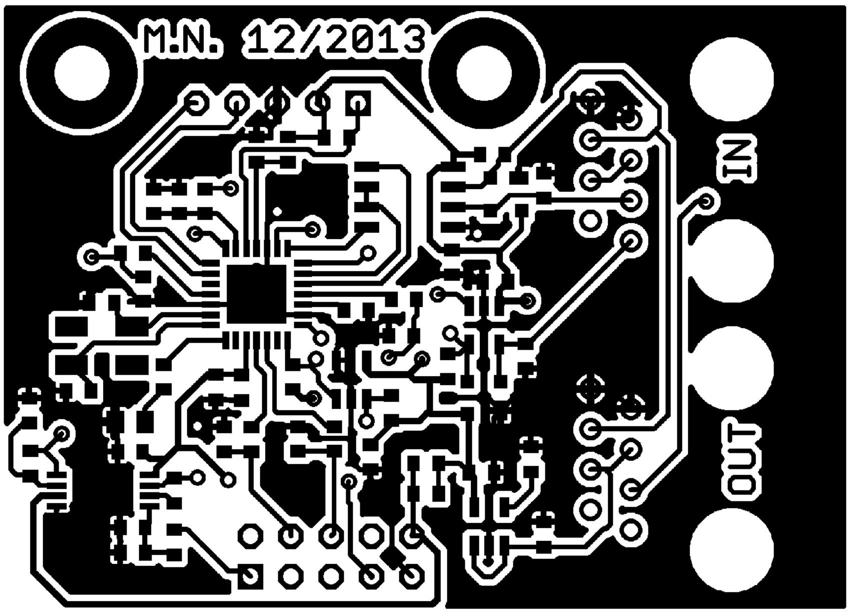


Figure 10.2: PCB Control Board - Top (1:2 ratio)

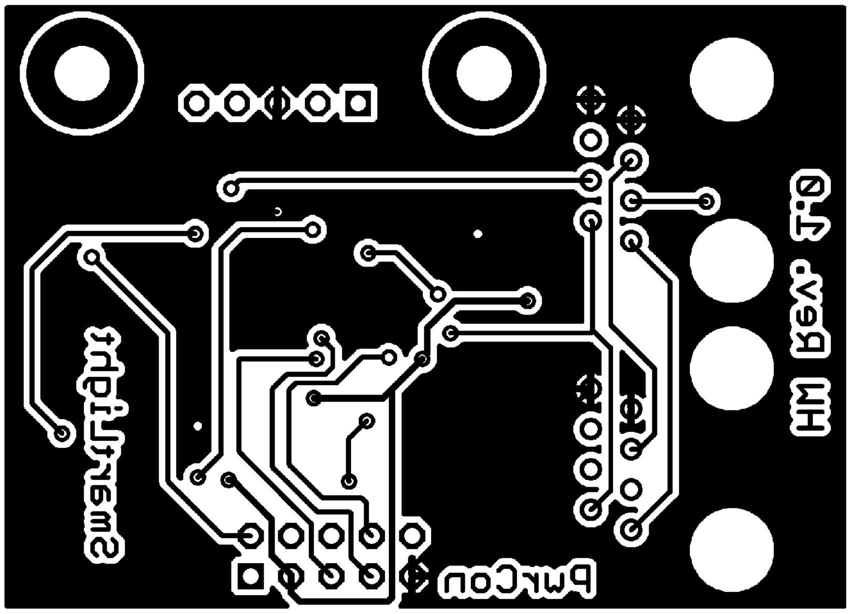


Figure 10.3: PCB Control Board - Bottom (1:2 ratio)

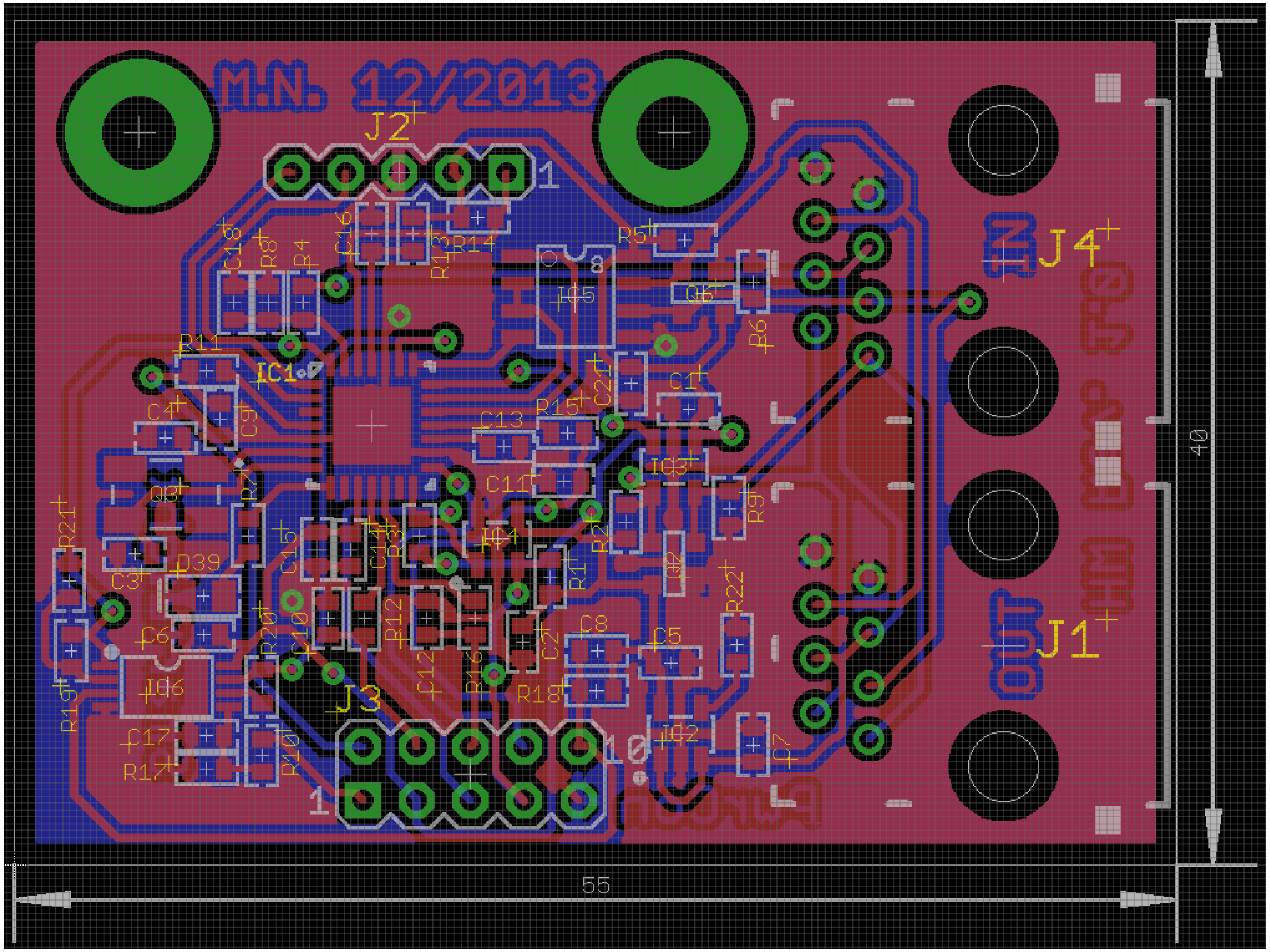


Figure 10.4: PCB Control Board - Assembly

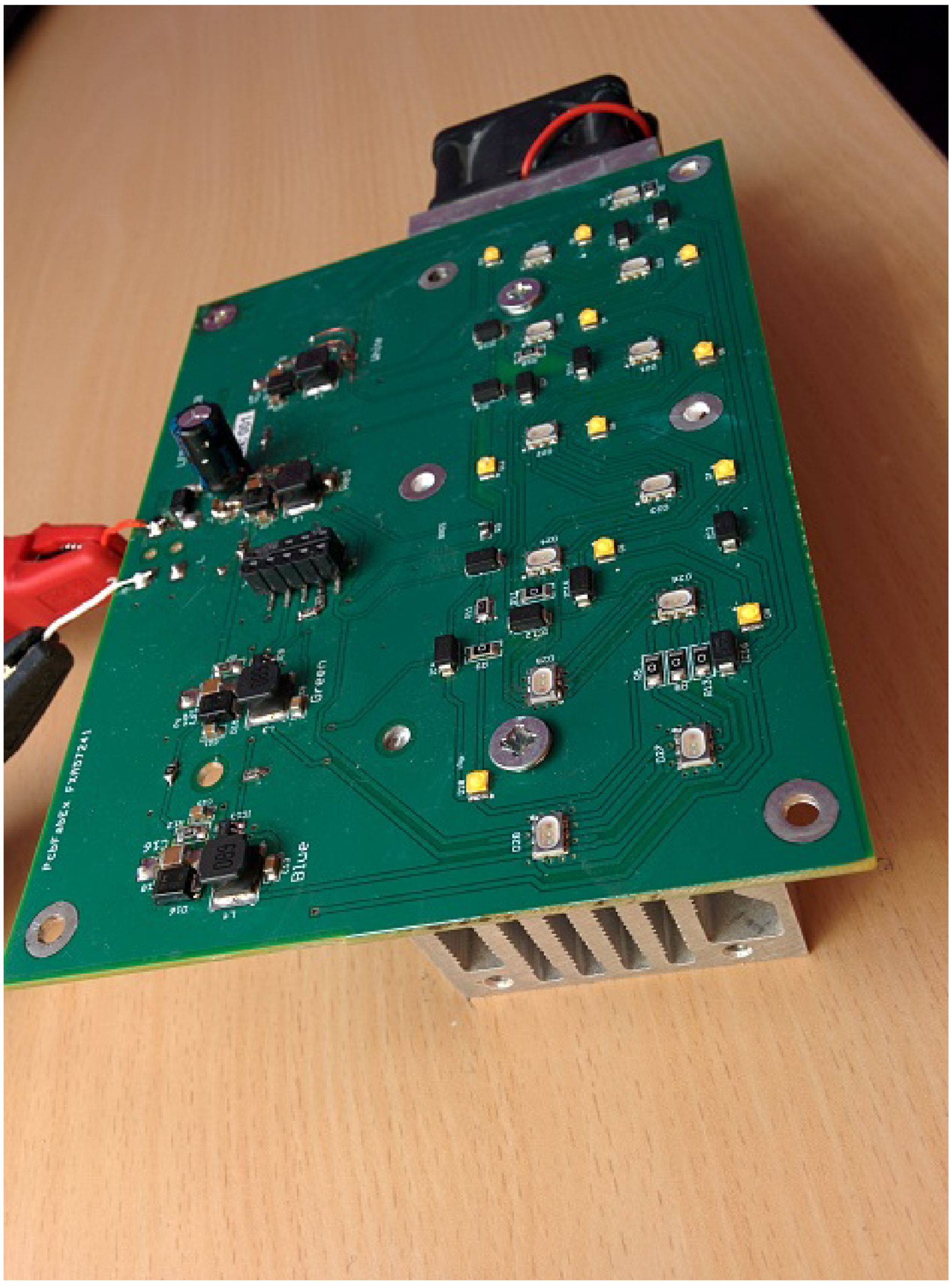


Figure 10.5: The Power Board (for testing purposes)

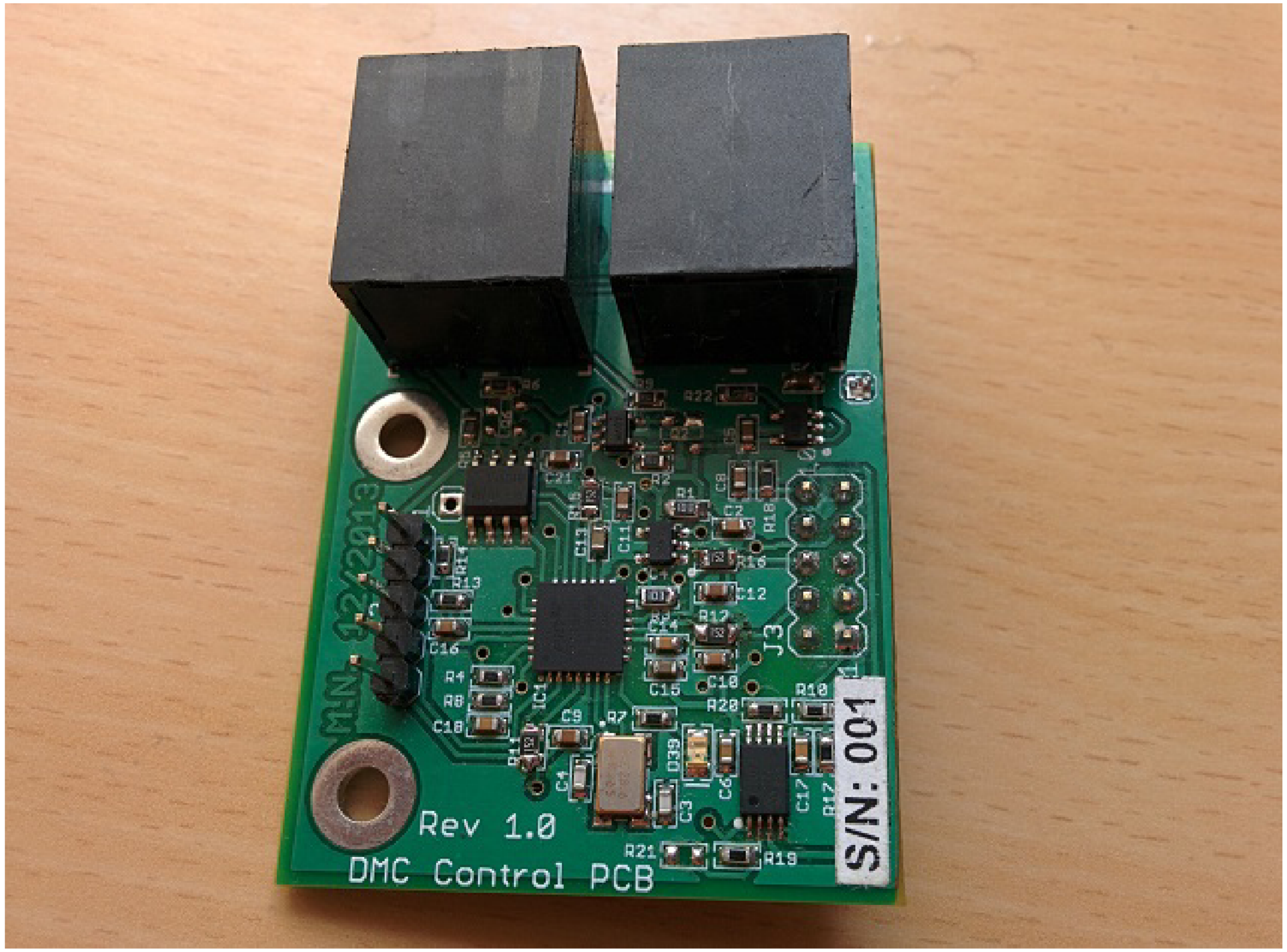


Figure 10.6: The Control Board

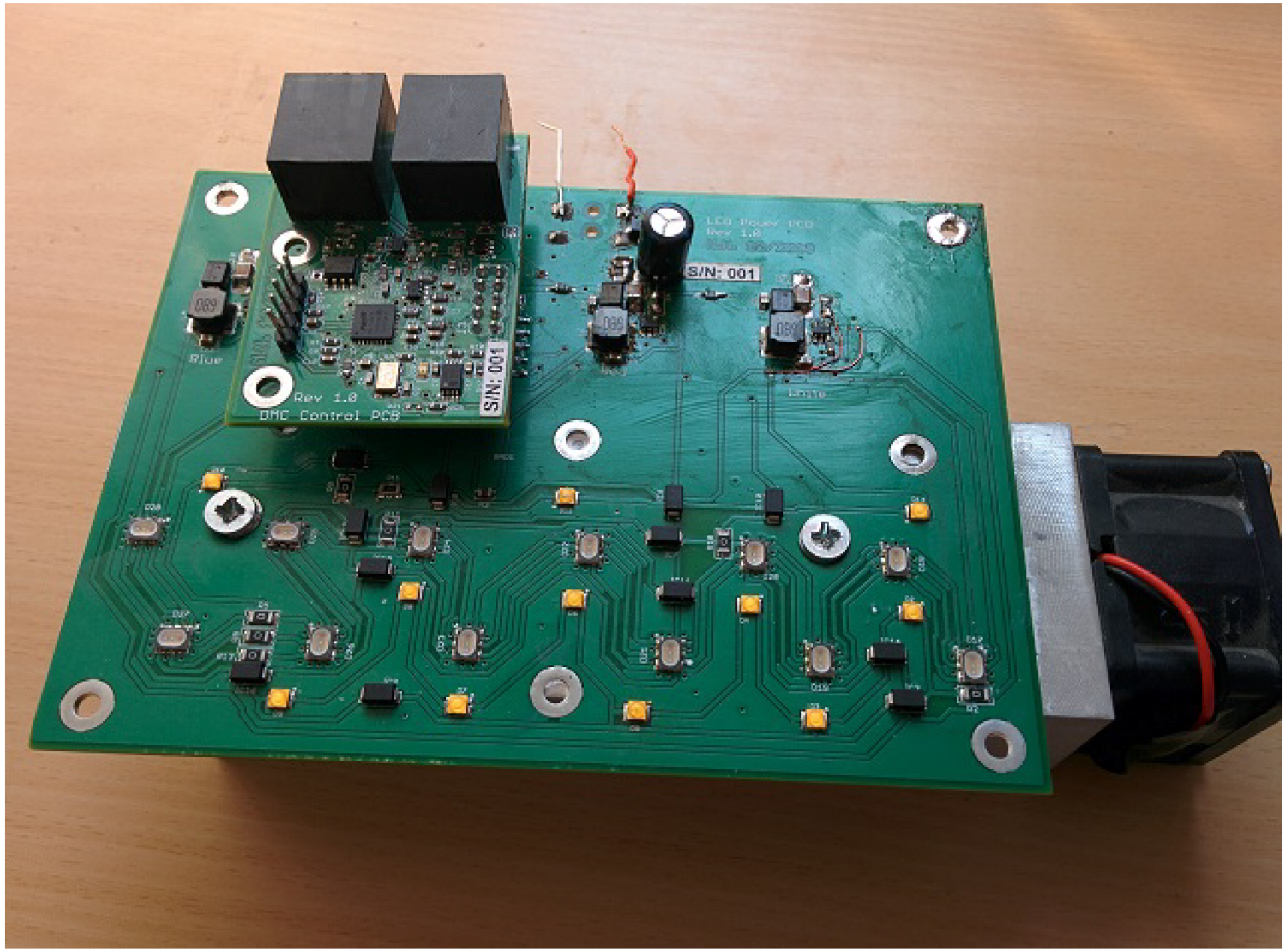


Figure 10.7: Both PCBs connected together