Maximilian W. Gotthardt

# Evaluation of protocols for control stage lighting

Bachelor Thesis in Computer Engineering

8 March 2022

# 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: | **8. 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 8 March 2022

# Abstract

about 1/2 page:

1. Motivation (Why do I care?)

2. Problem statement (What problem are I trying to solve?)

3. Approach (How did I go about it)

4. Results (What's the answer?)

5. Conclusion (What are the implications of the answer?)

In the field of lighting and stage technology, the challenge of controlling the individual installations, called 'fixtures', quickly and without complications is a recurring one. Established solutions are realized via cables.

However, due to the progress in radio technology, wireless solutions are becoming more and more common. Therefor is often expensive hardware needed. Parallel to this there is a fast growing market around creative and individually developed DIY projects, which have found their own niche. Durch niedrigpreisige While most commercial solutions still rely on expensive and complex wired control, it is particularly suitable for smaller projects to experiment with the new wireless technologies. In this thesis I try to implement a wireless solution, which does not need an IP-Layer using the popular platform ESP and the properitary protocol ESP-NOW to distribute the light information to each fixture. ESP-Now instead works more like an direct radio communication.

# Kurzfassung

- kabellose lösungen werden interessant

- chips werden günstiger

- für kleine projekte leider sehr teuere Hardware

- 802.11 wird als standard benutzt

- protokolle wie art-net könnten optimiert werden

- esp plattform bietet interssante möglichkeiten, wegen der geringen kosten der chips und esp-now

- entwicklung einer plattform die esp-now nutzt

- verschiedene ansätze studiert, wie broadcast und unicasts

- mit jeweils unterschiedlichen modifikationen

# Contents

# Contents

> The table of contents should fit on one page. When in doubt, adjust the `tocdepth` counter.

# Chapter 1

# Introduction

- general motivation for your work, context and goals.

- context: make sure to link where your work fits in

- problem: gap in knowledge, too expensive, too slow, a deficiency, superseded technology

- strategy: the way you will address the problem

- recommended length: 1-2 pages.

In the subject area
====
In the field of lighting and stage technology, the challenge of controlling the individual installations, called 'fixtures', quickly and without complications is a recurring one. Established solutions are realized via cables.

## 1.1 Motivation/Requirements

- Reliability

- .. and why 100% Reliability is not important (pyrotechnics)

- Lower latency

- Synchronisation

- higher update frequency

- Range

## 1.2   Challenges

- low cost

- (ESP Platform)

- DIY community

## 1.3   Problemstatement and Contribution WICHTIG

- open source available on github [link]

- thought-provoking impulse for different approaches

- Protocol auf DL Layer/App Layer Ebene

- Art-Net baseline

- simulativ und experimentel untersucht

## 1.4   Thesis Outline

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

    unsosliced Repetition

    blockack

- Evaluation of Error Control Mechanisms for 802.11b Multicast Transmissions

    packet loss rate

    ARQ, FEC

- ESP-NOW communication protocol with ESP32

    ESP-NOW details

- The Working Principles of ESP32 and Analytical Comparision of using Low-Cost Microcontroller Modules in Embedded Systems Design

    why the ESP32 is superior over arduino

- Adaptive Cross-Layer Protection Strategies for Robust Scqalable Video Transmissions Over 802.11 WLANs

- Voice Capacity of IEEE 802.11b, 802.11a and 802.11g Wireless LANs

# Chapter 2

# Related Work

- Wie der und der in Paper so gezeigt hat

- Auch Ding et al haben versucht

- ...

- 10 Paper

- halbe seite

Foo and bar [1] are of equal value. Thus, any can be used.
According to [2]

Wireless solutions for stage lighting are growing fast.

## 2.1  References

What follows is just a very quick refresher on how to use references. It is not a guide on scientific writing in general, nor copyright and plagiarism in particular. Please refer to an actual guide on technical writing and scientific practices to make sure you understand how, where, and when to cite.

Simply speaking, proper scientific writing has to deal with two closely related (but not identical) concepts:

a) Copyright

b) Plagiarism

Do not confuse the need for properly citing your sources as something related to copyright. Questions of (a) copyright or the corresponding national equivalent deal with who has the right to reproduce a certain text excerpt, an image, or something similar. Questions of (b) plagiarism deal with who came up with a certain idea or

insight, e.g., a certain finding, a certain concept, or a certain way of illustrating a concept. By way of analogy, consider a car: after buying a car you have the right to (a) do whatever you want with it, but you still cannot claim that you (b) invented it. Conversely, properly (b) crediting who invented your neighbor's car does not give you the right to (a) use it. Put yet another way, problem (a) is a legal one: to be allowed to publish a scientific work you (or, rather, your publisher) needs to have permission to reproduce it – or suffer legal consequences like heavy fines. Problem (b) is an academic one: claiming someone else's ideas as one's own is plagiarism; similarly, re-selling old ideas as new ones is self-plagiarism. Both incur heavy penalties like exclusion from schools and professional associations or being blacklisted from publishing with scientific outlets for any number of years.

You will need to address both problems in writing your thesis. Problem (a) can be addressed in two ways: First, by creating original content (that is, text or figures) yourself, which is always preferable as this gives you the freedom to present the content your way. Second, by obtaining a license to reproduce content (e.g., by way of buying a license or adhering to the terms of an existing copyleft license). Problem (b) can be addressed in two ways: First, presenting original ideas and insights (as you will do when presenting own results). Second, by clearly pointing out the (primary) source of an idea. The latter is the topic of this section.

In brief, use references whenever you cite from related work (either directly or indirectly), or when you build on related work (this includes their way of illustrating a particular concepts, in text form as well as in the overall design of a figure). Also use references to point a reader to related work. Clearly distinguish between these uses. Make it very clear which part of a statement a reference belongs to. Compare the following three, vastly different uses (where the cited idea appears in **boldface**):

- "**Foo and bar are of equal value. Thus, any can be used.**" [2], [3]

- According to [2] and [3], **foo and bar are of equal value, and any can be used**.

versus

- "**Foo and bar are of equal value**" [2], [3]. Thus, any can be used.

- According to [2] and [3], **foo and bar are of equal value**. From this it follows that any can be used.

versus

- Foo and **bar** [2], [3] are of equal value. Thus, any can be used.

- Foo and **bar** (detailed in [2] and [3]) are of equal value. Thus, any can be used.

versus

- **Foo** [2] and **bar** [3] are of equal value. Thus, any can be used.

- **Foo** (detailed in [2]) and **bar** (detailed in [3]) are of equal value. Thus, any can be used.

Never typeset a reference after the final full stop of a paragraph (or sentence) and expect your reader to figure out which part of the paragraph is an indirect citation and which part is original (i.e., your own) work. When paraphrasing longer passages of text, use an indirect citation. Make sure to clearly point out when you are finished paraphrasing, like so: *According to Akyildiz et al. [2], Foo and Bar can be characterized as follows. They are big. They are bright. The authors further argue that one can be substituted for the other. In the following I will go on to prove that this is not true.*

When citing more than a few pages worth of text, point the reader to the specific part you are referring to in your citation, like so: *In recent years, an increasing number of cyclists are switching from air filled tires to cement filled ones [4, Table IV].*

If a figure or a table is closely based on another one, make sure to cite its source, preferably in its caption, like so: *Figure 1 – the relation of ravens and writing desks (based on [4, Figure 42])*. Be aware that, while there is a well-established convention on how to illustrate a verbatim quote of text (by using quotation marks), there is no well-established convention for indicating that an image was copied verbatim. Thus, when citing a figure or table, you must explicitly state whether it was copied verbatim, ed, or whether it served as inspiration for your own.

Do not cite URLs. Content found there is not peer reviewed and it is likely to change during the lifetime of your work. For pointing a reader to interesting websites, use footnotes – but trust your reader to know how to use a web search engine.

Your text reads nicer if you do not use citations as a substitute for nouns (like this section did). Instead of *The benefits of cement filled tires has been shown by [2]*, consider writing *Akyildiz et al. [2] have shown the benefits of cement filled tires*. The `textcite` command makes this straightforward.

Make sure to read your bibliography section (that is, the typeset list of references) after you are done adding all citations to your text. Does it contain all information needed to uniquely identify to references you used? Do not trust BibTeX files you find on the web: Digital libraries frequently have their contents wrong, are missing information, or are using different field names than your bibliography style expects (leading to missing information in the typeset bibliography). To give a few examples: Check the authors' list (making sure all authors are listed in the same order and in the same way they are listed in the publication). Check the conference location (it's most likely not "New York, New York"). Check the publisher name (many digital libraries use a field that is not typeset by your bibliography style; have a look at the demo

bibliography in this template for how to deal with that). Check the page numbers (many digital libraries put "1–5" here despite the paper starting at a later page – or despite it not having any page numbers to begin with). Check the conference name, put its parts in a logical order, and lose the "in proceedings of" (it's not "Mobicom, in proceedings of, 1999 series MobiCom99" but "5th ACM International Conference on Mobile Computing and Networking (MobiCom 1999)".

triple-check all references

# Chapter 3

# Fundamentals

In this chapter the fundamentals required for understanding the different approaches in this thesis using are explained. 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..

refrence to table below

In order to understand the upcomming ESP-Now protocol we have to take a look at the Data Link Layer (DL) layer in 802.11. It is the second layer of the OSI model of computer networking illustrated in Table 3.1.

Rewrite introduction in chapter Fundamentals!

## 3.1   IEEE 802.11 Specification Family

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

TOLJA:  zu banal
darüber zu reden?

- 802.11 Wireless Local Area Network (WLAN)

- 802.15.1 Wireless Personal Area Network (WPAN)

- 802.15.4 Low-rate WPAN (LR-WPAN)

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

**Table 3.1** – OSI model

7

- 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 this BSS. This is the most common mode a WLAN is used.

- Independent BSS
  A network without a central station, where the network topology can flexible change over time. The communication happens directly between the Wireless Endsystems. Efficent routing can became 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.

### 3.1.1 Physical layer

In this thesis we sould take a breef look into the Physical Network Layer (PHY) of the IEEE 802.11 standard, which is the first layer of the OSI model 3.1. 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:

> TOLJA: Was soll ich zum PHY alles sagen?

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

- 802.11a and 802.11g
  with Orthogonal Frequency Division Multiplexing) (OFDM) data rates are increased up to 54 Mbit/s. 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.

In this thesis the rather basic 802.11b is used with a transmission rate of 1Mbit/s.

### 3.1.2   Data Link Layer

The DL Layer is the second lowest layer of the OSI Model 3.1 and is split in two sublayers. The Locig 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 multipoint communication over networks and it also adds frame sequence numbers. But in this thesis we focus on the other data link sublayer.

The Media Access Control (MAC) 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 WES were to transmit simultaneously. The MAC Protocol Data Unit is additional added inside of the PHY Payload. It contains the MAC Header and encapsulated in it the MAC Service Data Unit (MSDU).
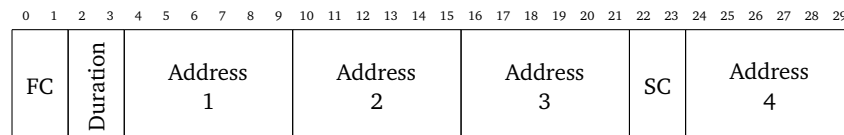
| 0 1 2 3 | 4 5 6 7 8 9 | 10 11 12 13 14 15 | 16 17 18 19 20 21 | 22 23 | 24 25 26 27 28 29 |
|---------|-------------|-------------------|-------------------|-------|-------------------|
| FC | Duration | Address 1 | Address 2 | Address 3 | SC | Address 4 |

**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 power to save energy, WES can defer access to the medium for this 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 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.

802.11ac and later using frame aggregation in order to reduce overhead.

### 3.1.3 Carrier Sense Multiple Access/Collision Avoidance

shot explaination of CSMA/CD

Multiple Access/Multiplexing: When Signals to/from different users share a common channel using time division methods (TDM/TDMA, CSMA) eigene Worte

DSSS: Usage of multiple antennas Direct Sequence Spread Spectrum. Spreading of the signal over a given bitsequence PN

Addressing
In a LAN environment, devices are logically separated using 48-bit globally unique MAC addresses: example In IPv4 networks (e.g. Internet), nodes are logically separated using 32-bit globally unique IP addresses: example

Routing
• Routing in a (W)LAN is based on MAC addresses, never IP addresses.
• A router (e.g. integrated with an access point) performs mapping between these two address types:

Address allocation
• MAC addresses are associated with the hardware devices.
• IP addresses can be allocated to (W)LAN devices either on a permanent basis or dynamically from an address pool using the Dynamic Host Configuration Protocol (DHCP).

Mesh networks

are able to relay frames from one device to another. • Provide coverage extension over multiple hops (e.g. Internet access) • Sufficient address information is required to be able to relay data from a source device to the ultimate destination (IP or MAC address). This can be used to extend the range from on Wireless Endsystem (WES) to another WES over some other WES. Since range isn't a critical parameter in this thesis, it hasn't to be further discussed.

Beacon Frames

contain the channel information found during passive scanning. Probe request are used in acting scanning.

Backoff:

random time delay to avoid collisions

DIFS/SIFS:

Delay between transmissions used for Carrier-sense multiple access with collision avoidance (CSMA/CA)

### 3.1.4 Data Link Transmissions

Text!

**Unicast**

The link layer unicast is used to send data over an single hop to the target WES destination. The link layer of each WES checks the destination MAC address in the link layer header and discards the frame if the destinatin address does not match its own address.
Unicast is by default reliable. E.g. the AP wants to transmit a packet to one specific WES
When the Unicast reaches the destination WES an acknowledgement frame is send back after the Short Inter Frame Spaces (SIFS) + backoff.

when to explain CSMA/DC?

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.

example

complete figure UC

$$\boxed{x_1}$$

**Figure 3.2** – Unicast Transmission

**Broadcast**

If a packet should be received from all WES's it can be distributed as broadcast. The MAC address of the destination address in the link layer is set 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 acknoledgements would be used, the acknoledgements would be sent by all nodes at the same time, because there is no mechanism in which order acknoledges should be answered. In addition, the sender of a broadcast does not know how many WESs he is addressing the packet to in the first place. Retransmitting acknoledgements would lead to massive collision and loss of acknoledgements. E.g. management information in a WLAN is sent in a broadcast mode, because it has to reach every WES and isn't worth to be acknoleged. `complete figure BC`

$$\boxed{x_1}$$

**Figure 3.3** – Broadcast Transmission

**Multicast**

`Mutlicast explain multicast mac address`

When the same packet should be transmitted to multiple WES's, but not to all, multicast can be used. Transmitting the same packet multiple times via unicast is wasteful. There are different approaches to realize acknowledgements for mutlicasts, they differ mainly by the respective field of application.

`examples for multicast ack + related work`

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. `complete figure MC`

$$\boxed{x_1}$$

**Figure 3.4** – 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.

### 3.2.1   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 transmitted over a differential pair of lines using the RS-485
voltage levels. The bus signal is updated with 44Hz. According to the specification
are XLR-5 type connectors are to be used. ⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯⎯ image der connectoren

Show Hardware e.g. DMX Plug

The endsystems are called fixture because it's most likely a lighting installation
which is mounted somewhere, this could be a moving-head, fresnel, spotlight, stro-
boscope 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 3.5.  The DMX controller
is in the begin 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.
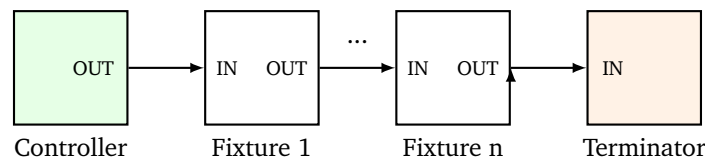


**Figure 3.5** – Block Diagram of an DMX Universe

The hole chain is called DMX-Universe and can contain a set of 512 channel. If
there is a need of more channels one needs more DMX universes - each channel
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 is to distinguish between e.g. a WiFi chan-
nel. Each endsystem is assigned at least one, but usually several channels. Every
endsystems knows which channel is intended for it, this must to be preset.

For example: An RGB-LED spotlight could have three channels, one for each
color. Due to the resolution of one byte, the individual colors can (theoretically) be
controlled in 256 different intensities. If the Channel 0-56 are already used, it could

be set to channel 57, 58, 59. Any other free channel range would also be possible, provided it is connected. There exist hardware with automatic-address-assignment.

Image of fixtures distributed on channel

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.

### 3.2.2 Art-Net

- Example of an wifi protocol

- 2.4 or 5GHz

- DMX-Like

- related work

Due the limitation of 512 channel for each universe there where protocols implemented using the Art-Net also called Art-Net DMX is

## 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 WiFi specifications, these chips are very affordable () and althrough the ongoing chip crysis (2021) there are 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 Kosten in €
2021 aufführen? Link?
Datum?

paper about esp above arduino

### 3.3.1 ESP32 Hardware

The chip ESP32 is quite common in DIY projects around everything from home automation to light installations and can baught on development boards, which are ready to use. The chip 3.6 is promoted with several features:

this is cited from: link
zum esp32 datasheet

- Fast CPU (2 cores at 240 MHz)

- 802.11 b/g/n with up to 150 Mbps (2.4GHz)

- Wifi 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
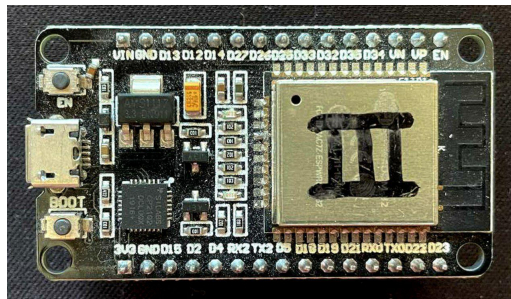


**Figure 3.6** – ESP32 Devboard (Devkit V1)

final words to the ESP32 Hardware - a lot of features for a cheap hardware

### 3.3.2   ESP-Now

ESP-NOW is a properitary protocol developed by Espressif. ESP-NOW is widely used in smart light, remote controlling, sensor, etc. It is a conectionless protocol, so the WES's are in Ad-Hoc mode insted 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, a ESP-Chip as gateway is needed to interact from the outside to the ESP-NOW communication.

cite ESP documentation website

Through the hardware limitation of the boards it can just be used on the 2.4 GHz frequncy band. ESP-NOW allows 10 ESPs for pairing with encryption and up to 20 without encryption. Espressif promises throuput of up to 30MBit/s with and a possible range of up to 1km. However *Roberto Pasic [5]* measured a range of the unmodified onboard antenna of the ESP32 and just got a *Roberto Pasic [5] stable commication up to 190m in open field*.

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

The default ESP-NOW bit rate is 1 Mbps it uses a channelwidth of 20MHz, there is no double channel (40Mbit/s or higher) used. But e.g. the low energy, high range protocol Long Range Wide Area Network (LoRaWAN) suffers from a to slow throuput for this application.

To undersant what ESP-NOW does it needs to take a look to the vendor-specific action frame transmiting ESP-NOW data.

> Explain Vendor Specific Frames!

cite somehow the ESP-NOW documentation pdf: ESP-IDF Programming Guide: ESP-NOW, source: `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html`

visualized in 3.2.

| 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

- **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 (0x18fe34), which is the first three bytes of MAC address applied by Espressif.

- **Random Value:** The Random Value filed is used to prevents 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.

Inside of the ESP-NOW frame 3.2 is the vendor specific content visualized in 3.3.

- **Element ID:** The Element ID field is set to the value (221), indicating the vendor-specific element.

| Element ID | Length | Org. Identifier | Type | Version | Body |
|------------|--------|-----------------|------|---------|------|
| 1 | 1 | 3 | 1 | 4 | 7 ∼ 250 |

**Table 3.3** – Vendor Specific Action Frame

- **Length:** The length is the total length of Organization Identifier, Type, Version and Body.

- **Organization Identifier:** The Organization Identifier contains a unique identifier(0x18fe34), 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.

> this is cited from espressif manual!!

It is worth to mention, that the vendor specific content 3.2 is allowed to contain up to 255 byte, but the sum over all values in 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, that broadcast is not supported in ESP-NOW, but it is. It seems that the documentation isn't complitly finished (or translated).

> 255 != 250 is it really that important?

### 3.3.3  ESP-Now vs Art-Net Baseline

> subsection can be on a wrong position!

> remove newpage command

- Network stack diagram

- baseline

> ESP-NOW Baseline Artnet should be moved to Design part??

# 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 the 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 | 802.11b/g/n PHY |

**Table 4.1** – Art-Net Layer compared with Slim Data Link Layer

The use of high-layer protocols, such as Art-Net Section 3.2.2, 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 User Datagram Protocol (UDP) (layer 4) for transporting the data. They both come withe additional headers. Such an overhead can lead to latency, channel congestion and packet loss.

In an ad-hoc network Section 3.1, on the other hand, packets can be sent directly on the MAC (layer 2) Table 4.1. With a payload of a few bytes to each WES, keeping overheat 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, from there finally the lighting technician from there must have everything in the range of vision to be able to intervene. One can therefore assume a simple star topology. In the following the

ESP-NOW Section 3.3.2 protocol 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.
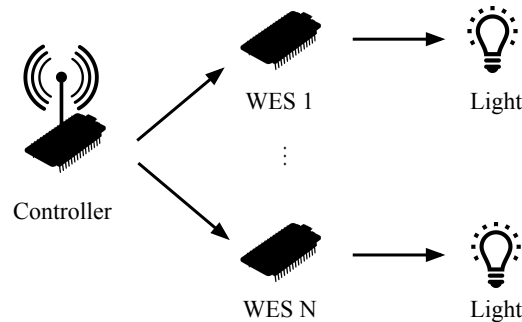


**Figure 4.1** – Topology of the Data Flow

For the purpose of this analysis, the controller transmits 20 Byte (analogue to 20 DMX channels) to every WESs. Four different metrics are considered, which where discussed in the requirements.

- **Latency**
  What is the latency from commanding the controller to the estimated reaction at the WES e.g. lighting of a light? For the sake of simplicity, delays caused by the microcontroller instruction set, data distribution and control of the light installation are neglected and the focus is placed only on the airtime.

- **Update Frequency**
  How often can the we update all WESs per second? This results in how smooth movement of moving heads are moving or how smooth the transition of the color of an LED can be performed.

- **Reliability**
  Wie sicher kommen die vom Controller gesendeten Daten bei den WESs an? 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**
  Are the signals sent to different WESs carried out at the same time? 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 1Mbit/s is rather easy calculated, every Byte (8 Bits) takes $8\mu s$.

For a full transmission the PHY and MAC preamble and header be transmitted twice, once for the data and once for the acknowledgement. The MAC body contains the payload, which depends of the needs of the addressed WES. In a perfect clean channel the sender hasn't to defer, but has to take a DCF Inter Frame Spaces (DIFS) plus a backoff. A perfect empty channel resets the CW to $CW_{min}$, which are 16 slots in 802.11b, so the average backoff should take $\frac{CW_{min}}{2}$ slots, with a slottime of $20\mu s$ follows an average backoff of $160\mu s$.

| Frame segment | Byte | Airtime in $\mu s$ |
|---|---|---|
| DIFS | - | 50 |
| Average Backoff | - | 160 |
| PHY header: PLCP preamble | 18 | 144 |
| PHY header: PLCP header | 6 | 48 |
| MAC headers | 28 | 224 |
| MAC body | 20 | 160 |
| **= tx time data** | | **786** |
| 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)

The total airtime of the transmission of data and ack, assuming the transmission arrived successfully, is $t_t x = 1100\mu s$. Strictly speaking, the light could also be changed before the acknolegement is sent, i.e. after $786\mu s$. The latency scales linearly, the delay to the n-th WES is:

$$\text{Airtime} = N \cdot t_{tx} = N \cdot 1100\mu s \qquad (4.1)$$

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

**Update Frequency**

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

$$\text{Frequency} = \frac{1}{N \cdot 1100\mu s} = \frac{9090}{N} Hz \tag{4.2}$$

For 10 WESs, addressed with respectively 20 Byte, it would still be 909Hz. This is far above the update frequency of DMX, but also very unrealistic and just intended to show, that it could theoretically be within the realm of possibility.

**Reliability**

One benefit of the unicast is the support of acknoledgements. The acknoledgements 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. This is because not only does each WES have to wait until its own packet has arrived, but until the packets have arrived at all the others. 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 and that for each WES.

$$\text{Airtime}_{sync} = 8 \cdot N \cdot 1100\mu s = N \cdot 8800\mu s \tag{4.3}$$

$$\text{Frequency}_{sync} = \frac{1}{N * 8800\mu s} = \frac{1136}{N} Hz \tag{4.4}$$

The idea of the slim unicast is, that a transmission to each device is very fast, because the transmitted payload small. However, since we are sending many small packets, it can be assumed that we will be sending a lot of overhead. So we playing

Discuss the inimportance of order of round robin in unicast

or buffering delay?

...buffering delay

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.

Unfortunately the ESP-Now protocol does not allow to control the number of retransmissions before the packet is discarded.

Is it tru, that retransmission can't be controlled in ESP-NOW?

### Slim Broadcast

The ESP-Now protocol supports both unicast and broadcast. Instead of transmitting every unicast after each other, Slim Broadcast transmitts 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 broadcasts has to be send to transmit containing 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. Only in the application layer do the WESs discard the incorrect broadcasts and read out their area from the entire payload.

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

quelle 200 byte

#### Latency

The latency of the broadcast is easier to calculate than that of unicast, because the acknoledgments 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 4.3.

| Frame segment | Byte | Airtime in $\mu s$ |
|---|---|---|
| DIFS | - | 50 |
| Average Backoff | - | 160 |
| PHY header: PLCP preamble | 18 | 144 |
| PHY header: PLCP header | 6 | 48 |
| MAC headers | 28 | 224 |
| MAC body | 200 | 1600 |
| **= tx time data** | | **2286** |

**Table 4.3** – Composition the Broadcast Airtime

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.
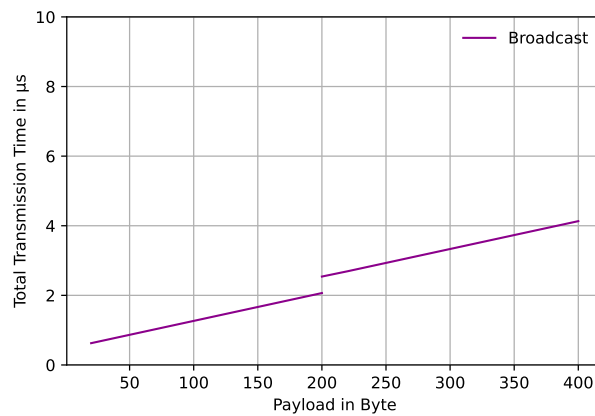
**Figure 4.2** – Transmission Time of Broadcasts Depending on Payload

**Update Frequency**

However, while compareing this to the latency of unicast 4.2, it becomes clear that the low overhead and the missing acknoledgements lead to a significantly higher rate. A complete pass, i.e. addressing all WESs with 20 bytes, is already an order of magnitude faster from a number of 10 WESs.

> it's ok to make an example here?

$$\text{Frequency}_{UC} = \frac{1}{10 \cdot 1100 \mu s} = 909 Hz \qquad (4.5)$$

$$\text{Frequency}_{BC} = \frac{1}{2286 \mu s} = 4347 Hz \qquad (4.6)$$

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 even clearer under real conditions.

**Reliability**

The huge advantage that the Slim Broadcast has over the Slim Unicast in terms of update frequency, comes at the cost of lower reliability. Broadcasts can't do acknolegements, a WES that has poor reception to the controller, will not receive packets and the controller cannot take countermeasures.

> ref to fundamentionals, datalink, broadcast

**Synchronisation**

Insead of transmitting to several fixtures after each other slim broadcast just transmitts to all fixtures at the same time. This solves the problem of synchronization for less than 200 channel. For more than 200 each WES has to wait until the last
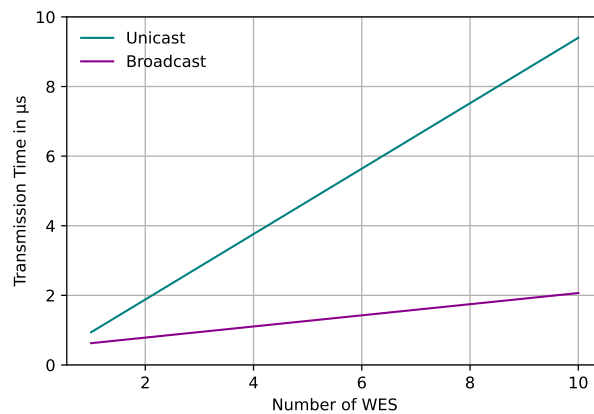
**Figure 4.3** – Transmission Time of Unicast vs Broadcast

broadcast is arrived, even if the broadcast must be discarded anyway because the required channel has already been arrived previous, deterministically calculated, the latency of all required broadcasts added up.

## Rapid Repetition

To improve the reliability of the slim broadcast, the same transmission can simply be repeated unsolicited. The idea is not to wait for a missing acknolegedment, but to increase the probability that one of the packets got through. The reliability of the Slim Broadcast with Rapid Repetition is improved with every rapid repetition (RR). In the formula below Equation (4.7), with RR set to zero, there happens no repetition.

$$\text{SR}_{RR}(RR) = 1 - (1 - \text{SR})^{RR+1} \tag{4.7}$$

$$\text{SR}_{RR}(0) = SR \tag{4.8}$$

$$\text{SR}_{RR}(1) = 1 - (1 - \text{SR})^2 \tag{4.9}$$

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, the same applies to the latency when synchronisation is required. However, in contrast to unicast, broadcast offers such shorter transmission times, that at least a few repetitions can be accepted. Figure 4.4

ist der Satz gramatisch flasch? Lass ihn einfach weg...

Is Rapid Repetition a appropriate name? Unsosliced Repetition is better siehe Paper?

Cite paper A First Implementation and Evaluation of the IEEE 802.11aa Group Addressed Transmission Service
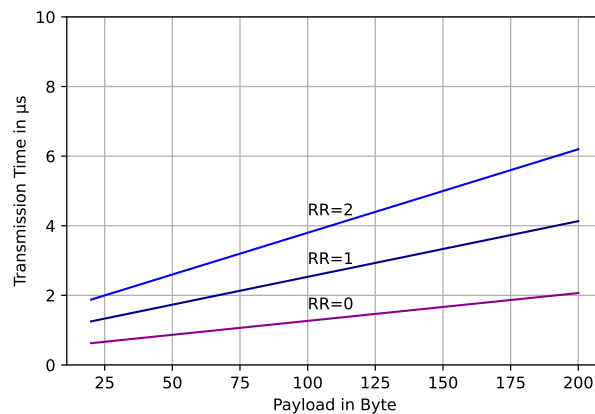
**Figure 4.4** – Transmission Time with with different sets of Rapid Repetition (RR)

## Delayed Rapid Repetition

To push the idea of rapid repetion even further, should also temporarily occuring noise 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 that at least one of the repetitions is not detected by a occuring noise noise is increased.

displaced repetition vs delayed repetition

By staggering the individual sequences, the update frequency is not affected, because just as many packets are sent as with Slim Broadcast RR. The latency, on the other hand, is significantly increased, especially when synchronisation is maintained. In the given example of Figure 4.5 the WESs has to wait for 5 times the duration of a broadcast transmission, instead of three.
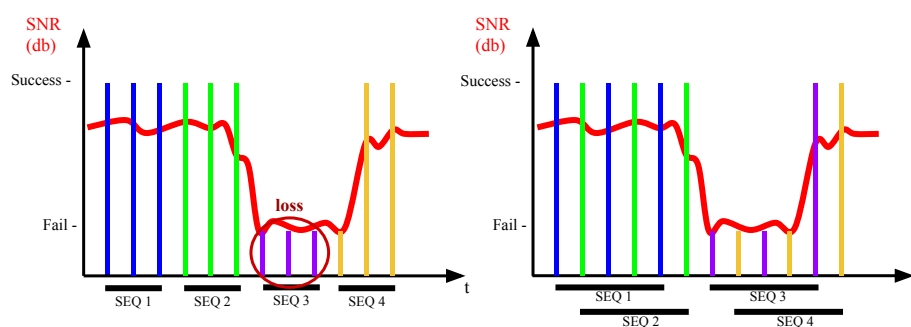


**Figure 4.5** – Rapid Repetition = 3 over a occuring noise

The delay of the Delayed Repetition 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.

With delayed repetition, improved reliability comes at the cost of latency, not update frequency. The measurement results 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

Check, because of split from testbed

This section explains how the test setup was distributed from the PC to all the chips and how the data was subsequently returned to the PC and gives a short hands on - which steps are necessary to get ESP-NOW running on theses chips.

The developer boad ESP32 Devkit V1 used in the experiments Chapter 5 for the collection of the data can be ordered cheaply from common (most favourably Chinese) websites. [3.79€, www.aliexpress.com, 3/3/22] The most accessible way to flash a chip is using the Arduino IDE, which can be easy configured for flashing ESPs. A more precise approach is to use the IDF of Espressif itself, which can also be easily made to work by installing the toolchain and build tools and an Plugin for your favorite IDE.

zeitangabe website aliexpress?
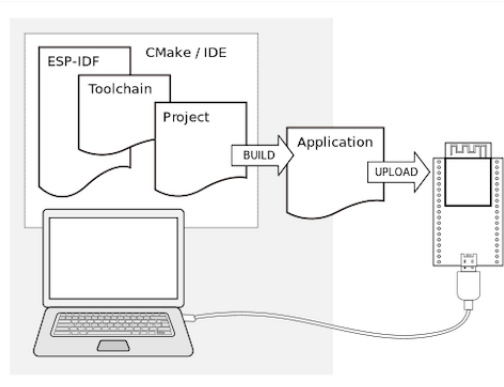
**Figure 4.6** – Flashing ESP. From the website of Espressif

Espressif has provided a user guide for the use of ESP-NOW. Unfortunately, some of the information is highly outdated, for example, it claims that broadcast is not supported. However, more detailed information can be found on the website [link]. `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html`

link und Quellenangabe: https://docs.espressif.com/idf/en/latest/esp32/getstarted/index.html

To use ESP-NOW on an ESP, only a few steps are necessary. The chip must activate Wifi 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.

for later use: ESP-NOW User Guide, V1, source: `https://www.espressif.com/en/support/documents/`

link einpflegen

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

**Listing 4.1** – Init ESP-NOW

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 within 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.

```
1 #ifndef MACLIST_H
2 #define MACLIST_H
3
4 uint8_t WES_MAC_1[6] = { 0xFC ,0xF5 ,0xC4 ,0x31 ,0x9A ,0x44 };
5 uint8_t WES_MAC_2[6] = { 0x24, 0x0A, 0xC4, 0x61, 0x19, 0x08 };
6 uint8_t BC_MAC[6]    = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
```

listing box prevent linebreaks on newpage

```
1 #include "maclist.h"
2
3 if (!esp_now_is_peer_exist(WES_MAC_1)) {
4   peer_info.channel = 13;                 // 1-14
5   memcpy(peer_info.peer_addr, WES_MAC_1, 6);
6   esp_err_t status = esp_now_add_peer(&peer_info);
7 }
8 if (ESP_OK == status) {                    // check success
9   Serial.println("[OK] Slave-peer added");
10 }
```

**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, a pointer to the payload to be transmitted 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 out. It is important to write as little code as possible in such a callback function. If you send the next package only after receiving the dispatch confirmation, then you can be sure that the packages are sent in the correct format, that the packages are sent in the correct order.

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

```
3
4    esp_err_t status = esp_now_send(WES_MAC_1,
5                                    (uint8_t *) &payload,
6                                    sizeof(payload));
7    if (ESP_OK == status) {                      // check success
8      Serial.println("[OK] ESP-NOW sending");
9    }
10
11 void onDataSent(const uint8_t *mac_addr, esp_now_send_status_t ⟍
       status) {
12   if (status == ESP_NOW_SEND_SUCCESS) {
13     Serial.println("[OK] ESP-NOW Send");
14   }
15 }
```

**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.

```
1 esp_now_register_recv_cb(OnDataRecv);
2
3 void OnDataRecv(const uint8_t *mac_addr, const uint8_t ⟍
       *incomingData, int data_len) {
4    if (incomingData[0] == 253) {              // setup data
5      applyMetaInformation(incomingData, data_len);
6      return;
7    }
8    if (incomingData[0] == 255) {              // verify data
9      applyPayload(incomingData, data_len);
10   }
11   if (incomingData[0] == 254) {              // return results
12     sendResultsToMaster();
13     return;
14   }
```

**Listing 4.4** – ESP-NOW Callback Functions

# Chapter 5

# Evaluation

## 5.1 Methodic

One challenge in programming the controller and the WESs, is the state-based approach and the fact that the WESs can only communicate restricted to 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 that the PC is connected to the microcontroller via wired Universal Asynchronous Receiver Transmitter (UART), this microcontroller is the controller. The controller then 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.
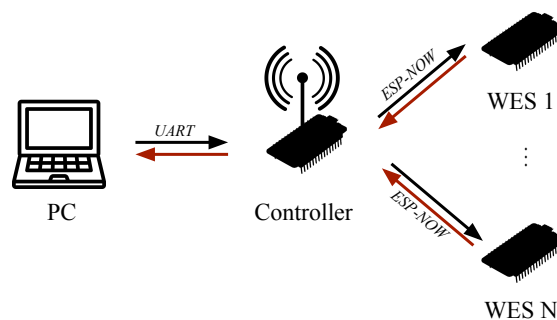


**Figure 5.1** – Data Flow for Measurement

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 | Wifi Channel of the Controller |
| WES_CHANNEL | 6 | Wifi 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_REPITITIONS. 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.
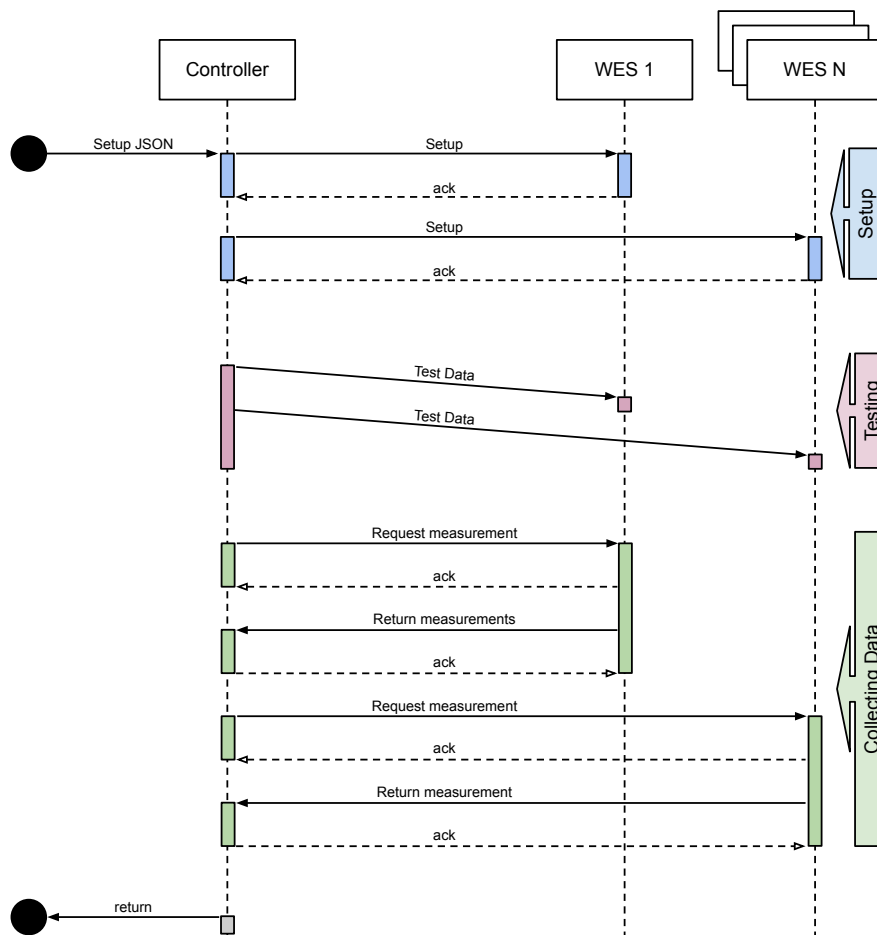


**Figure 5.2** – Sequence Diagram of the Measurment

## 5.2   Wireshark measurements

The Wireshark tool is suitable for recording traffic. In order to record packets outside
of a LAN, the WIFI 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. In Figure 5.3 it is

clear to see that 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 $696\mu s$. Adding the average backoff of a free channel ($160\mu s$) and the DIFS ($50\mu s$) gives $906\mu s$. In the calculation from Table 4.2, however, it was only $746\mu s$. The difference of $120\mu s$ can be... 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, even though I can only figure out an offset of 10 bytes.

> TOLJA!
> yes.... actually why?

> is this personal note OK?



```
▼ 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: 019f35a3dd1918fe340401ff0802030405060708090a0b0c0d0e0f10111213
          [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 Table 3.2 and Table 3.2 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 Byte 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 payload is filled with the flag ff Section 4.2 followed by the sequence number, in this example 0. The rest of the payload is filled with numbers, which are set to the value of the position in the payload.

> This is barely readable



**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 **??** 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.



**Figure 5.6** – Unicast Payload Analysis with Wireshark

## 5.3　Protocols under Study

### Slim Unicast vs Slim Broadcast

#### 5.3.0.1　Latency

#### 5.3.0.2　Reliability

> how to show 100% success ratio?

### Rapid Repetition

### Delayed Repetition

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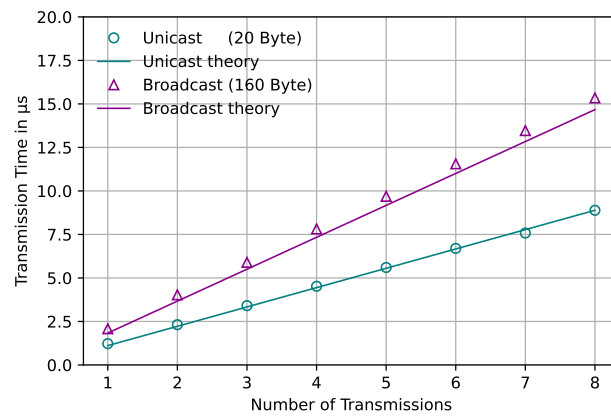
**Figure 5.7** – E.g. Transmission Time of Slim Unicast and Slim Broadcast
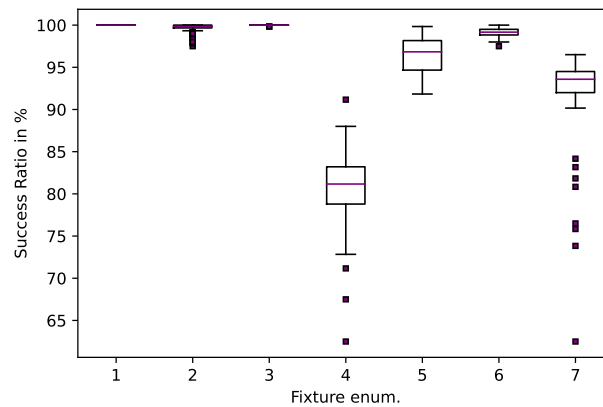


**Figure 5.8** – SR Broadcast

## 5.4  Results

Difference between Results and Discussion?

- Grafen miteinander verlgeichen?

- Which method had the best results?

# Chapter 6

# Conclusion & Discussion

- summarize again what your paper did, but now emphasize more the results, and comparisons

- write conclusions that can be drawn from the results found and the discussion presented in the paper

- future work (be very brief, explain what, but not much how, do not speculate about results or impact)

- recommended length: one page.

Why not 5GHz -> to expensive.

## Keep in Mind

- metrics (SR, Latency, ...)

- compare with art-net all the time

- wireshark

# 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 |
| **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 |
| **LLC** | Locig 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

table + listings same page?

# List of Listings

# Bibliography

[1] D. D. Husam Kareem, "The Working Principles of ESP32 and aAnalytical Comparison of using Low-Cost Microcontroller Modules in Embedded Systems Design," *4th International Conference on Circuits, Systems and Simulation*, vol. 40, no. 8, pp. 130–135, Aug. 2021. DOI: 10.1109/ICCSS51193.2021.9464217.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002. DOI: 10.1109/MCOM.2002.1024422.

[3] T. Arampatzis, J. Lygeros, and S. Manesis, "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks," in *13th Mediterrean Conference on Control and Automation*, Limassol, Cyprus, Jun. 2005, pp. 719–724. DOI: 10.1109/.2005.1467103.

[4] I. Dietrich and F. Dressler, "On the Lifetime of Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, pp. 1–39, Feb. 2009. DOI: 10.1145/1464420.1464425.

[5] 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.

# Todo list