

## **Bachelorarbeit**

Programmierung eines ESP32-Mikrocontrollers zur  
Überwachung einer Batteriespannung über die LoRaWAN  
Funktechnologie

Programming of an ESP32 microcontroller for monitoring  
a battery voltage via LoRaWAN radio technology

Ümmühan Carpisan

**Erstprüfer:** Prof. Dr.-Ing. Michael Karagounis

**Zweitprüfer:** Dipl.-Ing. Rolf Paulus

## **Kurzzusammenfassung**

### **Programmierung eines ESP32-Microcontrollers zur Überwachung einer Batteriespannung über die LoRaWAN Funktechnologie**

Im Rahmen dieser Abschlussarbeit wird die Kommunikation in LoRaWAN Funktechnologie getestet. Der Fokus ist hier die Programmierung eines ESP32-Mikrocontrollers, der LoRaWAN-Kommunikationsfähig ist und Spannungswerte an einer Batterie überwacht. Dabei ermöglicht es die Programmierung des ESP32, die Nutzdaten an einen Netzwerkserver zu senden. Der Abruf der Informationen erfolgt über einen Internetzugriff auf den Netzwerkserver. Zum Test wurde ein Labornetzgerät und eine Leiterplatte benutzt. Hierbei wurde die Spannung über einen ADC eingelesen und über LoRaWAN an den Netzwerkserver weitergeleitet.

## **Abstract**

### **Programming of an ESP32 microcontroller for monitoring a battery voltage via LoRaWAN radio technology**

As part of this thesis, communication via the LoRaWAN radio technology is tested. The focus here is on the programming of the ESP32 microcontroller that enables LoRaWAN communication and monitors the voltage values of a battery. The ESP32 programming allows the user data to be sent to a network server. Information is obtained from a network server by internet access. A laboratory power supply and printed circuit board were used for testing. Here the voltage was read in via an ADC and transferred via LoRaWAN to the network server.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Einführung in das Modulationsverfahren</b>	<b>2</b>
	2.1 <i>Analoge Modulation</i>	3
	2.2 <i>Digitale Modulation</i>	5
<b>3</b>	<b>LoRa</b>	<b>6</b>
	3.1 <i>Chirp Spread Spectrum</i>	6
	3.2 <i>Spreading Factor</i>	10
	3.3 <i>LoRa Interface</i>	13
<b>4</b>	<b>LoRaWAN</b>	<b>15</b>
	4.1 <i>LoRaWAN Kommunikation</i>	18
<b>5</b>	<b>The Things Network</b>	<b>22</b>
	5.1 <i>Endgerät Aktivierungen ABP und OTAA</i>	24
<b>6</b>	<b>I<sup>2</sup>C-Bus</b>	<b>28</b>
	6.1 <i>Elektrische Verbindung eines I<sup>2</sup>C-Busses</i>	29
	6.2 <i>Der Aufbau der I<sup>2</sup>C-Kommunikation</i>	30
	6.3 <i>Ablauf des I<sup>2</sup>C-Bus-Protokolls</i>	31
	6.4 <i>I<sup>2</sup>C-Bus Schreibmodus</i>	32
	6.5 <i>I<sup>2</sup>C-Bus Lesemodus</i>	32
<b>7</b>	<b>Werkzeuge für die Softwareimplementierung</b>	<b>33</b>
	7.1 <i>Arduino-IDE</i>	33
	7.1.1 <i>LMIC-Library</i>	33
	7.2 <i>WiFi LoRa 32 V2</i>	34
	7.2.1 <i>ESP32-Prozessor</i>	36
	7.3 <i>The Things Network</i>	42

7.4 Gateway NUCLEO-STM32F746ZG	43
7.5 Oszilloskop	45
7.6 Labornetzgerät	46
7.7 Bplaced	47
7.8 WinSCP	47
<b>8 Inbetriebnahme der ESP32 Programmierumgebung</b>	<b>48</b>
8.1 Entwicklungsumgebung Arduino	48
8.2 Analog to Digital Converter (ADC)	50
8.3 ADC Messung	51
8.4 Umwandlung des ADC-Werts in einen Spannungswert	53
8.5 GPIO-Ansteuerung	55
8.6 Spannungsteiler	56
8.7 12V Messung	58
<b>9 Programmierung der LoRa Schnittstelle</b>	<b>59</b>
9.1 Einrichtung The Things Network	59
9.2 Softwareimplementierung	62
9.3 TTN Decoder	72
9.4 Datentransfer zum Anwendungsserver	73
9.5 Webhook in TTN einrichten	74
<b>10 Inbetriebnahme Gesamtsystem</b>	<b>75</b>
10.1 Messdaten in TTN	76
10.2 Anwendungsserver	78
<b>11 Ausblick</b>	<b>79</b>
<b>12 Literaturverzeichnis</b>	<b>81</b>

# Abbildungsverzeichnis

Abbildung 1: Modulation	2
Abbildung 2: Amplitudenmodulation (AM)	3
Abbildung 3: Frequenzmodulation (FM)	4
Abbildung 4: Phasenmodulation (PM)	4
Abbildung 5: Digitale Modulation (ASK, FSK und PSK)	5
Abbildung 6: Chirp Spread Spectrum	7
Abbildung 7: CSS moduliertes und unmoduliertes Signal	7
Abbildung 8: Spektrogramm eines LoRa PHY-Paketstruktur	8
Abbildung 9: LoRa Uplink-Übertragung	9
Abbildung 10: LoRa Paketstruktur	13
Abbildung 11: LoRaWAN Architektur	18
Abbildung 12: Uplink und Downlink Kanäle (TTN EU863-870)	21
Abbildung 13: OTAA und -ABP Mode	24
Abbildung 14: Aktivierung ABP-Endgerät	25
Abbildung 15: I <sup>2</sup> C elektrische Verbindung	29
Abbildung 16: I <sup>2</sup> C Ablauf	31
Abbildung 17: WiFi LoRa 32 V2	34
Abbildung 18: Rückseite WiFi LoRa 32 V2	35
Abbildung 19: Vorderseite WiFi LoRa 32 V2	35
Abbildung 20: ESP32-Chip	36
Abbildung 21: LoRa-Chip SX1276	37
Abbildung 22: Eigenschaften der Produktvarianten SX1276/77/78/79	38
Abbildung 23: WiFi LoRa 32 V2 Pinout	41
Abbildung 24: TTN-Mapper	42
Abbildung 25: TTN-Mapper Signalstärke nach Farben	42
Abbildung 26: LoRaWAN Gateway (NUCLEO STM32F746ZG)	43
Abbildung 27: Oszilloskop (Rohde & Schwarz)	45
Abbildung 28: Labornetzgerät (Rohde & Schwarz)	46
Abbildung 29: Arduino IDE Installation	48
Abbildung 30: Grundfunktionen Arduino IDE	49
Abbildung 31: ADC-Stufenfunktion	50
Abbildung 32: Wifi LoRa 32 V2 (angeschlossen)	51

Abbildung 33: GPIO-Ansteuerung	55
Abbildung 34: Spannungsteiler Schaltung	56
Abbildung 35: TTN Registrierung 1	59
Abbildung 36: TTN Registrierung 2	60
Abbildung 37: TTN Endgerät Registrierung 1	60
Abbildung 38: TTN Endgerät Registrierung 2	61
Abbildung 39: TTN Endgerät Registrierung 3	62
Abbildung 40: TTN Decoder	72
Abbildung 41: Bplaced Einstellung 1	73
Abbildung 42: Bplaced Einstellung 2	73
Abbildung 43: Bplaced Einstellung 3	73
Abbildung 44: WinSCP Sitzung	74
Abbildung 45: TTN-Integration (Webhook) Einstellung	74
Abbildung 46: Ergebnis des Projekts	75
Abbildung 47: TTN Ergebnis	76
Abbildung 48: TTN (Gateway)	77
Abbildung 49: TTN-Mapper (Gateway aktiv)	77
Abbildung 50: TTN-Mapper (Gateway inaktiv)	77
Abbildung 51: Ergebnis auf der Webseite	78

## Tabellenverzeichnis

Tabelle 1: Modulierte Signale	9
Tabelle 2: Empfängerempfindlichkeit	19
Tabelle 3: ISM-Funkband	20
Tabelle 4: Standardkanäle EU863-870	20
Tabelle 5: Uplink-Kanäle	21
Tabelle 6: Downlink-Kanäle	21
Tabelle 7: Pinbelegung LoRa-Chip in ESP32	39
Tabelle 8: WiFi LoRa 32 V2 Technische Parameter	40
Tabelle 9: WiFi LoRa 32 V2 Elektrische Eigenschaften	41
Tabelle 10: Daten LoRaWAN Gateway (NUCLEO STM32F746ZG)	44
Tabelle 11: Technische Daten Oszilloskop (Rohde & Schwarz)	45
Tabelle 12: Technische Eigenschaften Labornetzgerät (Rohde & Schwarz)	46
Tabelle 13: SPI ESP32 - LoRa-Chip	64
Tabelle 14: DIO ESP32 - LoRa-Chip	65

## Formelverzeichnis

Formel 3.1: Anzahl Chips	8
Formel 3.2: Symbolperiode $T_s$	10
Formel 3.3: Bitrate $R_b$	10
Formel 3.4: Symbolrate $R_s$	11
Formel 3.5: Chiprate $R_c$	11
Formel 3.6: Chiprate $R_c$ (vereinfacht LoRa)	11
Formel 3.7: nominelle Bitrate $R_b$	11
Formel 3.8: Code Rate CR	11
Formel 3.9: Time On Air	14
Formel 3.10: Dauer der Präambel	14
Formel 3.11: Anzahl Präambel	14
Formel 3.12: Dauer der Payload	14
Formel 3.13: Anzahl der Payload Symbole	14
Formel 8.1: Ausgabewert Spannung	50
Formel 8.2: Basisformel Spannungsteiler	56
Formel 8.3: Spannungsteiler	56
Formel 8.4: Eingangsspannung	58



# Abkürzungsverzeichnis

---

## **A**

<b>ABP</b>	Activation By Personalization
<b>ACK</b>	Acknowledge
<b>ADC</b>	Analog to Digital Converter
<b>AM</b>	Amplitude modulation
<b>AppEUI</b>	Application Extended Unique Identifier
<b>AppKey</b>	Application Key
<b>AppsKey</b>	Application Session Key
<b>ARM</b>	Advanced RISC Machines
<b>ASK</b>	Amplitude Shift Keying

---

## **C**

<b>CAN</b>	Controller Area Network
<b>Chirp</b>	Compressed High Intensity Radar Puls
<b>CPU</b>	Central Processing Unit
<b>CSS</b>	Chirp Spread Spectrum

---

## **D**

<b>dB</b>	Decibel
<b>DevAddr</b>	Device Address
<b>DevEUI</b>	Device Extended Unique Identifier
<b>DIO</b>	Digital Input/Output

---

## **E**

<b>ETSI</b>	European Telecommunications Standards Institute
-------------	---

---

**F**

<b>FM</b>	Frequency modulation
<b>FSK</b>	Frequency Shift Keying
<b>FTP</b>	File Transfer Protocol

---

**G**

<b>GPIO</b>	General Purpose Input/Output
-------------	------------------------------

---

**I**

<b>I<sup>2</sup>C</b>	Inter-Integrated Circuit
<b>I<sup>2</sup>S</b>	Inter-IC Sound
<b>IC</b>	Integrated Circuit
<b>IDE</b>	Integrated Development Environment
<b>IoT</b>	Internet of Things
<b>IP</b>	Internet Protocol
<b>ISM-BAND</b>	Industrial, Scientific and Medical Band

---

**J**

<b>JTAG</b>	Joint Test Action Group
<b>JSON</b>	JavaScript Object Notation

---

**L**

<b>LED</b>	Light-Emitting Diode
<b>LoRa</b>	Long Range
<b>LoRaWAN</b>	Long Range Wide Area Network
<b>LPWAN</b>	Low Power Wide Area Network

---

**M**

<b>MAC</b>	Media Access Control
<b>MCU</b>	Microcontroller
<b>MOSI</b>	Master Output, Slave Input
<b>MISO</b>	Master Input, Slave Output
<b>MIC</b>	Message Integrity Code

---

**N**

<b>NwkSKey</b>	Network Session Key
----------------	---------------------

---

**O**

<b>OTAA</b>	Over The Air Activation
-------------	-------------------------

---

**P**

<b>PHY</b>	Physical Layer
<b>PHP</b>	Hypertext Preprocessor
<b>PM</b>	Phase modulation
<b>PSK</b>	Phase Shift Keying
<b>RTC</b>	Real Time Clock
<b>PWM-Kanäle</b>	Pulse Width Modulation

---

**S**

<b>SCL</b>	Serial Clock
<b>SDA</b>	Serial Data
<b>SDR</b>	Software Defined Radio
<b>SF</b>	Spreading Factor
<b>SFTP</b>	Secure File Transfer Protocol
<b>SPI</b>	Serial Peripheral Interface
<b>SRAM</b>	Static random-access memory

**SS** Slave Select

---

***T***

**TTN** The Things Network

---

***U***

**USB** Universal Serial Bus

---

***W***

**WinSCP** Windows Secure Copy

## Nomenklatur

$U_{in}$  Eingangsspannung

$U_{out}$  Ausgangsspannung

$U_{R1}$  Teilspannung

$R_2$  Widerstand unter Teilspannung

# 1 Einleitung

In Rahmen der Abschlussarbeit werden mithilfe der LoRaWAN Funktechnologie Messdaten erfasst und an einen LoRaWAN-Netzwerkserver gesendet. LoRaWAN ist eine sparsame und flächendeckende Funktechnologie in der Nachrichtentechnik. Mit dieser Technologie kann der Endnutzer viel Zeit und Ressourcen sparen. Außerdem ist diese Technologie für die Endnutzer sehr komfortabel. Der Zweck dieser Arbeit ist die Messung einer Batteriespannung. In diesem Projekt wurde ein spezieller ESP32-Mikrocontroller der Firma Espressif Systems verwendet, der einen integrierten ADC zur Erfassung der Daten besitzt und ein LoRaWAN-Interface zur Verfügung stellt. Es wurden P- und N-MOSFET Schalter für die Umsetzung eines Sleep-Modus integriert, damit das System effizienter wird. Nach der Messung der Batteriespannung werden die empfangenen Nutzdaten über LoRaWAN an einen Netzwerkserver gesendet. Motivation für die Entwicklung dieses Systems ist die Überwachung einer Boots-Batterie, um zu verhindern, dass sich die Batterie durch die minimale Nutzung des Bootes im Winter vollständig entleert. Durch das entwickelte System ist es möglich, den Batteriezustand jederzeit und ohne komplexe Eingriffe zu monitorieren. Hierbei spart der Nutzer den Aufwand, der mit der Messung der Batteriespannung vor Ort im Boot in Zusammenhang steht. Die Nutzdaten können im Netzwerkserver beobachtet werden. Zusätzlich wurde ein Anwendungsserver verwendet, der die empfangenen Nutzdaten durch ein Webinterface visualisiert. Die Softwareimplementierung erfolgt in der Entwicklungsumgebung Arduino-IDE. Mithilfe der LoRaWAN-Technologie können Daten mit einer hohen Reichweite erfasst und ermittelt werden. Die LoRa Funktechnologie ist noch im Entwicklungsstatus, besitzt ein hohes Potenzial und wird ständig erweitert.

## 2 Einführung in das Modulationsverfahren

Um Informationen zu einem beliebigen Standort zu senden oder Informationen zu erhalten, werden heutzutage Funktechnologien angewendet, die ohne Kabel Daten über die Luft übertragen. Die hier benutzten Verfahren werden Modulationsverfahren genannt. Bei den Modulationsverfahren werden Nutzsignale verändert bzw. moduliert von einem Träger übertragen [1]. Diese Nutzsignale können beispielsweise Dokumente, Musik, Video oder sonstige Daten beinhalten. Beim Senden des Nutzsignals werden Informationen in ein elektrisches Signal mit einer definierten Frequenz gewandelt. Bei der Modulation wird das Trägersignal mit definierter Frequenz vom Nutzsignal z.B. in seiner Frequenz geändert. Der Empfänger muss dieses modulierte Signal wieder in seine ursprüngliche Form zurückwandeln. Dieser Prozess wird Demodulation genannt. Für das Senden und Empfangen von Nutzdaten wird meist ein sinusförmiges Trägersignal verwendet (Abbildung 1). Das unmodulierte Trägersignal muss eine konstante Amplitude und Frequenz besitzen.

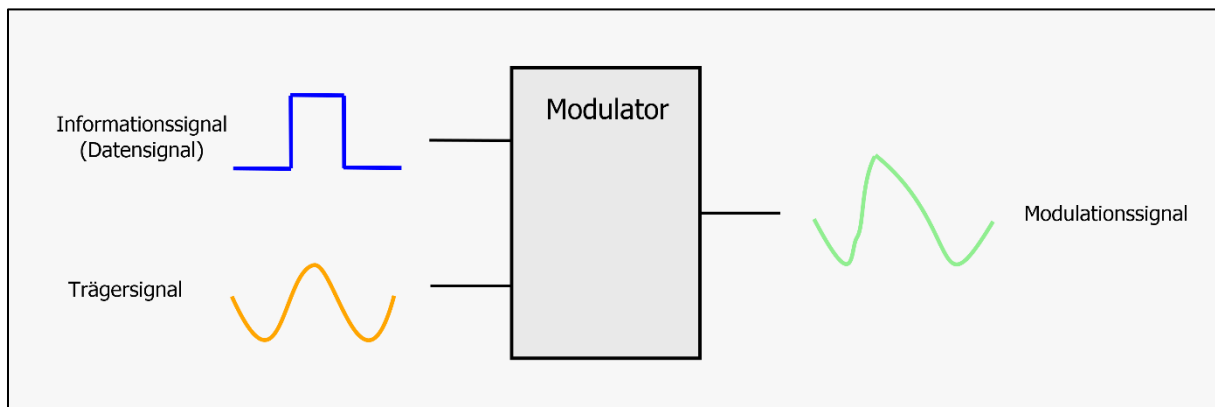


Abbildung 1: Modulation

In Abbildung 1 wird ein Informationssignal, ein Trägersignal und ein Modulationssignal dargestellt. Bei der Modulation wird das konstante Trägersignal und das empfangene Informationssignal im Modulator verarbeitet. Das Resultat ist das Modulationssignal, das dann wiederum vom Empfänger demoduliert wird. Es gibt vier Arten von Modulationsverfahren. Diese Modulationsverfahren sind die analoge Modulation, digitale Modulation, die Pulsmodulation und die Bandspreizmodulation.

## 2.1 Analoge Modulation

Die analoge Modulation umfasst die Amplitudenmodulation, Frequenzmodulation und die Phasenmodulation. Diese Modulationsarten werden häufig bei der Sprachkommunikation eingesetzt. Die Signalverläufe dieser Modulationsarten können in den Abbildungen 2, 3 und 4 betrachtet werden. Die Transformationen dieser analogen Modulationsarten sind unterschiedlich. Auf den Abbildungen ist oben das niederfrequente (NF-Signal) Informationssignal zu sehen. In der Mitte der Abbildungen ist das hochfrequente (HF-Signal) Trägersignal zu erkennen. Das dritte Signal ist das modulierte Signal.

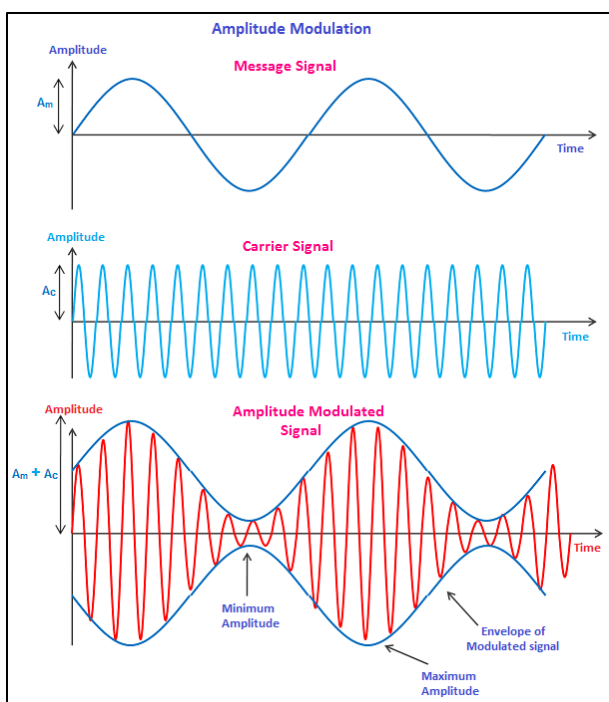


Abbildung 2: Amplitudenmodulation (AM) [2]

Bei der Amplitudenmodulation (siehe Abbildung 2) wird das Informationssignal (NF-Signal) mit dem Trägersignal (HF-Signal) multipliziert. Die Multiplikation erfolgt durch einen Oszillator. Das Trägersignal hat eine konstante Frequenz und Amplitude. Es wird vom Informationssignal gesteuert, sodass das Trägersignal seine Amplitude je nach der Stärke des Informationssignals verändert. Die Signale der Amplitudenmodulation können durch ungünstige Wetterbedingungen beeinflusst werden, was sich beim Empfänger als Rauschen bemerkbar macht.



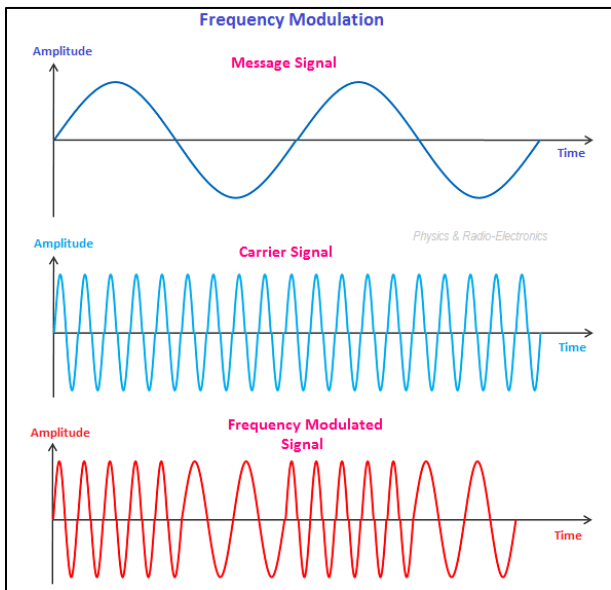


Abbildung 3: Frequenzmodulation (FM) [2]

Bei der Frequenzmodulation (Abbildung 3) wird das Informationssignal mit dem Trägersignal addiert. Dabei wird die Frequenz des Trägersignals im Rhythmus des Informationssignals verändert, während die Amplitude konstant bleibt. Die Frequenzmodulation ist amplitudenunabhängig, sodass es bei ungünstigen Wetterbedingungen keine Verfälschung des Signals entstehen.

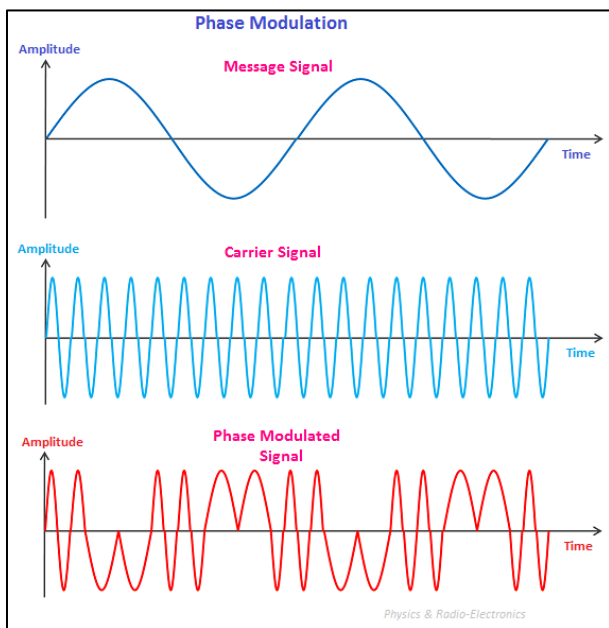


Abbildung 4: Phasenmodulation (PM) [2]

Die Phasenmodulation (Abbildung 4) ist ähnlich wie die Frequenzmodulation. Hierbei wird nur die Phase der Trägerfrequenz geändert. Die Phase des Trägersignals wird durch das Informationssignal bei der Phasenmodulation je nach Polarität des Informationssignals entweder in die eine oder in die entgegengesetzte Richtung verschoben.

## 2.2 Digitale Modulation

Die digitale Modulation ist ein Modulationsverfahren mit einem oder mit mehreren Trägersignalen. Das Modulationsverfahren mit einem Trägersignal umfasst die Amplitudenumtastung, die Frequenzumtastung und die Phasenumtastung (siehe Abbildung 5). Das Modulationsverfahren mit mehreren Trägersignalen enthält die orthogonale Amplitudenmodulation (Quadraturamplitudenmodulation) und das

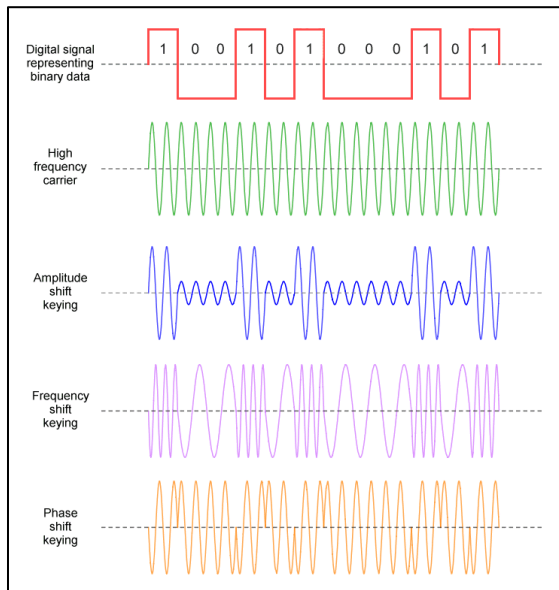


Abbildung 5: Digitale Modulation (ASK, FSK und PSK) [3]

Bei der Amplitudenmodulation ist das Trägerfrequenzsignal zeitkontinuierlich und das Informationssignal ist digital. Das Informationssignal steuert hier das Trägerfrequenzsignal. Ein Low-Pegel am Informationssignal bewirkt, dass die Amplitude des Trägerfrequenzsignals fast gleich 0 ist. Betrachtungsweise kann es auch als eine Null gesehen werden. Wechselt der Pegel des Informationssignals von Low auf High, so steigt die Amplitude des Trägerfrequenzsignals auf einen bestimmten Wert. Bei der Frequenzumtastung wird die Frequenz des zeitkontinuierlichen Trägerfrequenzsignals vom Informationssignal beeinflusst. Die Frequenz des Trägerfrequenzsignals nimmt bei einem High-Pegel des Informationssignals, einen bestimmten Wert an. Bei einem Low-Pegel wird die Frequenz des Trägerfrequenzsignals auf einen anderen Wert geändert. Bei der Phasenumtastung ist die Frequenz und die Amplitude des Trägerfrequenzsignals konstant. Ein High-Pegel am Informationssignal bewirkt, dass sich die Phase des Trägerfrequenzsignals um 180° verschiebt. Ein Low-Pegel des Informationssignals versetzt die Phase des Trägerfrequenzsignals wieder auf den Ursprungswert. Die orthogonale Amplitudenmodulation besitzt zwei Trägerfrequenzsignale, die zueinander 90° phasenverschoben sind, aber die gleiche Frequenz aufweisen. Diese Trägerfrequenzsignale werden durch zwei oder mehrere Informationssignale amplitudenmoduliert und zusammengeführt. Das orthogonale Frequenzmultiplexverfahren verwendet mehrere Trägerfrequenzsignale mit unterschiedlichen Frequenzen. Die Informationssignale werden hierbei in mehrere Signale geteilt. Diese Signale unterscheiden sich voneinander und werden in einzelne erzeugte Trägerfrequenzsignale unterteilt [3].

## 3 LoRa

LoRa (Long Range) ist eine Langdistanz Modulationstechnik [35]. Der Eigentümer dieses Protokolls ist das Unternehmen Semtech. LoRa ermöglicht eine drahtlose Kommunikation zwischen zwei Geräten, die voneinander sehr weit entfernt sind. Das verwendete Übertragungsverfahren basiert auf der Chirp Spread Spectrum (CSS) Technologie, die im Jahre 1940 entwickelt wurde [4]. Die Informationen werden mithilfe von Chirp-Impulsen durch Radiowellentechnologie übermittelt. Hierbei ist der Datentransfer bidirektional. Außerdem ist der Energiebedarf sehr gering. LoRa nutzt eine Bandbreite von 125kHz, 250kHz und 500kHz. Diese Bandbreiten können je nach Kontinent variieren. In Europa sind Bandbreiten zwischen 125kHz und 250kHz zugelassen. LoRa wird in Europa im ISM-Band im Frequenzbereich von 867MHz bis 868MHz betrieben. Die Übertragungsgeschwindigkeit beträgt hierbei 0,3 kbps bis 50 kbps. Die Benutzung dieses Modulationsverfahrens ist lizenzfrei.

### 3.1 Chirp Spread Spectrum

Chirp Spread Spectrum ist ein Modulationsverfahren, bei dem durch Frequenzspreizung die gesamte definierte Bandbreite verwendet wird. Das Wort Chirp ist eine Abkürzung für Compressed High Intensity Radar Puls [22]. Die Signale haben eine konstante Amplitude und variable Frequenz, die durch das gesamte Frequenzband linear von einem Ende zum anderen durchlaufen wird. In Abbildung 6 befinden sich zwei Diagramme, welche die Verläufe für up-chirp und down-chirp darstellen. Die x-Achse ist hier die Zeit und die y-Achse die Frequenz. Die y-Achse ist mit den Abschnitten  $f_{low}$ ,  $f_{center}$  und  $f_{high}$  beschriftet. Die Frequenz des Chirp-Impulses ist beim Anstieg (up-chirp) und beim Abstieg (down-chirp) zeitabhängig [7]. Die Geraden im up-chirp und down-chirp beschreiben den Vorgang, mit dem sich die Frequenz ändert. Die Verläufe werden Chirps bzw. auch Symbole genannt [5]. Die Chirp-Impulse werden mit orangenen Strichlinien getrennt im Diagramm dargestellt. Der Abstand zwischen den Strichlinien wird als Symbolperiode bezeichnet.

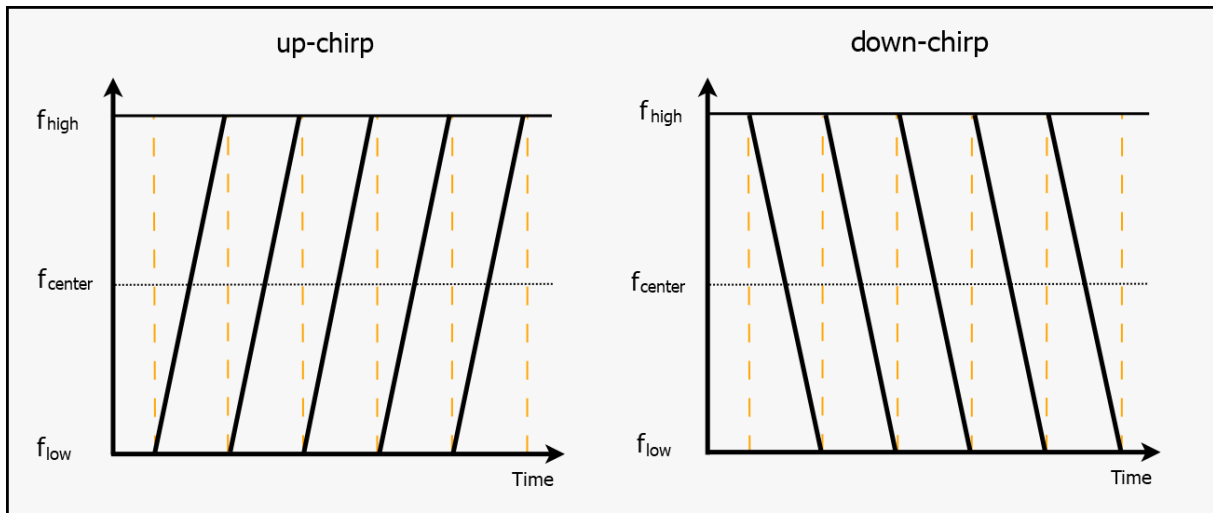


Abbildung 6: Chirp Spread Spectrum

Wenn sich die Frequenz von Low auf High ändert, so wird der Vorgang als up-chirp und im umgekehrten Fall als down-chirp bezeichnet. Wenn bei up-chirp das Symbol die höchste Frequenz  $f_{high}$  erreicht, springt die Frequenz zurück auf die niedrigste Frequenz  $f_{low}$  und der Prozess wiederholt sich kontinuierlich weiter. Der Ablauf eines down-chirps beginnt mit der maximalen Amplitude  $f_{high}$  und erreicht zeitabhängig die minimale Frequenz  $f_{low}$ . Nachdem die Frequenz  $f_{low}$  erreicht wird, erfolgt ein Sprung auf  $f_{high}$ . Dieser Prozess wird mehrmals wiederholt, bis ein komplettes Datenpaket moduliert wurde.

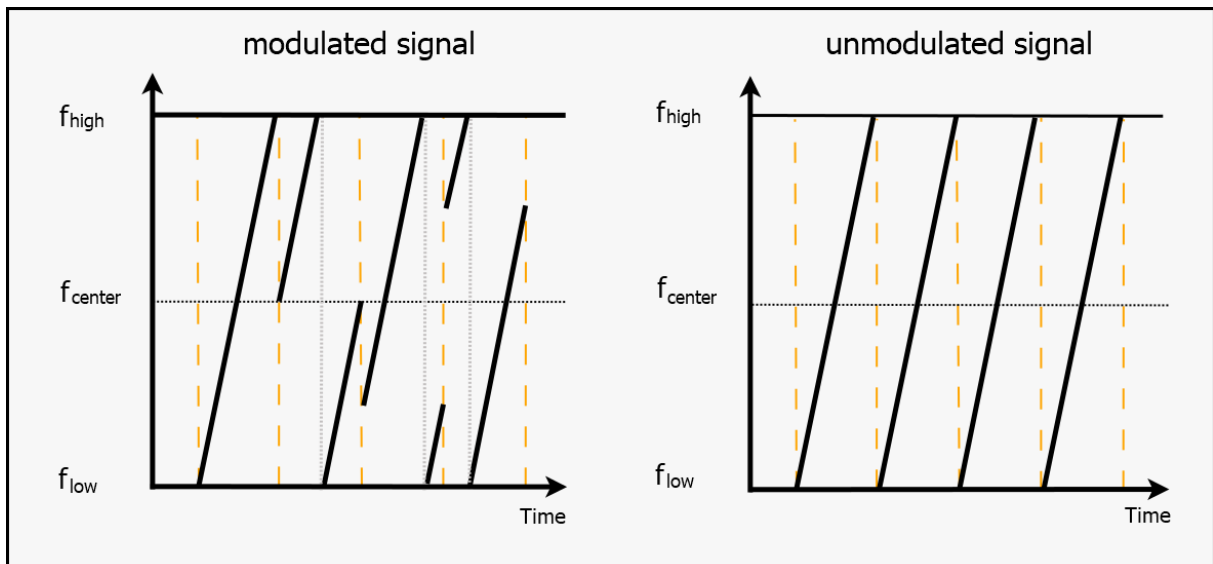


Abbildung 7: CSS moduliertes und unmoduliertes Signal

Die Abbildung 7 stellt die Verläufe der Chirp-Impulse dar. Auf der linken Seite wird ein moduliertes Signal und auf der rechten Seite unmodulierte up-chirp Impulse dargestellt. Die y-Achse ist mit den Variablen  $f_{low}$ ,  $f_{center}$  und  $f_{high}$  beschriftet. Dabei

werden die Informationen nicht in einem Schritt, sondern schrittweise moduliert. Die Informationen werden in Abhängigkeit vom Spreizfaktor (engl. Spreading Factor, SF) moduliert, der im nächsten Abschnitt 3.2 detailliert erklärt wird. Ein Symbol entsteht durch die Menge der Bits, die SF beschreibt [7]. Dabei kann SF die Werte 7, 8, 9, 10, 11 und 12 annehmen [4]. Die Symbole bestehen aus Chips, und je nach SF wird die Anzahl der Chips berechnet.

Die Anzahl der Chips berechnet sich zu [4]:

$$chips = 2^{SF} \quad (3.1)$$

In der LoRa-Datenübertragung werden Bits bei der Übertragung in physikalische Phänomene umgesetzt. In Abbildung 8 wird eine beispielhafte LoRa-Datenübertragung dargestellt, welche mithilfe der SDR (Software Defined Radio) - Software Gqrx aufgezeichnet worden ist. Mit dieser Software ist es möglich, empfangene Daten zu analysieren. Für die Analyse ist ein kompatibler RTL (Realtek RTL2832U-Chip) - SDR Stick mit einer externen Antenne für die Datenaufnahme nötig. Der RTL-Chip ist ein Signalkonverter, der die Funkwellen in Daten übersetzt und an der USB-Schnittstelle lesbar zur Verfügung stellt [6]. Die Radiosignale, welche sich in der Umgebung befinden, werden empfangen und in der SDR-Software bildlich dargestellt. Die bildliche Darstellung des zeitlichen Verlaufs der Frequenzänderung von

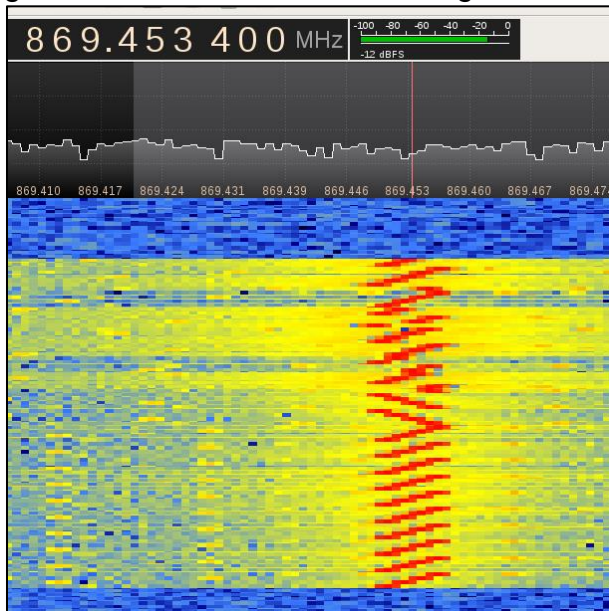


Abbildung 8: Spektrogramm eines LoRa PHY-Paketstruktur [10]

Signalen wird Spektrogramm genannt. In der Abbildung wurde die PHY (Physical Layer) - Paketstruktur in die folgenden drei Schritten unterteilt: Anfang der Bereitstellung, Auffindung des Datenbeginns und Nutzdaten. In der Abbildung 8 wird eine Datennachricht im Umfang eines Bytes gesendet. Folgende Parameter sind dazu angegeben: SF=12, Bandbreite=8 (engl. Bandwidth, BW) und Coderate=4/8 (engl. Code Rate, CR) [10]. Die Symbole werden in der Abbildung beginnend von unten nach oben als rote Striche dargestellt.

In Abbildung 9 wird im Spektrogramm die aufgezeichnete LoRa PHY-Paketstruktur dargestellt und beschriftet. Die x-Achse entspricht der Zeit und die y-Achse der Frequenz. Die Signale werden in blau dargestellt und mit Strichlinien wird die Symbolperiode kenntlich gemacht. Der Beginn einer LoRa Uplink-Übertragung besteht aus 10 Präambel up-chirps, welche den Beginn einer Kommunikation signalisieren. Anschließend folgen zwei down-chirps zu Synchronisationszwecken. Die restlichen Symbole entsprechen der Nutzlast. Die Nutzlast beinhaltet sechs modulierte Chirps. Nach der Formel 3.1 mit SF12 ergibt sich eine Chip Anzahl von 4096, wobei die Werte zwischen 0 bis 4096 in einem einzigen Symbol abgebildet werden können. Das erste Symbol fängt bei 40 Prozent des maximal möglichen Frequenzwertes an und entspricht somit 40% von 4096 bzw. 1638. Das Symbol mit dem Wert 1638 ergibt in binärer Kodierung 011001100110. Für ein verbessertes Verständnis des modulierten Signals wird eine beispielhafte Übertragung der Daten-Symbolen aus Tabelle 1 mit SF12 in Abbildung 9 dargestellt.

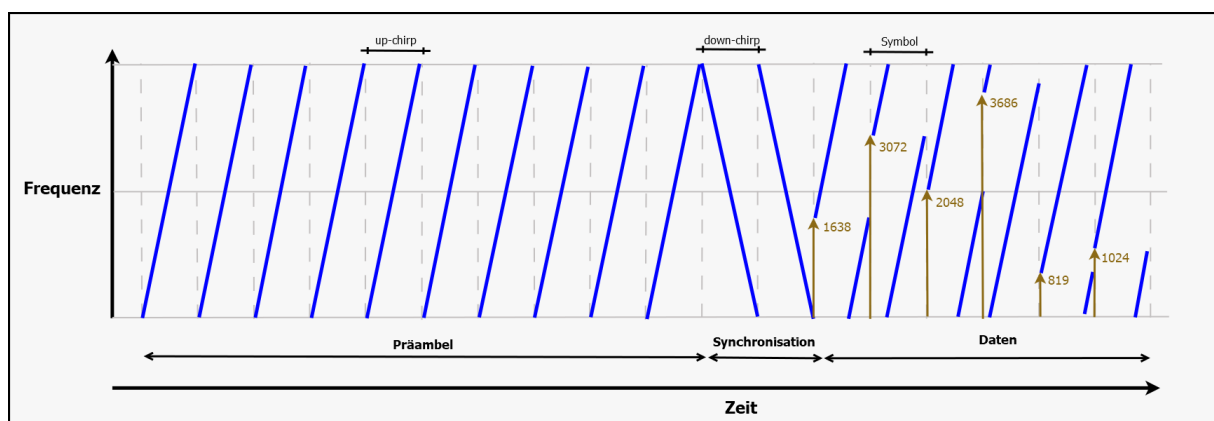


Abbildung 9: LoRa Uplink-Übertragung

### Modulierte Symbole (Abbildung 9)

Symbol	Frequenzverhältnis in %	Symbolwert	Binärwert
Symbol 1	40	1638	011001100110
Symbol 2	75	3072	110000000000
Symbol 3	50	2048	100000000000
Symbol 4	90	3686	111001100110
Symbol 5	20	819	001100110011
Symbol 6	25	1042	010000010010

Tabelle 1: Modulierte Signale

## 3.2 Spreading Factor

Der SF definiert im LoRa PHY die Anzahl der Bits, die ein Symbol kodieren kann, und liegt zwischen 6 und 12 Bits. Der SF bestimmt die Chiprate, die abhängig von der Geschwindigkeit der Datenübertragung ist [5].

Dabei beeinflusst SF die folgenden Kriterien:

- Datenrate
- Sendezeit
- Batterielaufzeit
- Empfängerempfindlichkeit

Die Lora-Modulation hat sechs SF nämlich SF7, SF8, SF9, SF10, SF11 und SF12. Der SF wird meist automatisch bestimmt und je nach Signalstärke erhöht oder verringert. Bei einer Bandbreite von 125kHz erreicht SF7 bei einem schwachen Signal eine Empfängerempfindlichkeit von -123dBm und der SF12 eine Empfängerempfindlichkeit von -137dBm. Je näher der Wert bei null liegt, desto besser ist die Empfindlichkeit. Für eine lange Batterielaufzeit ist ein niedriger SF wünschenswert. Höhere SF sorgen dafür, dass der Funksender länger aktiv bleibt, was die Batterielaufzeit reduziert [5]. Es gibt Unterschiede zwischen SF7 bis SF12 bei gleicher Datenmenge und Bandbreite. SF mit einer niedrigen Zahl erreichen eine höhere Datenübertragungsrate, aber eine niedrige Reichweite. Der schnellste SF ist SF7. Die höchste Reichweite erreicht SF12. Wenn die Zahl des SF erhöht wird, wird die Sendezeit verdoppelt und die Reichweite erhöht.

Die Symboldauer ( $T_s$ ) kann wie folgt berechnet werden. Die Angabe erfolgt dabei in der Einheit s (sekunde) [4]:

$$T_s = 2^{SF} / BW \quad (3.2)$$

Die Bitrate ( $R_b$ ) wird in der Einheit Bit/s angegeben und wird wie folgt definiert [4]:

$$R_b = SF * \frac{1}{\left[\frac{2^{SF}}{BW}\right]} \quad (3.3)$$

Die Symbolrate  $R_s$  wird in Symbols/s angegeben und berechnet sich wie folgt [4]:

$$R_s = \frac{1}{T_s} = \frac{BW}{2^{SF}} \quad (3.4)$$

Die Chiprate ( $R_c$ ) besitzt die Einheit chips/s und wird mit folgender Formel ermittelt [4]:

$$R_c = R_s * 2^{SF} \quad (3.5)$$

Die Chiprate kann durch die folgende Formel vereinfacht berechnet werden [4]:

$$R_c = BW \quad (3.6)$$

Die nominelle Bitrate für LoRa ergibt sich aus der folgenden Formel und wird in der Einheit Bit pro Sekunde angegeben [4]:

$$R_b = SF * \frac{\text{Rate Code}}{\left\lfloor \frac{2^{SF}}{BW} \right\rfloor} \quad (3.7)$$

Der Code Rate berechnet sich zu:

$$\text{Code Rate} = \frac{4}{4+CR} \quad (3.8)$$

Dabei ist der Code Rate (CR) ein ganzzahliger Wert zwischen 1- 4 und wird für die Fehlerkorrektur eingesetzt [36].

Im Folgenden werden nun in die obigen Formeln Werte eingesetzt. Dabei wird ein SF gleich 10, eine Bandbreite (BW) für Europa von 125kHz und ein CR gleich 1 festgelegt.

Nach der Formel 3.2 wird die Periodendauer wie folgt ermittelt.

$$T_s = 2^{10} / 125\text{kHz} = 0,008192\text{s}$$

Nach der Formel 3.3 ergibt sich als Resultat für die Bitrate:

$$R_b = 10 * \frac{1}{\left\lfloor \frac{2^{10}}{125\text{kHz}} \right\rfloor} = 1220,703 \text{ bit/s}$$



Die Symbolrate wird mit Hilfe von Formel 3.4 berechnet:

$$R_s = \frac{1}{0,008192s} = \frac{125kHz}{2^{10}} = 122,07 \text{ symbols/s}$$

Die Chiprate ergibt sich zu:

$$R_c = 122,07 \frac{\text{symbols}}{s} * 2^{10} = 125.000 \text{ chips/s}$$

Für die nominelle Bitrate in Formel 3.7 führen die Werte zu folgendem Ergebnis:

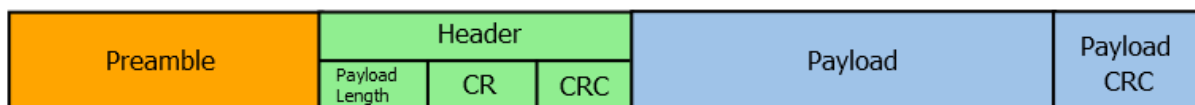
$$R_b = 10 * \frac{\frac{4}{4+1}}{\left[ \frac{2^{10}}{125kHz} \right]} = 976,562 \text{ bit/s}$$

Wie zu erkennen ist, beträgt die Bitrate ohne Fehlerkorrektur 1220,703 bit/s und die nominelle Bitrate mit Fehlerkorrektur 976,562 bit/s. Steigt der SF auf 12, beträgt die Bitrate 366,211 bit/s und die nominelle Bitrate 292,969 bit/s. Bei steigendem SF wird die Bitrate reduziert. Dabei steigt die Time On Air, und somit auch die maximal mögliche Entfernung vom Sender zum Empfänger [4].

### 3.3 LoRa Interface

Bei der Kommunikation über die Funktechnologie LoRa werden neben dem Versand von Datenpaketen auch andere Anforderungen berücksichtigt. Es ist zu gewährleisten, dass Datenpakete komplett von A nach B ohne Datenverlust gesendet werden. Dieser Prozess wird mithilfe der LoRa Paketstruktur vom Empfänger und vom Sender sichergestellt. Die LoRa Paketstruktur beinhaltet den Explicit header mode und den Implicit header mode (Abbildung 10).

#### Explicit header mode



#### Implicit header mode



Abbildung 10: LoRa Paketstruktur

Die LoRa Paket Struktur besteht aus drei Elementen. Diese Elemente sind die Präambel, der optionale Header und der Payload. Die **Präambel** dient dazu, den Beginn einer Kommunikation kenntlich zu machen und dadurch den Empfänger mit den eingehenden Daten zu synchronisieren. Bei einer Präambel werden insgesamt zwölf Symbole gesendet. Diese Symbole werden aus unmodulierten up-chirp Impulsen gebildet. Wichtig ist hierbei, dass die Anzahl der Symbole in der Präambel beim Sender und beim Empfänger identisch konfiguriert ist [8].

Im Explicit Mode wird ein Header vor der eigentlichen Payload versendet. Der Header gibt Informationen über die empfangene Nachricht an. Im Header befinden sich Datenfelder wie die Länge des Payloads, die CR und die zyklische Redundanzprüfung (CRC). Der Payload wird mittels CRC auf Fehler geprüft, die bei der Speicherung und bei der Übertragung entstanden sein könnten. Mit der CR wird der entstandene Fehler eines Payload korrigiert. Der **Payload** beinhaltet die eigentliche Information des Datenpakets [8]. Optional kann der **Payload CRC** aktiviert werden.

Das Senden der Nutzdaten benötigt eine Übertragungszeit, die auch als Time on Air bezeichnet wird. Außerdem bietet das The Things Network einen Rechner, der die Übertragungszeit ermitteln kann [9]. The Things Network ist ein LoRaWAN-Netzwerkserver, welcher für das Internet der Dinge (engl. Internet of Things, IoT) ausgelegt ist. Die Übertragungszeit wird mit der folgenden Formel bestimmt [8]:

$$T_{packet} = T_{preamble} + T_{payload} \quad (3.9)$$

Um die Übertragungszeit zu berechnen, müssen die Übertragungszeit der Präambel und Payload bekannt sein. Für die Übertragungszeit der Präambel kann folgende Gleichung verwendet werden [8].

$$T_{preamble} = (n_{preamble} + 4.25)T_s \quad (3.10)$$

$n_{preamble}$  ist dabei die Anzahl der Symbole in der Präambel und wird wie folgt bestimmt [8].

$$n_{preamble} = 12 \text{ Symbols} \quad (3.11)$$

Für die Ermittlung der Übertragungszeit des Payloads wird die Anzahl der Payload Symbole berechnet. Anschließend wird diese mit der Symbolzeit multipliziert.

$$T_{payload} = n_{payload} \times T_s \quad (3.12)$$

$$n_{payload} = 8 + \max \left( \text{ceil} \left[ \frac{8PL - 4SF + 28 + 16CRC - 20IH}{4(SF - 2DE)} \right] (CR + 4), 0 \right) \quad (3.13)$$

Die Größen in der Gleichung zur Berechnung von  $n_{payload}$  haben folgende Bedeutung [10]:

- PL: Datenmenge des Payloads in der Einheit Byte
- SF: Anzahl des Spreading Factors
- CRC: Wenn CRC aktiviert ist, ist der Wert gleich 1, andernfalls ist der Wert gleich 0
- IH: Der Wert ist gleich 1, wenn der implicit Header aktiv ist, andernfalls gleich 0

- DE: wenn LowDataRateOptimize aktiv ist, folgt eine 1, andernfalls eine 0. Die LowDataRateOptimize Option wird für die Optimierung der langsamen Datenrate (Symbolzeit  $T_s > 16\text{ms}$ ) verwendet
- CR: Dies sind Bits für die Fehlerkorrektur

Für die Berechnung der Symbolzeit  $T_s$  wird die Formel 3.2 angewendet. Nun kann die Time on Air berechnet werden.

## 4 LoRaWAN

LoRaWAN (Long Range Wide Area Network) basiert auf einem Wireless-Netzprotokoll und gehört in die Familie der Funktechnologie [13]. LoRaWAN ist im Gegensatz zu anderen Funktechnologien sehr stromsparend und besitzt zudem eine sehr hohe Reichweite. Dieses Netzprotokoll wird in der Telekommunikationsebene auf die Vermittlungsebene eingeordnet. Für die Spezifikationen der LoRaWAN-Technologie ist die Lora Alliance zuständig. Diese Spezifikation bietet eine drahtlose Kommunikation zwischen mehreren Endgeräten. LoRaWAN besitzt zudem ein Schichtprotokoll, das auf MAC (Media Access Control) basiert. Das LoRaWAN-Protokoll wurde im Jahr 2015 veröffentlicht und ist bis heute immer noch in der Entwicklungsphase. Zudem ist LoRaWAN frei zugänglich, ohne anfallende Lizenzkosten. Durch das LoRaWAN-Interface wird den Nutzer ermöglicht, die Internet of Things Anwendungen vereinfacht zu realisieren und zu testen. LoRaWAN ermöglicht eine sichere bidirektionale Kommunikation und eine Ende- zu Ende -Verschlüsselung [35]. Ziel ist es hier, bei einer hohen Reichweite Zeit und Kosten zu sparen und die Lebensqualität der Nutzer zu verbessern. In Zukunft könnte das LoRaWAN-Netzprotokoll für technische und wirtschaftliche Vorteile sorgen. Für die Anwendung des LoRaWAN wird Folgendes benötigt:

- **Endgerät**
- **Gateway**
- **Netzwerkserver**
- **Anwendung**

## Endgerät

Auf dem Endgerät sind in den meisten Fällen Sensoren integriert. Jedes Endgerät muss für die Teilnahme an einem LoRaWAN-Netzwerk ein LoRa Modul besitzen. Die Kommunikation zwischen den Endgeräten und dem LoRaWAN-Netzwerk verläuft über ein Gateway. Das Endgerät muss bei der Kommunikation eines der bidirektionalen Kommunikationsprotokolle der Klasse A, der Klasse B oder der Klasse C beherrschen, die im Folgenden vorgestellt werden. Je nach Kommunikationsart werden unterschiedliche Anwendungsfälle bearbeitet. Außerdem sind die Übertragungsgeschwindigkeiten der Kommunikationsprotokolle unterschiedlich.

Die **Klasse A** hat den niedrigsten Energieverbrauch gegenüber der Klasse B und C, weil es die meisten Zeit im Sleep-Modus verbringt. Das Endgerät ist öfters batteriebetrieben. Die Klasse A empfängt nur einen Downlink, wenn diese zuerst einen Uplink zum Server abgeschlossen hat. Daher hat diese Klasse eine hohe Downlink-Latenz. Meistens erfolgt die Anwendung der Kommunikationsart der Klasse A bei Überwachungen und bei Standortverfolgungen [11].

Die **Klasse B** öffnet zu definierten Zeiten weitere zusätzliche Übertragungsfenster. Da die Zeiten der Klasse B vorkonfiguriert sind und da sie keinen Uplink senden müssen, um einen Downlink zu empfangen, ist die Latenz bei der Klasse A geringer. Die Endgeräte mit dem Kommunikationsprotokoll der Klasse B verbrauchen mehr Energie und haben im Gegensatz zur Klasse A eine kürzere Akkulaufzeit. Geräte der Klasse B sind beispielsweise auf den Empfang von Temperaturinformationen und Daten von Verbrauchern spezialisiert [11].

Die **Klasse C** ist eine Erweiterung der Klasse A mit maximalem Übertragungsfenster. Das Empfangsfenster wird nur dann geschlossen, wenn ein Datenpaket gesendet wird. Die Geräte sind oft netzbetrieben und ermöglichen eine Kommunikation mit geringer Latenz. Aber sie verbrauchen wesentlich mehr Energie als die Klasse A. Anwendung finden Geräte der Klasse C oft bei der Steuerung von Straßenbeleuchtungen. Prinzipiell kann festgestellt werden, dass der Stromverbrauch der Endgeräte mit dem Kommunikationsprotokoll der Klasse C am größten ist [11].

## **Uplink- und Downlink Nachrichten**

Werden Datennachrichten von einem Endgerät über einen Gateway an den Netzwerkservers weitergeleitet, dann wird dies als **Uplink-Nachricht**, und der umgekehrte Fall wird als **Downlink-Nachricht** bezeichnet.

## **Gateway**

Ein Gateway ist das zentrale Gerät im LoRaWAN-Netzwerk und ist nur für die Weiterleitung der Nutzdaten zuständig. Das Gateway ermöglicht die Transferierung der Nachrichten von beliebigen LoRa Endgeräten oder vom Netzwerkservers, der diese Daten über die IP-Verbindung zur Verfügung stellt. Je nach Anwendung sind LoRaWAN-Gateways als Indoor-Gateway und als Outdoor-Gateway erhältlich. Das Outdoor-Gateway hat eine größere Reichweite als das Indoor-Gateway. Das Indoor-Gateway wird innerhalb von Gebäuden verwendet, wie z.B. bei mehrstöckigen Gebäuden oder in einem Keller. Die "Pigtail" Antenne ist entweder intern verbaut oder wird extern zur Verfügung gestellt. Das Outdoor-Gateway kann z.B. auf den Dächern von Gebäuden montiert werden. Eventuell bietet das Outdoor-Gateway nur eine interne verbaute Antenne. Natürlich ist es möglich, Indoor-Gateways auch als Outdoor-Gateways zu verwenden. Das Indoor-Gateway muss in einem Wasser- und Staub geschützte Gehäuse untergebracht werden, damit es als Outdoor-Gateway benutzt werden kann [12].

## **Netzwerkservers**

Der Netzwerkservers empfängt Nutzdaten von Gateways. Der Netzwerkservers muss vor der Anwendung eingerichtet werden. Anschließend folgt die Registrierung der Geräte.

## **Anwendungsservers**

Der Anwendungsservers wird mit dem LoRaWAN-Netzwerkservers gekoppelt. Der LoRaWAN-Netzwerkservers bietet nicht einen, sondern mehrere Anwendungsservers. Der Anwendungsservers verarbeitet die empfangenen Nachrichten und bildet diese als Messwerte bzw. als Diagramme ab. Im Anwendungsservers sind die Daten immer aufrufbar.

## 4.1 LoRaWAN Kommunikation

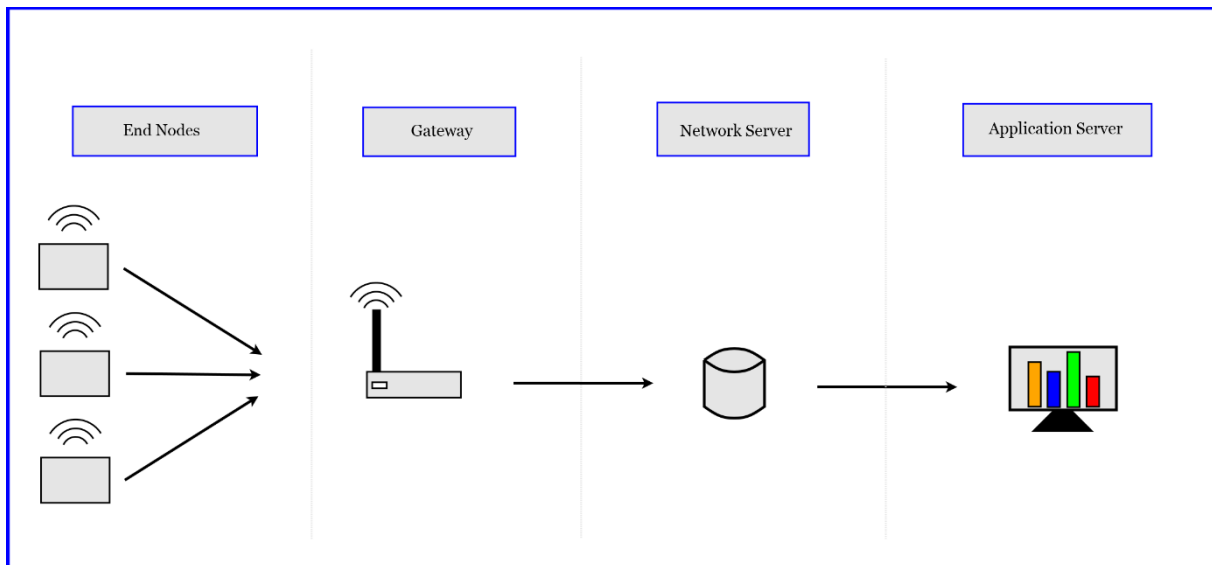


Abbildung 11: LoRaWAN Architektur

Die obige Abbildung 11 zeigt die Netzwerkarchitektur der LoRaWAN-Technologie. Im vorherigen Kapitel wurden die notwendigen Elemente für die Kommunikation beschrieben. Für die Anwendung reichen ein **LoRa-Endgerät**, ein **Gateway**, eine Internetverbindung für die **Netzwerkserver** und optional ein **Anwendungsserver** aus. Das **Endgerät** sendet Nutzdaten mittels LoRa an die Gateways, die am nächsten zum Endgerät sind. Das am nächsten liegende Gateway empfängt die Daten und leitet sie an entsprechende Netzwerkserver weiter. Anschließend werden die Daten vom Netzwerkserver an den entsprechenden Anwendungsserver gesendet.

Die Signale zwischen Gateway und Endgerät können eine Entfernung von bis zu 15 km erreichen und in ländlichen Gebieten in einem Radius von fünf km senden und empfangen. Empfehlenswert ist es, ein eigenes **Gateway** zu besitzen und dieses so nahe wie möglich an das Endgerät zu platzieren, um eine schnelle Übertragungsgeschwindigkeit zu erreichen. Natürlich spielt die Positionierung des Gateways eine große Rolle. Werden die Endgeräte beispielsweise in einem Gebäude angebracht, dann ist es sinnvoll, das Gateway horizontal in der mittleren Etage der Gebäude zu platzieren. Das Empfangs- und Sendesignal des Gateways breitet sich dann sowohl nach oben als auch nach unten im Gebäude aus. Dadurch erreichen Endgeräte im Gebäude eine bessere Signalstärke. Auch ist es möglich, das Gateway außerhalb der Gebäude zu montieren. Dabei muss die Antenne vertikal angepasst werden [14]. Um eine große Reichweite zu erzielen, sollte das Gateway so hoch wie möglich platziert

werden. Je höher das Gateway angebracht wird, desto größer wird die Reichweite. Eine gute Verbindung zwischen einem Endgerät und einem Gateway verbraucht wenig Energie und benötigt wenig Zeit für die Übertragung der Nutzdaten. Der Energieverbrauch hängt von der Entfernung zwischen Gateway und Endgerät ab. Bei maximaler Reichweite ist auch der Energieverbrauch am höchsten, was zu einer Verzögerung der Datenzustellung führt.

Die Tabelle 2 zeigt die Spreading Faktoren von 7 bis 12 bei maximaler Empfangsempfindlichkeit. Die Angaben beziehen sich auf die 125-kHz-Bandbreite. Der SF und die Signalstärke der empfangenen Nutzdaten können vom Netzwerkserver ausgelesen werden.

Spreading factor	Receiver sensitivity for bandwidth fixed at 125kHz
<b>SF7</b>	-123 dBm
<b>SF8</b>	-126 dBm
<b>SF9</b>	-129 dBm
<b>SF10</b>	-132 dBm
<b>SF11</b>	-134,5 dBm
<b>SF12</b>	-137 dBm

*Tabelle 2: Empfängerempfindlichkeit [5]*

In der LoRaWAN Netzwerk Architektur ist der Kern der **Netzwerkserver**. Hier werden nach erfolgreichem Routing vom Endgerät zum Netzwerkserver die Nutzdaten aus dem Netzwerkserver gelesen.

Der Netzwerkserver führt eine Verarbeitung der Kommunikation aus. Dabei übernimmt er Aufgaben wie z.B. die Überprüfung der Adresse des LoRa-Endgeräts, die Anpassung der Datenrate und die Auswahl des Gateways mit der stärksten RSSI (Received Signal Strength Indicator). Der RSSI ist ein Indikator für die Signalempfangsstärke. Der Netzwerkserver baut eine Kommunikation mit dem stärksten RSSI-Gateway. Das heißt, dass die Daten aller anderen Gateways vom Netzwerkserver gelöscht werden, da sie duplizierte Nachrichten des stärksten RSSI-Gateways darstellen [16].



## Regionale Parameter

LoRaWAN arbeitet auf Funkfrequenzen im 2,4-GHz-ISM-Band und 5-GHz-ISM-Band, wodurch die Benutzung lizenzfrei ist. LoRaWAN hat seine eigenen offiziellen regionalen Spezifikationen. Die verwendbaren ISM-Funkbänder variieren je nach Region. Die Regionen mit den zugehörigen ISM-Funkbändern sind in der Tabelle abgebildet [7].

Region	Frequency (MHz)
<b>Asia</b>	433
<b>Europe, Russia, India, Africa (parts)</b>	863-870
<b>US</b>	902-928
<b>Canada</b>	779-787
<b>Australia</b>	915-928
<b>China</b>	779-787, 470-510

Tabelle 3: ISM-Funkband [7]

Der ETSI-Standard [EN300.220] definiert die Nutzung des Funkspektrums für die europäischen Länder als EU863-870. Einige Länder haben mehrere Frequenzpläne, wie z.B. die Niederlande. Es gibt auch Länder außerhalb Europas, die denselben Frequenzplan wie die Länder in Europa verwenden. Ein Beispiel ist hier das Land Bahrain (BH). Einige Länder haben geringfügige Unterschiede in den Frequenzbereichen. In Georgien liegt der Frequenzbereich beispielsweise zwischen 863 und 873MHz [17].

In einem in Europa verwendeten Endgerät müssen die Kanäle des EU863-870-Bandes als Softwarecode implementiert werden. In der Tabelle 4 werden die drei europäischen Standardkanäle, die Bandbreiten, die LoRa-Datenraten und die Bitraten gezeigt.

Kanalfrequenz (MHz)	Bandbreite (kHz)	LoRa-Datenrate	Bitrate
868.10	125	DR0 - DR5	0,3 - 5 kbit/s
868.30	125	DR0 - DR5	0,3 - 5 kbit/s
868.50	125	DR0 - DR5	0,3 - 5 kbit/s

Tabelle 4: Standardkanäle EU863-870 [18]

Beim Senden von Daten vom Endgerät zum Netzwerkserver wird ein Standardkanal ausgewählt. Der Benutzer von LoRaWAN-Geräten der Version 1.0.x kann den Standardkanal nicht ändern. Die Kanäle und die entsprechenden Bandbreiten-spreizfaktoren sind in der folgenden Tabelle 5 dargestellt.

Channel	Uplink freq (MHz)	Spreading Factor & Bandwidth Range
0	868.1	SF7BW125 to SF12BW125
1	868.3	SF7BW125 to SF12BW125 and SFBW250
2	868.5	SF7BW125 to SF12BW125
3	867.1	SF7BW125 to SF12BW125
4	867.3	SF7BW125 to SF12BW125
5	867.5	SF7BW125 to SF12BW125
6	867.7	SF7BW125 to SF12BW125
7	867.9	SF7BW125 to SF12BW125
8	868.8	FSK

Tabelle 5: Uplink-Kanäle [7]

Der Downlink-Prozess fügt dem Uplink-Kanal zusätzlich zu den neun Kanälen einen weiteren Kanal hinzu (Tabelle 6). Abbildung 12 zeigt die Uplink-Frequenzen und Downlink-Frequenzen.

Download freq (MHz)	Spreading Factor & Bandwidth Range
	Uplink Channels 0-8
<b>869.525</b>	SF9BW125 (RX2 downlink only)

Tabelle 6: Downlink-Kanäle [7]

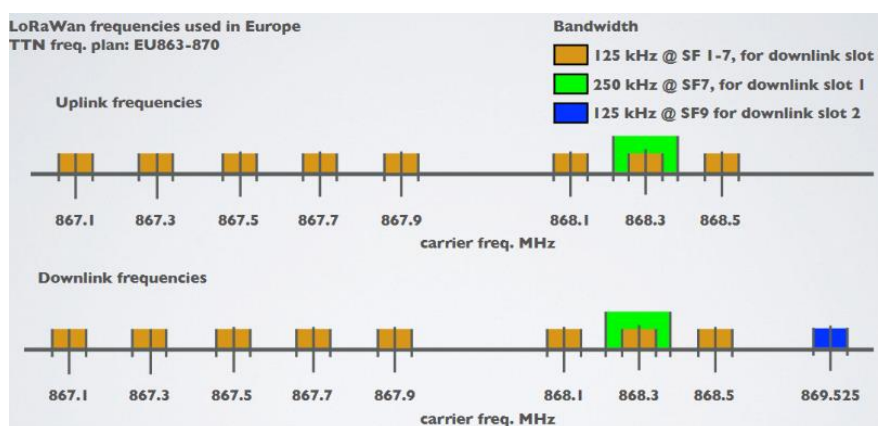


Abbildung 12: Uplink und Downlink Kanäle (TTN EU863-870) [7]

## 5 The Things Network

Dieses Kapitel beschreibt den Vorgang eines im ABP (Activation by Personalization) - Modus aktivierten Endgerät auf dem LoRaWAN-Netzwerk. Als Netzwerkservers wurde The Things Network (TTN) verwendet. The Things Network ist einer der bekanntesten LoRaWAN-Netzwerkservers. Im Jahr 2015 wurde TTN in Amsterdam gegründet. Die Gründer von TTN sind Wienke Giezman und Johan Stokking. Es ist kostenlos nutzbar und ist zudem ein Open-Source-Konzept. TTN hat derzeit weltweit 171.800 Mitglieder und verwaltet über 20.000 Gateways in 151 Ländern. In Deutschland sind bereits 2910 Gateways registriert [19].

TTN erfasst und verarbeitet LoRa-Daten von Endgeräten, die über LoRaWAN-Gateways gesendet werden. Die Nutzdaten werden optional vom Netzwerkservers an eine Anwendung übertragen und im Browser angezeigt.

### Anforderungen an das TTN

TTN erfordert zum Beginn die Erstellung eines Accounts im Netzwerk "thethingsnetwork.org". Auf der Webseite befindet sich rechts oben das Profilsymbol. Durch Anklicken des Profilsymbols erscheinen drei Auswahlkästchen. Wichtig ist hierbei das Auswahlkästchen "Console". Nach der Betätigung der Auswahlkästchen "Console" werden Netzwerkservers zur Verfügung gestellt, die sich auf den Kontinenten Europa, Amerika und Australien befinden. Die Betätigung auf einen Kontinent kann unter Anwendungen eine neue Application für TTN erstellt werden. Für die Datenverarbeitung und Datenübermittlung werden zuerst diese Informationen benötigt:

- **Application ID**
- **Application name**
- **Description**

Die drei Informationen sind zum Erstellen einer Applikation notwendig.

**Application ID** ist die Anwendungsidentifikation mit höchstens 36 Zeichen und ist frei wählbar. Dabei sind hier jedoch Sonderzeichen und Großbuchstaben nicht erlaubt.

**Application name** ist der Name für die Applikation. Der Name kann beliebig gewählt werden.

**Description** ist die Beschreibung der Anwendung und ist optional.

Als nächster Schritt folgt die Registrierung des Endgerätes. Ein Klick auf die Schaltfläche "+ Add End Device", öffnet eine neue Seite. Dabei werden zunächst folgende Parameter abgefragt:

- **Frequency Plan**
- **LoRaWAN Version**
- **Regional Parameters Version**

Der **Frequenzplan** definiert die Datenraten und die Kanäle und variiert je nach Region. Daher muss ein Frequenzplan gewählt werden, der den örtlichen Vorschriften entspricht.

Die **LoRaWAN Versionen** werden von Endgeräteherstellern festgelegt. Es gibt viele Versionskonfigurationen. Die LoRaWAN-Spezifikation 1.0.2 und die LoRaWAN-Spezifikation 1.0.3 werden am häufigsten verwendet [20].

Die **regionale Parameter Version** ist eine Kommunikationseinstellung und wird vom Gerätehersteller vorgegeben. Diese Kommunikationseinstellung ist je nach Kontinent unterschiedlich [17].

Weiterhin werden Aktivierungstypen als Radio Button abgefragt. TTN klassifiziert die Endgeräte in zwei Typen: ABP und Over-the-Air Activation (OTAA). Je nach Aktivierungsart werden dementsprechende Schlüssel angefordert.

## 5.1 Endgerät Aktivierungen ABP und OTAA

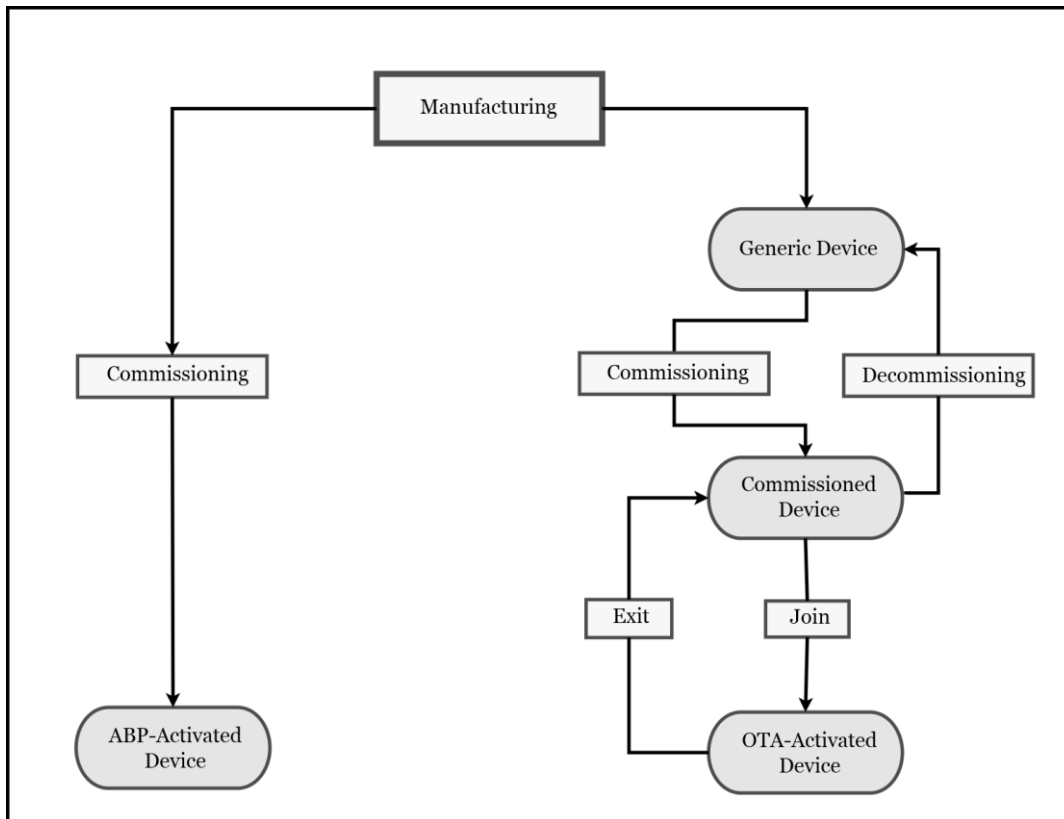


Abbildung 13: OTAA und -ABP Mode

Das Diagramm in Abbildung 13 zeigt zwei Arten von Endgeräten, links ein ABP-Endgerät und rechts ein OTAA-Endgerät. Es muss eine der beiden Optionen ausgewählt werden, um das Endgerät im TTN zu aktivieren.

Die **ABP-Aktivierung** bedeutet auf Deutsch "Persönliche Aktivierung". Dieser Modus ist mit jedem Gateway kompatibel. Die Sicherheit mit ABP-Aktivierung ist ausreichend, um Sensordaten oder Messwerte an einen Netzwerkserver zu senden. Das ABP-Verfahren verbindet sich ohne ein Login oder eine andere Anmeldeprozedur mit dem Netzwerk und unterstützt bidirektionale Kommunikation. Daher erfordert das ABP-Verfahren keinen Verbindungsschritt (Join Procedure). Das ABP-Endgerät fasst die Daten zusammen und verschlüsselt sie in ein einheitliches Datenformat (z.B. 16-Bit-Array). Der Payload wird dann wieder entschlüsselt und dem entsprechenden Applikationsserver übergeben. In der Grafik wird veranschaulicht, dass die ABP-Endgeräte direkt mit dem Netzwerk verbunden sind.

Die **OTAA-Endgeräte** führen einen sogenannten Join-Vorgang durch, um sich mit dem Netzwerk zu verbinden. Auf dem teilnehmenden Server müssen die OTAA-Schlüssel DevEUI, AppEUI und AppKey eingetragen werden. Diese Schlüssel werden auf Endgeräten und im beteiligten Server gespeichert. Bei einigen Endgeräten sind die Informationen bereits auf dem Gehäuse des Endgeräts als Aufkleber vorhanden. Wenn der Hersteller die Informationen nicht veröffentlicht, müssen die Schlüssel vom TTN-Server generiert und als Programmcode implementiert werden. Die Schlüssel DevEUI und AppEUI werden für Beitrittsanfragen (Join Request) verwendet. Diese Schlüssel werden zur Aktivierung des Endgeräts im Join-Server verwendet und sind somit auch bekannt. Die Aufgabe des Join-Servers ist es, mithilfe von AppEUI das richtige Netzwerk für das Endgerät zu finden. Der Application Session Key wird für den Applicationsserver benötigt, da die Daten weiter an den Anwendungsserver geleitet werden. Wenn die Join-Anfrage erfolgreich ist, dann wird eine Join-Accept-Nachricht an das Endgerät zurückgesendet. Die Join-Accept-Nachricht enthält die Geräteadresse (Device Adress) des Endgeräts. Die Kommunikation zwischen dem Endgerät und dem Netzwerkeserver wird dann erfolgreich initiiert [21].

Das Endgerät wurde in diesem Projekt im ABP-Modus betrieben. Zur Verständlichkeit wird zunächst die Verarbeitung im ABP-Modus beschrieben.

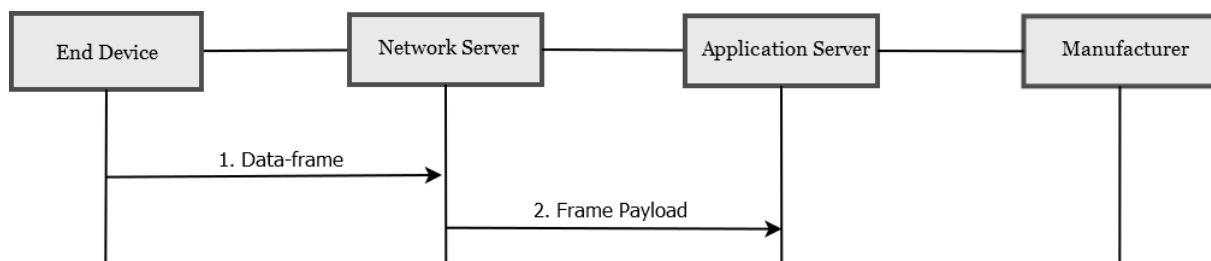


Abbildung 14: Aktivierung ABP-Endgerät

Dieses Diagramm zeigt die Funktion des ABP-Endgerätes in zwei Schritten. Zunächst muss das Endgerät mit den vom Netzwerkeserver und dem Applikationsserver benötigten Informationen erfolgreich konfiguriert werden. Anschließend werden die Nutzdaten beim Einschalten des Geräts verschickt.

## 1.Data-frame

Im ersten Schritt werden die Nutzdaten an den Netzwerkserver gesendet, sobald das Endgerät eingeschaltet wird. Diese Nutzdaten werden im Kontext eines AppsKey-Schlüssels, eines NwkSKey-Schlüssels und in den Message Integrity Code (MIC) verpackt. Der MIC wird anhand des NwkSKey-Schlüssels generiert. Nach dem Empfang der Nutzdaten im Netzwerkserver wird der NwkSKey-Schlüssel überprüft. Die Überprüfung erfolgt durch den MIC-Check im Netzwerkserver. Schlägt die Validierung fehl, wird der Vorgang abgebrochen. Dies könnte an einem unbekanntem Schlüssel oder einer missglückten MIC-Validierung liegen [21].

## 2.Frame Payload

Das Netzwerk gibt den Start einer LoRa-Sitzung erst nach Erhalt des ersten Datenpakets frei. Das Datenpaket muss vom Netzwerkserver akzeptiert werden, um die Kommunikation zwischen Netzwerkserver und Anwendungsserver starten zu können [21].

Der Aktivierungsmodus (ABP) benötigt die folgenden Informationen, bzw. der TTN-Server generiert die folgenden Informationen neu:

- **Device EUI**
- **Device address**
- **AppSKey**
- **NwkSKey**
- **End device ID**

**Device EUI** (Device Extended Unique Identifier) besteht aus 64 Bit bzw. aus 8 Byte [20]. Der Device EUI wird in der Regel vom Hersteller bereitgestellt. Andernfalls wird die Kennung serverseitig automatisch neu generiert.

**Device address** besteht aus 32 Bit und identifiziert ein Endgerät innerhalb des aktuellen Netzwerks. Dieser Schlüssel ist die einzige Adresse auf dem Netzwerkserver, die vom Benutzer nicht geändert werden kann. Als Programmcode muss dieser Schlüssel in der Entwicklungsumgebung eingetragen werden.

**AppSKey** (Application session key) ist der Anwendungssitzungsschlüssel. Der Schlüssel wird zur Sicherung der Nachrichten verwendet. Die Nutzdaten werden verschlüsselt, nachdem die Nachricht aus dem Endgerät gesendet wurde. Dieser Ablauf dient dazu, dass die Nachrichten nicht von Dritten gelesen werden können [15]. Die gesamte Verschlüsselung erfolgt in Bezug auf den NwkSKey.

**NwkSKey** (Network Session Key) ist der Netzwerksitzungsschlüssel. Dieser Schlüssel wird benötigt, um die Kommunikation zwischen dem Endgerät und dem Netzwerkserver aufbauen zu können. Die Integrität von Nutzdaten wird mittels MIC (Message Integrity Code) überprüft. Der Schlüssel-NwkSKey wird vom Netzwerkserver und von Endgeräten verwendet [15]. NwkSKey und AppSKey sind LoRaWAN-Sicherheitsschlüssel mit einer Größe von 128 Bit [22].

**End device ID** wird nach der Kennung der Schlüssel-Device-EUI automatisch ausgefüllt. Dieser Schlüssel ist eine Geräteerkennung und kann nicht auf anderen Endgeräten mit derselben Anwendung wiederverwendet werden.

Nach Eingabe aller erforderlichen Informationen kann das Endgerät registriert werden. Die Betätigung in die Schaltfläche "Register End Device" kann nach diesem Schritt erfolgen.



## 6 I<sup>2</sup>C-Bus

Der von der Firma Phillips entwickelte I<sup>2</sup>C-Bus ist ein serieller Datenbus. Die Abkürzung bedeutet I<sup>2</sup>C Inter-Integrated Circuit. Der I<sup>2</sup>C-Bus gehört in die Familie der Master-Slave Busse. Beim I<sup>2</sup>C-Bus verläuft die Kommunikation zwischen einem Mastergerät und mehreren Busteilnehmern, die als Slave agieren. Für die Kommunikation werden hier nur zwei Leitungen benötigt. Diese Leitungen sind die Datenleitung SDA (Serial Data) und die Taktleitung SCL (Serial Clock). Der I<sup>2</sup>C-Bus wird von der Firma Philips oftmals in TV-Geräten mit einzelnen integrierten Bauteilen eingesetzt. Die Datenübermittlungsrate des I<sup>2</sup>C-Busses variiert je nach an dem Bus beteiligtem Teilnehmer und kann maximal die Geschwindigkeiten 0,1 Mbit/s, 0,4 Mbit/s, 1,0 Mbit/s und 3,4 Mbit/s erreichen. Die Kommunikation verläuft in bidirektionaler Richtung. Beim I<sup>2</sup>C-Bus wird die Busteilnahme auf 128 Geräte begrenzt, da die Teilnehmendresse sieben Bit lang ist. Jedes Gerät muss also seine eigene sieben Bit lange Teilnehmendresse besitzen. Diese Adresse wird meistens vom Anwender selbst definiert. Der I<sup>2</sup>C-Bus erfordert zwingend für die Kommunikation einen Masterteilnehmer und mindestens einen Slave-Teilnehmer. Der Master steuert den I<sup>2</sup>C-Bus, indem er die Kommunikationsrichtung entweder auf Lesemodus (Read-Mode) oder auf Schreibmodus (Write-Mode) ändert. Des Weiteren übernimmt der Master die Taktleitung SCL. Der Lesemodus bewirkt, dass der Master Daten von den Slave-Schaltungsteilen empfängt. Beim Schreibmodus steuert der Master die Slave-Schaltungsteile, indem er Daten an die Teilnehmer sendet [23].

## 6.1 Elektrische Verbindung eines I<sup>2</sup>C-Busses

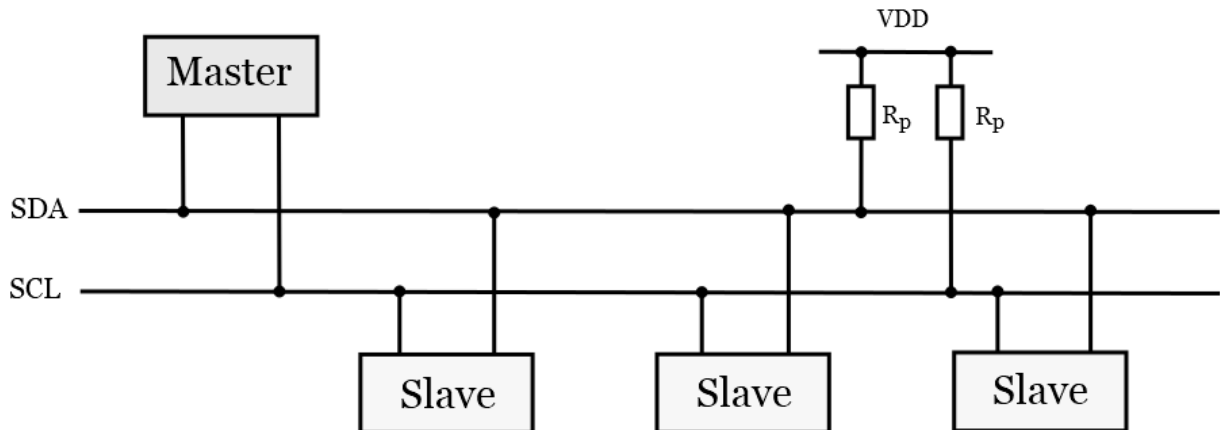


Abbildung 15: I<sup>2</sup>C elektrische Verbindung

Die Kommunikation über einen I<sup>2</sup>C-Bus erfolgt über eine elektrische Verbindung zwischen den Kommunikationsteilnehmern. Dazu müssen alle SDA-Pins und alle SCL-Pins der am I<sup>2</sup>C-Bus teilnehmenden Einheiten am Bus angeschlossen sein. Außerdem muss beim I<sup>2</sup>C-Bus ein Pullup-Widerstand zur Versorgungsspannung bestehen. Hierbei ist der Pegel Low am I<sup>2</sup>C-Bus dominant und I<sup>2</sup>C-Bus Teilnehmer konfigurieren ihre Pins auf Open-Drain.

## 6.2 Der Aufbau der I<sup>2</sup>C-Kommunikation

Der Master generiert eine Startbedingung und erzeugt auf der SCL-Taktleitung Impulse. Die am I<sup>2</sup>C-Bus angeschlossenen Slaves lesen den an der Leitung SDA anliegenden Wert, wenn der Pegel des Taktimpulses SCL den High-Wert annimmt. Der Master muss die Geschwindigkeit des Taktimpulses so anpassen, dass alle Teilnehmer am I<sup>2</sup>C-Bus mit der Geschwindigkeit arbeiten können. Das Protokoll des I<sup>2</sup>C-Busses besteht aus Datenbits, Bestätigungsbit, Startbedingung, Stoppbedingung und wiederholte Startbedingung. Das Senden eines Datenpakets am I<sup>2</sup>C-Bus beinhaltet neun Bits. Die ersten acht Bits sind die Nutzdaten und das letzte Bit ist die Bestätigung des Nutzpakets. Der Sender des Nutzpakets steuert den I<sup>2</sup>C-Bus und der Empfänger signalisiert am Bus eine Bestätigung. Das Nutzpaket wird vom Empfänger durch das Ziehen des SDA-Pegels auf Low bestätigt. Anschließend folgt nach einer Bestätigung ein weiteres Nutzpaket und dieser Vorgang verläuft so lange bis der Empfänger am I<sup>2</sup>C-Bus keine Bestätigung signalisiert. Der Master beendet die Kommunikation, indem er eine Stoppbedingung ausführt. Alle Teilnehmer überprüfen, nachdem die Stoppbedingung ausgeführt wurde, den I<sup>2</sup>C-Bus nach einer erneuten Startbedingung.

### 6.3 Ablauf des I<sup>2</sup>C-Bus-Protokolls

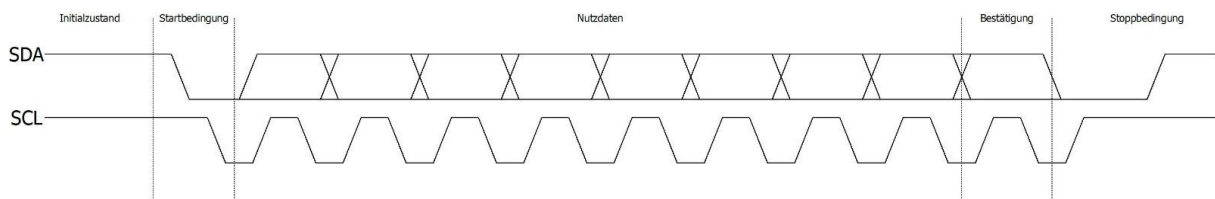


Abbildung 16: I<sup>2</sup>C Ablauf

Die Kommunikation auf dem I<sup>2</sup>C-Bus beginnt wie oben bereits erwähnt mit einer Startbedingung. Vor der Startbedingung ist der Bus im Ruhezustand und der SDA und SCL-Pegel liegen auf dem Buspegel High. Die Startbedingung erfolgt, wenn der Master die Leitung SDA auf Low zieht, während die Leitung SCL High ist. Beim I<sup>2</sup>C-Bus wird die Startbedingung nur vom Master signalisiert. Alle Busteilnehmer lesen die SDA-Leitung, nachdem eine Startbedingung ausgeführt wurde. Das Lesen der Leitung SDA erfolgt, wenn die Taktleitung einen High-Pegel aufweist. Es werden insgesamt acht Bits von allen Slaves gelesen. Die ersten am Bus signalisierten sieben Bits entspricht der I<sup>2</sup>C-Adresse. Alle Schaltungsteile vergleichen diese Adresse mit ihrer eigenen I<sup>2</sup>C-Adresse. Anschließend folgt das achte Bit. Das achte Bit stellt den Bus auf Lese- oder Schreib-Modus ein. Für das Lesen muss vom Master ein High-Pegel am Bus signalisiert werden und für das Schreiben ein Low-Pegel. Daraufhin folgt ein Bit, welches eine Bestätigung des Datenpakets beziehungsweise der I<sup>2</sup>C-Adresse darstellt. Alle nicht adressierten Teilnehmer versetzen sich in den Schlafmodus und warten erneut auf eine Startbedingung.

## 6.4 I<sup>2</sup>C-Bus Schreibmodus

Nachdem der Master die I<sup>2</sup>C-Adresse auf dem Bus übertragen hat und ein Teilnehmer diese I<sup>2</sup>C-Adresse bestätigt, kann die Kommunikation zwischen Master und Slave beginnen. Beim Schreibmodus sendet der Master Nutzdaten an den Slave. Ein Nutzpaket ist 1 Byte also 8 Bit lang. Der Slave liest jedes am Bus vorliegende Bit, wenn die Taktleitung SCL im High Zustand ist. Wenn ein Datenpaket erfolgreich vom Slave gelesen wurde, vergleicht der Slave intern das Datenpaket mit seinem Register. Es folgt nur dann eine Bestätigung vom Slave, wenn der Wert des Datenpakets mit dem Wert des Slave-Registers übereinstimmt. Das am Bus signalisierte Datenpaket kann einen der Slave Port-Pins ansprechen. Dieser Port-Pin kann durch das Datenpaket konfiguriert oder gesetzt werden. Beispielsweise kann eine LED des Slaves zum Leuchten gebracht werden, wenn das Datenpaket mit der Port Pin-Adresse des Slave-Registers übereinstimmt. Die Kommunikation dauert so lange, bis der Master keine weiteren Nutzdaten sendet und daraufhin die I<sup>2</sup>C-Kommunikation mittels Stoppbedingung beendet.

## 6.5 I<sup>2</sup>C-Bus Lesemodus

Beim Lesemodus übergibt der Master die SDA-Leitung an den Teilnehmer, der die auf dem Bus übertragenen I<sup>2</sup>C-Adresse mit seiner Busadresse verglichen und daraufhin eine Bestätigung auf dem Bus signalisiert hat. Nach der Bestätigung der I<sup>2</sup>C-Adresse ist der Master der Empfänger und der Slave der Sender des Datenpakets. Der Master muss nach dem Erhalt des Datenpakets am I<sup>2</sup>C-Bus eine Bestätigung senden. Der Slave überprüft den Wert des Bestätigungsbits und sendet weitere Nutzpakete nach dem Empfang der Bestätigung. Beispielsweise können am Slave angeschlossene Schalterstellungen oder der Wert eines Potentiometers bestimmt werden. Für das Beenden der I<sup>2</sup>C-Kommunikation darf der Master das Datenpaket nicht bestätigen. Es muss ein High-Pegel beim Bestätigungsbit folgen. Anschließend wird die Kommunikation mit einer Stoppbedingung beendet.

## 7 Werkzeuge für die Softwareimplementierung

Für die Umsetzung des Projekts wurden bestimmte Hardware- und Softwaretools benötigt. Die Programmierung erfolgte in der Entwicklungsumgebung Arduino-IDE. Zudem wurden zusätzliche Bibliotheken in die Entwicklungsumgebung Arduino-IDE eingebunden. Zur Übertragung der Messwerte wurde der ESP32-Mikrocontroller verwendet. Mithilfe von Laborgeräten wurden die Messungen und Tests durchgeführt. Die Nutzdaten wurden vom Endgerät über ein nahe gelegenes Gateway (NUCLEO-STM32F746GZ) an den Netzwerkserver gesendet.

### 7.1 Arduino-IDE

Die Arduino IDE ist eine der am weitesten verbreiteten Mikrocontroller-Entwicklungsumgebungen und Open-Source-Software. Die Entwickler sind Massimo Banzi und das Projektteam befindet sich in Ivrea, Italien. Die Programmiersprache der Arduino IDE basiert auf C/C++ [24]. Der Programmcode wird per USB-Schnittstelle auf das Modul übertragen und der Mikrocontroller per Bootloader geflasht. Arduino bietet eine große Anzahl an Bibliotheken.

Für die Entwicklungsumgebung Arduino-IDE sind im Internet verständlich definierte Informationen über Sprachreferenzen, Funktionen, Variablen und Strukturen zu finden. Arduino bietet ein Forum an, indem einzelne Themen diskutiert und gelöst werden können, welche die Implementierung eines Projektes erleichtern.

#### 7.1.1 LMIC-Library

Eine der wichtigsten Bibliotheken in der Arduino-Entwicklungsumgebung für die LoRaWAN Kommunikation ist die LMIC-Bibliothek. Diese Bibliothek stellt Funktionen für das LoRaWAN-Interface zur Verfügung. Die LMIC-Bibliothek unterstützt die LoRa Module Semtech SX1272 und SX1276. Diese Module sind auf Evaluationsboards integriert und besitzen die Kommunikationsprotokolle der Klasse A. Die Bibliothek unterstützt viele Regionalpläne wie EU 868, USA 915, AU 915, AS923, KR 920, IN 865.

## 7.2 WiFi LoRa 32 V2

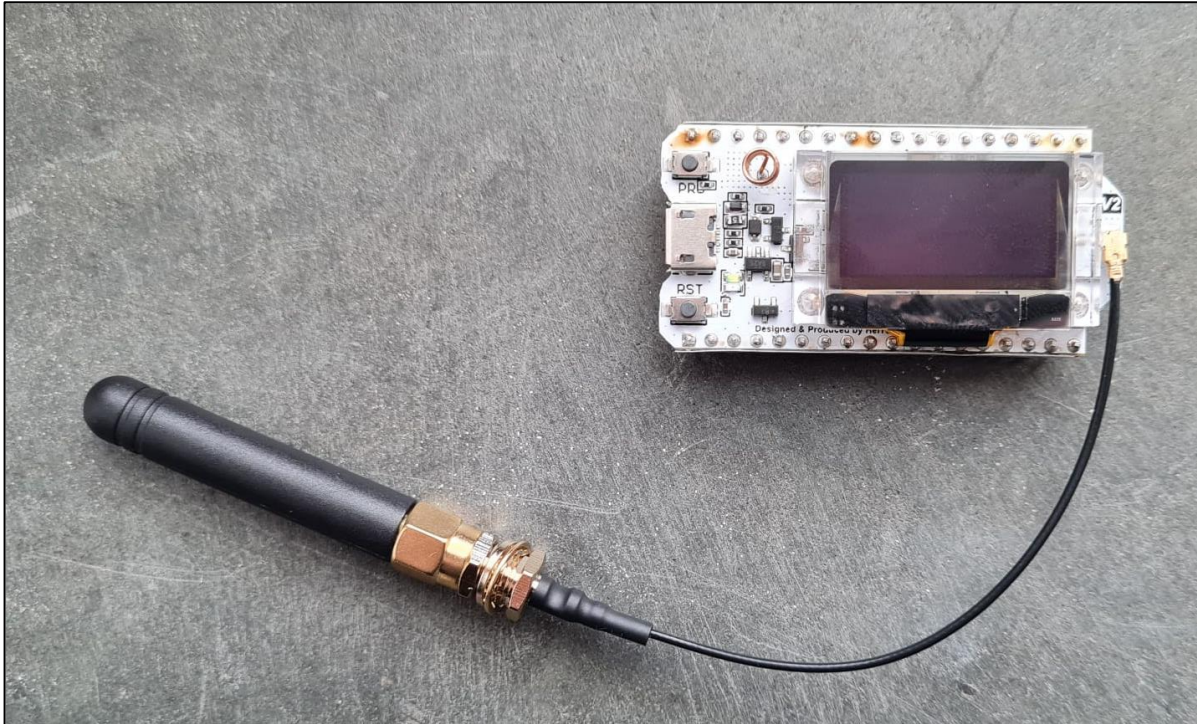


Abbildung 17: WiFi LoRa 32 V2

Als Endgerät wurde die neueste Version des integrierten LoRa-Moduls SX1276 ESP32 mit OLED-Display verwendet. Das Board WiFi LoRa 32 V2 wird vom Hersteller Heltec angeboten und stellt viele Ein- und Ausgangsports zur Verfügung. Der Prozessor dieses Moduls ist ein ESP32. Für die Funktechnik kommt der LoRa-Chip von Semtech Modell SX1276 im 868-MHz-Band EU 863-870 zum Einsatz. Die mitgelieferten Pins müssen selbst gelötet werden. Dieses Modul unterstützt die Arduino-Entwicklungsumgebung. Auf dem Modul befindet sich ein gelötetes 0,96 Zoll großes OLED-Display [28]. Neben LoRa unterstützt das Modul drahtlose Technologien wie WiFi und BLE (Bluetooth Low Energy). Das Debugging erfolgt über die JTAG-Schnittstelle. Drei UARTs (RS232, RS485 und IrDA) stehen für die serielle Kommunikation mit Geschwindigkeiten von bis zu 5 Mbit/s zur Verfügung. Die 868-MHz-Antenne ist extern und der IPEX-1-Eingang der Antenne ist an der Antennenbuchse des Moduls befestigt. Für die Spannungsversorgung gibt es eine interne Anschlussbuchse. Für diesen Anschluss ist eine 3,7-V-Lithiumbatterie geeignet. Außerdem ist es möglich, das Board mit einem 5V USB-Port zu versorgen. Das Board kann auch über den 3,3V Pin mit Spannung versorgt werden [27].

Der Preis des Produkts beträgt 30 Euro und der Lieferumfang beinhaltet:

- 1x Heltec WiFi LoRa 32 module
- 1x Pinout stickers
- 2x Header
- 1x Antenne

Vor- und Rückseite des Moduls

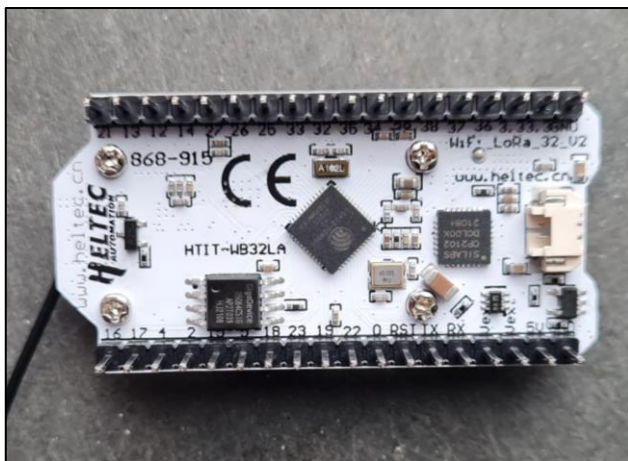


Abbildung 18: Rückseite WiFi LoRa 32 V2

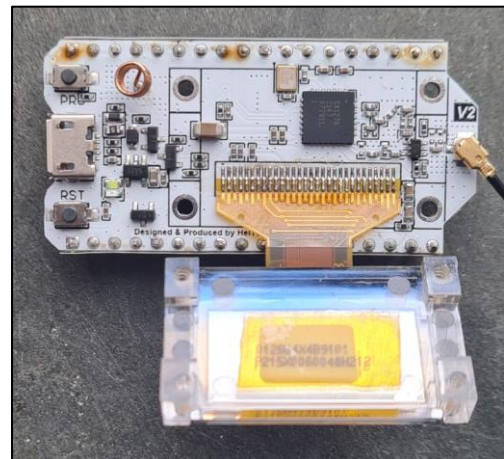


Abbildung 19: Vorderseite WiFi LoRa 32 V2

Das linke Bild (Abbildung 18) zeigt die Rückseite des Moduls. Auf der Rückseite befinden sich der ESP32-Chip, der SPI-Flash, die integrierte SH1.25-2-Batterieschnittstelle und der serielle USP-Chip CP2102. Das rechte Bild (Abbildung 19) entspricht der Vorderseite des Moduls. Auf der Vorderseite ist der LoRa-Chip SX1276, der sich unter dem Display befindet. Außerdem befinden sich auf der Vorderseite der Micro-USB-Anschluss und der IPEX-Antennenanschluss. HINWEIS: Zum Öffnen des Displays müssen die Schrauben gelöst werden.



## 7.2.1 ESP32-Prozessor



Abbildung 20: ESP32-Chip [25]

Der ESP32-Chip wird oben abgebildet (siehe Abbildung 20). Der Master-Chip ESP32 (Dual-Core 32-Bit MCU + ULP-Kern) wurde vom chinesischen Hersteller Espressif System hergestellt. Der Prozessor kann eine Frequenz von bis zu 240MHz erreichen. Der Chip verfügt intern über 520 KB SRAM und 448 KB ROM. Dabei unterstützt der Chip bis zu 4 x 16 MB externen Flash- und SRAM-Speicher. Darüber hinaus verfügt der ESP32-Prozessor über zwei I<sup>2</sup>C-Bus-Schnittstellen. Diese I<sup>2</sup>C-Pins können mit jeder I<sup>2</sup>C-Bus-kompatiblen Schnittstelle verbunden werden. Diese Pins können auf Master- oder Slave-Modus eingestellt werden. Außerdem bietet der Prozessor im Master- oder Slave-Modus eine Taktfrequenz zwischen 10kHz und 40kHz zur Verfügung [26].

Die Eigenschaften des ESP32-Prozessors lauten wie folgt [27]:

- 32-Bit Xtensa RISC-CPU: Dual-Core Mikrocontroller Tensilica Xtensa LX6
- Betriebsgeschwindigkeit von 160-240MHz
- 520 KB SRAM
- 448 KB ROM
- 16 KB SRAM (in der RTC)
- IEEE 802.11 b/g/n/e/l-WLAN
- Bluetooth 4.2
- 12-Bit-ADC mit 18 Kanälen
- 8-Bit-DAC mit 2 Kanälen
- 10 Touch-Sensoren
- Temperatursensor

- 36 GPIOs
- 4 x SPI
- 2 x I2C
- 2 x I2S
- 3 x UART
- 1 x CAN-Bus 2.0 (TWAI, kompatibel mit ISO 11898-1)
- SD-Speicherkarten-Unterstützung
- Betrieb mit 2,2-3,36 V
- RTC-Timer und Watchdog
- Hall-Sensor
- 16 PWM-Kanäle
- Ethernet-Schnittstelle
- Interner 8-MHz- und RC-Oszillator
- Kryptografische Hardware-Beschleunigung (AES, HASH, RSA, ECC, RNG)
- IEEE 802.11 Sicherheitsmerkmale

## LoRa-Chip SX1276



Abbildung 21: LoRa-Chip SX1276 [25]

In Abbildung 21 ist der LoRa-Chip SX1278 des Herstellers Semtech abgebildet. Für die LoRaWAN Kommunikation wird die SX1276-LoRa-Modulationstechnologie verwendet. Der Chip bietet eine große Reichweite an Breitbandkommunikation und hohe Störfestigkeit bei geringem Stromverbrauch. Der SX1276 kann eine Empfindlichkeit von mehr als -148dBm erreichen. Die maximale Sendeleistung beträgt bis zu +20dBm [8].

Wichtige Hauptmerkmale des verwendeten Bauteils werden im Folgenden gelistet [8]:

- LoRa-Modem
- 168dB maximales Link-Budget
- +20dBm maximale Sendeleistung
- Programmierbare Bitrate bis zu 300kbps
- Hohe Empfindlichkeit bis zu -148dBm
- Niedriger RX-Strom von 9,9mA
- Vollständig integrierter Frequenzsynthesizer mit einer Auflösung 61Hz
- FSK-, GFSK-, LoRa und OOK-Modulation
- Eingebauter Bit-Synchronisierer zur Taktrückgewinnung
- Präambel-Erkennung
- 127dB Dynamikbereich RSSI
- Automatische RF-Erkennung und Kanalerkennung (CAD) mit ultraschneller automatische Frequenzkontrolle (AFC)
- Packet Engine bis zu 256 Byte mit CRC

Das folgende Diagramm zeigt die Eigenschaften des Chiptyps SX1276, SX1277, SX1278 und SX1279. Das proprietäre (Eigentum) LoRa-Protokoll wird als ICs (Integrated Circuit) in verschiedenen Varianten vermarktet. Der Frequenzbereich des jeweiligen Landes muss berücksichtigt werden.

Part Number	Frequency Range	Spreading Factor	Bandwidth	Effective Bitrate	Est. Sensitivity
SX1276	137 - 1020 MHz	6 - 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
SX1277	137 - 1020 MHz	6 - 9	7.8 - 500 kHz	0.11 - 37.5 kbps	-111 to -139 dBm
SX1278	137 - 525 MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
SX1279	137 - 960MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm

Abbildung 22: Eigenschaften der Produktvarianten SX1276/77/78/79 [8]

Bei unveränderten Parametern gelten die folgenden elektrische Eigenschaften [8]:

- Spannungsversorgung = 3.3V
- Temperatur = 25°C
- Bandbreite (BW) = 125kHz
- Spreading Factor (SF) = 12
- Fehler Korrektur Code (EC) = 4/6
- Paket Fehlerrate (PER) = 1%
- CRC ist auf Payload aktiviert
- Ausgangsleistung = 13dBm bei der Übertragung
- Payload Größe = 64 Byte
- Präambel Anzahl = 12 Symbole (programmiertes Register PreambleLength= 8)
- Mit angepassten Impedanzen

### Pinout LoRa-Chip

In Tabelle 7 wird die Pinbelegung des ESP32 auf dem Heltec ESP32 LoRa (V2) Boards mit dem LoRa-Chip dargestellt.

ESP32 (GPIO)	LoRa-Chip SX1276
5	SCK
19	MISO
27	MOSI
18	CS
14	RESET
15	DIO0
13	DIO1
12	DIO2

Tabelle 7: Pinbelegung LoRa-Chip in ESP32

## Technische und Elektrische Daten WiFi LoRa 32 V2

Tabelle 8 zeigt die technischen Daten des ESP32-Moduls. Anschließend folgt Tabelle 9 mit den elektrischen Daten. Daraufhin wird das Pinout-Diagramm (Abbildung 23) dargestellt.

Resource	Parameter	
<b>Master Chip</b>	Esp32(240MHz Tensilica LX6 dual-core + ULP, 600 DMIPS)	
<b>Wireless Communication</b>	Wi-Fi	Bluetooth
	802.11 b/g/n	Bluetooth V4.2 BR/DER and Bluetooth LE specification
<b>Hardware Resource</b>	UART x 3; SPI x 2; I2C x 2; I2S x 1; 12-bits ADC input x 18; 8-bits DAC output x 2; GPIO x 22, GPI x6	
<b>FLASH</b>	4MB (64M-bits) SPI FLASH	
<b>RAM</b>	520KB internal SRAM	
<b>Interface</b>	Micro USBx 1; 18 x 2.54 pin x 2	
<b>Maximum Size</b>	51 x 25.5 x 10.6 mm	
<b>USB to Serial Chip</b>	CP2102	
<b>Battery</b>	3.7 Lithium(SH1.25 x2 socket)	
<b>Solar Energy</b>	x	
<b>Battery Detection Circuit</b>	√	
<b>External Device Power Control (Vext)</b>	√	
<b>Display Size</b>	0.96-inch OLED	
<b>Working Temperature</b>	-40 - 80°C	

Tabelle 8: WiFi LoRa 32 V2 Technische Parameter [28]

Electrical Features	Condition	Minimum	Typical	Maximum
Power Supply	USB powered (≥500mA)	4.7V	5V	6V
	Lithium powered (≥250mA)	3.3V	3.7V	4.2V
	3.3V (pin) powered (≥150mA)	2.7V	3.3V	3.5V
	5V (pin) powered (≥500mA)	4.7V	5V	6V
Power Consumption(mA)	WIFI Scan		115mA	
	WIFI AP		135mA	
Output	3.3V pin output			500mA
	5V pin output (USB powered only)		Equal to the input current	
	External device power control (Vext 3.3V)			350mA

Tabelle 9: WiFi LoRa 32 V2 Elektrische Eigenschaften [28]

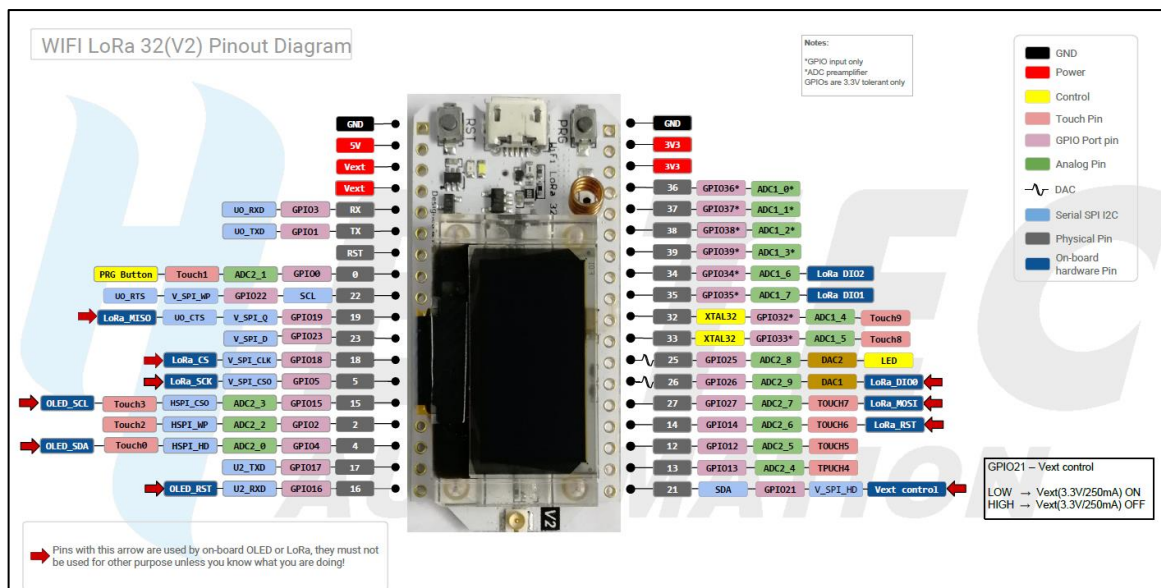


Abbildung 23: WiFi LoRa 32 V2 Pinout [28]

### 7.3 The Things Network

The Things Network ist ein Netzwerk, das auf LoRaWAN Funktechnologie basiert. Es ist ein "Low Power Wide Area Network" (LPWAN), mit dem sich Teilnehmer ohne WLAN, Mobilfunk oder Bluetooth mit dem Internet verbinden können. Das besondere an TTN ist, dass der Dienst kostenlos angeboten wird und Tests im Rahmen eines Projekts mittels einer übersichtlichen Benutzeroberfläche für den Benutzer stark vereinfacht werden. Die Umsetzung von IoT-Projekten sowie der Test und die Entwicklung verschiedener Projekte ist das Hauptziel der Community. TTN vermarktet auch seine eigenen Endgeräte (Sensoren) und Gateways unter der Marke „The Things“. Außerdem stellt der Netzwerkservers eine zusätzliche Webseite mit einer

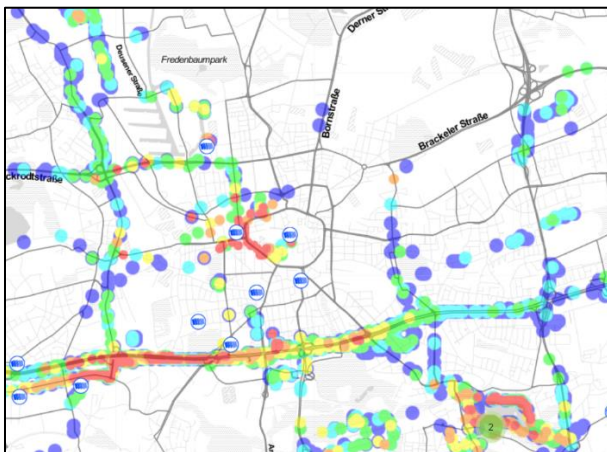


Abbildung 24: TTN-Mapper

Weltkarte bereit, welche die Standorte und die Lage der Gateways zeigt (siehe Abbildung 24). Darüber hinaus bietet TTN eine Forum-Webseite zur Diskussion einzelner Themen an.

Gateways werden als Punkte auf dem Mapper farblich abgebildet. Jeder Punkt zeigt die Signalstärke an einem Ort.

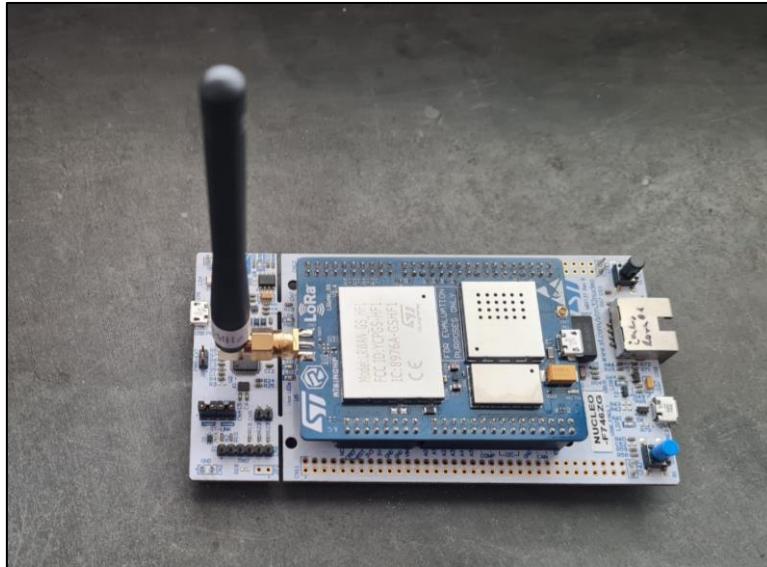
Colour	Signal
Red	> -100 dBm
Orange	-100 - -105
Yellow	-105 - -110
Green	-110 - -115
Cyan	-115 - -120
Blue	< -120 dBm

strong →  
← weak

Die Signalstärke wird in sechs Farben von Rot: >-100dBm (starkes Signal) bis Blau: <- 120dBm (schwaches Signal) eingeteilt (siehe Abbildung 25). Durch Anklicken eines Gateway auf der Weltkarte werden einige zusätzliche Eigenschaften angezeigt. Beispielsweise wird der Gerätenamen sowie das Datum und die Uhrzeit angegeben. Außerdem wird die Aktivität des Moduls angezeigt [29].

Abbildung 25: TTN-Mapper Signalstärke nach Farben

## 7.4 Gateway NUCLEO-STM32F746ZG



*Abbildung 26: LoRaWAN Gateway (NUCLEO STM32F746ZG)*

Das Modul NUCLEO-STM32F746ZG der Firma STMicroelectronics wurde Gateway fähig gemacht und in diesem Projekt eingesetzt. Dieses Modul besitzt einen leistungsstarken ARM Cortex-M7 32-Bit-RISC-Kern. Der Kern kann eine Frequenz bis zu 216MHz erreichen. Außerdem weist Cortex-M7 eine Gleitkommaeinheit mit einfacher Genauigkeit auf, der alle ARM Befehle zur Verarbeitung der Daten und Datentypen unterstützt. Das Modul besitzt mehrere I/O Pins, die sich auf der rechten und linken Seite des Boards befinden. Zudem bietet das Modul eine LAN-Schnittstelle, eine schwarze Hardware-Reset-Taste und eine blaue Taste. Die blaue Taste kann beispielsweise als Schalter zum Test des Programmcodes eingesetzt werden. Für das Debugging wird eine elektrische Verbindung zwischen einem Computer und diesem Modul benötigt. Der Anschluss für das Debugging befindet sich gegenüber der LAN-Schnittstelle. Das aufsteckbare ST-NUCLEO-LoRa-GW Modul wurde auf das Board NUCLEO-STM32F746ZG platziert. Für die Inbetriebnahme dieses aufsteckbaren Moduls wird eine externe 5V-Spannungsversorgung benötigt. Für die LoRaWAN Anwendung wurde ein leistungsfähiges LoRa-Basisband-Verfahren SX1301 vom Hersteller Semtech-Corporation verwendet. Die bidirektionale Kommunikation mit den Geräten in Klasse A und C wird hier unterstützt [30].



Das Modul deckt die LoRaWAN-Spezifikationen vollständig ab. Durch einen LAN-Anschluss kann das Modul NUCLEO-STM32F746ZG mit einem LoRaWAN-Server oder einem anderen spezifischen Server verbunden werden. Die Hardwaredetails zu diesem Board befinden sich in der Tabelle 10.

Peripherie		STM32F746ZG
Flash-Speicher		1024 KB
SRAM		320 KB
Timer	Allgemeiner Zweck	10
	Erweiterte Kontrolle	2
	Basis	2
	Niedrige Leistung	1
Kommunikations-schnittstellen	SPI/I <sup>2</sup> S	6/3 (Simplex)
	I <sup>2</sup> C	4
	USART/UART	4/4
	CAN	2
	USB OTG FS	Ja
	USB OTG HS	Ja
	SAI	2
	SPDIFRX	vier Eingänge
	SDMMC	Ja
	GPIOs (I/O Pins)	
FMC Speichercontroller		Ja
Ethernet		Ja
Zufallszahlgenerator		Ja
Kameraschnittstelle		Ja
Chrom-ART Beschleuniger (DMA2D)		Ja
LCD-TFT		Ja
12-Bit ADC		3
Anzahl an Kanälen		24
12-Bit DAC		Ja
Anzahl an Kanälen		2
Prozessorfrequenz (CPU)		216MHz
Betriebsspannung (UB)		1.7 zu 3,6V

*Tabelle 10: Daten LoRaWAN Gateway (NUCLEO STM32F746ZG) [30]*

## 7.5 Oszilloskop

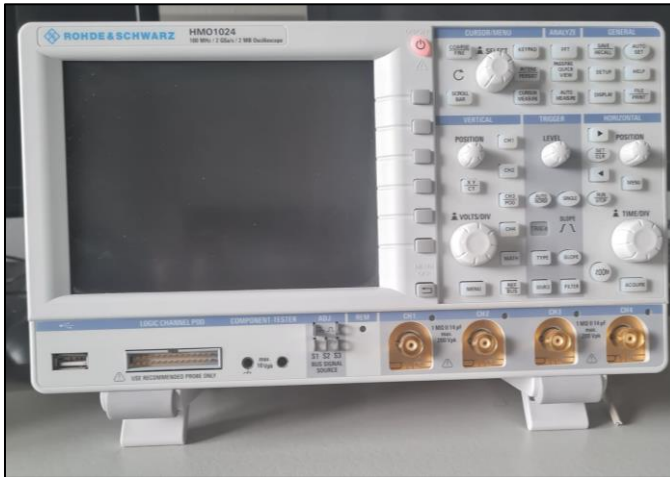


Abbildung 27: Oszilloskop (Rohde & Schwarz)

Die Analyse der elektrischen Signale wurde unter Verwendung eines Oszilloskops durchgeführt. Dieses Oszilloskop wurde von Rohde & Schwarz hergestellt und ist Teil der HMO-Serie. Die Displaygröße beträgt 165mm und hat eine Vertikalauflösung von 8-bit. Die Bandbreite des Oszilloskops beträgt 100MHz. Das elektrische

Signal wird als Graph auf dem Oszilloskop dargestellt. Signale können analysiert bzw. beobachtet werden. Beispielsweise werden Impulse von der I<sup>2</sup>C-Schnittstelle (SCL und SDA) auf dem Oszilloskop angezeigt.

Unten werden die technischen Daten aufgelistet:

Eigenschaft	Wert
Bandbreite	100MHz
Serie	HMO
Stromversorgung	Netzbetrieb
Anzeigentyp	Farb
Vertikalempfindlichkeit max.	1mV/div
Zeitbasis min.	2ns/div
Display Größe	165mm
Vertikalauflösung	8 bit
Abmessungen	285 x 175 x 140mm
Eingangskapazität	13 pF
Gewicht	2.5kg
Eingangsimpedanz	1 MΩ
Zeitbasis max.	50s/div
Anstiegszeit	<3.5ns
Sicherheitskategorie	CAT II 240 V
Messkategorie	CAT II
Sicherheitsspannung	240V

Tabelle 11: Technische Daten Oszilloskop (Rohde & Schwarz) [31]

## 7.6 Labornetzgerät



Abbildung 28: Labornetzgerät (Rohde & Schwarz)

Das Foto zeigt das Labornetzgerät des Herstellers Rohde & Schwarz, Modell HMC8041. Das Labornetzgerät versorgt Schaltungen, Platinen oder einzelne Bauteile mit einer definierten Spannung. Da die Aufgabe des Projekts die Überwachung einer 12V Autobatterie ist, wurde statt der 12V Autobatterie ein

Labornetzgerät mit 12V verwendet. Für die Durchführungen von Tests wurde das Labornetzgerät an den ADC-Eingang des Moduls angeschlossen. Dabei wurden die Werte des ADC beobachtet.

In der Tabelle 12 werden die technischen Eigenschaften dargestellt:

Eigenschaft	Wert
<b>Ausgangsspannung</b>	0 bis 30V
<b>Ausgangsstrom</b>	10A
<b>Tischnetzteilart</b>	Programmierbares Gleichspannungsnetzteil
<b>Anzahl der Ausgänge</b>	1
<b>Typ</b>	Digital
<b>Nennleistung</b>	100W
<b>Versorgungsspannung</b>	100 - 240V ac
<b>Anzahl der Anzeigen</b>	3
<b>Eingangsanschluss</b>	Wago-Stiftleiste
<b>Ausgangsanschluss</b>	4 mm Sicherheits Anschluss
<b>Temperatur min.</b>	0°C
<b>Temperatur max.</b>	+40°C
<b>Digital/Analog</b>	Digital

Tabelle 12: Technische Eigenschaften Labornetzgerät (Rohde & Schwarz) [32]

## 7.7 Bplaced

In diesem Projekt wurden die erfassten Messdaten auf einer erstellten Webseite als Benutzeroberfläche abgebildet. Hierfür wurde der Webserver Bplaced verwendet. Zunächst erfordert Bplaced eine Registrierung auf deren Webseite. Anschließend kann eine eigene Domain erstellt werden. Für die Datenübertragung ist ein FTP-Zugang (File Transfer Protocol) erforderlich. Der FTP-Zugang wird im Konto Bplaced registriert bzw. hinzugefügt. Um die Nutzdaten auf der erstellten Webseite zu sehen, wird ein FTP-Programm benötigt. Im FTP-Client werden die erforderlichen PHP-Daten hochgeladen. Dazu wurde die Software WinSCP eingesetzt. Für Domains mit wenig Speicherplatz steht der Bplaced Webserver kostenlos zur Verfügung.

## 7.8 WinSCP

Windows Secure Copy (kurz: WinSCP) ist eine kostenlose SFTP- und FTP-Client-Software. Diese Software erlaubt den Zugriff auf FTP-Server. Windows Secure Copy ermöglicht, Dateien im PHP-Format auf die selbst erstellte Webseite hochzuladen oder zu löschen.

Für den Start in WinSCP ist mindestens eine Sitzung notwendig. Die Anmeldedaten der Sitzung werden vom Webserver (FTP- Zugang von Bplaced) übernommen, danach folgt die Anmeldung am FTP-Server. Nach erfolgreicher Verbindung mit dem FTP-Server wird die Sitzung gestartet und die PHP-Dateien können hochgeladen werden.

## 8 Inbetriebnahme der ESP32 Programmierumgebung

In diesem Kapitel werden Testdurchführungen und Grundlagen des Projekts beschrieben. Dazu wurden einige der in Kapitel 7 beschriebenen Werkzeuge verwendet. Das ESP32-Board wurde eingesetzt, um Tests mit dem ADC, Spannungsmessungen bis zu 3,3V auf der Leiterplatte und die Steuerung von GPIOs durchzuführen. Anschließend wurde ein Spannungsteiler aus zwei Widerständen für die Messung von 12V eingesetzt. Für die Berechnung des Widerstands wurde eine Formel hergeleitet.

### 8.1 Entwicklungsumgebung Arduino

Die Softwareimplementierung dieses Projekts erfolgt auf der Entwicklungsumgebung Arduino-IDE. Im ersten Schritt erfolgt die Installation der Arduino Entwicklungsumgebung. Die Entwicklungsumgebung kann von der Webseite von Arduino heruntergeladen werden. Auf der Homepage befinden sich die aktuellen Versionen für Betriebssysteme wie Windows, Linux und Mac OS X. Die Arduino IDE ist Open-Source und ist somit für den privaten Gebrauch kostenlos verfügbar.

Nach erfolgreicher Installation werden Treiber für das ESP32-Modul installiert. Anschließend steht das ESP32-Modul mit seinen Funktionen vollständig zur Verfügung. Im nächsten Schritt muss die Entwicklungsumgebung mit dem ESP32-Board gekoppelt werden. Dazu wird die Arduino IDE gestartet und der Menüpunkt Datei mit dem Unterpunkt Voreinstellung ausgewählt. Es erscheint ein neues Fenster mit vielen Optionsmöglichkeiten. Für das Koppeln des ESP32-Moduls mit der Entwicklungsumgebung Arduino IDE muss im Textfeld "Zusätzliche Boardverwalter-URLs" der Weblink "[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)" eingetragen werden. Danach wird das Voreinstellungsfenster mit Betätigung des OK-Buttons geschlossen.

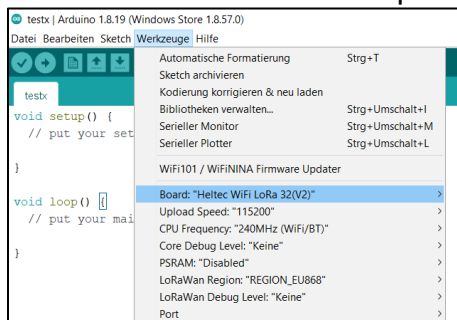


Abbildung 29: Arduino IDE Installation

Im nächsten Schritt wird in der Menüleiste das Menü Werkzeuge ausgewählt. Anschließend wird der Menüpunkt "Boardverwalter", der sich in der

Menüleiste "Board" befindet, ausgewählt. Danach öffnet sich ein neues Fenster mit einem Suchfeld. In das Suchfeld wird Esp32 eingetragen, und es erscheint eine Auswahlbox von ESP32 mit einem Installationsbutton. Nun wird der Installationsbutton betätigt. Im nächsten Schritt erscheint in der Menüleiste „Board“ unter dem Menüpunkt "Boardverwaltung" das neue Menüfeld "ESP32 Arduino". In diesem befindet sich eine Liste aller unterstützten ESP32-Module. In dieser Menüliste wird "Heltec WiFi LoRa 32(V2)" ausgewählt. Danach wird das Board mit einem USB-Kabel mit dem Computer verbunden. Unter dem Menüpunkt "Werkzeuge" wird der richtige Anschluss (Port) ausgewählt. Nun kann die Programmierung beginnen.

## Grundfunktionen Arduino IDE

```
void setup() {  
  Anweisung  
}  
  
void loop() {  
  Anweisung  
}
```

In Abbildung 30 ist die grobe Programmstruktur dargestellt. In diesem Code-Auszug ist die minimal erforderliche Funktionalität der Arduino-Programmierungsumgebung enthalten. Die Funktion `setup()` führt die Anweisungen einmal beim Start aus, und die Funktion `loop()` ist eine Schleife, die endlos wiederholt wird.

Abbildung 30: Grundfunktionen Arduino IDE

### Die Funktion `setup()`

Die Funktion `setup()` sollte immer im Programmcode vorhanden sein, auch wenn keine Anweisungen erforderlich sind. Die Aufgabe dieser Funktion ist die Initialisierung von Mikrocontroller-Funktionen und Werten. In der Funktion `setup()` kann z.B. ein Pin eines Mikrocontrollers durch die Funktion `pinMode()` als Ein- oder Ausgang konfiguriert werden.

### Die Funktion `loop()`

Nach der Funktion `setup()` wird die Funktion `loop()` ausgeführt. Der in der Funktion `loop()` enthaltene Code, wird ständig wiederholt. Dadurch können z.B. LEDs am Mikrocontroller ständig mit einer Funktion ein- oder ausgeschaltet werden. Die Funktion `loop()` stellt den Hauptbestandteil von Arduino-Programmen dar.

## 8.2 Analog to Digital Converter (ADC)

Die Spannungsmessungen in diesem Projekt erfolgten über den ADC-Eingang des ESP32-Moduls. Der ADC wird verwendet, um die analoge Spannung in eine digitale Bitfolge umzuwandeln. Der ADC kann prinzipiell auch für die Auslesung von Sensoren mit analogem Signalausgang verwendet werden, um dabei physikalische Größen wie

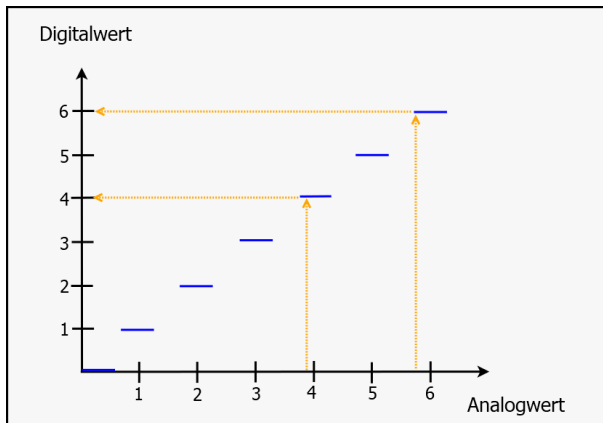


Abbildung 31: ADC-Stufenfunktion

z.B. Temperatur oder Druck aufnehmen zu können. Die Ausgangskennlinie des ADCs ist eine Art Stufenfunktion als Funktion des analogen Eingangssignal. Dabei wird jedem Amplitudenwert des analogen Eingangssignals ein genau definierter diskreter Wert zugeordnet. In Abbildung 31 ist die Stufenfunktion bzw. Treppenfunktion eines ADCs dargestellt.

Die Anzahl der Schritte in der Treppenfunktion über dem gesamten Eingangsbereich hängt von der Auflösung des ADCs ab. Der ESP32-Prozessor besitzt eine 12-Bit-Auflösung mit 18 verschiedenen ADC-Eingangskanälen. Die 12-Bit-Auflösung hat  $2^{12}$  bzw. 4096 Stufen [26]. Die Messungen können als ganzzahlige Werte zwischen 0 und 4095 aufgenommen werden. Dabei darf der Spannungspegel des Eingangssignals zwischen 0V und 3,3V liegen. Das bedeutet, dass ein Wert von 0 gleich 0V und ein Wert von 4096 gleich 3,3V ist. Der Spannungsschritt pro Teilung beträgt  $3,3V/4096=0,806mV$ . Die Berechnung des Ausgabewertes des ADCs für einen gegebenen Spannungswert wird mit Hilfe Formel 8.1 durchgeführt.

$$\frac{\text{Auflösung des ADC}}{\text{Systemspannung}} = \frac{\text{Ausgabewert des ADC}}{\text{Angelegte Spannung}} \quad (8.1)$$

Mit Hilfe der Gleichung wird ein Beispiel für einen Ausgabewert des ADCs von 2500 berechnet.

$$\frac{4096}{3,3V} = \frac{2500}{\text{Angelegte Spannung}}$$

$$\text{Angelegte Spannung} \approx 2,014 V$$

### 8.3 ADC Messung

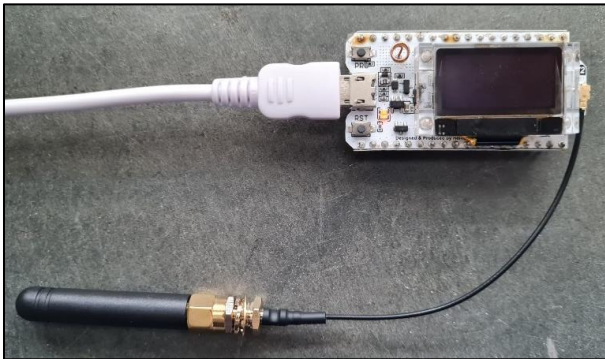


Abbildung 32: Wifi LoRa 32 V2 (angeschlossen)

Zunächst wurden Tests mit dem ADC durchgeführt und Werte ausgegeben. Das Ziel dieses Tests war es, die gelieferten Werte des ADCs zu beobachten und auf Plausibilität zu prüfen. Die Ergebnisse werden über den seriellen Monitor der Arduino IDE angezeigt. Das Board wird über einen

USB-A-zu-Micro-B-Anschluss mit dem Computer verbunden. Nach Anschluss an den Computer leuchtet auf dem Board eine orangefarbene LED (siehe Abbildung 32).

Das Board wurde auf einem Steckbrett platziert. Dabei wurde das Modul über die USB-Verbindung mit 5V versorgt. Ein Labornetzgerät wird auf eine definierte Spannung zwischen 0V und 3,3V eingestellt und mit dem ADC2\_CH3-Pin (GPIO15) des Boards verbunden. Es werden ADC-Werte zwischen 0 und 4095 erwartet.

```
const int Messpin = 15;
int Lesewert;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  Lesewert = analogRead(Messpin);
  Serial.println(Lesewert);
  delay(1000);
}
```

Zur Erstellung eines neuen Projektes wird zunächst die Arduino-IDE gestartet. Mit dem in der linken Abbildung angezeigten Code werden Messungen durchgeführt. Zuerst werden die notwendigen Variablen definiert. Außerhalb einer Funktion definierte Variablen sind für alle Funktionen sichtbar. Ansonsten

sind die Variablen nur innerhalb der entsprechenden Funktion sichtbar. Die **definierten Variablen** sind:

- const int Messpin
- int Lesewert

Für die Analog-Digital-Wandlung wird ein Datentyp Integer “**const int Messpin**“ deklariert. Const kann mit jedem Datentyp verwendet werden. Wenn eine Deklaration mit const erfolgt, kann der einmal festgelegte Wert nicht geändert werden ("read-only"). Die Zuweisung eines neuen Wertes an eine Konstante führt zu einem Compiler-Fehler.



Die Variable **“int Lesewert“** enthält das Ergebnis des vom ADC gelieferten Wertes.

Um die Konsole zu starten, muss in die Funktion `setup()` die Baudrate implementiert werden. In diesem Projekt wird die Baudrate mit der Funktion `Serial.begin(speed)` festgelegt.

- `Serial.begin(115200)`

In der Funktion `loop()` werden Werte am Pin GPIO15 gemessen und am seriellen Monitor ausgegeben. Zur Ermittlung der Werte am angegebenen ADC-Pin wurde die folgende Funktion verwendet:

- `analogRead(Messpin)`

Diese Funktion liest den ADC-Kanal aus, der am Pin ADC2\_CH3 angeschlossen ist. Der Wert wird der definierten Variablen **“int Lesewert“** zugewiesen. Danach wird der Messwert mit der Funktion `Serial.println(Lesewert)` an die Konsole übermittelt.

Mit der Funktion `delay(ms)` wird das Programm pausiert. Die Variabel in der Funktion definiert die zu pausierende Zeit in Millisekunden.

Nachdem der Programmcode debuggt wurde, konnten die Werte auf dem seriellen Monitor beobachtet werden. Durch die Einstellung des Labornetzgerätes von 0V bis 3,3V stieg der ausgelesene Wert von 0 auf 4095. Je nach Schalterstellung des Labornetzgerätes war der entsprechende Wert auf dem seriellen Monitor abgebildet.

## 8.4 Umwandlung des ADC-Werts in einen Spannungswert

In diesem Teil wird der verwendete Programmcode erweitert. Hier werden die vom ADC gelieferten Werte in Spannungswerte umgerechnet.

Der Hardwarekonfiguration bleibt hierbei unverändert.

```
const int Messpin = 15;
const int Wert = 4096;
const float Vref = 3.3;
float Spannung;

int Lesewert;

void setup() {
  Serial.begin(115200);
}

void loop()
{
  Lesewert = analogRead(Messpin);
  Spannung = (Lesewert*Vref/Wert);
  Serial.print(Spannung);
  delay(1000);
}
```

In der Software werden drei weitere Variablen deklariert. Diese sind:

- const int Wert = 4096
- const float Vref = 3.3
- float Spannung

Der “**const int Wert**“ steht für die Auflösung des ADCs im ESP32 und hat einen Wert von 4096.

Der “**const float Vref**“ ist die Systemspannung des ESP32, welche 3,3V beträgt.

In der Variable “**float Spannung**“ wird das Rechenergebnis gespeichert, welches der angelegten Spannung entsprechen sollte.

Die Funktion setup() bleibt unverändert.

Die Formel zur Umwandlung von ADC-Werten in Spannungswerte wurde in der Funktion loop() umgesetzt. In der Funktion loop() werden die vom ADC gelieferten Werte eingelesen und der Variable “int Lesewert“ zugewiesen.

Um ADC-Werte in Spannungswerte umzuwandeln, wurde die Formel 8.1 verwendet. Für die Umsetzung der Formel wurden die in der Software deklarierten Variablen

eingesetzt. Nach der Berechnung wird das Ergebnis der Variablen "Spannung" zugewiesen. Anschließend werden die Ergebnisse auf dem seriellen Monitor in 1-Sekunden-Schritten angezeigt (siehe untere Abbildung).

Wert 2683	Spannung: 2.16 V
Wert 2544	Spannung: 2.05 V
Wert 2543	Spannung: 2.05 V
Wert 2579	Spannung: 2.08 V
Wert 2181	Spannung: 1.76 V
Wert 1939	Spannung: 1.56 V

Das linke Bild zeigt einen Ausschnitt der Ausgabe auf dem seriellen Monitor. Gemessen wurde eine Abweichungen von 0,1V bis 0,7V. Beispielsweise konnte der ADC die Spannung 3,2V und 3,3V

nicht unterscheiden. In beiden Fällen wurde der Ausgangswert 4095 generiert. Außerdem konnte die Spannung 0,5V nicht von der Spannung 0V unterschieden werden. Dabei wurde in diesem Spannungsbereich der Wert 0 ausgegeben. Die ADC ESP32-Messwerte sind nicht genau (siehe ESP32-Datenblatt) [26]. Die Gründe für diese Abweichungen sollen in einer anderen Arbeit untersucht werden.

## 8.5 GPIO-Ansteuerung

Mit der Funktion `digitalWrite(pin,value)` können GPIOs gesetzt werden. Beispielsweise kann die auf dem Board integrierte LED mit dieser Funktion ein oder ausgeschaltet werden.

Das ESP32-Board wird auf einer Leiterplatte platziert. Über den USB-Anschluss wird das Modul mit 5V versorgt. Der GPIO 34 wird mit einem Oszilloskop überwacht.

```
#define GPIO34 34

void setup() {
  Serial.begin(9600);
  pinMode(GPIO34, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(GPIO34, HIGH);
  delay(5000);
  digitalWrite(GPIO34, LOW);
  delay(5000);
}
```

Die GPIO-Steuerung ist im linken Programmcode dargestellt. Der GPIO34 wird am Anfang definiert. Die Startfunktion `setup()` konfiguriert den GPIO34 als Ausgang. Die Konfiguration erfolgt über die Funktion `pinMode()`. In der Funktion `loop()` wird der Pin eingeschaltet und nach fünf Sekunden wieder ausgeschaltet. Der

Programmcode wird so oft durchlaufen, bis das Programm gestoppt oder beendet wird. Das Oszilloskop zeigt an, dass der GPIO34 alle 5 Sekunden einen Pegelwechsel ausführt. Der Pegelwechsel erfolgt zwischen 0V und 3,3V (siehe Abbildung 33).

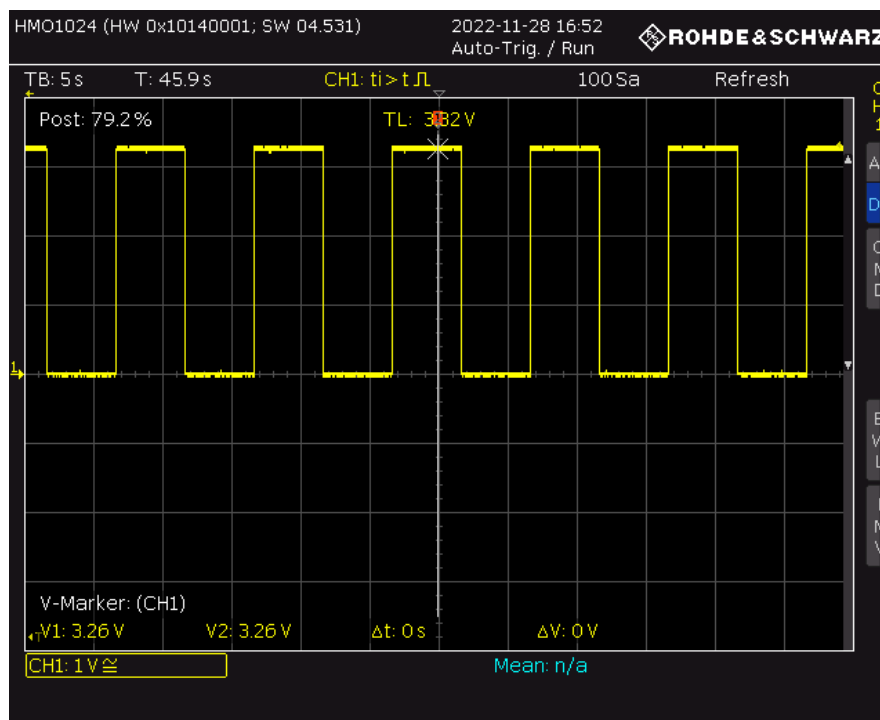


Abbildung 33: GPIO-Ansteuerung

## 8.6 Spannungsteiler

In diesem Projekt wird ein Spannungsteiler verwendet. Mit einem Spannungsteiler können Spannungen auf dem Board gemessen werden. Eine Spannungsreglung kann mittels eines Spannungsteilers erfolgen. Er besteht aus zwei in Reihe geschalteten

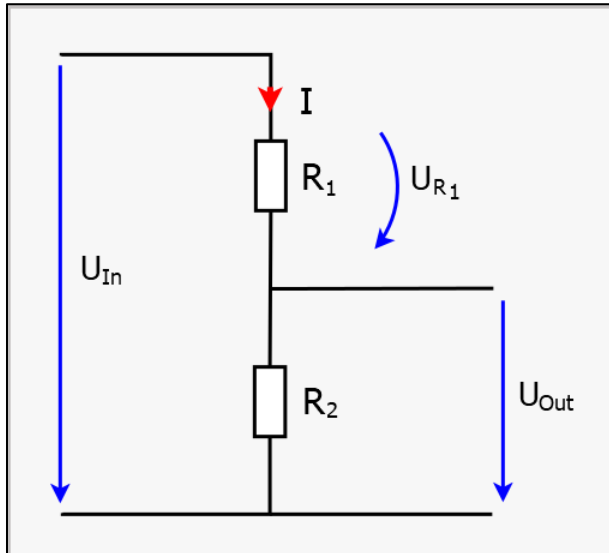


Abbildung 34: Spannungsteiler Schaltung

Widerständen (siehe Abbildung 34). Ein Spannungsteiler hat eine Eingangsspannung  $U_{in}$  und eine Ausgangsspannung  $U_{out}$ . Da die Widerstände  $R_1$  und  $R_2$  in Reihe geschaltet sind, fließt durch beide Widerstände der gleiche Strom. Allerdings wird die Eingangsspannung  $U_{in}$  durch die Widerstände  $R_1$  und  $R_2$  in die Teilspannungen  $U_{R1}$  und  $U_{out}$  aufgeteilt. Also entspricht die Summe der Teilspannung  $U_{R1}$  und der Teilspannung  $U_{out}$  die Eingangsspannung.

Es ist hier wichtig, einen hochohmigen Wert für den Gesamtwiderstand  $R_1$  plus  $R_2$  zu wählen, damit wenig Strom durch den Spannungsteiler fließt und somit die überwachte Batterie nur geringfügig belastet wird. Der Gesamtwiderstand sollte ungefähr zwischen  $10k\Omega$  und  $100k\Omega$  liegen. Um den Widerstand auszuwählen, wird eine Berechnung durchgeführt. Bei einer Halbierung der Eingangsspannung werden zwei Widerstände mit gleichem Ohm-Wert verwendet.

Die Spannungsteileregeln lautet:

$$\frac{\text{Teilspannung}}{\text{Gesamtspannung}} = \frac{\text{Widerstand unter Teilspannung}}{\text{Gesamtwiderstand}} \quad (8.2)$$

Daraus wird folgende Formel für die Spannungsteiler abgeleitet.

$$U_{out} = \frac{R_2}{R_1 + R_2} * U_{in} \quad (8.3)$$

Für die Berechnung müssen zunächst drei Werte vorliegen. Der Wert der Ausgangsspannung  $U_{out}$  ist bekannt, da der Mikrocontroller nur maximal 3,3V akzeptiert. Die maximale Eingangsspannung  $U_{in}$  beträgt 14,3V. Denn die durchschnittliche Maximalspannung bei laufendem Motor einer 12-V-Starterbatterie beträgt 14,3V. Der Wert des Widerstands  $R_2$  wurde mit  $15k\Omega$  gewählt. Der gesuchte Widerstand ist also  $R_1$ . Der Widerstand wird nach der Spannungsteiler Regel berechnet. Gegeben sind:

$$\text{Eingangsspannung } U_{in} \text{ (Gesamtspannung)} = 14,3V$$

$$\text{Ausgangsspannung } U_{out} \text{ (Teilspannung)} = 3,3V$$

$$R_2 \text{ (Widerstand unter Teilspannung)} = 15k\Omega$$

Die vorgegebenen Werte werden in die Gleichung 8.3 eingesetzt:

$$\frac{3,3V}{14,3V} = \frac{15K\Omega}{R_1 + 15K\Omega}$$

Daraus ergibt sich:

$$R_1 = 49900\Omega$$

Alle erforderlichen Werte des Spannungsteilers sind ermittelt. Nun kann der Spannungsteiler eingesetzt werden.

## 8.7 12V Messung

Nach Dimensionierung des Spannungsteilers kann eine Spannung von 12V gemessen werden.

Im praktischen Teil werden das ESP32-Modul und der Spannungsteiler auf einer Leiterplatte platziert. Das Labornetzteil wird an den positiven (12V) und negativen (GND) Pol des Spannungsteilers angeschlossen. Die Ausgangsspannung wird mit dem ADC-Eingang des Boards verbunden. Der ADC-Eingang befindet sich auf dem Modul GPIO15 (ADC2\_CH3).

```
const int Messpin = 15;
const int Wert = 4096;
const float Vref = 3.3;
float Spannung;

int Lesewert;

void setup() {
  Serial.begin(115200);
}

void loop()
{
  Lesewert = analogRead(Messpin);
  Spannung = (Lesewert*Vref/Wert);
  Serial.print(Spannung);
  delay(1000);
}
```

Im Programmcode wird der gelieferte Wert ausgewertet. Anschließend wird die Eingangsspannung umgerechnet und ausgegeben. Zunächst wird die serielle Kommunikation in der Funktion setup() gestartet. Die Eingangsspannung wird in der Funktion loop() berechnet und ausgegeben. Zuerst wird der Wert des ADCs ermittelt. Daraufhin wird mit der entsprechenden Formel 8.4 die Eingangsspannung ermittelt. Die Formel für die zu berechnende Eingangsspannung lautet:

$$\frac{\text{Ausgabe des ADC} * \text{Systemspannung}}{\text{Auflösung des ADC}} * \left(\frac{R1 + R2}{R2}\right) = \text{Angelegte Spannung} \quad (8.4)$$

## 9 Programmierung der LoRa Schnittstelle

In diesem Kapitel wird die vollständige Implementierung des Projekts vorgestellt. Das Endgerät muss zunächst beim TTN registriert werden. Der benötigte Schlüssel wird bei TTN generiert. Diese Schlüssel werden für den Programmcode in der Arduino-IDE benötigt. Sobald die Registrierung bei TTN abgeschlossen ist, beginnt die Programmierung mit der Entwicklungsumgebung Arduino IDE. Daraufhin wird das fertige Programm auf das Board hochgeladen. Anschließend wird das ESP32-Modul auf die fertige Leiterplatte platziert. Die Platine wird mit einer Spannung von 12V versorgt, und dabei wird die Versorgungsspannung gemessen. Die gemessenen Messdaten werden an den Netzwerkserversender gesendet. Zum Schluss sendet der Netzwerkserversender die Nutzdaten an die erstellte Webseite.

### 9.1 Einrichtung The Things Network

The Things Network ist ein Netzwerk aus Servern, welche die Kommunikation mit LoRa ermöglicht. Ausführlich wurde das The Things Network in Kapitel 5 behandelt. Als erstes wird ein persönliches Konto auf der Website "www.thethingsnetwork.org" registriert. Nun kann das Endgerät registriert werden.

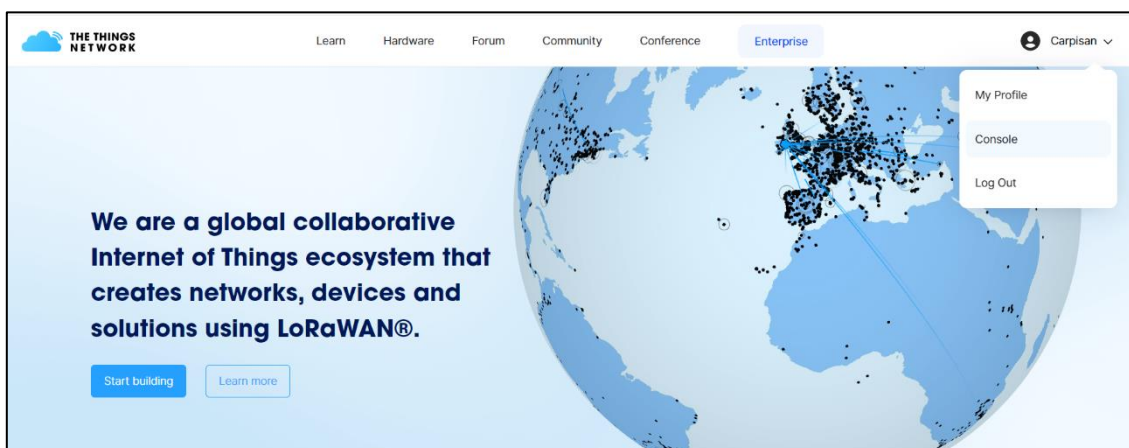


Abbildung 35: TTN Registrierung 1

Zunächst wird oben rechts das Profilsymbol betätigt (siehe Abbildung 35). Anschließend werden Optionen angezeigt. Die Option "Console" wird ausgewählt. Danach erfolgt eine Weiterleitung auf eine neue Seite. Auf dieser Seite befindet sich eine Weltkarte (siehe Abbildung 36).





Abbildung 36: TTN Registrierung 2

Auf der Karte befinden sich drei blaue Punkte, die ausgewählt werden können. Diese Punkte befinden sich auf den Kontinenten Europa, Nordamerika und Australien. Für dieses Projekt wurde Europa ausgewählt, da der Test in Deutschland durchgeführt wird und der Datentransfer entsprechend schnell ist.

Als nächstes wurde eine Abfrage mit Anmeldedaten angezeigt. Anschließend wurden die Login-Daten eingegeben und bestätigt. Weiterhin erscheint ein neues Fenster mit zwei Konfigurationsoptionen (siehe Abbildung 37). Die linke Seite dient der Endgerätekonfiguration und die rechte Seite der Gatewaykonfiguration. Hier wurde die Schaltfläche Endgerätkonfiguration ausgewählt. Für die Benutzer mit eigenem Gateway ist es besser, zuerst das Gateway zu konfigurieren und dann das Endgerät.

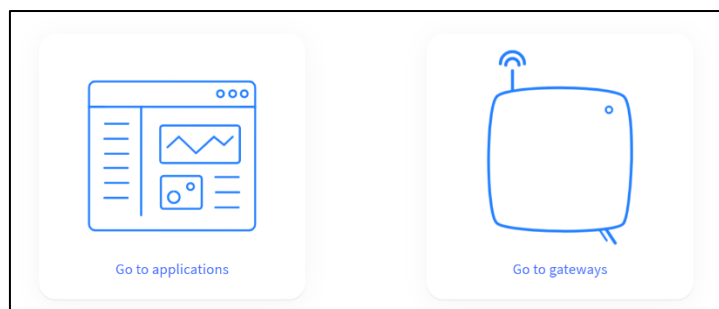


Abbildung 37: TTN Endgerät Registrierung 1

Nun erscheint eine neue Seite. Dort wird die Schaltfläche “+Create application“ angeklickt. Anschließend wird eine neue Seite zum Erstellen einer Applikation geöffnet (siehe Abbildung 38).

The image shows a web form for creating a new application. It contains three input fields: 'Application ID' with the value 'wifilora32', 'Application name' with the value 'Application 1', and 'Description' with the value 'Wifi LoRa 32 V2, Batterieüberwachung'. Below the description field is a small text note: 'Optional application description; can also be used to save notes about the application'. At the bottom of the form is a blue button labeled 'Create application'.

Abbildung 38: TTN Endgerät Registrierung 2

Auf dieser Seite werden Application ID, Application Name und Description ausgefüllt. Abschließend wurde der Button "Create application" betätigt.

Als nächstes taucht eine neue Seite auf. Hier wird unten rechts der Button mit der Bezeichnung "+Add End Device" angeklickt.

Es werden drei Parameter abgefragt. Das erste ist ein Dropdown-Listefeld mit dem Label "Frequency plan". Hier wird die Auswahloption für Deutschland "Europa 869-870 MHz (SF9 für RX2 - recommended)" ausgewählt. Als nächstes wird die LoRaWAN-Version abgefragt. Dabei stehen viele Auswahlmöglichkeiten als Dropdown-Listfelder zur Verfügung. Die LoRaWAN-Version "LoRaWAN Specification1.0.3" wurde ausgewählt, da das Gerät bei dieser Version die Kommunikationsprotokolle der Klasse A unterstützt. Das dritte Dropdown-Listefeld "Regional Parameters Version" wird automatisch vom Server ausgefüllt, nachdem die LoRaWAN-Version ausgewählt wurde. Anschließend wird die Aktivierungsart als Radiobutton abgefragt. Für das Projekt wurde der Aktivierungstyp ABP-Mode ausgewählt.

DevEUI ⓘ

70 B3 D5 7E D0 05 43 78  1/50 used

Device address ⓘ \*

26 0B 2D A8

AppSKey ⓘ \*

36 5D A3 0D B3 75 BA FE 45 C9 90 18 E4 07 4D 0A

NwkSKey ⓘ \*

F5 AF BA E8 6B 56 84 3A CE 6D A1 7E 2F 6D 29 CD

End device ID ⓘ \*

eui-70b3d57ed0054378

This value is automatically prefilled using the DevEUI

Abbildung 39: TTN Endgerät Registrierung 3

In Abbildung 39 werden die benötigten Schlüssel dargestellt. Der Server benötigt diese Schlüssel, um das ABP-Endgerät zu konfigurieren. Abschließend wird die Schaltfläche "Register end Device" betätigt. Nun ist die Endgerätekonfiguration erfolgreich abgeschlossen.

## 9.2 Softwareimplementierung

Dieser Abschnitt gibt einen Überblick auf den Programmcode für die LoRa Kommunikation. Bei Inaktivität befindet sich der Mikrocontroller im Sleep-Modus. Der Sleep-Modus wird beim Ablauf eines internen Timers deaktiviert. Nachdem der Mikrocontroller aus dem Sleep-Modus aufgewacht ist, beginnt die Kommunikation zwischen dem ESP32-Chip und dem LoRa-Chip. Für diese Kommunikation sind SPI- und DIO-Pins vorgesehen. Der GPIO12 ist am Mikrocontroller als Eingang für die Spannungsmessung konfiguriert. Weiterhin wird der GPIO2 auf High gesetzt, um den Spannungsteiler zu aktivieren. Danach erfolgt die Spannungsmessung über GPIO12. Anschließend werden GPIO12 und GPIO2 auf Low gesetzt. Nun werden die Messdaten vom Endgerät an den TTN-Server gesendet. Bei erfolgreicher Übertragung geht das Board in den Tiefschlaf und der Timer wird zurückgesetzt. Bei einer guten Verbindung zwischen Endgerät und Gateway dauert der Sendevorgang zwei Sekunden.

Dieser Abschnitt beinhaltet den Programmcode und den Ablauf des Projekts. Zunächst wird die LMIC-Bibliothek in der Entwicklungsumgebung Arduino eingebunden. Anschließend wird die Frequenzanpassung für die Konfiguration der Bibliothek eingestellt. Für die Frequenzanpassung wird zuerst der Ordner "libraries" im Hauptordner von Arduino geöffnet. Weiterhin wird der Ordner "LMIC\_library" und anschließend der Ordner "projekt\_config" ausgewählt. Eine Header-Datei mit gleichem Ordernamen wird im Ordner "projekt\_config" geöffnet (siehe Abbildung unten). In dieser Datei sind die Frequenzbänder aufgelistet. In dieser Datei ist die dritte Zeile "#define CFG\_us915 1" auskommentiert. Das bedeutet, dass die Bibliothek auf das US-Frequenzband eingestellt ist. Da dieses Projekt nicht im US-Frequenzband arbeitet, wurde die dritte Zeile "#define CFG\_us915 1" kommentiert. Anschließend wird die zweite Zeile "#define CFG\_eu686 1" auskommentiert, um die europäischen Frequenzbänder verwenden zu können. Abschließend wird die Header-Datei gespeichert.

```
// project-specific definitions
#define CFG_eu868 1
// #define CFG_us915 1
// #define CFG_au915 1
// #define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP /* for as923-JP */
// #define CFG_kr920 1
// #define CFG_in866 1
#define CFG_sx1276_radio 1
// #define LMIC_USE_INTERRUPTS
```

Nach der Frequenzeinstellung beginnt die Programmierung. Zunächst wurden die benötigten Variablen definiert.

```
#define MessPin      13
#define SchalterPin  2
#define LED          25

#define Vref         3.3
#define R1           49900
#define R2           15000

#define INTERVALL    60

#define SET(x, y)    digitalWrite(x, y)
#define GET(x)       digitalRead(x)
#define AN           HIGH
#define AUS          LOW

#define MODUS(x, y)  pinMode(x, y)
#define AUSG         OUTPUT
#define EING         INPUT
#define TIME(z)      delay(z)

#define Schl         5
```

## Aktivierungsschlüssel ABP-Type

Als nächstes wurden die Schlüssel eingefügt, die für die Kommunikation zwischen dem Endgerät und dem Server benötigt werden. Die Schlüssel NWKSKY, APPSKY und DEVADDR wurden vom Server selbst generiert und als Programmcode eingefügt.

```
static const PROGMEM ul_t NWKSKY[16] = {0x16, 0x35, 0xE2, 0xE6, 0x82, 0x93, 0x14, 0x1D, 0xF5, 0x12, 0x54, 0x7C, 0xED, 0x24, 0x3F, 0xAB};
static const ul_t PROGMEM APPSKY[16] = {0xC3, 0x9B, 0xC7, 0xC9, 0x8C, 0x6D, 0x15, 0xC7, 0x23, 0x39, 0x45, 0xA2, 0x7E, 0xA6, 0xA7, 0x81};
static const u4_t DEVADDR = 0x260BF103;
```

## SPI-Schnittstelle

Der folgende Codeabschnitt zeigt, wie die serielle Kommunikation und die SPI-Schnittstelle in der setup() Funktion gestartet werden. Durch die Funktion SPI.begin()

```
void setup()
{
    Serial.begin(115200);
    SPI.begin(5, 19, 27, 18);
}
```

werden SPI-Pins eingetragen. Im Funktion (siehe Programmcode) sind vier Zahlen, die zu jeweiligen Pins gehören (siehe Abbildung 23). Durch diese Pins wird die Verbindung zwischen dem ESP32-Mikrocontroller

und dem LoRa-Chip hergestellt. Die Pins: MOSI, MISO, SCK und NSS sorgen für die Verbindung. Laut Schematic Diagramm [28] sind die Pins MOSI, MISO und SCK (ESP32-Mikrocontroller und LoRa-Chip) direkt miteinander verbunden. Der NSS-Pin des LoRa-Chips wird mit dem CS-Pin des ESP32 gekoppelt. Dieser NSS-Pin befindet sich beim ESP32 auf GPIO18 (siehe Tabelle 14). Die Tabelle 13 zeigt die ESP32-SPI-Pins.

Beschreibung	GPIO
<b>SCK</b>	5
<b>MISO</b>	19
<b>MOSI</b>	27

Tabelle 13: SPI ESP32 - LoRa-Chip

## DIO-Pins

```
const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 14,
    .dio = {26, 34, 35},
};
```

Die Bibliothek benötigt Zugriff auf bestimmte DIO-Pins (Digital I/O). Diese DIO-Pins werden beispielsweise benötigt, um die Übertragung von LoRa-Datenpaketen zu initiieren und zu

beenden. Die Bezeichnung der lmic\_pins-Struktur darf nicht geändert werden, damit die Bibliothek auf das Pin-Mapping zugreifen kann. Alle nicht belegten bzw. nicht

benötigten Pins werden mit der Definition "LMIC\_UNUSED\_PIN" versehen. In der Tabelle 14 ist die Pinbelegungen des LoRa-Chips dargestellt.

Beschreibung	GPIO
<b>NSS</b>	18
<b>DIO 0</b>	26
<b>DIO 1</b>	35
<b>DIO 2</b>	34
<b>RST</b>	14

Tabelle 14: DIO ESP32 - LoRa-Chip

Der Mikrocontroller wurde auf eine für das Projekt angefertigte Leiterplatte platziert. Die Platine enthält alle notwendigen Komponenten eines Spannungsteilers zur Reduzierung der Eingangsspannung, einen DC-DC-Wandler zur Stromversorgung des Moduls und einige Schutzkomponenten. Diese Platine wurde durch einen anderen Studierenden angefertigt und für Testzwecke diesem Projekt zur Verfügung gestellt.

```

MODUS(SchalterPin, AUSG);
SET(SchalterPin, AN);
Serial.println("SchalterPin AN ");

Serial.println("Wartezeit für SchalterPin AN");
TIME(50); //Wartezeit

//Wifi LoRa 32 wacht auf, LED Pin 25 AN
MODUS(LED, AUSG);
SET(LED, AN);
Serial.println("LED AN");
TIME(2);

```

Der folgende Programmcode entspricht der setup() Funktion. In diesem Code-Abschnitt wird das Modul aufgeweckt bzw. der Schlafvorgang unterbrochen. Infolgedessen schaltet sich der Ausgangspin GPIO2 des ESP32-Moduls direkt auf High. Der GPIO2 dient zur

Aktivierung des Spannungsteilers. Für die Aktivierung wird eine kurze Wartezeit eingesetzt. Die Ergebnisse werden mit der Funktion "Serial.print()" auf dem seriellen Monitor angezeigt. Bei aktiviertem ESP32-Modul geht die weiße LED an und im Sleep-Modus wieder aus. Das Ein- und Ausschalten der LED dient zur Überwachung der aktiven Zeit des Moduls.

```

Serial.println("init..");
// LMIC init
os_init();

Serial.println("LMIC reset..");
LMIC_reset();

```

Die Funktionen os\_init() und LMIC\_reset() wurden links im Programmcode gelistet. Mit der Funktion os\_init() erfolgt die Initialisierung. Die LMIC-Bibliothek stellt auch API-Funktionen bereit, um den MAC (Media Access Control) - Zustand zu steuern

und Protokollaktionen auszulösen. Der MAC ist eine Ebene im Schichtenmodell des

LoRaWANs. Die Funktion LMIC\_reset() ist eine API-Funktion und wird für den MAC-Reset und zur Verwerfung der anstehenden Datenübertragungen verwendet.

```
#ifndef PROGMEM
uint8_t appskey[sizeof(APPSKEY)];
uint8_t nwkskey[sizeof(NWKSKEY)];
memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
#else
// If not running an AVR with PROGMEM, just use the arrays directly
LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
#endif
```

Der folgende Codeabschnitt enthält Sitzungsparameter, die in der Funktion LMIC\_setSession() gesetzt werden. Diese werden beim Start des Programms konfiguriert und sind für den

Netzwerkbeitritt erforderlich. Die Definition PROGMEM ist ein variabler Modifikator, der die Daten nicht im SRAM sondern im Flash bzw. im Programmspeicher speichert.

Der Codeabschnitt (siehe unterer Programmcode) konfiguriert insgesamt neun LoRa-Kanäle. Diese Kanäle sind identisch mit dem Frequenzplan EU868. Die Konfiguration erfolgt erst nach der Funktion LMIC\_setSession(). Die Kanäle sind von null bis acht nummeriert. Die maximale Frequenz beträgt 868,8MHz und die minimale Frequenz 867,1MHz. Die Funktion DR\_RANGE\_MAP() zeigt die aktivierte Datenrate für diesen Kanal an. Alle Kanäle außer Kanal acht unterstützen die Datenraten SF7 bis SF12. Für den neunten Kanal werden nur die Kanalnummer und die Frequenz benötigt. Die Bandnamen BAND\_CENTI und BAND\_MILLI sind für Arbeitszyklen vorgesehen. Das Tastverhältnis beträgt 0,1% bis 10%. Der Bandname BAND\_CENTI hat die Einschalt Dauergrenze von 1% und der Bandname BAND\_MILLI hat die Einschalt-dauergrenze von 0,1%. Der Bandname BAND\_DECI wird hier nicht verwendet, da der Bandname ein Arbeitszykluslimit von 10% hat. Das Tastverhältnis basiert auf der richtigen Kanalauswahl [7]. Die Sendeleistung für alle Bänder beträgt 14 dBm bzw. 25 mW. Anschließend wird die Funktion LMIC\_selectSubBand(1) für EU868 verwendet. Für Frequenzplan US915 und AU915 wird die Funktion LMIC\_selectSubBand(0) null eingesetzt.

```
#if defined(CFG_eu868)
LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI); // g-band
LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK, DR_FSK), BAND_MILLI); // g2-band
```

In der unteren Abbildung befindet sich der Programmcode des Empfangsfensters zwei. Für die RX2-Window-Downlink Pakete verwendet TTN einen Spreading Factor neun und eine Bandbreite von 125kHz. Die Datenrate und Sendeleistung werden mit der Funktion `LMIC_setDrTxpow(DR_SF7,14)` eingestellt und die Sendeleistung beträgt 14dBm.

```
// TTN benutzt SF9 für RX2 Fenster.
LMIC.dn2Dr = DR_SF9;

// Datenrate und Sendeleistung für Uplink einstellen
LMIC_setDrTxpow(DR_SF7,14);
```

Nachdem der Sleep-Modus des Moduls mit der Setup-Funktion deaktiviert wurde, werden die Variablen neu initialisiert. Das bedeutet, dass gespeicherte Variablen nicht mehr existieren. Dieser Vorgang würde dann zu hohen Sendezeiten führen.

```
RTC_DATA_ATTR lmic_t RTC_LMIC;
```

Um dieses Problem auf ESP32 zu lösen, werden für eine schnelle Sendezeit einige

Variablen in einem Zwischenspeicher abgelegt. Dafür wird der RTC-Speicher verwendet. Der RTC speichert Daten während der Sleep-Modus aktiv ist. Der RTC-Speicher wird als Variable deklariert. Danach folgt eine Funktion, welche die LMIC-Struktur speichert. Die Funktion ist als `SaveLMICToRTC()` konfiguriert (Codeabschnitt unten). Wie der Funktionsname andeutet, werden hier Daten aus der LMIC-Struktur im RTC-Speicher gespeichert [34].

```
void SaveLMICToRTC(int deepsleep_sec)
{
  Serial.println(F("Save LMIC to RTC"));
  RTC_LMIC = LMIC;
  // ESP32 kann die millis während des DeepSleep nicht verfolgen und es gibt keine Möglichkeit, die millis nach dem DeepSleep zu erweitern.
  // Dafür DutyCycles zurücksetzen // DutyClye = Einschaltdauer
  unsigned long now = millis();
  // EU Like Bands
  #if defined(CFG_LMIC_EU_like) // CFG Region Wert eu868 = 1
    Serial.println(F("Reset CFG_LMIC_EU_like band avail"));
    for (int i = 0; i < MAX_BANDS; i++) // MAX_BANDS=4
    {
      ostime_ correctedAvail = RTC_LMIC.bands[i].avail - ((now / 1000.0 + deepsleep_sec) * OSTICKS_PER_SEC);
      if (correctedAvail < 0)
      {
        correctedAvail = 0;
      }
      RTC_LMIC.bands[i].avail = correctedAvail;
    }
    RTC_LMIC.globalDutyAvail = RTC_LMIC.globalDutyAvail - ((now / 1000.0 + deepsleep_sec) * OSTICKS_PER_SEC);
    if (RTC_LMIC.globalDutyAvail < 0)
    {
      RTC_LMIC.globalDutyAvail = 0;
    }
  #else
    Serial.println(F("Keine DutyCycle Neuberechnungsfunktion!"));
  #endif
}
```



```

void LoadLMICFromRTC()
{
    Serial.println(F("Load LMIC from RTC"));
    LMIC = RTC_LMIC;
}

```

Die abgebildete Funktion wird verwendet, um die LMIC aus dem RTC zu laden.

```

if (RTC_LMIC.seqnoUp != 0)
{
    LoadLMICFromRTC();
}

```

Des Weiteren befindet sich in der Setup-Funktion eine if-Anweisung. In dieser Anweisung wird überprüft, ob die LoRa Daten vom RTC geladen werden sollen.

Die Messwerte werden vom ADC eingelesen und über das LoRaWAN-Netzwerk gesendet. Dazu wird ein Payload verwendet, um Messdaten zu übertragen. Das Payload enthält Nutzdaten im 16-Bit-Array-Datenformat. Die Nutzdaten werden auf dem Endgerät verschlüsselt und an den entsprechenden Server gesendet. Sobald die Messdaten beim TTN-Server ankommen, werden die Nutzdaten entschlüsselt. Dieser Prozess entspricht dem Payload-Encoding und -Decoding. Im Programmcode wird die Batteriespannung in der Funktion `do_send()` gemessen und an den TTN-Server bei Sendestart gesendet. Die Funktion `do_send()` wird in der Funktion `setup()` eingesetzt.

```

float readVoltage(){
    #define constant 14.27; // Maximale Spannung
    float ADCvoltage =(float)analogRead(MessPin)/4095*constant;
    return ADCvoltage;
}

```

Der folgende Codeabschnitt dient zum Einlesen der angelegten Spannung. Diese Funktion überwacht die Spannung am ADC-Eingang GPIO15 in fest-

gelegten Intervallen. Das Intervall wird mit Hilfe eines ganzzahligen (integer) Datentyps auf eine Stunde festgelegt. Die Funktion `readVoltage()` wird in der Funktion `do_send()` aufgerufen und schließlich in der Funktion `setup()` ausgeführt. Die Nutzdaten werden in der Funktion `do_send()` enkodiert und nach dem Empfang über TTN wieder dekodiert.

Der Payload Encoder ist in der Abbildung unten dargestellt. Die gemessene Spannung wird zunächst dem float-Datentyp "LeseWert\_" zugewiesen. Die erste Codezeile konvertiert die Variable "LeseWert\_" in einen Integer-Datentyp um. Der Wert der Batteriespannung muss als Ausgabewert mit Komma ausgedrückt werden (z.B. 12,42V). Deshalb muss der Messwert für einen integer-Wert mit 100 multipliziert werden. Dabei ergibt sich beispielsweise bei einem Messwert von 12,42 1242. Dieser Wert entspricht zwei Bytes. Das Payload wird in der zweiten Zeile definiert: "uint8\_t payload [2] ". Da die Messwerte 16 Bit enthalten, werden sie über zwei Einträge des Arrays geteilt. Das Teilen erfolgt über Bit-Shifting. Im ersten Array-Eintrag befinden sich die ersten 8-Bits, und die restlichen 8-Bits befinden sich im zweiten Array-Eintrag. Die Dekodierung erfolgt im TTN. Wenn kein Decoder im TTN-Server vorhanden ist, werden die Messdaten in hexadezimaler Darstellung angezeigt. Um den Dezimalwert

```
int adclesen = ((int)(LeseWert_ * 100))
uint8_t payload[2];

payload[0] = adclesen >> 8;
payload[1] = adclesen;

MODUS(SchalterPin, AUSG);
SET(SchalterPin, AUS);
```

zu erhalten, wird das Ergebnis mit einem Decoder vom Formattyp "Custom Javascript Formatter" dekodiert. Der Decoder konvertiert also das hexadezimale Payload in eine Dezimalzahl.

```
LMIC_setTxData2(1, payload, sizeof(payload), 0);
```

Anschließend wird die Funktion LMIC\_setTxData2 aufgerufen. Die Funktion LMIC\_setTxData2 bereitet die Nutzdaten für die Übertragung vor. Der erste Parameter 1 steht für den Port, über den die Übermittlung abgewickelt wird. Der Port kann auch geändert werden. Dadurch kann der Payload-Decoder die Daten entsprechend des Ports verarbeiten. Die zweite Variable ist die Nutzlast, die in den Array-Bytes enthalten ist. Als nächstes kommt die Nutzlastgröße. Der letzte Parameter ist für die Bestätigung (ACK) zuständig. Der Wert null bedeutet, dass kein ACK erforderlich ist. Ansonst ist der Wert immer eins.

```

//TiefSchlaf Funktion
void DeepSleepESP()
{
    // MessPin 2 auf 0
    MODUS(MessPin, AUSG);
    SET(MessPin, AUS);
    Serial.println("MessPin AUS");
    TIME(2);

    // LED Pin 25 auf 0
    MODUS(LED, AUSG);
    SET(LED, AUS);
    Serial.println("LED AUS");
    TIME(2);

    Serial.println(F("Gehe JETZT Schlafen"));
    PrintRuntime();
    Serial.flush();

    //Sleep-Modus
    esp_sleep_enable_timer_wakeup(INTERVALL * 1000000);
    esp_deep_sleep_start();
}

```

Der links dargestellte Programmabschnitt entspricht der Funktion für die Aktivierung des Sleep-Modus. Vor dem Sleep-Modus sind GPIO2 und GPIO25 auf Low gesetzt. Die Verwendung der Funktion PrintRuntime() gibt die gesamte Laufzeit des aktiven Zustand des Mikroprozessors an. Das heißt, dass diese Zeit der benötigten Ausführungszeit des Mikrocontrollers entspricht. Die Funktion Serial.flush() wartet so lange

bis der Transfer der seriellen Daten abgeschlossen ist. Anschließend wird ein Interrupt-Timer aktiviert, der auf eine Stunde festgelegt ist. Der Interrupt-Timer wird durch einen RTC-Controller realisiert. Nachdem die zeitgesteuerte Weckfunktion konfiguriert wurde, wird die Sleep-Modus-Funktion aktiviert. Die Funktion void DeepSleepESP() wird in der loop-Funktion ausgeführt.

```

void loop()
{
    static unsigned long lastPrintTime = 0;

    const bool timeCriticalJobs = os_queryTimeCriticalJobs(ma2osticksRound((INTERVALL * 1000)));
    TIME(5);
    if (!(timeCriticalJobs && GoSleep == true && !(LMIC.opmode & OP_TXRXPEND))
    {
        Serial.print(F("Kann schlafen gehen \n"));
        LoraWANPrintLMICopmode(); //Gibt Betriebs-MODUS aus
        SaveLMICtoRTC(INTERVALL); // Speichert LMIC -> RTC
        DeepSleepESP(); // Funktion für Schlafmodus
    }
}

```

In der Loop-Funktion befindet sich der eigentliche Prozessablauf (Programmcode links). Dabei wird mit der if-Anweisung überprüft, ob die Funktion

DeepSleepESP() ausgeführt werden kann bzw. ob der Sleep-Modus aktiviert werden kann. Bei der if-Anweisung ist es wichtig, dass der Typ Bool GoSleep = true ist. Der Typ Bool GoSleep wird erst true, wenn in der Funktion onEvent() der Define EV\_TXCOMPLETE ausgeführt wird. Die Funktion onEvent() enthält viele Defines und prüft den Status des ausgehenden Datenpakets. Nachdem der Betriebsmodus abgearbeitet wurde, wird der Sleep-Modus mit der Funktion DeepSleepESP() aktiviert. Anschließend gibt diese Funktion den aktuellen Status auf dem seriellen Monitor aus. Wenn der Mikrocontroller jedoch die Sleep-Modus-Funktion nicht aktivieren kann, erfolgt die Ausführung der Instruktionen in der else if-Anweisung, und der serielle Monitor gibt den Status "Kann nicht schlafen" aus. In diesem else if-Zweig wird anhand der Betriebsmodus überprüft, was der Grund des inaktiven Sleep-Modus ist.

```

case EV_TXCOMPLETE: //Die über LMIC_getTxData() aufbereiteten Daten wurden gesendet
    Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
    if (LMIC.txrxFlags & TXRX_ACK)
        Serial.println(F("Received ack"));
    if (LMIC.dataLen)
    {
        Serial.print(F("Received "));
        Serial.print(LMIC.dataLen);
        Serial.println(F(" bytes of payload"));
    }
    GoSleep = true;
    break;

```

Der linke Programmcode zeigt ein Abschnitt der Funktion onEvent(). Der Define EV\_TXCOMPLETE, wird erst ausgeführt, wenn die Nutzdaten gesendet worden sind.

Anschließend wird die Bool Variable GoSleep in diesem Define auf true gesetzt. Es gibt viele Defines in der Funktion onEvent(), die in einer switch-Anweisung überprüft werden. Die switch-Anweisung ist sehr nützlich für komplexe Vorgänge. Dabei ist die Funktionsweise sehr simpel. Alle notwendigen Defines werden in Case-Anweisungen abgearbeitet, und die break-Anweisung beendet eine bestimmte Anweisung. Hier werden folgende Defines (Status) geprüft:

- EV\_JOINING: Beginn des Netzwerkaufbaus
- EV\_JOINED: Erfolgreiche Verbindung mit dem Netzwerk
- EV\_JOIN\_FAILED: Verbindung mit dem Netzwerk fehlgeschlagen
- EV\_TXCOMPLETE: Die Daten wurden erfolgreich gesendet
- EV\_RXCOMPLETE: Der Downstream Daten wurde empfangen
- EV\_LOST\_TSYNC: Die Zeitsynchronisation ist verloren gegangen. Tracking oder Ping muss neu gestartet werden.
- EV\_RESET: Die Sitzung wird zurückgesetzt und neu gestartet
- EV\_LINK\_DEAD: Verbindung mit dem Netzwerk fehlgeschlagen wegen Zeitüberschreitung

Für die serielle-Schnittstelle wird der Define "LMIC\_DEBUG\_LEVEL 2" in die folgende Zeile im Ordner "project\_config" hinzugefügt:

```

// project-specific definitions
#define CFG_eu868 1
//#define CFG_us915 1
//#define CFG_au915 1
//#define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP
//#define CFG_kr920 1
//#define CFG_in866 1
#define CFG_sx1276_radio 1
//#define LMIC_USE_INTERRUPTS

#define LMIC_DEBUG_LEVEL 2

```

```

Ich bin jetzt aufgewacht
SchalterPin AN
Wartezeit für SchalterPin AN
LED AN
init..
RXMODE_RSSI
LMIC reset..
Load LMIC from RTC
Messung startet
13.43 V
SchalterPin AUS
Wartezeit bis SchalterPin AUS
RXMODE_SINGLE, freq=869525000, SF=9, BW=125, CR=4/5, IH=0
Save LMIC to RTC
Kann schlafen gehen
LMIC.opmode: OF_RXCTX_OF_NEXTCHNL Save LMIC to RTC
Reset CFG_LMIC_EU_like band avail
MessPin AUS
LED AUS
Sehe JETZT Schlafen
Runtime: 2 seconds

```

In der linken Abbildung ist die Ausgabe des seriellen Monitors an Port COM3 dargestellt. Das Programm wird debuggt, und die Ergebnisse werden auf dem seriellen Monitor angezeigt. Am ADC-Eingang wurde eine Spannung angelegt. Der Mikrocontroller wird vom Computer über den USB-Port mit Strom versorgt.

### 9.3 TTN Decoder

```

Setup
Formatter type *
Custom Javascript formatter
Formatter code *
1 function Decoder(bytes, port) {
2   var adclesen = (bytes[0]<<8 | bytes[1]);
3
4
5   return {
6     Volt: (adclesen)/100,
7   };
8 }
9

```

Abbildung 40: TTN Decoder

Die Nutzdaten werden im ESP32 encodiert und im Netzwerkserver decodiert. Dabei werden die in mehrere Arrays-Einträge aufgeteilten Daten in der Entwicklungsumgebung mit dem TTN-Decoder zu einem einzigen Wert rekonstruiert. Hier werden die ersten 8

Bits im TTN-Decoder nach links verschoben, und die restlichen Bits werden mit einer 'AND'-Operation angehängt. Zum Beispiel entspricht ein hexadezimaler Payload von 18 6E 12,54.

Die auf dem Netzwerkserver empfangenen Messdaten werden nicht langfristig gespeichert. Stattdessen werden die Daten an den Applikationsserver übertragen. In diesem Projekt ist der Anwendungsserver die erstellte Webseite (siehe Kapitel 10 Abschnitt 2).

## 9.4 Datentransfer zum Anwendungsserver

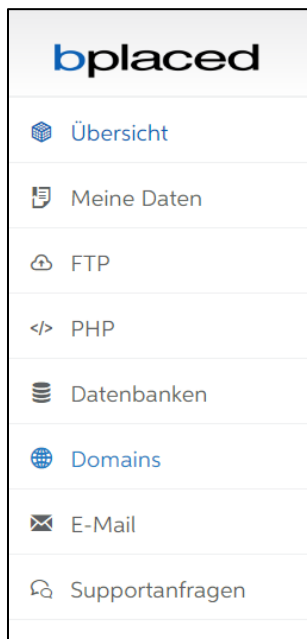


Abbildung 41: Bplaced Einstellung 1

Zur Übertragung der Benutzerdaten vom TTN an den Anwendungsserver wird ein Kopplungsmechanismus benötigt, welche als Integrationen bezeichnet werden. Beispiele für Integrationen sind MQTT, Webhooks, Azure IoT oder LoRa Cloud [33]. Für dieses Projekt wurde die Intergration Webhooks verwendet. Um die Webhook-Integrationsvorlage anzuwenden, muss zunächst ein benutzerdefinierter Webserver eingerichtet werden. In diesem Projekt wurde der Webserver Bplaced verwendet. Die Datenübertragung startet unmittelbar nach dem Empfang der Daten im http-Request auf dem TTN-Server, wobei die Daten mittels http-Post an die Webseite versendet werden. Hierbei wird ein Konto auf der Webseite "https://my.bplaced.net/" eingerichtet. Nach dem

Einloggen im Bplaced befinden sich auf der linken Seite Menüs, die durch Symbole kategorisiert sind (siehe Abbildung 41). Im ersten Schritt wird das Symbol mit der Weltkugel angeklickt.

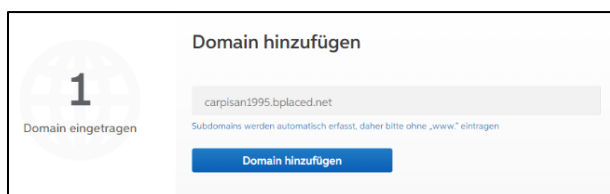


Abbildung 42: Bplaced Einstellung 2

Hier wird die folgende Domain erstellt (siehe Abbildung 42).



Abbildung 43: Bplaced Einstellung 3

Anschließend wird das FTP-Menü ausgewählt. Hier wird ein FTP-Zugang erstellt (siehe Abbildung 43). Der FTP-Zugang ist für die Datenübertragung notwendig. Anschließend wird die FTP-Client Software WinSCP gestartet. Um

Daten herunterladen zu können, muss eine Anmeldung am FTP-Server erfolgen. Für die Anmeldung wird oben links die Schaltfläche "Neue Sitzung" gewählt. Daraufhin wird eine Seite geöffnet (siehe Abbildung 44). Die erforderlichen Schlüssel werden eingegeben und schließlich wird die Schaltfläche "Anmelden" angeklickt. Danach erscheint ein neues Fenster, in dem das Passwort abgefragt wird. Nach Eingabe des

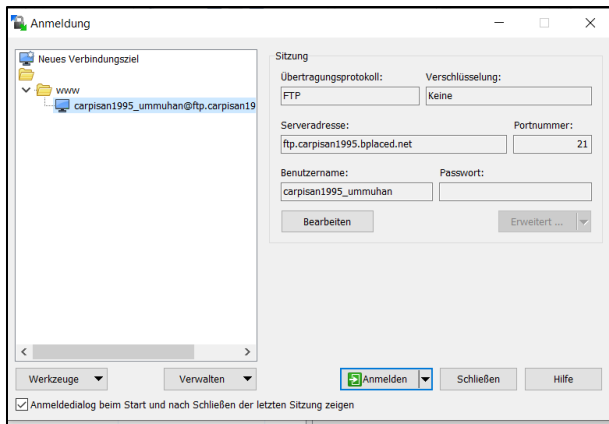


Abbildung 44: WinSCP Sitzung

jeweiligen Anbieters (Bplaced) bereitgestellt. Als nächstes wird die Webhook-Integration in TTN eingerichtet.

## 9.5 Webhook in TTN einrichten

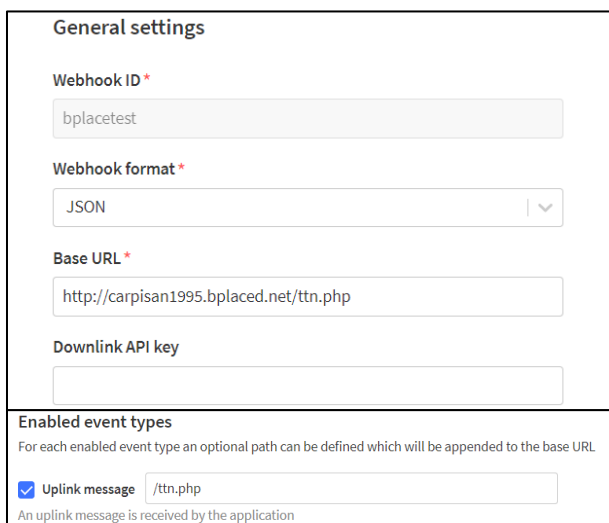


Abbildung 45: TTN-Integration (Webhook) Einstellung

JSON ausgewählt. Im Basis-URL Textfeld wird der Link der erstellten Webseite eingesetzt und am Ende "/ttn.php" hinzugefügt (siehe Abbildung 45). Danach wird weiter nach unten gescrollt und die Checkbox Uplink message angeklickt. Neben der Anknickbox befindet sich ein Textfeld, das mit "/ttn.php" ausgefüllt wird. Abschließend wird die Schaltfläche "Save changes" betätigt.

Passworts wird die Sitzung gestartet. Anschließend werden die erforderlichen PHP-Daten heruntergeladen. Zum Speichern der Daten wird eine SQL-Datei benötigt. Dazu wird die SQL-Datei in den Datenbank-Server importiert. Für dieses Projekt wurde die SQL-Datei über phpMyAdmin importiert. Für die Anmeldung wurden Zugangsdaten des

Anmeldung wurden Zugangsdaten des



## 10 Inbetriebnahme Gesamtsystem

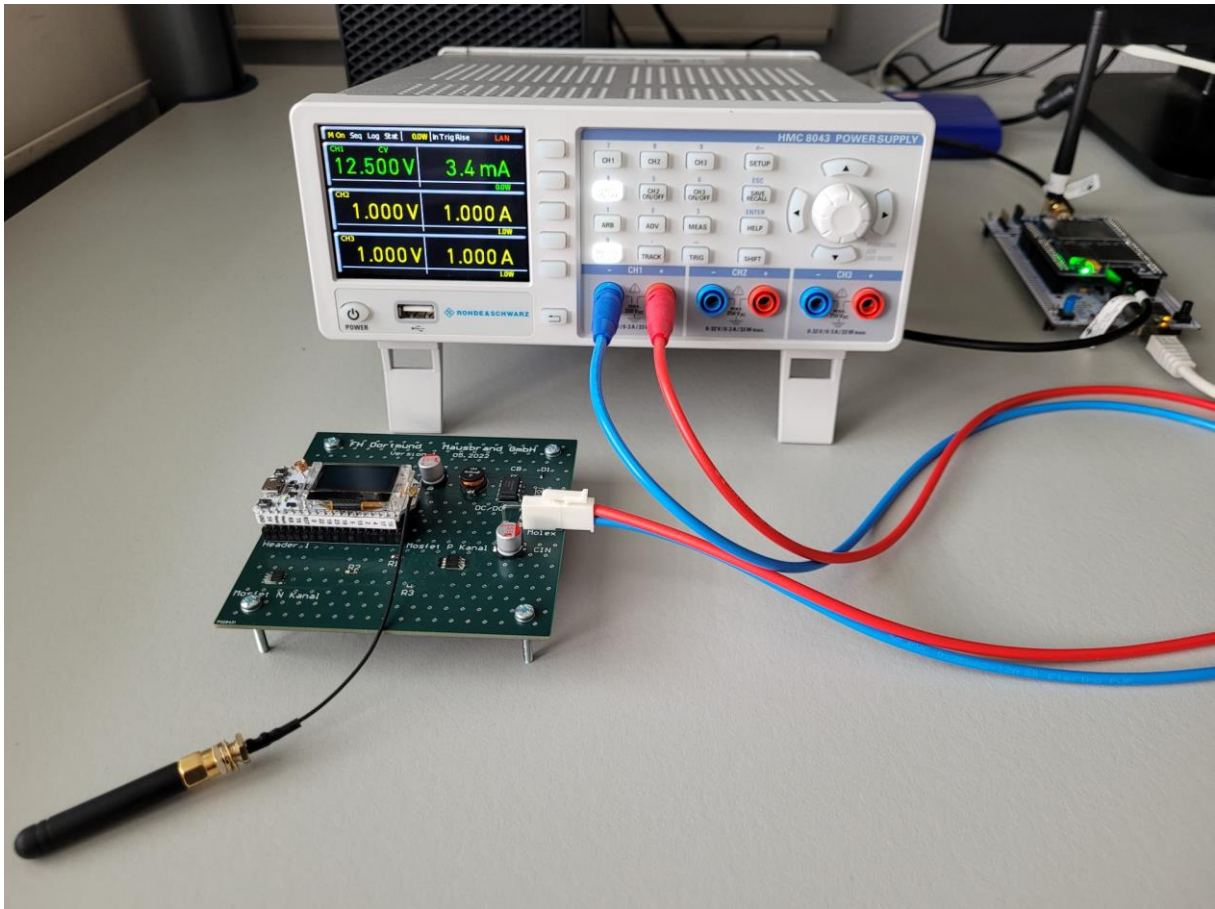


Abbildung 46: Ergebnis des Projekts

In diesem Kapitel werden die Projektergebnisse vorgestellt. Das Foto zeigt das gesamte System bestehend aus ESP-Modul, Testplatine, Netzteil und LoRaWAN Gateway. Das WiFi LoRa 32 V2 Board wurde auf die fertige Platine platziert. Der Mikrocontroller wird über einen DC/DC Wandler versorgt, der mit einer beispielhaften Spannung von 12,5V aus einem Labornetzgerät gespeist wird. Gleichzeitig misst der Mikrocontroller dabei auch die Eingangsspannung. Das Gateway befindet sich auf der rechten Seite. Dadurch können die Messdaten vom Endgerät an den Netzwerkeserver mit guter Signalstärke gesendet werden.



## 10.1 Messdaten in TTN

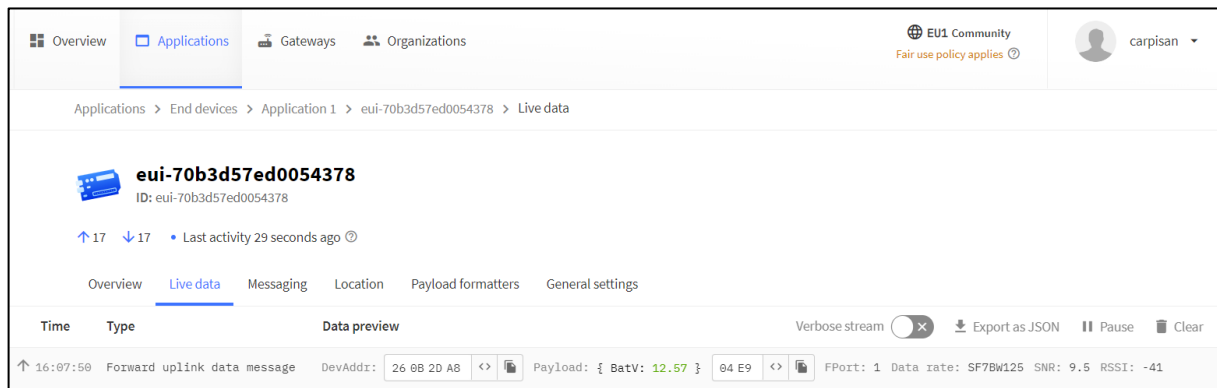


Abbildung 47: TTN Ergebnis

In der Abbildung wird gezeigt, dass die Messdaten erfolgreich auf dem TTN ankommen (siehe Abbildung 47). Der Status der Batterie kann unter dem “Live Data“ Menü beobachtet werden. Außerdem werden weitere Informationen angezeigt. Diese sind (links nach rechts):

- Time
  - Die Uhrzeit der empfangenen Nutzdatei
- Type
  - Uplink- oder Downlink
- Device Adress
  - Die Gerätadresse des Endgeräts
- Payload
  - Entschlüsselte (decodierte) Datei “BatV: 12.57“ und verschlüsselte (encodierte) Nutzdatei “04 E9“
- FPort
  - Port Nummer der empfangenen Datei
- Data rate
  - Datenrate
- SNR
  - Signal-Rausch-Verhältnis (Signal-to-Noise Ratio)
- RSSI
  - Indikator für die Empfangsfeldstärke (Received Signal Strength Indication)

Anschließend werden die Messdaten an die Webseite übertragen.

Die Abbildung 48 ist ein Ausschnitt des "Live Data" Menüs auf dem, TTN-Server. Dieses Menü ist dem Gateway zugeordnet. Hier wird die jeweilige Device Adresse des Endgeräts angezeigt. Außerdem werden auch Informationen wie Datenrate, SNR und Signalstärke zwischen dem Endgerät und dem Gateway eingeblendet.

Time	Type	Data preview
↓ 16:08:23	Send downlink message	Tx Power: 16.15 Data rate: SF7BW125
↑ 16:08:22	Receive uplink message	DevAddr: 26 0B 2D A8 <> FCnt: 16 FPort: 1 Data rate: SF7BW125 SNR: 9.5 RSSI: -45
↓ 16:07:50	Send downlink message	Tx Power: 16.15 Data rate: SF7BW125
↑ 16:07:50	Receive uplink message	DevAddr: 26 0B 2D A8 <> FCnt: 15 FPort: 1 Data rate: SF7BW125 SNR: 9.5 RSSI: -41

Abbildung 48: TTN (Gateway)

Nachdem der Standort des Gateways im TTN eingetragen wurde, wurde der Gateway-Standort unter dem Link "<https://ttnmapper.org/heatmap/>" sichtbar [29].

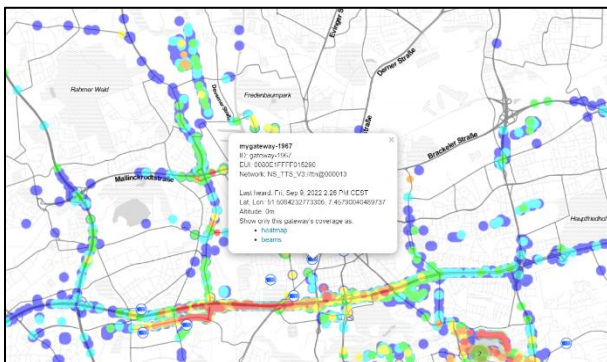


Abbildung 49: TTN-Mapper (Gateway aktiv)

In Abbildung 49 ist der TTN-Mapper abgebildet, und das Gateway ist auf der Karte sichtbar. Wenn das Gateway aktiv ist, wird der Standort des Gateways durch ein blaues Symbol dargestellt. Andernfalls wird das Symbol des Gateways rot dargestellt (siehe Abbildung 50).

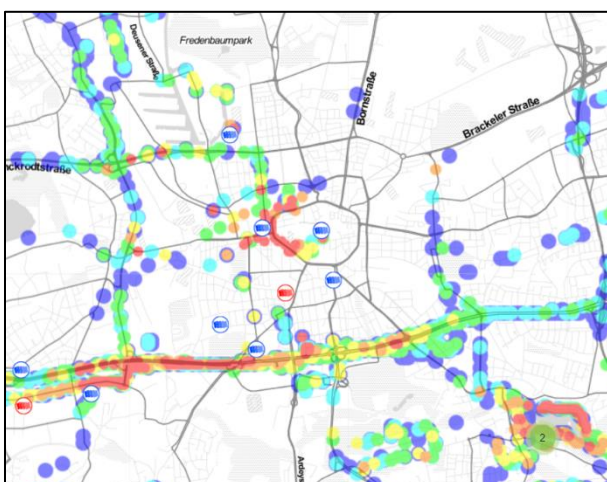
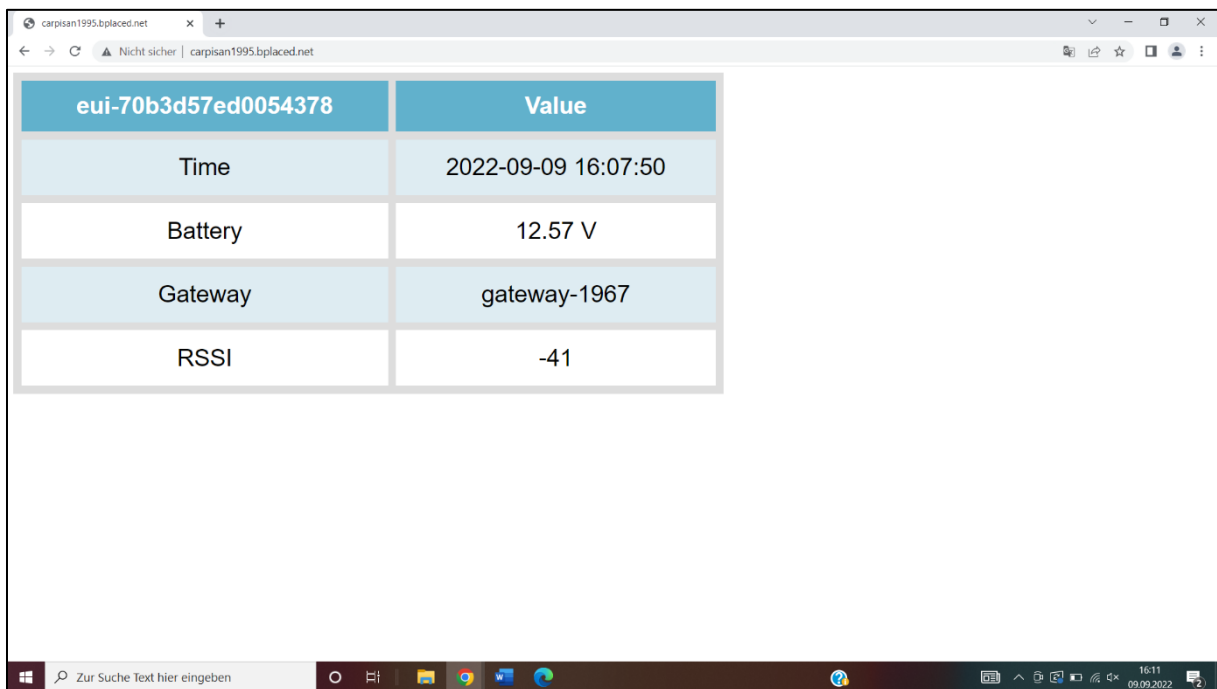


Abbildung 50: TTN-Mapper (Gateway inaktiv)

## 10.2 Anwendungsserver

Sobald der TTN-Server die Messdaten erhält, werden die Daten an die erstellte Webseite übertragen. Um die empfangenen Daten zu betrachten, muss in einem Browser der Link "carpisan1995.bplaced.net" aufgerufen werden. Diese Webseite präsentiert Informationen in tabellarischer Form. Es zeigt nicht nur den Batteriestatus, sondern auch die letzte aktualisierte Uhrzeit des gesendeten Gateways an. Außerdem werden das Gateway und die Signalstärke abgebildet. Ein Screenshot der Webseite ist in der Abbildung 51 zu sehen.



The screenshot shows a web browser window with the address bar displaying "carpisan1995.bplaced.net". The main content area contains a table with the following data:

eui-70b3d57ed0054378	Value
Time	2022-09-09 16:07:50
Battery	12.57 V
Gateway	gateway-1967
RSSI	-41

Abbildung 51: Ergebnis auf der Webseite

## 11 Ausblick

Das Ziel dieses Projekts war es, eine LoRaWAN Kommunikation zwischen einem ESP32-Endgerät und einem TTN-Netzwerkserver aufzubauen. Der Zweck der Kommunikation bestand darin, die 12-V-Batteriespannung am ADC-Eingang des Mikrocontrollers zu messen und die Messdaten verschlüsselt an den Netzwerkserver zu senden. Der Programmcode für diesen Prozess wurde in der Arduino-Entwicklungsumgebung implementiert. Außerdem wurde für die LoRaWAN Kommunikation in der Arduino-Entwicklungsumgebung eine Bibliothek namens LMIC-Library verwendet.

Das Projekt begann zunächst mit der Erarbeitung eines theoretischen Verständnisses. Anschließend folgte die Umsetzung in die Praxis. Der theoretische Teil umfasste Recherchen des Projekts. Zunächst wurde das LoRa-Modulationsverfahren vorgestellt. Das Modulationsverfahren umfasste die Chirp Spread Spectrum Techniken, die LoRa Paketstruktur und die Umsetzung der Datenübertragung in physikalische Ereignisse. Danach wurden Recherchen zu LoRaWAN durchgeführt, beispielsweise zum Kommunikationsmechanismus der drahtlosen LoRaWAN-Technologie und die dafür notwendigen Elemente. Bei dieser Kommunikation ist der Netzwerkserver ein wesentlicher Bestandteil. In diesem Projekt wurde The Things Network als LoRa-Netzwerkserver verwendet. Die Registrierung eines Moduls, das hier als Endgerät agiert, muss im Netzwerkserver erfolgen. Dadurch werden vom Netzwerkserver die ABP-Aktivierungsschlüssel generiert, die in die Entwicklungsumgebung codiert werden können. Das Projekt kann mit Hilfe geeigneter Hilfswerkzeuge gestartet werden, die in Kapitel 7 beschrieben sind.

Der fertige Programmcode wurde auf den Mikrocontroller heruntergeladen und dieser auf der fertigen Platine platziert. Die Platine wurde mit elektrischen Komponenten zur Verfügung gestellt. Im aktiven Zustand wurde die Eingangsspannung 12V gemessen und an den Netzwerkserver gesendet. Anschließend konnten die Messdaten auf dem Server beobachtet werden.

Zusätzlich wurde ein Applikationsserver für die Messdaten erstellt, da die Messdaten nicht auf dem TTN-Server gespeichert werden können. Der Anwendungsserver stellt eine Webseite dar, auf der die Messergebnisse jederzeit verfügbar sind.

Zusätzlich wurde ein I<sup>2</sup>C-Interface implementiert. In diesem Projekt sollte die Batteriespannung mit einem IC überwacht werden und dazu sollte das I<sup>2</sup>C-Interface zum Einsatz kommen. Jedoch wurde im späteren Verlauf entschieden, dass ein Spannungsteiler für die Messung eingesetzt wird.

Das Projekt kann softwareseitig und hardwareseitig erweitert werden. Es gab einige Abweichung bei den Spannungsmesswerten. Diese Abweichungen könnten durch eine Eichung minimiert werden. Der Programmcode setzt die Datenübertragung per LoRaWAN Kommunikation um und kann mit zusätzlichen Funktionen erweitert werden. Darüber hinaus können Dinge wie Temperatur- und Luftdruckmessungen implementiert werden. Dazu existieren entsprechende Sensoren z.B. BMP280, der mit hoher Messgenauigkeit misst und über I<sup>2</sup>C-Kommunikation ausgelesen werden kann. Für die Platine kann auch ein Gehäuse angefertigt werden, dass Vorschriften in Bezug auf Dichtigkeit gegen das Eindringen von Staub und Wasser erfüllt. Das LoRaWAN-Netzwerk verbreitet sich weltweit immer mehr und kann hardware- oder softwareseitig mit immer mehr Funktionen erweitert werden.

## 12 Literaturverzeichnis

- [1] elektronik-kompendium.de-1 Modulation / Modulationsverfahren  
Verfügbar am 06.07.2022  
<https://www.elektronik-kompendium.de/sites/kom/0211195.htm>
- [2] Physics and Radio-Electronics Amplitudenmodulation  
Verfügbar am 06.07.2022  
<https://www.physics-and-radio-electronics.com/blog/amplitude-modulation/>
- [3] TechnologyUK Digital Modulation  
Verfügbar am 10.11.2022  
<https://www.technologyuk.net/telecommunications/telecom-principles/digital-modulation-part-one.shtml>
- [4] Semtech-1 AN1200.22, LoRa™ Modulation Basics  
Verfügbar am 28.04.2022  
<https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf>
- [5] The Things Network-1 Spreading Factors  
Verfügbar am 26.04.2022  
<https://www.thethingsnetwork.org/docs/lorawan/spreading-factors/>
- [6] bremerfunkfreude.de DVB-T RTL-Stick SDR Anleitung  
Verfügbar am 29.11.2022  
<http://bremerfunkfreunde.de/index.php/sdr/sdr-hardware/rtl-sdr/rtl-sdr-anleitung>
- [7] Mobilefish LoRa  
Verfügbar am 07.07.2022  
<https://lora.readthedocs.io/en/latest/>

- [8] Semtech-2 SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver  
Verfügbar am 03.04.2022  
[https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKffvaF\\_Fkpgp5kzjiNyiAbqcpqh9qSjE](https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKffvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE)
- [9] The Things Network-2 LoRaWAN airtime calculator  
Verfügbar am 24.07.2022  
<https://www.thethingsnetwork.org/airtime-calculator>
- [10] revspace.nl DecodingLora  
Verfügbar am 07.08.2022  
<https://revspace.nl/DecodingLora>
- [11] The Things Network-3 Device Classes  
Verfügbar am 03.02.2022  
<https://www.thethingsnetwork.org/docs/lorawan/classes/>
- [12] The Things Network-4 LoRaWAN Architecture  
Verfügbar am 04.03.2022  
<https://www.thethingsnetwork.org/docs/lorawan/architecture/>
- [13] elektronik-komendium.de-2 LoRa / LoRaWAN - Long Range Wide Area Network  
Verfügbar am 28.06.2022  
<https://www.elektronik-kompendium.de/sites/kom/2203171.htm>
- [14] LineMetrics GmbH Für maximale LoRaWAN Reichweite: Gateways und Sensoren richtig positionieren  
Verfügbar am 28.05.2022  
<https://www.linemetrics.com/de/praxistipps-noch-mehr-lorawan-reichweite/>

- [15] LoRa Alliance-1      LoRaWAN™ 1.0.3 Specification  
Verfügbar am 02.01.2022  
<https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf>
- [16] MOKOSmart      LoRaWAN-Anwendungsfälle auf dem IoT-Markt!  
Verfügbar am 29.01.2022  
<https://www.mokosmart.com/de/lorawan-one-of-the-most-interesting-iot-technologies-on-the-market/>
- [17] LoRa Alliance-2      RP002-1.0.0 LoRaWAN Regional Parameters  
Verfügbar am 17.12.2021  
[https://lora-alliance.org/wp-content/uploads/2019/11/rp\\_2-1.0.0\\_final\\_release.pdf](https://lora-alliance.org/wp-content/uploads/2019/11/rp_2-1.0.0_final_release.pdf)
- [18] The Things Network-5      Regional Parameters  
Verfügbar am 19.07.2022  
<https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/>
- [19] The Things Network-6      The Things Network  
Verfügbar am 23.08.2022  
<https://www.thethingsnetwork.org/>
- [20] The Things Industries      LoRaWAN Version  
Verfügbar am 19.06.2022  
<https://www.thethingsindustries.com/docs/reference/glossary/#lorawan-version>
- [21] LoRa Alliance-3      TS2-1.1.0 LoRaWAN Backend Interfaces Specification  
Verfügbar am 07.03.2022  
[https://lora-alliance.org/wp-content/uploads/2020/11/TS002-1.1.0\\_LoRaWAN\\_Backend\\_Interfaces.pdf](https://lora-alliance.org/wp-content/uploads/2020/11/TS002-1.1.0_LoRaWAN_Backend_Interfaces.pdf)



- [22] The Things Network-7 Security  
Verfügbar am 29.08.2022  
<https://www.thethingsnetwork.org/docs/lorawan/security/>
- [23] NXP Semiconductors The I<sup>2</sup>C-Bus Specification and User Manual  
Verfügbar am 30.01.2022  
<https://www.pololu.com/file/0J435/UM10204.pdf>
- [24] Wikipedia Arduino (Plattform)  
Verfügbar am 30.07.2022  
[https://de.wikipedia.org/wiki/Arduino\\_\(Plattform\)#:~:text=9%20Einzelnachweise-,Geschichte,dem%20g%C3%BCnstigeren%20ATmega8%20basierte%2C%20entwickelt.](https://de.wikipedia.org/wiki/Arduino_(Plattform)#:~:text=9%20Einzelnachweise-,Geschichte,dem%20g%C3%BCnstigeren%20ATmega8%20basierte%2C%20entwickelt.)
- [25] content.instructables.com Verfügbar am 07.10.2022  
<https://content.instructables.com/ORIG/FK3/LFD8/JQKU0VBP/FK3LFD8JQKU0VBP.png>
- [26] Espressif Systems ESP32 Series Version 3.6  
esp32Datenblatt.pdf
- [27] ESP32 DAS OFFIZIELLE ESP32-HANDBUCH  
Verfügbar am 10.10.2022  
[https://www.user.tu-berlin.de/lieske/dEIn\\_Labor/Smart%20Home/ESP32/das-offizielle-esp32-handbuch-ebook.pdf](https://www.user.tu-berlin.de/lieske/dEIn_Labor/Smart%20Home/ESP32/das-offizielle-esp32-handbuch-ebook.pdf)
- [28] Heltec Automation WiFi LoRa 32 (V2.1) Phaseout  
Verfügbar am 01.07.2022  
<https://heltec.org/project/wifi-lora-32/>

- [29] ttnmapper.org TTNMapper  
Verfügbar am 18.10.2022  
<https://ttnmapper.org/heatmap/>
- [30] STmicroelectronics ARM®-based Cortex®-M7 32b MCU+FPU,  
462DMIPS, up to 1MB Flash/320+16+ 4KB RAM,  
USB OTG HS/FS, ethernet, 18 TIMs, 3 ADCs, 25  
com itf, cam & LCD  
Verfügbar am 22.09.2022  
<https://www.st.com/resource/en/datasheet/stm32f746zg.pdf>
- [31] RS Components GmbH-1 Rohde & Schwarz HMO1024 Digital-Oszilloskop, 4-  
Kanal Analog, 100MHz  
Verfügbar am 25.09.2022  
<https://de.rs-online.com/web/p/oszilloskope/8711310>
- [32] RS Components GmbH-2 Rohde & Schwarz Digital Labornetzgerät 100W  
Verfügbar am 25.08.2022  
<https://de.rs-online.com/web/p/labornetzgeraete/1448196>
- [33] The Things Network-8 Applications & Integrations  
Verfügbar am 15.07.2022  
<https://www.thethingsnetwork.org/docs/applications-and-integrations>
- [34] The Things Network- 9 The Things Network Forum  
Verfügbar am 25.02.2022  
<https://www.thethingsnetwork.org/forum/t/how-to-persist-lmic-otaa-parameters-with-an-esp32/35032>

- [35] The Things Network-10      What are LoRa and LoRaWAN  
Verfügbar am 26.05.2022  
<https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/>
- [36] mobilefish.com              LORA/LORAWAN TUTORIAL 14  
Verfügbar am 15.11.2022  
[https://www.mobilefish.com/download/lora/lora\\_part14.pdf](https://www.mobilefish.com/download/lora/lora_part14.pdf)

## Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass die von mir vorgelegte Arbeit selbstständig und ohne unzulässige fremde Hilfe erstellt worden ist. Alle verwendeten Quellen sind in der Arbeit so aufgeführt, dass Art und Umfang der Verwendung nachvollziehbar sind.

---

Ort, Datum

---

Unterschrift