Maximilian W. Gotthardt

Evaluation of protocols for control stage lighting

Bachelor Thesis in Computer Engineering

27 March 2022

Please cite as:
Maximilian W. Gotthardt, “Evaluation of protocols for control stage lighting,” Bachelor Thesis (Bachelorarbeit), Institute of Telecommunication Systems, Technische Universität Berlin, Germany, March 2022.

TKN Telecommunication Networks Group

Fachgebiet Telekommunikationsnetze
Technische Universität Berlin, Germany

Einsteinufer 25 · 10587 Berlin · Germany

https://www.tkn.tu-berlin.de/

Evaluation of protocols for control stage lighting

Bachelor Thesis in Computer Engineering

vorgelegt von

Maximilian W. Gotthardt

geb. am 13. July 1993
in Berlin

angefertigt in der Fachgruppe

Fachgebiet Telekommunikationsnetze

Institut für Telekommunikationsnetze
Technische Universität Berlin

Betreuer: **Anatolij Zubow**
Gutachter: **Falko Dressler**
Thomas Sikora

Abgabe der Arbeit: **27. März 2022**

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde.
Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

(Maximilian W. Gotthardt)
Berlin, den 27 March 2022

Abstract

In the field of lighting and stage technology, the challenge of controlling the individual installations, quickly and without complications is a recurring one. Established solutions are realized via cables using the DMX-512a protocol. However, due to the progress in radio technology wireless solutions based on IEEE 801.11 are becoming more and more common. But with these it is a challenge to send control signals, to many individual stations and to update them multiple times a second. Additionally, a low latency and, of course, a certain reliability must be guaranteed. This is challenged because at typical locations, such as a concert, there are often very busy frequency bands. This thesis therefore examines an approach attempting to reduce a large part of the overhead, by having the stations communicate with each other on an Ad-Hoc network at the data link layer, in order to develop an improved wireless protocol. Different approaches with broadcast and unicast are evaluated analytically and experimentally.

It can be shown that the overhead has a great influence on the transmission and that it is particularly extreme, because so many individual stations are often only controlled with very short control signals. The results show that the gain in throughput means that packets can be sent redundantly, which also leads to improvements in reliability.

Kurzfassung

Im Bereich der Licht- und Bühnentechnik stellt sich immer wieder die Herausforderung, die einzelnen Anlagen schnell und unkompliziert anzusteuern. Etablierte Lösungen werden über Kabel mit dem DMX-512a Protokoll realisiert. Durchundersant die Fortschritte in der Funktechnik setzen sich aber immer mehr drahtlose Lösungen, hauptsächlich auf Basis von IEEE 801.11 durch.

Allerdings ist es eine Herausforderung, Steuersignale an viele einzelne Stationen zu senden und dies mehrmals pro Sekunde. Außerdem muss eine geringe Latenzzeit und natürlich eine gewisse Zuverlässigkeit gewährleistet sein. Dies wird dadurch erschwert, dass an typischen Standorten, wie z.B. einem Konzert, die Frequenzbänder oft stark ausgelastet sind. In dieser Arbeit wird ein Ansatz untersucht, der versucht, einen großen Teil des Overheads zu reduzieren, indem die Stationen über ein Ad-Hoc-Netz auf der Datenverbindungsschicht miteinander kommunizieren, um ein verbessertes Protokoll zu entwickeln, welches möglichst geringen Einfluss auf das Medium hat. Verschiedene Ansätze mit Broadcast und Unicast werden analytisch und experimentell bewertet.

Es zeigt sich, dass der Overhead einen großen Einfluss auf die Übertragung hat und dass im Fall der Lichttechnik-Steursignale besonders extrem ausfällt, weil viele Einzelstationen oft nur mit sehr kurzen Steuersignalen angesteuert werden. Die Ergebnisse zeigen, dass ein Gewinn an Durchsatz bedeuten kann, dass Pakete redundant gesendet werden können, was wiederum zu einer Verbesserung der Zuverlässigkeit führt.

Contents

Chapter 1

Introduction

One part of lighting technology is about controlling lights or installations to match a stage for a given event. These installations can be moving heads, strobes or even fog machines. Wireless networks in general are becoming increasingly popular, also in the field of lighting technology, here they are more flexible and also more cost-effective than wired solutions. However, they also have their own limitations, in this thesis, protocols are introduced to address some of these.

Motivation and Requirements

In order to improve the properties of radio networks, a logic link layer approach is chosen in this thesis, to distribute the control signals to the individual stations, This promises better performance due to reduced overhead.

The properties of the wired DMX-512a protocol solution are taken as a baseline. The packets are to be distributed to all stations with an equally large update frequency, the latency between sending the control signal and switching the light should be as short as possible, the stations should control the lights synchronously and the reliability should still be as high as possible. The range should be at least 100 m and the hardware used should be as cost-effective as possible.

Challenges

There will always be a possibility to build an even more reliable system with extremely complex hardware, however, the focus of this thesis is to achieve results with low-cost hardware that can be compared to the existing DMX-512a solution. The ESP-NOW protocol used here by the manufacturer Espressif is unfortunately proprietary and had additionally to be investigated due to the partly inaccurate or outdated

documentation. Also, the complete control process finds place over microcontrollers, which have to be programmed in a fundamentally different way than ordinary programs.

Contribution

In this thesis a number of protocols are being developed on the data link layer/application layer, which offer promising improvements. It is shown that broadcasts are preferable to unicasts and also become significantly more robust through adaptations. These were investigated simulatively and experimentally and compared with existing solutions. The test suite used on the test hardware is publicly available on GitHub, as well as the collected measurement data. [1]

update link, tkn gitlab link...

Thesis Outline

The thesis is divided into three main chapters. Chapter 3 explains the 802.11 specifications family, the physical and the data link layer. Existing light protocols are explained in order to better understand the new protocols that will be introduced later, in addition, the EPS32 hardware and the associated ESP-NOW protocol are briefly shown.

Chapter 4 then specifies the various protocols developed and makes analytical assumptions. And it is explained how to work with the hardware used.

Chapter 5 then explains the methodology, how the measurement data was collected with a technical implementation. Wireshark's measurements are analyzed and used to better understand the protocols. Afterwards, the protocol data is further evaluated and compared with the analytical approaches and with each other.

[1] `www.github.com`

Chapter 2

Related Work

The trend towards wireless solutions in all areas also affects stage technology. However, wireless solutions often have lower throughput, latency and reliability than their wired counterparts.

Cost-effective solutions work with the 802.11 protocol, which is widely used. H. Kareem and D. Dunaev [1] observed that *the recently growing demand for the control and automation of a wide variety of devices and gadgets has led to a rapid expansion in embedded systems market*. When selecting the platform for the experiments in this thesis, the ESP32 was chosen, which was compared analytically along with Arduino embedded systems from the two, with the result, *to highlight the advantages of the ESP32 in designing embedded systems compared to similar boards*.

Due to the fact that many individual stations are controlled in stage lighting, often with very short signals, this can sometimes lead to an extreme overhead. *Overhead is* according to Y. Xiao [2] *one of the fundamental problems of MAC inefficiency, and it includes MAC header, frame check sequence, and physical header.* But also interframe spaces can be classified as overhead.

The impact of transmission overhead is also researched by Nurul Sarkar [3] he pointed out the DCF inefficiency and developed a protocol that *is a simple packet scheduling mechanism that can be used to reduce transmission overheads of DCF and to improve the performance. The key idea is to create a temporary buffer unit at the MAC layer for each active connection on the network where multiple packets are accumulated and combined into a single large packet*. In this thesis broadcasts are used to address multiple stations in order to reduce overhead compared to multiple unicast transmissions. Nurul Sarkar [3] also took a look into *its impact on throughput for a single-user wireless ad hoc network*, which were used in this thesis.

Addressing to multiple stations is often realized with multicasts, because, unlike broadcasts, acknowledgements can be sent. One approach to this is presented by Pablo Salvador et. al. [4] they experimentally investigate the IEEE 802.11aa GATS

and test block acknowledgements. Also unsolicited retries where researched by them: *the idea is to improve reliability with a very somple scheme, wich does not requre a closed loop between the sender and the receivers(s), and therefore the price to pay is efficiency*. A method that was also used in this thesis to improve reliability.

Other approaches use PCF to transmit real-time traffic for video or audio transmissions, for example, Mihaela et al. [5] use these *since it is the most efective scenario for video transport over 802.11 WLANS*. But they also say that simple *retransmission is not very effective in combating long burst of packet losses*, which are circumvented in this thesis by delayed repetition.

Chapter 3

Fundamentals

In this chapter explains the fundamentals required for understanding the different approaches in this thesis. This contains basic knowledge of the physical- and data link layer, which are located in the first and second layer of the Open Systems Interconnection (OSI) Model.

It also explains what types of transmissions exist, introduces the hardware used in this thesis and explains the ESP-NOW protocol running on this hardware.

Application layer
Presentation layer
Session layer
Network layer
Data Link layer
Physical layer

Table 3.1 – OSI model

3.1 IEEE 802.11 Protocol

The Institut of Electrical and Electronics Engineers (IEEE) 802 is a family of standards dealing with area networks of different kinds.

- 802.11 Wireless Local Area Network (WLAN)
- 802.15.1 Wireless Personal Area Network (WPAN)
- 802.15.4 Low-rate WPAN (LR-WPAN)
- 802.16 Wireless metropolitan area network (WMAN)

For this thesis is the focus set to the 802.11, because of the accessability and wide functionality. There are two Basic Service Set (BSS) defined:

- Infrastructure BSS
  A central element manages the network and all the traffic goes through. Every Station (STA) must always communicate via the Access Point (AP) and never directly - exceptional: Direct Link Mode. An initial association must take place to use the Infrastructure BSS.

- Independent BSS
  A network without a central station, where the network topology can flexibly change over time. The communication happens directly between the Wireless Endsystems. Efficent routing can become a problem in more complex topologys.

The most common use in 802.11 is the Infrastructure mode, which is commonly used in office and home enviroments.

Physical layer

In this thesis a brief look into the Physical Network Layer (PHY) of the IEEE 802.11 standard, which is the first layer of the OSI model 3.1, should be taken. This layer provides mechanical, electrical and other functional tools to activate or deactivate physical connections, maintain them and transmit bits over them. These can be, for example, electrical signals, optical signals (fiber optics, lasers) or electromagnetic waves (wireless networks). There are several complements to the 802.11 standard, the most common are:

- 802.11b
  Supports larger bitrates with Direct Sequence Spread Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS) as modulation from 1Mbps to 11Mbbps. It uses the 2.4 GHz ISM band.

- 802.11a and 802.11g
  With Orthogonal Frequency Division Multiplexing (OFDM) data rates is increased by up to 54 Mbps. Where 802.11a is in the 5GHz ISM band 802.11g uses the 2.4GHz ISM band.

- 802.11n
  It also uses OFDM and improves with additionaly Multiple Input-Multiple Output (MIMO), channel bonding and frame aggregation to increase the bandwidth and decrease the overhead. Using 2.4 GHz and 5GHz ISM band.

- 802.11ac
  Support of wider channel and out of it higher bitrates. It also includes features like Multi-User MIMO. It only uses the 5 GHz ISM band.

- 802.11ax
  Like 802.11ac but with additional use of the 6GHz ISM band and better power control. Also called WiFi6.

There are a few more specifications, in this thesis the rather basic 802.11b is used with a transmission rate of 1Mbps.

Data Link Layer

The Data Link Layer (DL) Layer is the second-lowest layer of the OSI Model 3.1 and is split into two sublayers. The Logic Link Control (LLC) sublayer which multiplex protocols over the MAC layer while transmitting and to de-multiplex the protocols while receiving. LLC provides the hop-to-hop flow and error control, allows multi-point communication over networks and adds frame sequence numbers. But thesis focuses on the other data link sublayer.

The Media Access Control (MAC) is the second sublayer and includes network protocols that regulate how multiple computers share the physical transmission medium they use. Without regulation, collisions and data loss would occur in the shared medium if several stations were to transmit simultaneously. The MAC Protocol Data Unit is aditionally added inside the Physical Network Layer (PHY) Payload. It contains the MAC Header and encapsulated in it the MAC Service Data Unit (MSDU).

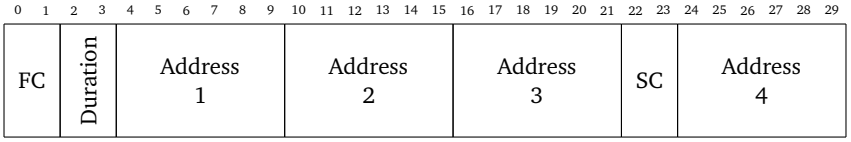

Figure 3.1 – MAC header of a WLAN frame

Payload and FCS are missing

- **Frame Control Field:** Discribes the Type of frame:
  - 00 Manegement Frame
  - 01 Control Frame
  - 10 Data Frame
- **Duration:** Contains the Network Allocation Vector (NAV) value, specifies the transmission time required for the frame. In order to save energy, stations can defer access to the medium for the given duration.
- **Address fields:** Certain address fields are specified by the relative position of the address field. Not every address field is needed by certain frames. Each device is associated with a unique MAC address.
  - Basic Service Set Identifier (BSSID)

  - Source Address
  - Destination Address
  - Transmitting STA Address
  - Receiving STA Address

- **Sequence Control:** Sequence number of the current frame modulo 4096.
- **MAC Payload:** The actual payload information of the MAC layer. The actual payload can differ, because the headers of the LLC and ip etc. has to be subtracted.
- **Frame Check Frequence:** The sender calculates the checksum for the entire data block and appends it to the end of the block.

Carrier Sense Multiple Access/Collision Avoidance

Carrier-sense Multiple Access with Collision Avoidance (CSMA/CA) is composed of:

- **CS (Carrier Sense):** Each station checks whether the medium is free before transmitting
- **MA (Multiple Access)**: Several stations share one medium, which can also be a cable.
- **CD (Collision Detection):** A schedule prevents two stations from starting their transmission at the same time.

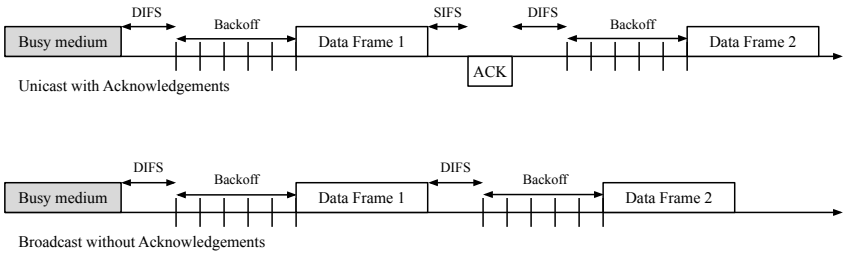

Figure 3.2 – CSMA/CA with and without acknowledgements

Before a station transmitts data it has to check or listen to the medium to avoid collision resulting in packet loss. Only when the medium is free, the station waits for a DCF Inter Frame Spaces (DIFS) and a random backoff. The backoff is a random time within a Contention Window (CW) to prevent all stations from transmitting immediately after the DIFS. The CW consists of several slots, each consisting of $20\mu s$ in 802.11b. If a transmission was not successful, the station doubles the CW, this doubles the average waiting time and is intended to prevent re-collisions. The

minimum CW in 802.11b is set to 16 slots. Before sending an acknowledgement, the station only waits for a Short Inter Frame Spaces (SIFS), which is significantly shorter than the DIFS because the acknowledgement frame is prioritised before the next frame.

Data Link Transmissions

When packets are sent on the data link layer, they do not contain the headers to the layers above, such as TCP/IP. Therefore, IP addresses cannot be used to transmit packets to another network. There are three basic types of data link transmissions: unicast, broadcast and multicast.

Unicast

The link layer unicast is used to sent data over a single hop to the target Wireless Endsystem (WES) destination. The link layer of each WES checks the destination MAC address in the link layer header and discards the frame if the destination address does not match its own address. It is therefore a direct communication between the transmitter and the WES.

Unicast is by default reliable. When the Unicast reaches the destination WES without missing or broken bytes an acknowledgement frame is send back after the SIFS. If the acknowledgement is not successfully received by the sender, the sender will repeat the transmission for a given number. When the number is exceeded, the packet could not be delivered. If the number is set to zero, the unicast can be considered as non-reliable. After each unsuccessful delivery, the size of the CW is doubled.

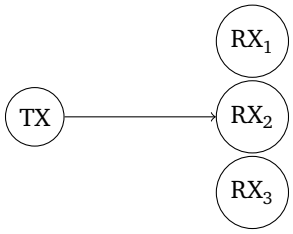

Figure 3.3 – Unicast Transmission

Broadcast

If a packet should be received from all WESs it can be distributed as a broadcast, while setting the MAC address of the destination address in the link layer to the common broadcast address, which is ff:ff:ff:ff:ff:ff.

In contrast to unicast, broadcast is not reliable. This is mainly because the packet is addressed to all nodes at the same time, and if link layer acknowledgements would be used, the acknowledgements would be sent by all nodes at the same time, because there is no mechanism in which order acknowledges should be answered. Retransmitting all acknowledgements at the same time would lead to massive collision and loss of acknowledgements. In addition, the sender of a broadcast does not know how many WESs he is addressing the packet to in the first place.

For example, in a WLAN management information is sent as a broadcast, because it has to reach every station and isn't worth to be acknowleged.

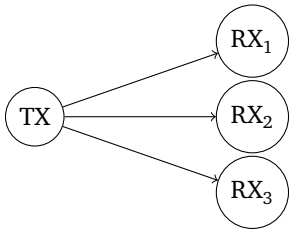

Figure 3.4 – Broadcast Transmission

Multicast

When the same packet should be transmitted to multiple WESs, but not to all, multicast can be used. Transmitting the same packet multiple times via unicast is wasteful. For this purpose, there is a corresponding multicast addresses. They exist only as destination addresses.

With multicast, the use of acknowledgements is possible because the sender knows how many receivers there are. There are different approaches to realize acknowledgements for mutlicasts, they differ mainly by the respective field of application, an approach for robust multicast in video transmissions is given by Mihaela van der Schaar et al. in [6]. Level 2 multicast is often used for large files in audio or video streams, where a big amount of data is distributed and multiple clients listen simultaneously.

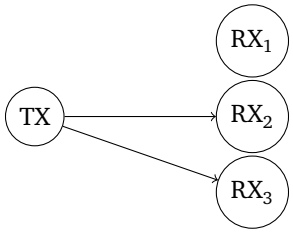

Figure 3.5 – Multicast Transmission

3.2 Light protocols

There are several lighting protocols that are used. The field of application ranges from wired CAN buses over ethernet cables to wireless WLAN networks. To give a short insight, some of the most important protocols are explained below.

DMX-512a

Digital Multiplex (DMX) 512a, is the current industry standard for stage lighting. It is based on Controller Area Network (CAN), therefore it uses wires. Physically is the DMX protocol is transmitted over a differential pair of lines using the RS-485 voltage levels. The bus signal is updated with 44Hz. According to the specification XLR-5 type connectors are to be used (Figure 3.6[2]).

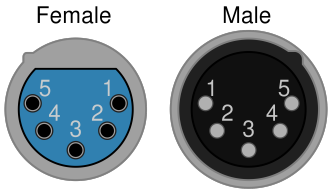

Figure 3.6 – XLR5 Pinout - Officialy Used for DMX

The endsystems in DMX512 are called fixture because it's most likely a lighting installation which is mounted somewhere. This could be a moving-head, fresnel, spotlight, stroboscope or any other light installation. It could also be a fog machine that emits fog on an appropriate signal.

All devices are daisy-chained together visualized in 3.7. The DMX controller is in the beginning of each chain. The receiving endsystems are chained behind each other from output to input. A terminator, specified in the DMX specification, is to be connected to the final output.

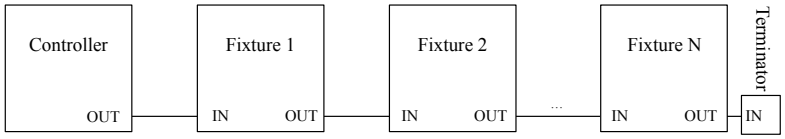

Figure 3.7 – DMX Topology

The hole chain is called DMX-Universe and can contain a set of 512 channels. If there is a need of more channels one needs more DMX universes - each channel

[2]Image available at `https://upload.wikimedia.org/wikipedia/commons/thumb/1/1c/XLR5_pinouts.svg/2560px-XLR5_pinouts.svg.png`

consists of one byte. Due to the fact that a DMX universe always has its own bus, starting from a new controller can lead to inconveniences.

A channel in the event technology/CAN-Bus is to distinguish between e.g. a Wi-Fi channel. Each endsystem is assigned at least one but usually several channels. Every endsystem knows which channel is intended for it, which must be must be preset.

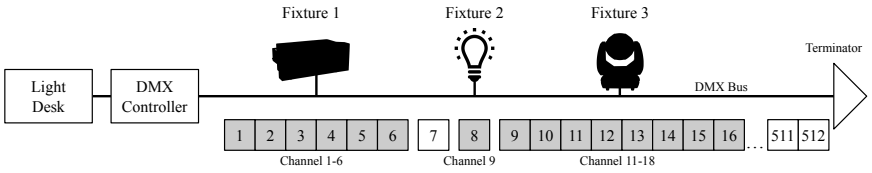

Figure 3.8 – DMX Chain Example

In the illustrated example Figure 3.8 the first fixture occupies channels 1-6, while channel 7 is unoccupied. The second fixture only is a dimmer that can be dimmed in 256 steps, therefore it only needs one channel, the 8th. The third fixture is a moving head, which occupies channels 9-18 directly behind the dimmer. The remaining channels are still unoccupied. The terminator is connected to the end of the DMX bus. If one byte does not have a high enough resolution, merging two bytes to a dual channel of 2 bytes or 65536 different values insted of 256. E.g., the 360-degree rotation of a moving head is described roughly in the first byte and in fine steps in the other.

Since DMX is unidirectional it can be assumed that the endsystems generally only receive or forward (daisy chain) control signals sent from the control console. This is a major limitation of DMX, beside of the rather small universe size. It is also not reliable, the use of fire installations is therefore considered too dangerous.

Art-Net

To deal with the limitations of DMX, some further developments have been established, the most common being Art-Net. Art-Net is a protocol that moves away from the CAN bus and distributes the data via Ethernet in a Local Area Network (LAN). The approach remains the same, the individual stations are controlled with a constant update frequency. There is no longer a limitation of 512 channel universes. A universe at Art-Net describes 32,768 DMX universes over a single network, far more than is realistically used. In standard mode, the packets are distributed to the stations with unicasts, by means of connectionless User Datagram Protocol (UDP).

Because Art-Net is a network protocol, it is also able and designed to distribute packets via WLAN. The lighting console talks to an AP via Ethernet or WLAN and this access point forwards the control signals to the individual station. The choice of

WLAN specification is mainly determined by the stations, so it is just limited to 2.4 GHz or 5 GHz. Special hardware that supports Art-Net at the factory is expensive because of the development costs and the relatively low quantities.

3.3 ESP Platform

Almost every 802.11 capable Microcontroller Unit (MCU) could be picked for this research. But there are several reasons why the ESP Platform from Espressif is a valid choice. There are several chips provided by Espressif with Wi-Fi specifications, these chips are very affordable [3] and although the ongoing chip crysis there are still easy to get, in contrast of the also very popular Chips from the manufacturer Arduino, which are also more expansive. Espressif supports an own development IDF to flash the chips, with minor tweeks it's also to use the Arduino IDE.
However, the properitary protocol ESP-Now which, just supported in the ESP Ecosystem, is discussed below and has promising properties for a solid and fast realisation of a low level protocol.

ESP32 Hardware

The chip ESP32 is quite common in DIY projects around everything from home automation to light installations and can be bought as development boards, which are ready to use.

The chip 3.9 is promoted with several features: [4]

- Fast CPU (2 cores at 240 MHz)
- 802.11 b/g/n with up to 150 Mbps (2.4GHz)
- Wi-fi Multimedia (WMM)
- Immediate Block ACK
- Automatic Beacon monitoring (hardware TSF)
- Virtual Wi-Fi Interfaces
- Simultaneous support for Infrastructure, SoftAP, and Promiscuous modes
- Bluetooth v4.2 BR/EDR and Bluetooth LE
- Advanced Peripheral Interfaces: GPIO, ADC, DAC, touch sensors, hall sensor, SPI, I2S, I2C, UART, CAN, RMT (TX/RX), Motor/LED PWM

[3] 3.86 at `www.aliexpress.com`, 10.3.2022

[4] The most recent ESP32 datasheet can be found the documentation page of Espressif, here used: V3.3
`https://www.espressif.com/en/support/documents/technical-documents`

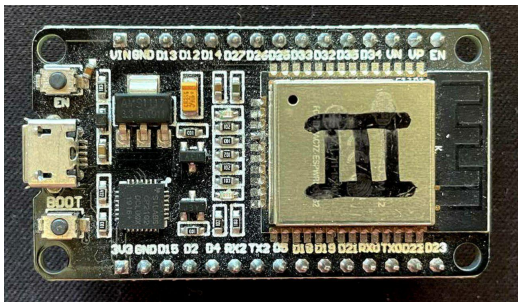

Figure 3.9 – ESP32 Devboard (Devkit V1)

It can be said that the ESP32 is a low-cost 32-bit microcontroller, which is quite powerful despite its low power consumption. The extensive set of features makes it suitable for integration into a wide range of smart environments. [1]

ESP-Now

ESP-NOW is a properitary protocol developed by Espressif. *ESP-NOW is widely used in smart light, remote controlling, sensor, etc.* [5] It is a connectionless protocol, so the WESs are in Ad-Hoc mode instead of STA. It is just supported on the ESP8266, ESP32 and ESP32s, all chipsets from Espressif, but they are compatible with each other. Because of this, an ESP-Chip is needed as a gateway to interact from the outside to the ESP-NOW communication.

Through the hardware limitation of the boards it can just be used on the 2.4 GHz frequency band. ESP-NOW allows 10 ESPs for pairing with encryption and up to 20 without encryption. Espressif promises throughput of up to 30 MBit/s and a possible range of up to 1 km. However, Roberto Pasic et al. [7] measured a range of the unmodified onboard antenna of the ESP32 and just got a *stable communication up to 190 m in open field*.

The focus of the ESP-NOW protocol is on low power consumption. A connectionless communication between WESs not only saves energy during the authentication process, Additionally, the commonication is direct and not over an access point, through the properties of ad-hoc networks. The protocol has a limitation of a limited payload of 250 bytes for each transmission. It also has much less overhead, which results in shorter airtime, fewer disturbances and also less power consumption through the antenna (latter beeing not relevant for this thesis). There is no TCP/IP header to be transmitted. For very small payloads, this offset can become disproportional.

The default ESP-NOW bit rate is 1 Mbps it uses a channel width of 20MHz, there is no double channel (40Mbit/s or higher) used. But e.g., the low energy, high

[5] `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html`

range protocol Long Range Wide Area Network (LoRaWAN) suffers from a too slow throuput for this application.

To understand what ESP-NOW does one must take a look to the vendor-specific action frame transmiting ESP-NOW data, these are Action Frames desinged for vendor-specific signaling. The compositions of the frame are broken down in more detail in Table 3.2 and Table 3.3 [6]

MAC Header	Category Code	Org.	Random Values	Vendor Specific Content	FCS
24	1	3	4	7 ~ 255	4

Table 3.2 – ESP-NOW Frame Format

check out at the end, if formatation is broken

- **MAC Header:** As ESP-NOW is connectionless, the MAC header differs from that of standard frames.
- **Category Code:** The Category Code field is set to the value(127) indicating the vendor-specific category.
- **Organization Identifier:** The Organization Identifier contains a unique identifier (0×18fe34), which is the first three bytes of MAC address applied by Espressif.
- **Random Value:** The Random Value field is used to prevent relay attacks.
- **Vendor Specific Content:** The Vendor Specific Content contains vendor-specific fields (table 3.3)
- **Frame Check Sequence:** Used for error correction in layer 2.

Element ID	Length	Org. Identifier	Type	Version	Body
1	1	3	1	4	7 ~ 250

Table 3.3 – Vendor Specific Content

- **Element ID:** The Element ID field is set to the value (221), indicating the vendor-specific element.
- **Length:** The length is the total length of Organization Identifier, Type, Version and Body.

[6] Also from 5

- **Organization Identifier:** The Organization Identifier contains a unique identifier(0×18fe34), which is the first three bytes of MAC address applied by Espressif.
- **Type:** The Type field is set to the value (4) indicating ESP-NOW.
- **Version:** The Version field is set to the version of ESP-NOW.
- **Body:** The Body contains the ESP-NOW data.

It is worth to mention, that the vendor specific content (3.2) is allowed to contain up to 255 bytes. But the sum over all values (3.3), if the body would contain the maximum of 250 bytes, leads to a total of 260 bytes. The values are from the documentation of ESP-NOW from Espressif. They also claim in the ESP-NOW documentation, that broadcast is not supported, but it is. It seems that the documentation [7] isn't completely finished (or translated).

[7] Also from 5

Chapter 4

Proposed Approach

Different approaches are presented and discussed in this chapter using data link unicast or broadcast in different specifications, these were also empirically tested and evaluated in Chapter 5. At the end of this chapter, the test setup will be introduced and it is briefly explained how the chips are programmed.

4.1 Design

Art-Net	Slim Application
UDP	
IP	
802.11 DL/Unicast	802.11 DL/UC or BC
802.11* PHY	802.11* PHY

Table 4.1 – Art-Net Layer compared to Slim Data Link Layer

The use of high-layer protocols, described in Art-Net Section 3.2, in lighting technology involves a considerable overhead. Because the lighting console does not talk directly to the WES, communication must be controlled via an AP, which means that the Internet Protocol (IP) (layer 3) must be used for addressing and UDP (layer 4) for transporting the data. They both come with additional headers. Such an overhead can lead to more latency, higher channel congestion and packet loss.

Ad-hoc networks seen in Section 3.1, on the other hand, packets can be sent directly on the MAC (layer 2). With a payload of a few bytes to each WES keeping overhead small can be quite important. Complexity problems as often typical in ad-hoc networks are not to be assumed, since the controller at the light desk must normally stand in line of sight to the individual WES. Because the lighting technician also has to keep an eye on everything from there.

One can therefore assume a simple star topology. In the following the ESP-NOW protocol (Section 3.3) was chosen to distribute the packets low-level, because even if it was not developed for this purpose the specification fits quite well to the requirements.

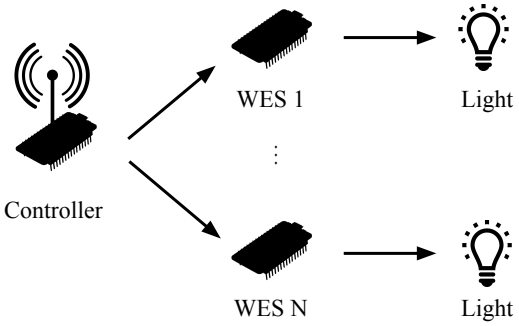

Figure 4.1 – Topology of the Data Flow

For the purpose of this analysis, the controller transmits 20 bytes (analogue to 20 DMX channels) to 8 WESs. The dataflow is shown in Figure 5.1, both the controller and the WESs are implemented with the ESP32 harware. Four different metrics are considered, which where discussed in the requirements.

- **Latency**
  The time that elapses between sending the command to the controller, transmitting it to the WESs and switching the lights. For the sake of simplicity, delays caused by the microcontroller and control of the light installation are neglected and the focus is placed on 802.11 affected parameters. Low latency is important so that the lighting technician can intervene live in the light show, and the lights do react in time.

- **Update Frequency**
  The frequency with which the controller can send the control signals every WES. This results in how continous movement of moving heads are moving, how fluid the transition of the color can be performed and how exact the control can be adapted to a musical beat.

- **Reliability**
  The security with which the data sent by the controller actually arrives at the WESs. Lack of reliability can result in two WESs positioned next to each other not behaving the same because one of them only receives half of the signals.

- **Synchronisation**
  The synchronous control of the lights by the WES. For synchronisation, each

WES must wait a buffering delay until the last WES has also received a signal. If two of the WESs are to be controlled simultaneously, but the signal was transmitted one after the other, they have to wait for each other so that the lights change at the same time.

Slim Unicast

The implementation that probably comes closest to Art-Net's is to replace the TCP packets sent by Art-Net to the respective IP-Address of the WESs with unicasts to the MAC-address of the WES. Of course, the MAC address of all WESs must be known and they must all be paired with the controller, but this process is just analogous to mapping the corresponding IP addresses after dialing the WESs into a WLAN.

Latency

Latency describes the time between a command and the expected response, here considered as airtime. The airtime using 1 Mbit/s is rather easy calculated, every byte (8 Bits) takes 8 μs.

For a full transmission the PHY and MAC preamble and header has to be transmitted twice, once for the data and once for the acknowledgement. The MAC body contains the payload, which depends upon the addressed WES. In a perfect clean channel the sender does not need to defer, but has to take a DIFS plus a backoff. A perfect empty channel resets the CW to CW_{min}, which are 31 slots in 802.11b and a slot time of 20 μs an average delay of 370 μs is estimated.

$$\frac{CW_{min}}{2} = \frac{31}{2} = 16.5 \tag{4.1}$$

$$20\mu s \cdot 16.5 = 370\mu s \tag{4.2}$$

The total airtime of the transmission of the data and acknowledgement, assuming the transmission arrived successfully, is $t_t x = 1100\mu s$. Strictly speaking, the light could change before the acknowledgement is sent, i.e., after $956\mu s$. The latency scales linearly, the delay on the n-th WES is:

$$\text{Airtime} = N \cdot t_{tx} = N \cdot 1270\mu s \tag{4.3}$$

In order to avoid unnecessary load on the radio channel, Art-Net transmits only the changes. In the worst case, however, changes affect all WESs at the same time.

Frame segment	Byte	Duration in μs
DIFS	-	50
Average Backoff	-	370
PHY header: PLCP preamble	18	144
PHY header: PLCP header	6	48
MAC headers	28	224
MAC body	20	160
= tx time data		**956**
SIFS	-	10
PHY header: PLCP preamble	18	144
PHY header: PLCP header	6	48
MAC headers, no MAC body	18	112
= tx time ack		**314**

Table 4.2 – Composition of the Total Airtime (tx + ack)

Update Frequency

Following the approach of DMX and updating the 'bus' every 44Hz, would be achieved by sending the packets round-robin via unicast. With a correspondingly high number of WESs, this could be challenged with a transmission speed of 1 MBit/s, it also scales linearly with each additional WES. In a labor steril empty channel, denying all side latencies, the airtime for N WESs could be:

$$\text{Frequency} = \frac{1}{N \cdot 1270\mu s} = \frac{787.4}{N} Hz \tag{4.4}$$

For 10 WESs, addressed with respectively 20 bytes, it would still be 787.4 Hz. This is far above the update frequency of DMX with 44Hz, but also very unrealistic and just intended to show, that it could theoretically be within the realm of possibility.

Reliability

A benefit of the unicast is the support of acknowledgements. The acknowledgements trigger a retransmission if no packet has arrived, therefore a controlled light will receive its signal in any case. So the reliability should be very good.

Synchronisation

Synchronising the unicast transmissions costs a lot of latency, through buffering delay. This is because not only does each WES need to wait until its own packet has arrived, but until all the packets have arrived at each WES. The implementation is chosen in such a way that each WES knows at which position of the round-robin it is and delays the execution of the successfully received transmission until the last WES

has also received its signal. The delay must then be calculated deterministically. In ESP-NOW, the default is set to 8 retransmissions, so in the worst case it is assumed that a packet is sent 8 times for each WES.

$$\text{Airtime}_{sync} = 8 \cdot N \cdot 1270\mu s = N \cdot 10160\mu s \tag{4.5}$$

$$\text{Frequency}_{sync} = \frac{1}{N * 10160\mu s} = \frac{98.42}{N} Hz \tag{4.6}$$

The idea of the slim unicast is, that a transmission to each device is very fast, because the transmitted payload is small. However, since we are sending many small packets, it can be assumed that we will be sending a lot of overhead. So it's playing off reliability against transmission speed. It can be said that synchronisation is a feature that should be dispensed with in the slim unicast for the sake of latency. An approach that has not been explored is to reduce the number of retransmissions.

Slim Broadcast

The ESP-Now protocol supports both unicast and broadcast. Instead of transmitting every unicast after each other, Slim Broadcast transmits a broadcast with the payload of all channels at the same time to all fixtures. If there is a need for more than 250 byte (DMX channel) a second broadcast has to be sent to transmit the missing data. To achieve this, each WES must be told in advance at which position in the payload its data is located. The broadcast must also spend one byte of payload for the sequence number. In the application layer the WESs discard the incorrect broadcasts or read out their assigned section from the entire payload.

For the unicast the payload was assumed to be 20 bytes for each WES, the same amount is assumed for each WES in the Slim Broadcast calculations. The maximum payload of the broadcast is reduced to 200 bytes, instead of the possible 250 bytes, because when it is close to the 250 byte limit, reliability is supposed to collapse.

cite something 250 byte max payload ESP-NOW

Latency

The latency of the broadcast is easier to calculate than that of unicast, because the acknowledgements are no longer necessary. In return, the payload of a single transmission increases. Assuming 10 WESs are to be addressed, each with 20 byte payload visualized in 4.3.

Due to the fact that a second broadcast is needed if the maximum payload of ESP-NOW is exceeded, the expected transmission time is not continuous, as shown in Figure 4.2. It is easy to see that the additional overhead caused by adding another package creates noticeable latency.

Frame segment	Byte	Duration in μs
DIFS	-	50
Average Backoff	-	330
PHY header: PLCP preamble	18	144
PHY header: PLCP header	6	48
MAC headers	28	224
MAC body	200	1600
= tx time data		**2456**

Table 4.3 – Composition of the Broadcast Airtime

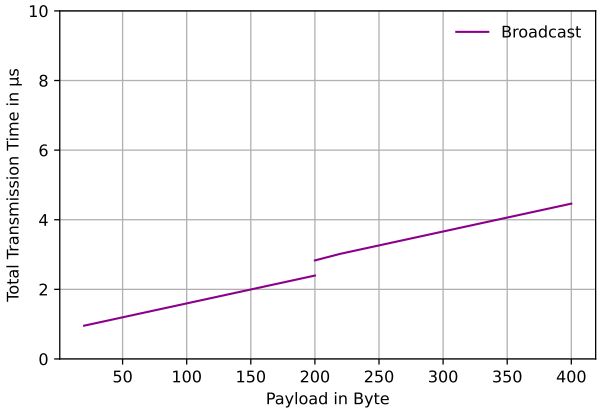

Figure 4.2 – Transmission Time of Broadcasts Depending on Payload

Update Frequency

However, while compareing the latency of the unicast to the latency of the broadcast visualized in (4.3), it becomes clear that the low overhead and the missing acknowledgements lead to a significantly higher rate. Even with a WES, the transmission time is somewhat shorter due to the lack of the acknowledgement, a complete cycle, i.e., addressing all WESs with 20 bytes, is already an order of magnitude faster from a number of 10 WESs.

$$\text{Frequency}_{UC} = \frac{1}{10 \cdot 1270\mu s} = 78.7Hz \tag{4.7}$$

$$\text{Frequency}_{BC} = \frac{1}{2456\mu s} = 407.2Hz \tag{4.8}$$

Even if these values are only remotely comparable with real measurement data, it is clear, that the throughput is significantly higher with broadcast than with unicast. This effect should be seen clearer under real conditions.

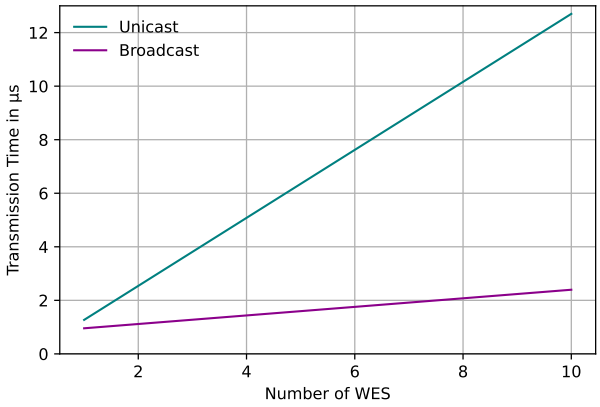

Figure 4.3 – Transmission Time of Unicast vs Broadcast

Reliability

The huge advantage that the Slim Broadcast has over the Slim Unicast in terms of update frequency, comes with the cost of lower reliability. Broadcasts can't use acknowlegements, a WES that has poor reception to the controller, will not receive packets. The controller cannot take countermeasures as already discribed in Section 3.1.

Synchronisation

Insead of transmitting to several fixtures after each other slim broadcast transmits to all fixtures at the same time. This solves the problem of synchronisation for less than 200 channels. For more than 200 each WES has to wait until the last broadcast has arrived. In contrast to unicast, the buffering delay can be calculated almost deterministically for broadcast due to the absence of automatic retransmissions.

Rapid Repetition

To improve the reliability of the slim broadcast, the same sequence can simply be repeated unsolicited. The idea is not to wait for a missing acknowledgement, but to increase the probability that one of the packets got through. The reliability of the Slim Broadcast with Rapid Repetition improves with every rapid repetition (RR). In the Equation (4.9), with RR set to zero no repetition occurs, the results are visualized in Figure 4.4.

Cite paper A First Implementation and Evaluation of the IEEE 802.11aa Group Addressed Transmission Service

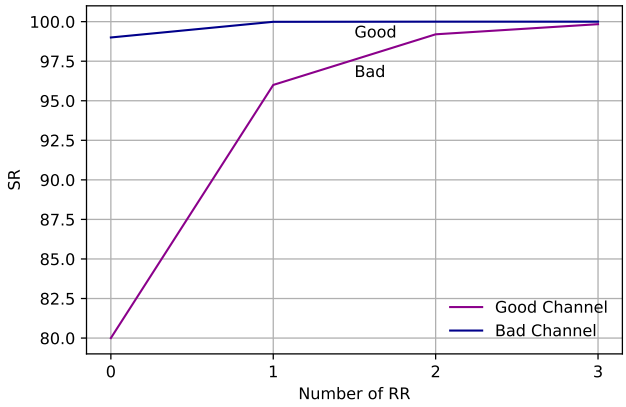

Figure 4.4 – Success Ratio increase with increasing number of RR

$$\mathrm{SR}_{RR}(RR) = 1-(1-\mathrm{SR})^{RR+1} \tag{4.9}$$

$$\mathrm{SR}_{RR}(0) = SR \tag{4.10}$$

$$\mathrm{SR}_{RR}(1) = 1-(1-\mathrm{SR})^{2} \tag{4.11}$$

The RR makes it possible to reach WESs with poor reception much more reliably. However, WESs that have very good reception also receive the same packet redundantly. The update frequency of a BC with RR must be divided by the number of repetitions, compared to one without repetitions, and the latency multiplies when using synchronisation. However, in contrast to unicast, broadcast offers much shorter transmission times, that at least a few repetitions can be accepted. Figure 4.5 shows the multiplicative offset, for each further retransmission.

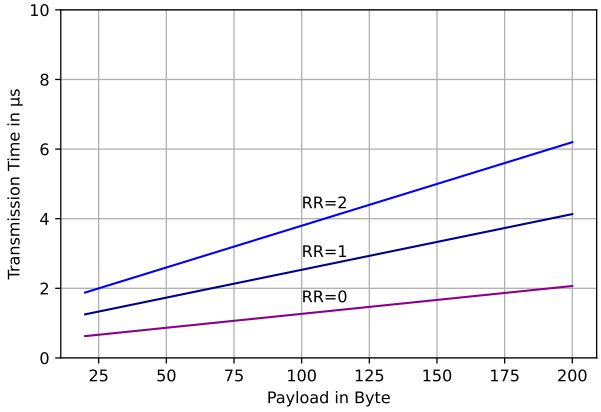

Figure 4.5 – Transmission Time with different sets of RR over rising Payload

Delayed Rapid Repetition

To push the idea of rapid repetion even further, long burst of packet loss be taken into account. This can cause all the repetitions to be captured at once. If the individual repetitions of the first sequence are sent displaced together with those of the second sequence, then the probability increases that at least one of the repetitions is not affected by long bursts of packet loss.

Staggering the individual sequences, does not affect the update frequency at all, because just as many packets are sent as with Slim Broadcast using RR. The latency, on the other hand, is significantly increased, especially when synchronisation is maintained. In the given example of Figure 4.6 the WESs has to wait for 5 times the duration of a broadcast transmission, instead of three.

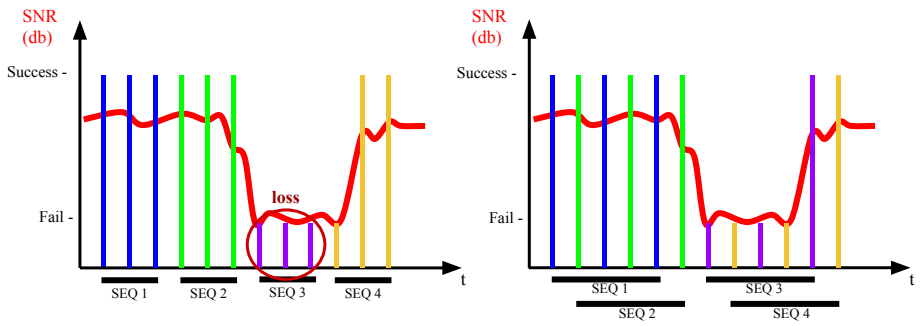

Figure 4.6 – Rapid Repetition = 3 over a occuring noise

The delay of the Delayed Rapid Repetition (DR) can also be extended considerably. In extreme cases, it could be set to the number of sequences, which is of course impractical, but interleaving two or three sequences could help counteract persistent noise.

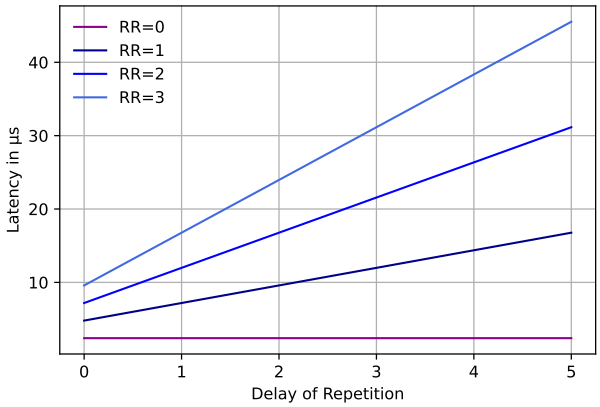

Figure 4.7 – Latency of DR, transmitting 200 Bytes

With DR, improved reliability comes at the cost of latency, not update frequency. Because no more packets are sent, only the order is shifted. If no repetition is applied, the delayed rapid repetition has no influence, which is why the graph in Figure 4.7 remains constant for RR=0. The greater the RR and the delay of the DR, the more packets were send between the first and the last packet of each sequence.

The measurement results introduced in Chapter 5 will show whether the use of delayed repetition can reduce the number of rapid repetitions while maintaining or even improving reliability.

4.2 Implementation

The developer board ESP32 Devkit V1 used in the experiments (Chapter 5) and the collection of the data can be ordered cheaply [8] from common (most favourably Chinese) websites. The most accessible way to flash a chip is using the Arduino IDE, which can be configured for flashing ESPs. A more precise approach is to use the IDF of Espressif itself, which can also be made to work by installing the toolchain and build tools and a Plugin for your favorite IDE. An illustration can be found in Figure 4.8 [9].

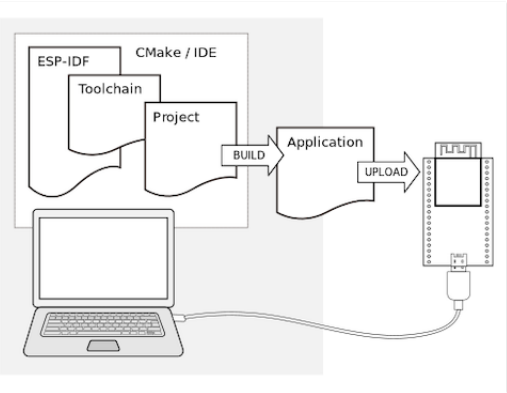

Figure 4.8 – Flashing the ESP

Espressif has provided a user guide for the use of ESP-NOW [10], but unfortunately it contains some outdated information, for example, claiming that broadcast is not supported. However, more detailed information can be found on the website.

[8] 3.79 €, www.aliexpress.com, ESP32 Devboard V1, 3/3/22

[9] `Imageisavalibleathttps://docs.espressif.com/projects/esp-idf/en/latest/esp32/_images/what-you-need.png,3.3.2022`

[10] Available at `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html`

Only a few steps are necessary, to use ESP-NOW on an ESP. The chip must activate Wi-Fi and put it in STA mode and ESP-NOW can be activated. The two libraries "esp_wifi.h" and "esp_now.h" are required for this.

```
WiFi.mode(WIFI_STA);
esp_now_init();
```

Listing 4.1 – Init ESP-NOW

check formatation in the final version

Later, the individual MAC addresses of the WESs must be saved. These are created in a separate header file and can then be added during the setup of the chip. The MAC address is needed to add the respective WES to the peerlist. However, the WES does not have to be switched on or has to be in range for this, it is more a case of making the WES known to the controller. The Controller can store up to 20 devices in his peerlist at the same time.

listing box prevent linebreaks on newpage

```
#ifndef MACLIST_H
#define MACLIST_H

uint8_t WES_MAC_1[6] = { 0xFC ,0xF5 ,0xC4 ,0x31 ,0x9A ,0x44 };
uint8_t WES_MAC_2[6] = { 0x24, 0x0A, 0xC4, 0x61, 0x19, 0x08 };
uint8_t BC_MAC[6]    = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
```

```
#include "maclist.h"

if (!esp_now_is_peer_exist(WES_MAC_1)) {
  peer_info.channel = 13;                    // 1-14
  memcpy(peer_info.peer_addr, WES_MAC_1, 6);
  esp_err_t status = esp_now_add_peer(&peer_info);
}
if (ESP_OK == status) {                      // check success
  Serial.println("[OK] Slave-peer added");
}
```

Listing 4.2 – Add Peers

Sending an ESP-NOW unicast is performed with the method esp_now_send(). The MAC address of the recipient is passed, also a pointer to the payload and its length. In the case of a broadcast, the address field is filled with the broadcast address (ff:ff:ff:ff:ff:ff). The function esp_now_register_send_cb can be used to check whether the cast was successfully sent. It is important to write as little code as possible in such a callback function. Sending the next package only after receiving the dispatch confirmation, ensures that the packets are sent in the correct order.

```
void metaInformationToSlaves(const uint8_t *peer_addr, ↘
    struct_advanced_meta metaData) {
  esp_now_register_send_cb(onDataSent);     // register callback

  esp_err_t status = esp_now_send(WES_MAC_1,
                                  (uint8_t *) &payload,
                                  sizeof(payload));
  if (ESP_OK == status) {                   // check success
    Serial.println("[OK] ESP-NOW sending");
  }

void onDataSent(const uint8_t *mac_addr, esp_now_send_status_t ↘
    status) {
  if (status == ESP_NOW_SEND_SUCCESS) {
    Serial.println("[OK] ESP-NOW Send");
  }
}
```

Listing 4.3 – Send ESP-NOW Cast UC/BC

Working with mirocontrollers requires a non continous program-flow, instead it's event-based. For the individual WESs, a callback function is registered after setup, it's called when an ESP-NOW transmission is passed on to the application layer.

```
esp_now_register_recv_cb(OnDataRecv);

void OnDataRecv(const uint8_t *mac_addr, const uint8_t ↘
    *incomingData, int data_len) {
  if (incomingData[0] == 253) {             // setup data
    applyMetaInformation(incomingData, data_len);
    return;
  }
  if (incomingData[0] == 255) {             // verify data
    applyPayload(incomingData, data_len);
  }
  if (incomingData[0] == 254) {             // return results
    sendResultsToMaster();
    return;
  }
```

Listing 4.4 – ESP-NOW Callback Functions

Chapter 5

Evaluation

5.1 Methodology

One challenge in programming the controller and the WESs, is the state-based approach and the fact that the WESs can only communicate restricted with each other. There were therefore two types of state machines, one for the controller and one for the WESs.

The testbed Figure 5.1 is set up so that the PC is connected to the microcontroller via wired Universal Asynchronous Receiver Transmitter (UART), this microcontroller is the controller. The controller communicates exclusively via ESP-NOW with the WESs, these then control in different ways with the stage lights and motors. The WESs never respond to the controller for the actual application, the communication is unidirectional. However, in order to evaluate the test data, the WESs are put into a mode in which they send the data to the controller, also via ESP-NOW. From there, they are also sent to the PC via UART.

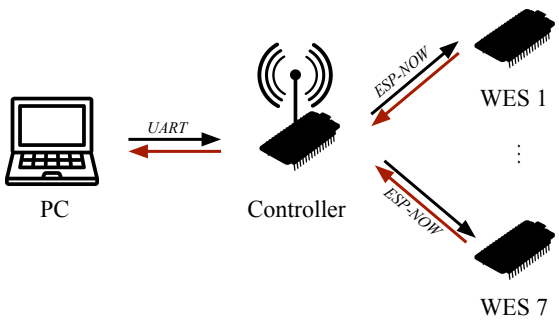

Figure 5.1 – Data Flow for Measurement

The experiment is started from the PC, a JSON is then transmitted to the controller via UART using a Python script. The JSON contains all the parameters that are needed to carry out a test. At the time the JSON string is transmitted, the controller should be idle so that the packet is read correctly.

Variable	Example	Explaination
VERBOSE	0	Enable VERBOSE
DEBUG	0	Enable DEBUG
TIMESTAMP_UART	0	Enable UART timestamps
SEQUENCE_REPITITIONS	200	Sequences per experiment
FULL_REPETITIONS	1000	Repeations of the experiment
MASTER_CHANNEL	6	Wi-fi Channel of the Controller
WES_CHANNEL	6	Wi-fi Channel of the WES
WAIT_AFTER_SEQ	0	delay between sequences
WAIT_AFTER_REP_EXP	2000	delay between experiments
IS_BROADCASTING	1	BC:=1, UC:=0
RAPID_REPITITION	2	BC: Rapid Repetitions
CHANNEL_TOTAL	160	BC: Addressed Payload
BROADCAST_FRAME_SIZE	200	BC: Maximum Payload/Brodcast
UNICAST_FRAME_SIZE	20	UC: Payload/Unicast
WES_COUNT	6	UC: WES Count
AIRTIME	0	Capture airtime

Table 5.1 – JSON Experiment Setup

When the controller has received its JSON, it must go through three states, regardless of further input from the PCFigure 5.2.

Setup

After the start-up, the controller waits for a JSON. When the JSON is received, it sets the corresponding variables. It then forwards these to the individual WESs using unicast. To be absolutely sure that the respective WES has received the setup information, the number of retransmits in the application layer is set to infinity. It distributes the data round-robin, the MAC addresses of all WESs are hardcoded, but could also be transmitted via JSON. If the first byte of the setup unicast is set to 253, the WES knows that it is a metadata packet and handles it accordingly in its callback.

Testing

When the controller has ensured that each WES has received the measurement data, it starts transmitting the dummy test data. Sequences are then sent according to the variable SEQUENCE_REPETITIONS. The duration varies greatly depending on the number of sequences and the protocol used.

Collecting

When the controller has processed all sequences, it makes a request to one of the WESs, again with the help of a 100% reliable unicast forced on the application layer. This then transmits the measured values and in turn ensures that these have also arrived at the controller. The measured values are then transmitted to the PC and the experiment is repeated until the required number of experiments has been completed.

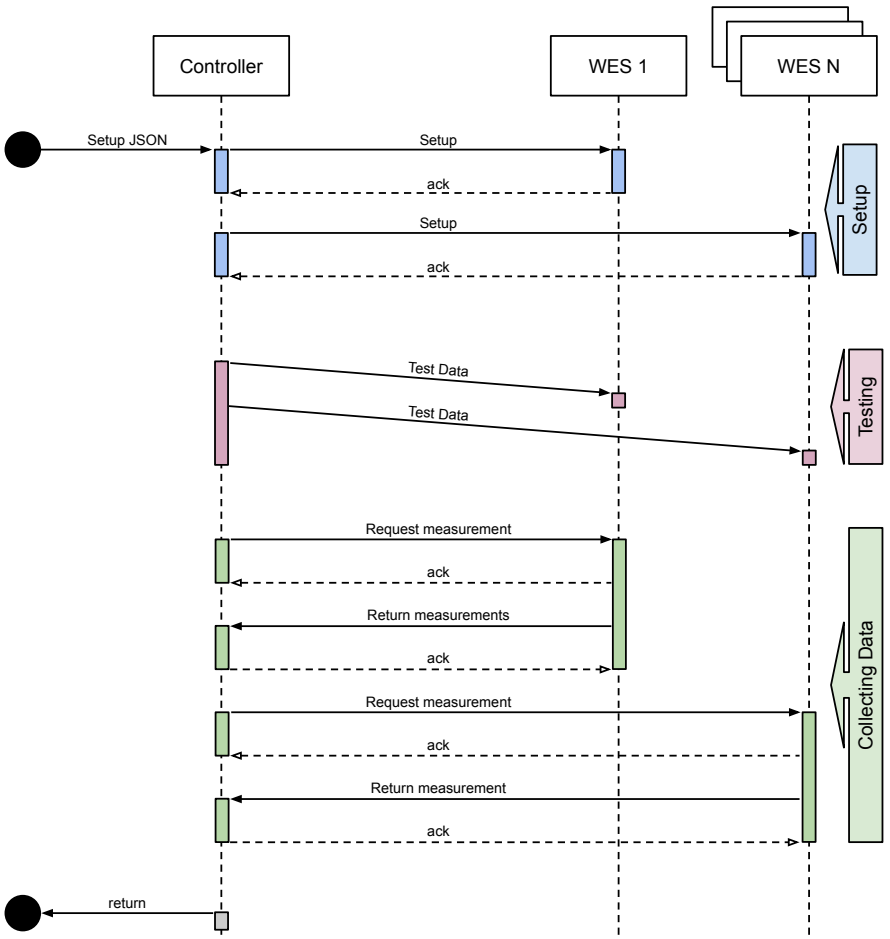

Figure 5.2 – Sequence Diagram of the Measurment

5.2 Wireshark's measurements

The Wireshark tool is suitable for recording traffic. In order to record packets outside a LAN, the WI-FI card must be set to monitor mode. In this mode, frames from an

ad-hoc network can also be sniffed, as is the case with ESP-NOW. Figure 5.3 shows hat each unicast is followed by an acknolegement, as mentioned in the Chapter 3.

No.	Time	Source	Destination	Lengtl	Info
1	0.000000	Espressi_31:69:0c	Espressi_31:9a:44	119	Action, SN=23, FN=0, Flags=
2	0.000012		Espressi_31:69:0c…	70	Acknowledgement, Flags=....
3	0.001224	Espressi_31:69:0c	Espressi_31:9a:44	119	Action, SN=24, FN=0, Flags=
4	0.001258		Espressi_31:69:0c…	70	Acknowledgement, Flags=....
5	0.002315	Espressi_31:69:0c	Espressi_31:9a:44	119	Action, SN=25, FN=0, Flags=
6	0.002350		Espressi_31:69:0c…	70	Acknowledgement, Flags=....
7	0.003408	Espressi_31:69:0c	Espressi_31:9a:44	119	Action, SN=26, FN=0, Flags=
8	0.003443		Espressi_31:69:0c…	70	Acknowledgement, Flags=....
9	0.004522	Espressi_31:69:0c	Espressi_31:9a:44	119	Action, SN=27, FN=0, Flags=
10	0.004534		Espressi_31:69:0c…	70	Acknowledgement, Flags=....
11	0.005600	Espressi_31:69:0c	Espressi_31:9a:44	119	Action, SN=28, FN=0, Flags=
12	0.005634		Espressi_31:69:0c…	70	Acknowledgement, Flags=....

Figure 5.3 – Unicast Transmissions Recording from Wireshark

A look into the data frame Figure 5.4 also shows the measured airtime of 696μs. In the calculation from Table 4.2, however, it was only 536μs. Adding the average backoff of a free channel (160μs) and the DIFS (50μs) gives 906μs. Wireshark also recognises the category code of the Vendor Specific Action Frame that ESP-NOW uses. The payload of 20 bytes assumed for unicast in the experiment is given here as 31 bytes, presumably it has to do with the implementation of the action frame shown in Table 3.3.

```
▾ 802.11 radio information
    PHY type: 802.11b (HR/DSSS) (4)
    Short preamble: False
    Data rate: 1,0 Mb/s
    Channel: 1
    Frequency: 2412MHz
    Signal strength (dBm): -71 dBm
    TSF timestamp: 3013560838338
  ▸ [Duration: 696µs]
▾ IEEE 802.11 Action, Flags: ........C
    Type/Subtype: Action (0x000d)
  ▾ Frame Control Field: 0xd000
      .... ..00 = Version: 0
      .... 00.. = Type: Management frame (0)
      1101 .... = Subtype: 13
    ▸ Flags: 0x00
    .000 0001 0011 1010 = Duration: 314 microseconds
    Receiver address: Espressi_31:9a:44 (fc:f5:c4:31:9a:44)
    Destination address: Espressi_31:9a:44 (fc:f5:c4:31:9a:44)
    Transmitter address: Espressi_31:69:0c (fc:f5:c4:31:69:0c)
    Source address: Espressi_31:69:0c (fc:f5:c4:31:69:0c)
    BSS Id: Broadcast (ff:ff:ff:ff:ff:ff)
    .... .... .... 0000 = Fragment number: 0
    0000 0001 1111 .... = Sequence number: 31
    Frame check sequence: 0xaad022ed [unverified]
    [FCS Status: Unverified]
▾ IEEE 802.11 Wireless Management
  ▾ Fixed parameters
      Category code: Vendor Specific (127)
      OUI: 18:fe:34 (Espressif Inc.)
    ▾ Data (31 bytes)
        Data: 019f35a3dd1918fe340401ff080203040506070809 0a0b0c0d0e0f10111213
        [Length: 31]
```

Figure 5.4 – Unicast Transmission Radio Information from Wireshark

A look at the 31 byte "payload" shows that ESP-NOW specific parts of wireshark have not been parsed correctly. From the representation **??** and **??** it can be deduced:

That the first 4 bytes of the payload are still the random values and, according to Espressif, do not belong to the Vendor Specific Content. These 4 bytes together with the following 1 byte (Element ID), 1 byte (length, interestingly specified as 25

bytes instead of 20), Organisation Identifier (3 bytes), Type (1 byte) and Version (1 byte) make up the difference of 11 bytes to the actual payload. The real payload is filled with the self defined flag 0xff Section 4.2 followed by the sequence number 0x00. The rest of the payload is filled with numbers, which are set to the value of the position in the payload 0x02,0x03,..,0x13.

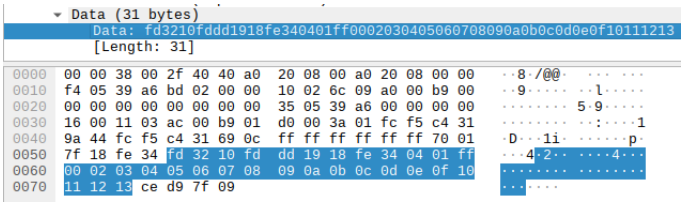

Figure 5.5 – Unicast Payload Analysis with Wireshark

For completeness, here is a recording of the broadcast traffic. The difference to the unicast traffic in Figure 5.4 is, that the destination address is bundled in a transmission of 160 bytes instead of being distributed in 8x20 byte packets. As already mentioned, the acknoledgments are not possible with the broadcast.

No.	Time	Source	Destination	Lengtl	Info
1	0.000000	Espressi_31:69:0c	Broadcast	259	Action, SN=2994, FN=0, Flags=........C
2	0.002087	Espressi_31:69:0c	Broadcast	259	Action, SN=2995, FN=0, Flags=........C
3	0.004136	Espressi_31:69:0c	Broadcast	259	Action, SN=2996, FN=0, Flags=........C
4	0.006022	Espressi_31:69:0c	Broadcast	259	Action, SN=2997, FN=0, Flags=........C

Figure 5.6 – Unicast Payload Analysis with Wireshark

5.3 Protocols under Study

Slim Unicast vs Slim Broadcast

In the design part it became clear that it is much more efficient to reach many individual WESs with one broadcast instead of many individual unicasts. Tracking the transmissions with Wireshark also confirms this assumption Figure 5.7.

The theoretical assumptions clearly correspond to the measured values, Additional latencies due to processing in the microcontroller or at the antenna are therefore hardly significant. The update interval for the unicast increases sharply as the number of stations increases, whereas the broadcast increases only slowly. With unicast, it can be seen that the measured values fluctuate around the values assumed in theory. This is because these are exemplary values from the Wireshark measurement Figure 5.3 and these are not averaged over all experiments, the fluctuation can be explained by the fact that the backoff is randomly selected from the CW. Measurments of the transmission time for broadcasts with smaller payloads

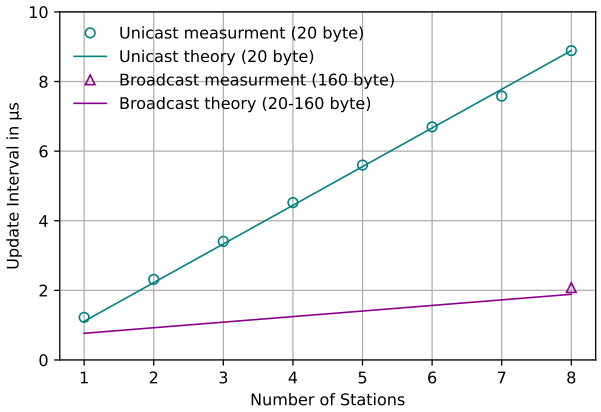

Figure 5.7 – E.g. Transmission Time of Slim Unicast and Slim Broadcast

where not made. Therefore, a broadcast is clearly superior to an unicast in terms of latency, update frequency and synchronisation.

Slim Unicast can only make a difference through its reliability. The channel in the experiment was free, so no retransmissions had to be sent, which would have delayed the transmission time even more. When implementing slim unicast, it makes sense to do without synchronisation, because the buffering delay would be too long. However, it is also difficult to estimate how many transmissions can be saved if only changed values trigger a transmission, but it makes sense to simulate a stress test because critical errors become apparent precisely when packets are lost and all values are changed.

It can be said that Slim Broadcast is the superior design due to the much lower part of overhead, because the data throughput is significantly higher. The WESs can also be synchronised more easily because the transmissions reach many people at the same time. The weak point, however, is reliability.

Rapid Repetition

Success Ratio is a good metric to measure reliability, it is calculated from the number of successfully received packets and the total number of packets sent. In a test setup of 7 distributed WESs, some placed close to the controller, some at a slightly greater distance, this naturally varies greatly Figure 5.8. The measurement was carried out in a flat distributed over different rooms. Several WLANs are on the 2.4GHz band too and lead to indifferences. For the slim broadcast, 160 byte packets were distributed in 600 sequences, the experiment was repeated 1000 times.

It shows that especially WESs 4 and WES 7 have poor reception. The reception of other WESs is much better. WES1, WES2 and WES3 where in the same room as the controller, WES1 in particular was only a few centimetres away from the

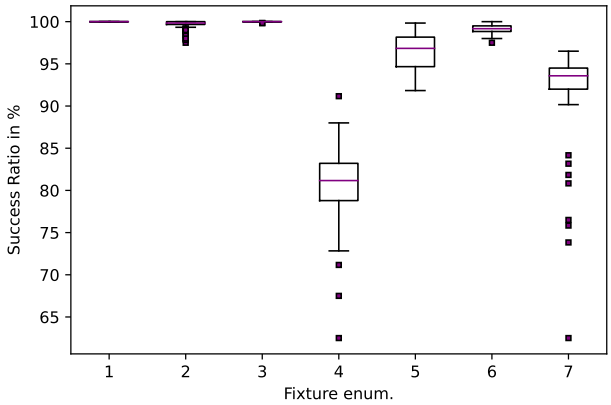

Figure 5.8 – SR of Broadcasts for 7 WESs

transmitter and actually has no packet loss either. It should be said that with the Slim Unicast 100% of the packets arrived and there was no loss of data. One approach to improving reliability at the expense of latency was rapid repetitions.

With RR, it must be weighed up how many repetitions are sensible, because at some point the performance beyond reliability is impaired too much. WES4 has the weakest reception and is supposed to represent a kind of worst-case scenario. The measurement data of the same measurement as for Figure 5.8 are used as a basis, so that a comparability is given. With RR=0, the unchanged SR is taken over, with RR=X, X transmissions are always linked in pairs to one with an or.

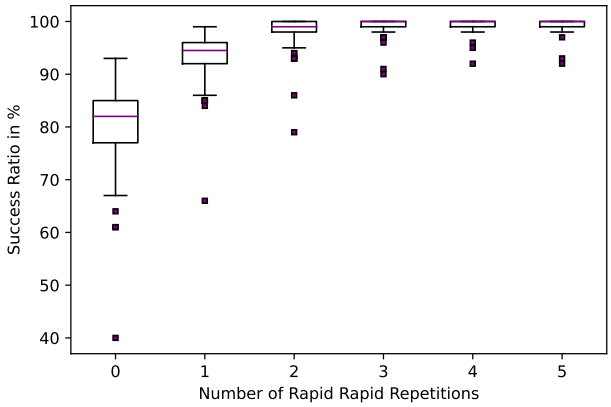

Figure 5.9 – SR of WES4 with altered RRs

It becomes clear that even a few repetitions lead to a significantly better SR. But it also becomes clear that this is no longer improved as much after two repetitions.

However, a reliability of 100% as with unicast is not necessarily achieved. A look at the measurement data shows why Figure 5.10. The given excerpt is from

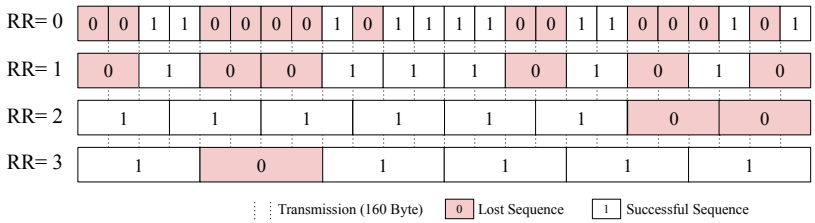

Figure 5.10 – Bitwise Example of RR (WES=4, SeqNr.=9-24, ExpNr.=1)

an actual measurement of the WES4, the channel here is really very bad and there is massive packet loss. When the repetitions are increased from RR=0 to RR=1, the loss of packets is slightly dampened. However, it is noticeable that even with RR=3, one sequence is still lost, despite the 4x costs of latency and throughput. The problem is that faulty packets often come clustered and the repetitions still fall within the range of the bad channel. Erwähnenswert ist, dass einzelne Sequenzen bei RR=2 ankommen, obwohl sie bei RR=3 verloren werden. This can also be seen in Figure 5.10, the second and third sequence of RR=2 happily get over the channel interference, whereas the second sequence of RR=3 falls right into the interference and is therefore lost. Of course, this is rather rarely the case, which can be seen in the significantly increased success ratio in Figure 5.9.

Delayed Repetition

To circumvent this circumstance, the sequences can be nested Figure 4.6, this is only possible in combination with the Rapid Repetitions. Subsequently swapping the sequence numbers makes it possible to make an evaluation on the same data. WES4 is chosen again because the packet loss is very high.

update Graphic!

In Figure 5.11 it is quite clear that the success ratio increases the longer the delay of the repetition. However, the improvement converges quite quickly. It can therefore be assumed that the success ratio is noticeably improved with a constant update frequency, at least with a buffering delay of 1, is the effect significant. It also shows that the use of the buffering delay does not improve the success ratio so much that one less repetition could be sent. In addition, the use of buffering delay has a cost in latency, as mentioned in FIGURE. . It must therefore be weighed up whether the additional latency justifies the increased success ratio.

How significant - after the plot is updated, give an exact example

link to graphic of buffering delay latency theory

address syncronization problem?

5.4 Results

In the measurements, the unicast approach with 20 bytes each or a broadcast with 160 bytes were sent to the 7 WES. The metrics latency, update frequency, reliability and synchronisation are compared in the table Table 5.2.

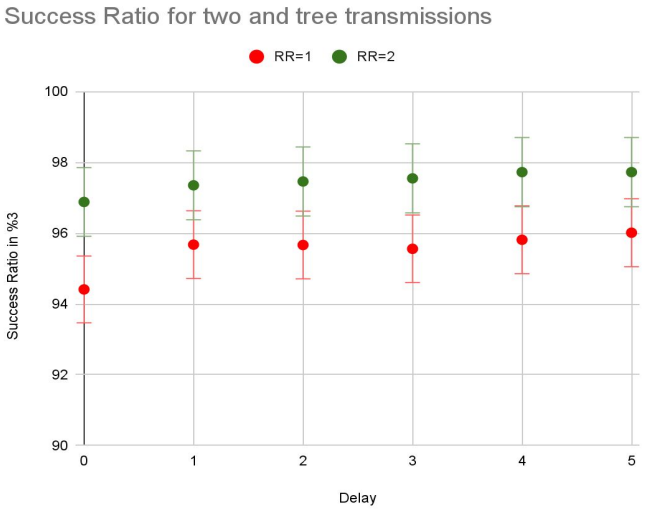

Figure 5.11 – SR for Buffering Delay with/without RR for WES4

Slim Protocol	Latency in μs	Update Frequency in Hz	Reliability in % Average	WES 4	Sync
UC	696 - 59128	16	100.00	100.00	no
UC sync	7581 - 59128	16	100.00	100.00	yes
BC	1816	479	95.45	80.50	yes
BC 1xRR	3903	239.5	98.37	93.00	yes
BC 2xRR	5990	159.7	99.40	98.22	yes
BC 3xRR	8077	119	99.62	99.04	yes
BC 1xRR+DR	XXXX	239.5	XX.00	XX.00	yes
BC 2xRR+DR	XXXX	159.7	XX.00	XX.00	yes
BC 3xRR+DR	XXXX	119	XX.00	XX.00	yes
DMX (base)	22700	44	100.00	100.00	yes

Table 5.2 – Comparison of the measured results

It quickly becomes apparent that the Slim Unicast, both with and without synchronisation, is more reliable than the broadcast, but also has much better latencies and update frequency. These values become particularly extreme when the lights are to react synchronously. In the absence of synchronisation, the values vary greatly, because the latency varies depending on whether the WES is in the first or last position of the round-robin and on the number of retransmissions that have to be made if a packet is lost.

With the solutions that use a broadcast, it becomes clear that synchronisation between the WESs is always given, because all WESs receive the signal at the same time. The difference is mainly due to the DIFS and the backoff that have to be waited for before the second sequence can be sent. But the reliability especially for WES4

is too low (80.5%). The use of rapid repetitions also shows that the reliability of the weaker WES4 in particular is extremely improved. With 3 rapid repetitions, the reliability reaches a range that is also acceptable for the WES4 (99%).

Finally, there is the use of delayed repetition (DR) to broadcasts with rapid repetition (RR). Here, only the delayed repetition with the interlacing of a sequence is shown, as the effect is greatest here. It is noticeable that the latency due to the retention of the packets is no longer close to the update frequency, as with the other broadcasts, but even higher. The effect of the delayed repetition fades into the background when the RR are set to 3.

Compared with DMX as a baseline, it shows that with this specific setup, the broadcast is also good with 3 RR, whether with or without DR. Only the reliability is below that of DMX, albeit only just, because even the WES with the worst reception has a success ratio of over 99%. The unicast already reaches its limits with this small setup, even if synchronisation were to be dispensed with.

Chapter 6

Conclusion & Discussion

This thesis introduces various protocols optimised for lighting technology. These were investigated analytically and experimentally. A special feature of the protocols is that they send packets on the data link layer and have and have additional adaptations in the application layer to improve reliability.

The analytical examination showed that the unicast scales poorly with many stations, because the packets only have a few bytes of payload, the additional overhead is too great. The calculations have shown that even with a few stations, a single broadcast packet distributes the control signals twice as fast. In practice, this effect could be even more pronounced, because an error-free channel without need of retransmissions, was always assumed.

With broadcast, on the other hand, there is no reliability. The assumptions for the duration of the distribution of all control signals to the stations, were made with a perfectly clean channel, but in a realistic scenario, packet loss must be expected. Rapit repetition in the application layer is intended to counteract this. By sending the same sequence multiple times, lost packets can be compensated for. If the channel is unclean over a longer period of time, the Delayed Rapid Repetition, is supposed to close these gaps by staggering the sequences.

The measurements confirmed the analyses. The unicast packets arrived successfully at the stations 100% of the time. With the broadcast packets, there were significant losses recorded. It had become apparent that reception was particularly poor at one of the stations, this station was then selected to simulate a bad-case scenario. The use of rapid repetition led to a significant increase in reliability at this station. However, up to three retransmissions were necessary to bring the success ratio of the weakest station to over 99%. Delayed Rapid Repetitions, with interleaving the sequences, on the other hand, had a rather small effect on the success ratio, especially when many repetitions were sent. The additional latency caused by the Delayed Rapid Repetition does not justify the small improvement in the success ratio

and is therefore not recommended. In conclusion, it can be said that 100% reliability in lighting technology is desirable, but if the update frequency is high enough, the next sequence will transmit a new or unchanged control signal shortly afterwards anyway. A correspondingly high update frequency reduces the influence of missing individual packets.

For the test set-up, a separate procedure protocol was developed, which should guarantee the smooth running of the test. One difficulty was to ensure that the stations were always in the correct state. A controller sent 160 bytes of control information per sequence to 8 different stations. This was sent either in 20 byte packets as unicast or as a single 160 byte packets as broadcast. This, of course, corresponds to a very small stage setup. A DMX Universe is 512 bytes, so it would be interesting to see how the protocols would behave if they had to distribute the data to 30 stations. In reality, the size of the control signals of a single station also varies greatly, which has not been investigated here.

The ESP-NOW protocol was used to implement the transmissions on the data link layer. A connectionless protocol, which is proprietary, but allows the development of a simple and robust prototype, originally optimised for low energy. It only runs on the low-cost platform of the manufacturer Espressif. It has some limitations that make it difficult to use in practice. The payload of a transmission may not exceed 250 bytes, that if the control signal exceeds 250 bytes, two broadcasts must be sent, provided with an ID. Also limiting is, that no more than 20 stations can be paired with the controller. Because the protocol only runs on the ESP platform, the transmitter must also be an ESP chip. This limits the choice of physical layers. Thus, the behaviour of the 5GHz frequency band was not investigated. In the tests, only 1 Mbps was transmitted, the outdated IEEE 802.11b was used, as the investigation with wireshark showed. According to the data sheet, the ESP chip used can achieve data rates of up to 150 Mbps.

The prototype protocols were able to show that transmissions in the data link layer can work properly, however, tests with larger setups and a non-proprietary protocol are still pending.

List of Abbreviations

AP	Access Point
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
CAN	Controller Area Network, *when referring to the bus protocol*
CSMA/CA	Carrier-sense Multiple Access with Collision Avoidance
CW	Contention Window
DIFS	DCF Inter Frame Spaces
DL	Data Link Layer
DMX	Digital Multiplex
DSSS	Direct Sequence Spread Spectrum
FHSS	Frequency Hopping Spread Spectrum
IEEE	Institut of Electrical and Electronics Engineers
IP	Internet Protocol
LAN	Local Area Network
LLC	Logic Link Control
MAC	Media Access Control
MCU	Microcontroller Unit
MIMO	Multiple Input-Multiple Output
MSDU	MAC Service Data Unit
NAV	Network Allocation Vector
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open Systems Interconnection
PHY	Physical Network Layer
SIFS	Short Inter Frame Spaces
STA	Station
UART	Universal Asynchronous Receiver Transmitter
UDP	User Datagram Protocol
WES	Wireless Endsystem
WLAN	Wireless Local Area Network

List of Figures

List of Tables

List of Listings

Bibliography

[1] D. D. Husam Kareem, "The Working Principles of ESP32 and Analytical Comparison of using Low-Cost Microcontroller Modules in Embedded Systems Design," *4th International Conference on Circuits, Systems and Simulation (ICCSS)*, pp. 130–135, Mar. 2021. DOI: 10.1109/ICCSS51193.2021.9464217.

[2] Y. Xiao, "IEEE 802.11 Performance Enhancement via Concatenation and Piggyback Mechanisms," *IEEE Transactions on Wireless Communications*, pp. 2182–2192, Nov. 2005. DOI: 10.1109/TWC.2005.853875.

[3] N. I. Sarkar, "The impact of transmission overheads on IEEE 802.11 throughput: analysis and simulation," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, pp. 49–55, Jan. 2011.

[4] L. C. P. Salvador F. Gringoli and P. Serrano, "A First Implementation and Evaluation of the IEEE 802.11aa Group Addressed Transmission Service," *ACM SIGCOMM Computer Communication Review*, vol. 4, pp. 35–41, Jan. 2014. DOI: 10.1145/2567561.2567567.

[5] S. C. Mihaela van der Schaar Santhana Krishnamachari and X. Xu, "Adaptive Cross-Layer Protection Strategies for Robust Scalable Video Transmission over 802.11 WLANs," *IEEE Journal on Selected Areas in Communications*, no. 8, pp. 1752–1763, Jan. 2004. DOI: 10.1109/JSAC.2003.815231.

[6] T. P. Jérome Lacan, "Evaluation of Error Control Mechanisms for 802.11b Multicast Transmissions," vol. 6, 2006. DOI: 0-7803-9550-6/06.

[7] K. A. Roberto Pasic Ivo Kuzmanov, "ESP-NOW communication protocol with ESP32," *Journal of Universal Excellence*, vol. 40, no. 8, pp. 53–60, Feb. 2021. DOI: 10.37886/ip.2021.019.